

The L^AT_EX 2_ε Sources

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

2005/12/01

Contents

a	ltdirchk.dtx	1
1	L^AT_EX System Dependent Initialisations	1
2	Initialisation	2
2.1	INITEX	2
2.2	Some bits of 2e	3
3	texsys.cfg	4
3.1	texsys.cfg	4
3.2	UNIX (web2c)	5
3.3	UNIX (other)	6
3.4	MSDOS (emtex)	6
3.5	MSDOS (other)	6
3.6	VMS (DECUS T _E X, PD VMS 3.6)	6
3.7	VMS (???)	6
3.8	MACINTOSH (OzT _E X 1.6)	7
3.9	MACINTOSH (other)	7
3.10	FAKE EXAMPLE	7
4	Setting \@currdir	8
5	Setting \input@path	9
6	Filename Parsing	10
7	T_EX Versions	11

8	ltxcheck.tex	12
b	ltplain.dtx	13
9	Plain T_EX	13
c	ltvers.dtx	25
10	Version Identification	25
d	ltdefns.dtx	27
11	Definitions	27
11.1	Initex initialisations	27
11.2	Saved versions of T _E X primitives	27
11.3	Command definitions	28
11.4	Robust commands and protect	36
11.5	Internal defining commands	38
11.6	Commands for Autoloading	40
e	ltdllc.dtx	41
12	Counters	41
f	ltcntrl.dtx	43
13	Program control structure	43
g	lterror.dtx	47
14	Error handling	47
14.1	General commands	47
14.2	Specific errors	52
h	ltpar.dtx	57
15	Paragraphs	57
15.1	Implementation	57
i	ltspc.dtx	59

16 Spacing	59
16.1 User Commands	59
16.2 Chris' comments	59
16.3 Some immediate actions	61
16.4 The code	62
16.5 Vertical spacing	66
16.6 Horizontal space (and breaks)	68
 j ltlogos.dtx	 71
17 Logos	71
 k ltfiles.dtx	 72
18 File Handling	72
18.1 Safe Input Macros	78
18.2 Listing files	80
 l ltoutenc.dtx	 82
19 Font encodings	82
19.1 Removing encoding-specific commands	84
19.2 The order of declarations	85
19.3 Docstrip modules	85
19.4 Definitions for the kernel	86
19.4.1 Declaration commands	86
19.4.2 Hyphenation	92
19.4.3 Miscellania	93
19.4.4 Default encodings	93
19.4.5 Math material	95
19.5 Definitions for the OT1 encoding	96
19.6 Definitions for the T1 encoding	98
19.7 Definitions for the OMS encoding	102
19.8 Definitions for the OML encoding	103
19.9 Definitions for the OT4 encoding	103
19.10Definitions for the TS1 encoding	105
 20 Package files	 109
20.1 The fontenc package	109
20.2 The textcomp package	111
20.2.1 Supporting oldstyle digits	118
20.2.2 Subset encoding defaults	119
 m ltcounts.dtx	 121
21 Counters and Lengths	121
21.1 Environment Counter Macros	121

n ltnlength.dtx	125
22 Lengths	125
o ltfssbas.dtx	126
23 Autoloading parts of NFSS	126
24 Preliminary macros	126
25 Macros for setting up the tables	127
26 Selecting a new font	132
26.1 Macros for the user	132
26.2 Macros for loading fonts	136
27 Assigning math fonts to <i>versions</i>	141
p ltfsstrc.dtx	146
28 Introduction	146
29 A driver for this document	146
30 The Implementation	147
31 Handling Options	147
32 Macros common to fam.tex and tracefnt.sty	149
32.1 General font loading	149
32.2 Math fonts setup	154
32.2.1 Outline of algorithm for math font sizes	154
32.2.2 Code for math font size setting	155
32.2.3 Other code for math	156
33 Scaled font extraction	158
33.1 Sizefunctions	165
q ltfsscnp.dtx	169
34 Compatibility code for NFSS release 1	169
r ltfssdcl.dtx	174
35 Interface Commands	174
s ltfssini.dtx	196

36 NFSS Initialisation	196
36.1 Providing math <i>versions</i>	196
36.2 Miscellaneous	197
 t fontdef.dtx	 202
37 Introduction	202
38 Customization	202
39 The docstrip modules	203
40 A driver for this document	203
41 The fonttext.ltx file	204
41.1 Encodings	204
41.2 Defaults	205
42 The fontmath.ltx file	206
42.1 The font encodings used	206
42.1.1 Symbolfont and Alphabet declarations	207
42.2 Math font sizes	207
42.3 The math symbol assignments	208
42.3.1 The letters	208
42.3.2 The digits	209
42.3.3 Punctuation, brace, etc. keys	209
42.3.4 Delimitercodes for characters	210
42.4 Symbols accessed via control sequences	210
42.4.1 Greek letters	210
42.4.2 Ordinary symbols	211
42.4.3 Large Operators	212
42.4.4 Binary symbols	212
42.4.5 Relations	213
42.4.6 Arrows	214
42.4.7 Punctuation symbols	215
42.4.8 Math accents	215
42.4.9 Radicals	215
42.4.10 Over and under something, etc	216
42.4.11 Delimiters	216
42.5 Math versions of text commands	217
42.6 Other special functions and parameters	218
42.6.1 Biggggg	218
42.6.2 The log-like functions	218
42.6.3 Parameters	218
 43 Default cfg files	 218
 u preload.dtx	 220

44 Overview	220
45 Customization	220
46 Module switches for the DOCSTRIP program	221
47 A driver for this document	221
48 The code	221
 v ltfntcmd.dtx	 224
49 Introduction	224
50 The implementation	226
51 Initialization	231
 w ltpageno.dtx	 232
52 Page Numbering	232
 x ltxref.dtx	 233
53 Cross Referencing	233
53.1 Cross Referencing	233
53.2 An extension of counter referencing	235
 y ltmiscen.dtx	 237
54 Miscellaneous Environments	237
54.1 Environments	237
54.2 Center, Flushright, Flushleft	241
54.3 Verbatim	243
 z ltmath.dtx	 246
55 Math setup	246
55.1 Math commands based on plain $\mathrm{T}_{\mathrm{E}}\mathrm{X}$	246
55.1.1 The log-like functions	246
55.1.2 Biggggg	247
55.1.3 The UNSORTED Rest	247
55.2 Math Environments	250
55.3 External options to the standard document classes	254
55.3.1 Left equation numbering	254
55.3.2 Flush left equations	254

A	ltlists.dtx	256
56	List, and related environments	256
56.1	List and Trivlist	257
56.2	Vertical Spacing (skips)	258
56.3	Penalties	258
56.4	Horizontal Spacing (dimens)	258
56.5	Default Values	259
56.6	Itemize and Enumerate	268
B	ltboxes.dtx	271
57	L^AT_EX Box commands	271
57.1	Some low-level constructs	281
C	lftab.dtx	282
58	Tabbing, Tabular and Array Environments	282
58.1	tabbing	282
58.2	array and tabular environments	290
D	ltpictur.dtx	304
59	Picture Mode	304
59.1	Curves	323
E	ltthm.dtx	326
60	Theorem Environments	326
F	ltsect.dtx	330
61	Sectioning Commands	330
61.1	The Title	330
61.2	Sectioning	331
61.2.1	Initializations	337
61.3	Table of Contents etc.	337
61.3.1	Convention	337
61.3.2	Commands	337
G	ltoat.dtx	340
62	Floats	340
62.1	Floating Environments	340
62.2	Footnotes	351

H	ltxglo.dtx	357
63	Index and Glossary Generation	357
I	ltxbibl.dtx	359
64	Bibliography Generation	359
64.1	Default definitions	362
J	ltxpage.dtx	363
65	Page styles and related commands	363
65.1	Page Style Commands	363
65.2	How a page style makes running heads and feet	363
65.3	marking conventions	363
K	ltxoutput.dtx	366
66	Output Routine	366
66.1	Floats	366
66.1.1	Kludgeins	407
66.1.2	Float control	409
66.1.3	Float placement parameters	418
L	ltxclass.dtx	421
67	Introduction	421
68	User interface	421
68.1	Option processing	422
69	Class and Package interface	422
69.1	Class name and version	422
69.2	Package name and version	423
69.3	Requiring other packages	423
69.4	Declaring new options	424
69.5	Safe Input Macros	424
70	Implementation	425
70.1	Hooks	435
70.2	Providing shipment	436
71	After Preamble	438
M	ltxhyphen.dtx	440

N	ltfinal.dtx	442
72	Final settings	442
72.1	Debugging	442
72.2	Typesetting parameters	442
72.3	Lccodes for hyphenation	442
72.4	Hyphenation	444
72.5	Font loading	444
72.6	Input encoding	444
72.7	Lccodes and uccodes	446
72.8	Applying Patch files	447
72.9	Freeing Memory	448
72.10	Initialise file list	449
72.11	Dumping the format	449
O	ltpatch	450
	Index	451

File a

ltdirchk.dtx

1 L^AT_EX System Dependent Initialisations

This file implements the semi-automatic determination of various system dependent parts of the initialisation. 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`, `\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 initialisation 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 , $3.0 \leq v \leq 3.14$
 $\langle undefined \rangle$ 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.
...
```

```
1.3 \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.
! .
...
```

```
1.3 \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 Initialisation

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

2.1 INITEX

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

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.

```

6 <initex>\fi
7 \catcode'\{=1
8 \catcode'\}=2
9 \catcode'\#=6
10 \catcode'\^=7
11 \chardef\active=13
12 \catcode'\@=11
13 \countdef\count@=255
14 \let\bgroup={ \let\egroup=}
15 \ifx\@input\@undefined\let\@input\input\fi
16 \ifx\@end\@undefined\let\@end\end\fi
17 \chardef\@inputcheck0
18 \chardef\sixt@n=16
19 \newlinechar'\^^J
20 \def\typeout{\immediate\write17}
21 \def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
22 \do\#\do\^{\do\_ \do\% \do\~}
23 \def\@makeother#1{\catcode'#1=12\relax}
24 \def\space{ }
25 \def\@tempswafalse{\let\if@tempswa\iffalse}
26 \def\@tempswatrue{\let\if@tempswa\iftrue}
27 \let\if@tempswa\iffalse
28 \def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
29 \iterate \let\iterate\relax}
30 \let\repeat\fi
31 </initex>

```

2.2 Some bits of 2e

```

32 <*2ekernel>
33 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}
34 \long\def\@firstoftwo#1#2{#1}
35 \long\def\@secondoftwo#1#2{#2}

```

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

```

36 \def\ProvidesFile#1{%
37 \begingroup
38 \catcode'\ 10 %
39 \ifnum \endlinechar<256 %
40 \ifnum \endlinechar>\m@ne
41 \catcode\endlinechar 10 %
42 \fi
43 \fi
44 \@makeother\%
45 \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}]
46 \def\@providesfile#1[#2]{%
47 \wlog{File: #1 #2}%
48 \@addtofilelist{ #2}%
49 \endgroup}
50 \long\def\@addtofilelist#1{}
51 \def\@empty{}
52 \catcode'\%=12
53 \def\@percentchar{%}
54 \catcode'\%=14
55 \let\@currdir\@undefined

```

```

56 \let\input@path\@undefined
57 \let\filename@parse\@undefined

\strip@prefix

58 \def\strip@prefix#1>{}
59 \</2ekernel>

```

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.

```

60 \*docstrip
61 \openin15=texsys.cfg
62 \ifeof15
63 \typeout{** Writing a default texsys.cfg}
64 \immediate\openout15=texsys.cfg
65 \begingroup
66 \catcode'\^^M\active%
67 \let^^M\par%
68 \def\reserved@a#1^^M{%
69   \def\reserved@b{#1}%
70   \ifx\reserved@b\reserved@c\endgroup\else%
71     \immediate\write15{#1}%
72     \expandafter\reserved@a\fi}%
73 \def\reserved@d#1START^^M{\let\do\@makeother\dospecials\reserved@a}%
74 \catcode'\%=12
75 \def\reserved@c{%END}
76 \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.

```
77 %\def\@currdir{./}
```

```
78 %\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 `/`.

```
79 % \def\@currdir{./}
80 % \def\input@path{%
81 %   {/usr/local/lib/tex/inputs/distrib/}%
82 %   {/usr/local/lib/tex/inputs/contrib/}%
83 %   {/usr/local/lib/tex/inputs/local/}%
84 % }
```

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.

```
85 % \def\@currdir{./}
86 % \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.

```
87 % \def\@currdir{./}
88 % \def\input@path{%
89 %   {c:/tex/inputs/distrib/}%
90 %   {c:/tex/inputs/contrib/}%
91 %   {c:/tex/inputs/local/}%
92 % }
```

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.

```
93 % \def\@currdir{[] }
94 % \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:

```
95 % \def\@currdir{[] }
96 % \def\input@path{%
97 %   {tex_inputs:}%
98 %   {SOMEDISK:[SOME.TEXT.DIRECTORY]}%
99 % }
```

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.

```
100 % \def\@currdir{:}
101 % \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.

```
102 % \def\@currdir{:}
103 % \def\input@path{%
104 %   {Hard-Disk:Applications:TeX:TeX-inputs:}%
105 %   {Hard-Disk:Applications:TeX:My-inputs:}%
106 % }
```

3.10 FAKE EXAMPLE

This example is for an operating system that has filenames of the form `<area>name`. For maximum compatability 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:

```
107 % \def\@dir#1#2 {%
108 %   \@d@r{#1}#2..\@nil}
109 % \def\@d@r#1#2.#3.#4\@nil{%
110 %   <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
111 %
112 % \def\@currdir{\@dir{}}
113 % \def\input@path{%
114 %   {\@dir{area.one}}}%
115 %   {\@dir{area.two}}}%
116 % }
```

END

```
117 \immediate\closeout15
```

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

```
118 \else
119 \typeout{** Using the existing texsys.cfg}
120 \closein15
121 \input texsys.cfg
122 \fi
123 </docstrip>
```

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

```
124 <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.

```
125 \begingroup
126 \count@ \time
127 \divide \count@ 60
128 \count2 = - \count@
129 \multiply \count2 60
130 \advance \count2 \time
```

`\today` The current date and time stamp.

```
131 \edef \today {%
132   \the \year / \two@digits {\the \month} / \two@digits {\the \day} : %
133   \two@digits {\the \count@} : \two@digits {\the \count2}}
```

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

```
134 \immediate \openout15 = texsys.aux
135 \immediate \write15 {\today ^^J}
136 \immediate \closeout15 %
```

#1 is the file to try, #2 is what to do on success, #3 on failure.

```
137 \def \IfFileExists #1 #2 #3 {%
138   \openin \@inputcheck #1 %
139   \ifeof \@inputcheck
140     #3 \relax
141   \else
142     \read \@inputcheck to \reserved@a
143     \ifx \reserved@a \today
144       \typeout {#1 found} #2 \relax
145     \else
146       \typeout {BAD: old file \reserved@a (should be \today)} %
147       #3 \relax
148     \fi
149   \fi
150   \closein \@inputcheck}
151 \endlinechar = -1
```

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

```
152 \ifx \@currdir \@undefined
153   \IfFileExists {./texsys.aux} {\gdef \@currdir {./}} %
154   {\IfFileExists {[]texsys.aux} {\gdef \@currdir {[]}} %
155   {\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.

```

156 \ifx\@currdir\@undefined
157   \global\let\@currdir\@empty
158   \typeout{^^J^^J%
159     !! No syntax for the current directory could be found^^J%
160   }%
161 \fi

```

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

```

162 \else
163   \IfFileExists{\@currdir texsys.aux}{-}{%
164     \edef\reserved@a{\errhelp{%
165       texsys.cfg specifies the current directory syntax to be^^J%
166       \meaning\@currdir^^J%
167       but this does not work on this system.^^J%
168       Remove texsys.cfg and restart.}}\reserved@a
169   \errmessage{Bad texsys.cfg file: \noexpand\@currdir}\@end}

```

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

```

170 \fi

171 \immediate\closeout15 %
172 \endgroup

173 \typeout{^^J^^J%
174   \noexpand\@currdir set to:
175   \expandafter\strip@prefix\meaning\@currdir.^^J%
176 }

```

Stop here if the file is being used unstripped.

```

177 <*\docstrip>
178 \relax\endinput
179 </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_ε 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_ε. This will check, amongst 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.

```

180 \typeout{^^J%
181   Assuming \noexpand\openin and \noexpand\input^^J%
182   \ifx\input@path\@undefined

```

```

\input@path has not been pre-defined.
183     have the same search path.^^J%
184     \else
\input@path has been defined in texsys.cfg.
185     have different search paths.^^J%
186     LaTeX will use the path specified by \noexpand\input@path:^^J%
187     \fi
188     }

```

6 Filename Parsing

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

```

189 \ifx\filename@parse\@undefined
190   \def\reserved@a{.}\ifx\@currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like
UNIX...
191   \typeout{^^JDefining UNIX/DOS style filename parser.^^J}
192   \def\filename@parse#1{%
193     \let\filename@area\@empty
194     \expandafter\filename@path#1/\}
Search for the last /.
195   \def\filename@path#1/#2\{%
196     \ifx\#2\%
197       \def\reserved@a{\filename@simple#1.\}%
198     \else
199       \edef\filename@area{\filename@area#1/}%
200       \def\reserved@a{\filename@path#2\}%
201     \fi
202     \reserved@a}
203   \else\def\reserved@a{[]}\ifx\@currdir\reserved@a
\filename@parse was not specified in texsys.cfg, but \@currdir looks like
VMS...
204   \typeout{^^JDefining VMS style filename parser.^^J}
205   \def\filename@parse#1{%
206     \let\filename@area\@empty
207     \expandafter\filename@path#1]\}
Search for the last ].
208   \def\filename@path#1]#2\{%
209     \ifx\#2\%
210       \def\reserved@a{\filename@simple#1.\}%
211     \else
212       \edef\filename@area{\filename@area#1]}%
213       \def\reserved@a{\filename@path#2\}%
214     \fi
215     \reserved@a}
216   \else\def\reserved@a{:}\ifx\@currdir\reserved@a

```

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

```

217 \typeout{^^JDefining Mac style filename parser.^^J}
218 \def\filename@parse#1{%
219 \let\filename@area\@empty
220 \expandafter\filename@path#1:\}

```

Search for the last :

```

221 \def\filename@path#1:#2\{%
222 \ifx\#2\%
223 \def\reserved@a{\filename@simple#1.\}%
224 \else
225 \edef\filename@area{\filename@area#1:}%
226 \def\reserved@a{\filename@path#2\}%
227 \fi
228 \reserved@a}

```

```

229 \else

```

\filename@parse was not specified in texsys.cfg. So just make a simple parser that always sets \filename@area to empty.

```

230 \typeout{^^JDefining generic filename parser.^^J}
231 \def\filename@parse#1{%
232 \let\filename@area\@empty
233 \expandafter\filename@simple#1.\}
234 \fi\fi\fi

```

\filename@simple is used by all three versions. Finally we can split off the extension.

```

235 \def\filename@simple#1.#2\{%
236 \ifx\#2\%
237 \let\filename@ext\relax
238 \else
239 \edef\filename@ext{\filename@dot#2\}%
240 \fi
241 \edef\filename@base{#1}}

```

Remove a final dot, added earlier.

```

242 \def\filename@dot#1.\{#1}
243 \else

```

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

```

244 \typeout{^^J^^J%
245 \noexpand\filename@parse was defined in texsys.cfg:^^J%
246 \expandafter\strip@prefix\meaning\filename@parse.^^J%
247 }
248 \fi

```

7 T_EX Versions

\@TeXversion T_EX versions older than 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 T_EX3.14 would be detected

as T_EX3, but L^AT_EX only needs to take account of T_EX's older than 3, or between 3 and 3.14.

```
249 \ifx\@TeXversion\undefined
250   \ifx\undefined\inputlineno
251     \def\@TeXversion{2}
252   \else
253     {\catcode'\^^J=\active
254      \def\reserved@a#1#2\@@{\if#1\string^3\fi}
255      \edef\reserved@a{\expandafter\reserved@a\string^^J\@@}
256      \ifx\reserved@a\empty\else\gdef\@TeXversion{3}\fi}
257   \fi
258 \fi

259 </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

lplain.dtx

9 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 (*2kernel | autoload)
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 \outer\def^L{\par}% ascii form-feed is \outer\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 conict 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

\@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
```

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 conicts may occur, especially when two or more macro packages are being used at the same time.

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

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
```

`\m@ne` The constant `-1`.

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

`\wlog` Write on log file (only)

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

`\count@` Here are abbreviations for the names of scratch registers that don’t need to be allocated.

```
\dimen@ 41 \countdef\count@=255
\dimen@i 42 \dimendef\dimen@=0
\skip@ 43 \dimendef\dimen@i=1 % global only
\toks@ 44 \dimendef\dimen@ii=2
45 \skipdef\skip@=0
46 \toksdef\toks@=0
```

`\newcount` Now, we define `\newcount`, `\newbox`, etc. so that you can say `\newcount\foo` and `\foo` will be defined (with `\countdef`) to be the next counter.

`\newdimen` To find out which counter `\foo` is, you can look at `\allocationnumber`.

`\newskip` Since there’s no `\boxdef` command, `\chardef` is used to define a `\newbox`, `\newinsert`, `\newfam`, and so on.

`\newhelp`

`\newtoks`

L^AT_EX change: remove `\outer` from `\newcount` and `\newdimen` (FMi) This is necessary to use `\newcount` inside `\if...` later on. Also remove from `\newskip`, `\newbox` `\newwrite` and `\newfam` (DPC) to save later redefinition.

```
47 \def\newcount{\alloc@0\count\countdef\insc@unt}
48 \def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
49 \def\newskip{\alloc@2\skip\skipdef\insc@unt}
50 \def\newmuskip{\alloc@3\muskip\muskipdef\@ccclvi}
51 \def\newbox{\alloc@4\box\chardef\insc@unt}
52 \def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
53 \def\newtoks{\alloc@5\toks\toksdef\@ccclvi}
```

`\newread`

```
\newwrite 54 \def\newread{\alloc@6\read\chardef\sixt@@n}
55 \def\newwrite{\alloc@7\write\chardef\sixt@@n}
```

L^AT_EX defines `\newfam` in `ltfss.dtx`.

```
\def\newfam{\alloc@8\fam\chardef\sixt@@n}
```

`\newlanguage`

```
56 \def\newlanguage{\alloc@9\language\chardef\@ccclvi}
```

`\alloc@`

```
57 \def\alloc@#1#2#3#4#5{\global\advance\count1#1\@ne
58 \ch@ck#1#4#2% make sure there's still room
59 \allocationnumber\count1#1%
60 \global#3#5\allocationnumber
61 \wlog{\string#5=\string#2\the\allocationnumber}}
```

`\newinsert`

```
62 \def\newinsert#1{\global\advance\insc@unt \m@ne
63 \ch@ck0\insc@unt\count
64 \ch@ck1\insc@unt\dimen
65 \ch@ck2\insc@unt\skip
66 \ch@ck4\insc@unt\box
67 \allocationnumber\insc@unt
68 \global\chardef#1\allocationnumber
69 \wlog{\string#1=\string\insert\the\allocationnumber}}
```

`\ch@ck`

```
70 \</2kernel | autoloading>
71 \<2kernel | autoloading | autoerr>
72 \gdef\ch@ck#1#2#3{%
73 \ifnum\count1#1<#2\else
74 \!autoloading \errmessage{No room for a new #3}%
75 \<autoloading> \@autoerr\ch@ck#1#2#3%
76 \fi}
77 \</2kernel | autoloading | autoerr>
78 \<2kernel | autoloading>
```

`\maxdimen` Here are some examples of allocation.

```
\hideskip 79 \newdimen\maxdimen \maxdimen=16383.99999pt % the largest legal <dimen>
80 \newskip\hideskip \hideskip=-1000pt plus 1fill % negative but can grow
```

```

\p@
\z@ 81 \newdimen\p@ \p@=1pt % this saves macro space and time
\z@skip 82 \newdimen\z@ \z@=0pt % can be used both for 0pt and 0
\voidb@x 83 \newskip\z@skip \z@skip=0pt plus0pt minus0pt
84 \newbox\voidb@x % permanently void box register

```

```
85 \message{compatibility for TeX 2, }
```

If this file is used in an old T_EX we define the new features of T_EX 3.0 as simple macros or counters so that files that uses these features can be processed in such an environment (They will however produce some other results).

```
86 \ifx\@undefined\inputlineno
87 \newcount\inputlineno

```

This could be used to detect that an old T_EX is in force

```
88 \inputlineno-1
```

Extra test for MLTeX 2, RmS 91/11/07.

```

89 \ifx\@undefined\language
90 \newcount\language
91 \fi
92 \newcount\lefthyphenmin
93 \newcount\rightthyphenmin
94 \newcount\errorcontextlines
95 \newcount\holdinginserts
96 \newdimen\emergencystretch
97 \newcount\badness
98 \let\noboundary\relax
99 \newcount\setlanguage
100 \fi

```

Assign initial values to T_EX's parameters

```
101 \message{parameters,}
```

All of T_EX'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.

```

102 \pretolerance=100
103 \tolerance=200 % INITEX sets this to 10000
104 \hbadness=1000
105 \vbadness=1000
106 \linepenalty=10
107 \hyphenpenalty=50
108 \exhyphenpenalty=50
109 \binoppenalty=700
110 \relpenalty=500
111 \clubpenalty=150
112 \widowpenalty=150
113 \displaywidowpenalty=50
114 \brokenpenalty=100
115 \predisplaypenalty=10000

\postdisplaypenalty=0
\interlinepenalty=0

```

```

\floatingpenalty=0, set during \insert
\outputpenalty=0, set before TeX enters \output
116 \doublehyphendemerits=10000
117 \finalhyphendemerits=5000
118 \adjdemerits=10000

\looseness=0, cleared by TeX after each paragraph
\pausing=0
\holdinginserts=0
\tracingonline=0
\tracingmacros=0
\tracingstats=0
\tracingparagraphs=0
\tracingpages=0
\tracingoutput=0
119 \tracinglostchars=1

\tracingcommands=0
\tracingrestores=0
\language=0
120 \uchyph=1

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

\endlinechar='^M % INITEX does this
\newlinechar=-1 \LaTeX\ sets this in ltdefs.dtx.
123 \delimiterfactor=901

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

In LATEX we don't want box information in the transcript unless we do a full
tracing.
124 \showboxbreadth=-1
125 \showboxdepth=-1
126 \errorcontextlines=-1

127 \hfuzz=0.1pt
128 \vfuzz=0.1pt
129 \overfullrule=5pt
130 \maxdepth=4pt
131 \splitmaxdepth=\maxdimen
132 \boxmaxdepth=\maxdimen

```

```

\lineskiplimit=0pt, changed by \normalbaselines
133 \delimitershortfall=5pt
134 \nulldelimiterspace=1.2pt
135 \scriptspace=0.5pt

\mathsurround=0pt
\predisplaysize=0pt, set before TeX enters $$
\displaywidth=0pt, set before TeX enters $$
\displayindent=0pt, set before TeX enters $$
136 \parindent=20pt

\hangindent=0pt, zeroed by TeX after each paragraph
\hoffset=0pt
\voffset=0pt

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

137 \parskip=0pt plus 1pt
138 \abovedisplayskip=12pt plus 3pt minus 9pt
139 \abovedisplayshortskip=0pt plus 3pt
140 \belowdisplayskip=12pt plus 3pt minus 9pt
141 \belowdisplayshortskip=7pt plus 3pt minus 4pt

\leftskip=0pt
\rightskip=0pt
142 \topskip=10pt
143 \splittopskip=10pt

\tabskip=0pt
\spaceskip=0pt
\xspaceskip=0pt
144 \parfillskip=0pt plus 1fil

\normalbaselineskip We also define special registers that function like parameters:
\normallineskip 145 \newskip\normalbaselineskip \normalbaselineskip=12pt
\normallineskiplimit 146 \newskip\normallineskip \normallineskip=1pt
147 \newdimen\normallineskiplimit \normallineskiplimit=0pt

\interfootlinepenalty
148 \newcount\interfootnotelinepenalty \interfootnotelinepenalty=100

Definitions for preloaded fonts

\magstephalf
\magstep 149 \def\magstephalf{1095 }
150 \def\magstep#1{\ifcase#1 \@m\or 1200\or 1440\or 1728\or
151 2074\or 2488\fi\relax}

Macros for setting ordinary text

```

```

\frenchspacing
\nonfrenchspacing 152 \def\frenchspacing{\sfcode'\.\@m \sfcode'\?\@m \sfcode'\!\@m
153 \sfcode'\:\@m \sfcode'\;\@m \sfcode'\,\@m}
154 \def\nonfrenchspacing{\sfcode'\.3000\sfcode'\?3000\sfcode'\!3000%
155 \sfcode'\:2000\sfcode'\;1500\sfcode'\,1250 }

\normalbaselines
156 \def\normalbaselines{\lineskip\normallineskip
157 \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}

\M Save a bit of space by using \let here.
\I 158 \def^^M{\ } % control <return> = control <space>
159 \let^^I^^M % same for <tab>

\lq
\rq 160 \def\lq{' }
161 \def\rq{' }

\lbrack
\rbrack 162 \def\lbrack{[ }
163 \def\rbrack{[ ]}

\aa These are not from plain.tex but they are similar to other commands found here
\AA and nowhere else, being alternate input forms for characters.
164 \def \aa {\r a}
165 \def \AA {\r A}

\endgraf
\endline 166 \let\endgraf=\par
167 \let\endline=\cr

\space
168 \def\space{ }

\empty This probably ought to go altogether, but let it to the LATEX version to save space.
169 \let\empty@empty

\null
170 \def\null{\hbox{}}

\bgroup
\egroup 171 \let\bgroup={
172 \let\egroup=}

\obeylines In \obeylines, we say \let^^M=\par instead of \def^^M{\par} since this allows,
\obeyspaces for example, \let\par=\cr \obeylines \halign{...
173 {\catcode'\^^M=\active % these lines must end with %
174 \gdef\obeylines{\catcode'\^^M\active \let^^M\par}%
175 \global\let^^M\par} % this is in case ^^M appears in a \write
176 \def\obeyspaces{\catcode'\ \active}
177 {\obeyspaces\global\let =\space}

```

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

```
178 \long\def \loop #1\repeat{%
179   \def\iterate{#1\relax % Extra \relax
180     \expandafter\iterate\fi
181   }%
182   \iterate
183   \let\iterate\relax
184 }
```

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

```
185 \let\repeat=\fi
```

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

`\nointerlineskip`

```
\offinterlineskip 186 \def\nointerlineskip{\prevdepth-\@m\p@}
187 \def\offinterlineskip{\baselineskip-\@m\p@
188   \lineskip\z@ \lineskiplimit\maxdimen}
```

`\vglue`

```
\hglue 189 \def\vglue{\afterassignment\vgl@ \skip@=}
190 \def\vgl@{\par \dimen@ \prevdepth \hrule \@height\z@
191   \nobreak\vskip\skip@ \prevdepth\dimen@}
192 \def\hglue{\afterassignment\hgl@ \skip@=}
193 \def\hgl@{\leavevmode \count@ \spacefactor \vrule \@width\z@
194   \nobreak\hskip\skip@ \spacefactor\count@}
```

L^AT_EX defines `~` in `ltdefns.dtx`.

`\slash`

```
195 \def\slash{/\penalty\exhyphenpenalty} % a '/' that acts like a '-'
```

`\break`

```
\nobreak 196 \def\break{\penalty-\@M}
\allowbreak 197 \def\nobreak{\penalty \@M}
198 \def\allowbreak{\penalty \z@}
```

`\filbreak`

```
\goodbreak 199 \def\filbreak{\par\vfil\penalty-200\vfilneg}
200 \def\goodbreak{\par\penalty-500 }
```

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

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

`\removelastskip`

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

```

\smallbreak
\medbreak 203 \def\smallbreak{\par\ifdim\lastskip<\smallskipamount
\bigbreak 204 \removeatlastskip\penalty-50\smallskip\fi}
205 \def\medbreak{\par\ifdim\lastskip<\medskipamount
206 \removeatlastskip\penalty-100\medskip\fi}
207 \def\bigbreak{\par\ifdim\lastskip<\bigskipamount
208 \removeatlastskip\penalty-200\bigskip\fi}

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

\underbar Due to LATEX's redefinition of \underline plain TEX's \underbar can be done in
a simpler fashion (but do we need it at all?).
210 \def\underbar#1{\underline{\sbox\tw@{#1}\dp\tw@\z@\box\tw@}}

\strutbox LATEX sets \strutbox in \set@fontsize.
\strut 211 \newbox\strutbox
212 \def\strut{\relax\ifmmode\copy\strutbox\else\unhcopy\strutbox\fi}

\hidewidth For alignment entries that can stick out.
213 \def\hidewidth{\hskip\hideskip}

\narrower
214 \def\narrower{%
215 \advance\leftskip\parindent
216 \advance\rightskip\parindent}

LATEX defines \ae and similar commands elsewhere.
217 \chardef\%='\%
218 \chardef\&='\&
219 \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
220 \def\leavevmode{\unhbox\voidb@x}

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

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

\oalign
\o@lign 223 \def\oalign#1{\leavevmode\top{\baselineskip\z@skip \lineskip.25ex%
\oalign 224 \ialign{##\crr#1\crr}}}}
225 \def\o@lign{\lineskiplimit\z@ \oalign}
226 \def\oalign{\lineskiplimit-\maxdimen \oalign}

```

`\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.

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

`\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`).

```
229 \def\ltx@sh@ft #1{%
230   \dimen@ #1%
231   \kern \strip@pt
232     \fontdimen1\font \dimen@
233 } % kern by #1 times the current slant
```

\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 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.

```
234 \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.

```
235 \def\dotfill{%
236   \leavevmode
237   \cleaders \hb@xt@ .44em{\hss.\hss}\hfill
238   \kern\z@}
```

\LaTeX sets `\sfcode x=1000` for all `x`, except that `\sfcode X=999` for upper-case letters. The following changes are needed:

```
239 \sfcode'\=0 \sfcode\'=0 \sfcode'\]=0
```

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

Definitions related to output

`\magnification` doesn’t work in \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:

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



```

\showoutput
\loggingoutput 241 </2ekernel | autoload>
                242 <*2ekernel | autoerr>
                243 \gdef\loggingoutput{\tracingoutput\@ne
                244     \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
                245 \gdef\showoutput{\loggingoutput\showoverfull}
                246 </2ekernel | autoerr>
                247 <autoload>\def\showoutput{\@autoerr\showoutput}

\tracingall
\loggingall 248 <*2ekernel | autoerr>
            249 \gdef\loggingall{\tracingcommands\tw@\tracingstats\tw@
            250     \tracingpages\@ne\tracinglostchars\@ne
            251     \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
            252     \errorcontextlines\maxdimen\loggingoutput}
            253 \gdef\tracingall{\loggingall\showoverfull}
            254 </2ekernel | autoerr>
            255 <autoload>\def\tracingall{\@autoerr\tracingall}

            LATEX change: \showhyphens Defined later.
            Punctuation affects the spacing.

256 <*2ekernel | autoload>
257 \nonfrenchspacing
258 </2ekernel | autoload>

```

File c
ltvers.dtx

10 Version Identification

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

```

\fmtname
\fmtversion 1 {*2ekernel}
2 \def\fmtname{LaTeX2e}
3 \edef\fmtversion{2005/12/01}

```

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:)).

```

4 \iftrue
5 \def\reserved@a#1/#2/#3\@nil{%
6   \count@ \year
7   \advance\count@-#1\relax
8   \multiply\count@ by 12\relax
9   \advance\count@\month
10  \advance\count@-#2\relax}
11 \expandafter\reserved@a\fmtversion\@nil

```

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

```

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

This startup banner may be further modified by the code in `ltfinal.dtx` if a patch file is present.

```

28 \everyjob{\typeout{\fmtname
29 \langle autoload \space(autoload version)\%
30                                     \space\langle \fmtversion\rangle}}
31 \immediate\write16{\fmtname

```

```

32 <autoload>\space(autoload version)%
33                                     \space<\fmtversion>}
34 </2kernel>

```

File d

ltxdefs.dtx

11 Definitions

This section contains commands used in defining other macros.

```
1 \langle *2ekernel \rangle
```

11.1 Initex initialisations

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

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

`\typeout` Display something on the terminal.

```
3 \def\typeout#1{\begingroup\set@display@protect
4   \immediate\write\@unused{#1}\endgroup}
```

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

```
5 \newlinechar'\^^J
```

11.2 Saved versions of T_EX primitives

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

`\@@par`

```
6 \let\@@par=\par
7 %\let\@@input=\input      %%% moved earlier
8 %\let\@@end=\end          %%%
```

`\@@hyph` 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.

There is a need for a robust command for a discretionary hyphen since its exact representation depends on the glyphs available in the current font. For example, with suitable fonts and the T1 font encoding it is possible to use hanging hyphens.

A suitable robust definition that allows for many possible types of font and encoding may be as follows:

```
\DeclareRobustCommand {\-}{\%
\discretionary {%
  \char \ifnum\hyphenchar\font<\z@
    \defaultshyphenchar
  \else
    \hyphenchar\font
  \fi
}{\}{\}%}
```

}

The redefinition (via `\let`) of `\-` within tabbing also makes the use of a robust command advisable since then any redefinition of `\-` via `\DeclareRobustCommand` will not cause a conflict.

Therefore, macro writers should be hereby warned that these internals will probably change! It is likely that a future release of L^AT_EX will make `\-` effectively an encoding specific text command.

```
9 \let\@hyph=\-          % Save original primitive definition
10 \def\-\{\discretionary{-}{-}{-}}
```

`\@dischyph`

```
11 \let\@dischyph=\-
```

`\@italiccorr` Save the original italic correction.

```
12 \let\@italiccorr=\/
```

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

```
\@depth 13 \def\@height{height} \def\@depth{depth} \def\@width{width}
\@width 14 \def\@minus{minus}
\@minus 15 \def\@plus{plus}
\@plus
```

`\hb@xt@` The next one is another 100 tokens worth.

```
16 \def\hb@xt@\{\hbox to}
```

```
17 \message{hacks,}
```

11.3 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}[i]{DEF1}{DEF2}`

equivalent to:

`\newcommand{\FOO}[i]{DEF1} \def\endFOO{DEF2}`

(or the appropriate star forms).

`\renewenvironment`

Obvious companion to \newenvironment.

`\@cons` : See description of \output routine.

`\@car` `\@car T1 T2 ... Tn\@nil == T1` (unexpanded)

`\@cdr` `\@cdr T1 T2 ... Tn\@nil == T2 ... Tn` (unexpanded)

`\typeout` `{message}`

Produces a warning message on the terminal.

`\typein` `{message}`

Types message, asks the user to type in a command, then executes it

`\typein` `[\CS]{MSG}`

Same as above, except defines \CS to be the input instead of executing it.

`\typein`

```
18 \def\typein{%
19   \let\@typein\relax
20   \@testopt\@xtypein\@typein}
```

```
21 \def\@xtypein[#1]#2{%
22   \typeout{#2}%
23   \advance\endlinechar\@M
24   \read\@inputcheck to#1%
25   \advance\endlinechar-\@M
26   \@typein}
```

`\@namedef`

```
27 \def\@namedef#1{\expandafter\def\csname #1\endcsname}
```

`\@nameuse`

```
28 \def\@nameuse#1{\csname #1\endcsname}
```

`\@cons`

```
29 \def\@cons#1#2{\begingroup\let\@elt\relax\xdef#1{#1\@elt #2}\endgroup}
```

`\@car`

`\@cdr` 30 \def\@car#1#2\@nil{#1}

31 \def\@cdr#1#2\@nil{#2}

```

\@carcube \@carcube T1 ... Tn\@nil = T1 T2 T3 ,  $n > 3$ 
32 \def\@carcube#1#2#3#4\@nil{#1#2#3}

\@onlypreamble This macro adds its argument to the list of commands stored in \@preamblecmds
\@preamblecmds to be disabled after \begin{document}. These commands are redefined to gener-
ate \@notprerr at this point.
33 \def\@preamblecmds{}
34 \def\@onlypreamble#1{%
35   \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
36     \@preamblecmds\do#1}}
37 \@onlypreamble\@onlypreamble
38 \@onlypreamble\@preamblecmds

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

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

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

\new@command
45 \def\new@command#1{%
46   \@testopt{\@newcommand#1}0}

\@newcommand Handling arguments for \newcommand.
\@argdef 47 \def\@newcommand#1[#2]{%
\@xargdef 48   \kernel@ifnextchar [{\@xargdef#1[#2]}%
49     {\@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.
50 \long\def\@argdef#1[#2]#3{%
51   \@ifdefinable #1{\@yargdef#1\@ne{#2}{#3}}}
Handle the second optional argument.
52 \long\def\@xargdef#1[#2][#3]#4{%
53   \@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.

```

The `\autoglobal` command below is only used in the autoload format. If it is `\global` then a global definition will be made.

```
54 <autoload>\autoglobal
55   \expandafter\def\expandafter#1\expandafter{%
56     \expandafter
57     \@protected@testopt
58     \expandafter
59     #1%
60     \csname\string#1\endcsname
61     {#3}}%
```

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

```
62   \expandafter\@yargdef
63   \csname\string#1\endcsname
64   \tw@
65   {#2}%
66   {#4}}}
```

`\@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.

```
67 \long\def\@testopt#1#2{%
68   \kernel@ifnextchar[{#1}{#1[{#2}]}}}
```

`\@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 `\@x@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.

```
69 \def\@protected@testopt#1{%%
70   \ifx\protect\@typeset@protect
71     \expandafter\@testopt
72   \else
73     \@x@protect#1%
74   \fi}
```

`\@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.

`\@yargdef` 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_EX 2.09 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 ag 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.

```

75 \long \def \@yargdef #1#2#3{%
76   \ifx#2\tw@
77     \def\reserved@b##11{####1}}%
78   \else
79     \let\reserved@b\@gobble
80   \fi
81   \expandafter
82     \@yargdef \expandafter{\number #3}#1%
83 }

```

The \autoglobal command below is only used in the autoloading format. If it is \global then a global definition will be made.

```

84 \long \def \@yargdef#1#2{%
85   \def \reserved@a ##1##2##3{%
86     \autoglobal
87     \expandafter\def\expandafter#2\reserved@b ##1#1%
88     }%
89   \l@ngrel@x \reserved@a 0##1##2##3##4##5##6##7##8##9##1%
90 }

```

\@reargdef

```

91 \long\def \@reargdef#1[#2]{%
92   \@yargdef#1\@ne{#2}}

```

\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.)

```

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

```

\renew@command

```

94 \def\renew@command#1{%
95   \begingroup \escapechar\m@ne\xdef\@gtempa{\string#1}\endgroup
96   \expandafter\@ifundefined\@gtempa
97     {\@latex@error{\noexpand#1undefined}\@ehc}%
98   \relax
99   \let\@ifdefinable\@rc@ifdefinable
100   \new@command#1}

```

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

```

\@ifdefinable 101 \long\def \@ifdefinable #1#2{%
\@rc@ifdefinable 102   \edef\reserved@a{\expandafter\@gobble\string #1}%
103   \@ifundefined\reserved@a
104     {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
105     \ifx \reserved@b\@qend \@notdefinable\else
106       \ifx \reserved@a\@qrelax \@notdefinable\else
107         #2%
108       \fi

```

```

109         \fi}%
110         \@notdefinable}
    Saved definition of \@ifdefinable.
111 \let\@@ifdefinable\ifdefinable
    Version of \@ifdefinable for use with \renewcommand. Does not do the check
    this time, but restores the normal definition.
112 \long\def\@rc@ifdefinable#1#2{%
113     \let\@ifdefinable\@@ifdefinable
114     #2}

\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.
115 \def\newenvironment{\@star@or@long\new@environment}

\new@environment

116 \def\new@environment#1{%
117     \@testopt{\@newenva#1}0}

\@newenva

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

\@newenvb

120 \def\@newenvb#1[#2][#3]{\@newenv{#1}{[#2][[#3]]}}

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

\renew@environment

122 \def\renew@environment#1{%
123     \@ifundefined{#1}%
124         {\@latex@error{Environment #1 undefined}\@ehc
125         }\relax
126     \expandafter\let\csname#1\endcsname\relax

127 \autoload\aut@global
128 \expandafter\let\csname end#1\endcsname\relax
129 \new@environment{#1}}

\@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.
130 \long\def\@newenv#1#2#3#4{%

```

```

131 \@ifundefined{#1}%
132   {\expandafter\let\csname#1\expandafter\endcsname
133     \csname end#1\endcsname}%
134   \relax
135 \expandafter\new@command
136   \csname #1\endcsname#2{#3}%

137 \autoload\aut@global
138   \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}

\newif And here's a different sort of allocation: For example, \newif\iffoo creates
\footrue, \foofalse to go with \iffoo.

139 \def\newif#1{%
140   \count@\escapechar \escapechar\m@ne
141 \autoload\aut@global
142   \let#1\iffalse
143   \@if#1\iftrue
144   \@if#1\iffalse
145   \escapechar\count@}

\@if

146 \def\@if#1#2{%
147 \autoload\aut@global
148   \expandafter\def\csname\expandafter\@gobbletwo\string#1%
149     \expandafter\@gobbletwo\string#2\endcsname
150     {\let#1#2}}

\providecommand \providecommand takes the same arguments as \newcommand, but discards them
if #1 is already defined. Otherwise it just acts like \newcommand. This imple-
mentation 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.

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

\provide@command

152 \def\provide@command#1{%
153   \begingroup
154     \escapechar\m@ne\edef\@gtempa{\string#1}%
155   \endgroup
156   \expandafter\@ifundefined\@gtempa
157     {\def\reserved@a{\new@command#1}}%
158     {\def\reserved@a{\renew@command\reserved@a}}%
159   \reserved@a}%

\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 pack-
age 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.

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

```

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

```

161 \onlypreamble\CheckCommand

\check@command
162 \def\check@command#1#2#{\@check@c#1{#2}}
163 \onlypreamble\check@command

\@check@c \CheckCommand itself just grabs all the arguments we need, without actually look-
ing 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.
164 \long\def\@check@c#1#2#3{%
165   \expandafter\let\csname\string\reserved@a\endcsname\relax
166   \renew@command\reserved@a#2{#3}%
167   \@ifundefined{\string\reserved@a}%
168     {\@check@eq#1\reserved@a}%
169     {\expandafter\@check@eq
170       \csname\string#1\expandafter\endcsname
171       \csname\string\reserved@a\endcsname}}
172 \onlypreamble\@check@c

\@check@eq Complain if #1 and #2 are not \ifx equal.
173 \def\@check@eq#1#2{%
174   \ifx#1#2\else
175     \@latex@warning@no@line
176       {Command \noexpand#1 has
177         changed.\MessageBreak
178         Check if current package is valid}%
179   \fi}
180 \onlypreamble\@check@eq

\@gobble The \@gobble macro is used to get rid of its argument.
\@gobbletwo 181 \long\def \@gobble #1{}
\@gobblefour 182 \long\def \@gobbletwo #1#2{}
183 \long\def \@gobblefour #1#2#3#4{}

\@firstofone Some argument-grabbers.
\@firstoftwo 184 \long\def \@firstofone#1{#1}
\@secondoftwo 185 \long\def \@firstoftwo#1#2{#1}
186 \long\def \@secondoftwo#1#2{#2}

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

\@thirdofthree Another grabber now used in the encoding specific section.
188 \long\def \@thirdofthree#1#2#3{#3}

\@expandtwoargs A macro to totally expand two arguments to another macro
189 \def\@expandtwoargs#1#2#3{%
190   \edef\reserved@a{\noexpand#1{#2}{#3}}\reserved@a}

\@backslashchar A category code 12 backslash.
191 \edef\@backslashchar{\expandafter\@gobble\string\\}

```

11.4 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 five possible values:

- `\relax`, for normal typesetting. So `\protect\foo` will execute `\foo`.
- `\string`, for writing to the screen. So `\protect\foo` will write `\foo`.
- `\noexpand`, for writing to a file. So `\protect\foo` will write `\foo` followed by a space.
- `\@unexpandable@protect`, for writing a moving argument to a file. So `\protect\foo` will write `\protect\foo` followed by a space. This value is also used inside `\edefs`, `\marks` and other commands which evaluate their arguments fully.
- `\@unexpandable@noexpand`, for performing a deferred write inside an `\edef`. So `\protect\foo` will write `\foo` followed by a space. If you want `\protect\foo` to be written, you should use `\@unexpandable@protect`. (Removed as never used).

```
\@unexpandable@protect  These commands are used for setting \protect inside \edefs.
\@unexpandable@noexpand 192 \def\@unexpandable@protect{\noexpand\protect\noexpand}
                        193 %\def\@unexpandable@noexpand{\noexpand\noexpand\noexpand}
```

```
\DeclareRobustCommand  This is a package-writers command, which has the same syntax as \newcommand,
\declare@robustcommand 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 `\relax` to `\@empty` will cause `_` to loop forever. It will also break lots of other things, such as protected `\ifmodes` inside `\haligns`. If you really 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`.

```
194 \def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
195 \def\declare@robustcommand#1{%
196   \ifx#1\@undefined\else\ifx#1\relax\else
197     \@latex@info{Redefining \string#1}%
198   \fi\fi
199   \edef\reserved@a{\string#1}%
200   \def\reserved@b{#1}%
201   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%

202 \autoload\aut@global
203   \edef#1{%
204     \ifx\reserved@a\reserved@b
205       \noexpand\x@protect
206       \noexpand#1%
207     \fi
208     \noexpand\protect
209     \expandafter\noexpand\csname
210       \expandafter\@gobble\string#1 \endcsname
211   }%
212   \let\@ifdefinable\@rc@ifdefinable
213   \expandafter\new@command\csname
214     \expandafter\@gobble\string#1 \endcsname
215 }

\@x@protect
\@x@protect 216 \def\x@protect#1{%
217   \ifx\protect\@typeset@protect\else
218     \@x@protect#1%
219   \fi
220 }
221 \def\@x@protect#1\fi#2#3{%
222   \fi\protect#1%
223 }

\@typeset@protect
224 \let\@typeset@protect\relax
```

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

`\set@typeset@protect` 225 `\def\set@display@protect{\let\protect\string}`
226 `\def\set@typeset@protect{\let\protect\@typeset@protect}`

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

227 `\def\protected@edef{%`
228 `\let\@protect\protect`
229 `\let\protect\@unexpandable@protect`
230 `\afterassignment\restore@protect`
231 `\edef`
232 `}`
233 `\def\protected@xdef{%`
234 `\let\@protect\protect`
235 `\let\protect\@unexpandable@protect`
236 `\afterassignment\restore@protect`
237 `\xdef`
238 `}`
239 `\def\unrestored@protected@xdef{%`
240 `\let\protect\@unexpandable@protect`
241 `\xdef`
242 `}`
243 `\def\restore@protect{\let\protect\@protect}`

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

244 `\set@typeset@protect`

11.5 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,

245 `\def\@ifundefined#1{%`
246 `\expandafter\ifx\csname#1\endcsname\relax`
247 `\expandafter\@firstoftwo`
248 `\else`
249 `\expandafter\@secondoftwo`
250 `\fi}`

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

251 `\edef\@qend{\expandafter\@cdr\string@end\@nil}`
252 `\edef\@qrelax{\expandafter\@cdr\string\relax\@nil}`

`\@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.

253 `\long\def\@ifnextchar#1#2#3{%`
254 `\let\reserved@d=#1%`
255 `\def\reserved@a{#2}%`
256 `\def\reserved@b{#3}%`
257 `\futurelet\@let@token\@ifnch}`

`\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.

```
258 \let\kernel@ifnextchar\@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`.

```
259 \def\@ifnch{%
260   \ifx\@let@token\@sptoken
261     \let\reserved@c\@xifnch
262   \else
263     \ifx\@let@token\reserved@d
264       \let\reserved@c\reserved@a
265     \else
266       \let\reserved@c\reserved@b
267     \fi
268   \fi
269   \reserved@c}
```

`\@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.

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

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

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

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

```
\makeatother 272 \def\makeatletter{\catcode'\@11\relax}
273 \def\makeatother{\catcode'\@12\relax}
```

`\@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`.

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

`\@dblarg`

```
\@xdblarg 275 \long\def\@dblarg#1{\kernel@ifnextchar[#{#1}{\@xdblarg{#1}}}
276 \long\def\@xdblarg#1#2#{#1}[#{#2}]{#2}}
```


`\@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.

```
277 \def\@sanitize{\@makeother\ \@makeother\\\@makeother\$\@makeother\&%
278 \@makeother\#\@makeother\^\@makeother\_ \@makeother\%\@makeother\~}
```

`\@onelevel@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 oat specifiers, this should be done when they are defined.

```
279 \def \@onelevel@sanitize #1{%
280   \edef #1{\expandafter\strip@prefix
281           \meaning #1}%
282 }

283 </2ekernel>
```

11.6 Commands for Autoloading

```
284 < *autoload>
```

`\aut@global` This command is only defined in the ‘autoload’ format. It is normally `\relax` but may be set to `\global`, in which case `\newif` and the commands based on `\newcommand` will all make global definitions.

```
285 \let\aut@global\relax
```

`\@autoload` This macro is only defined in the ‘autoload’ format. It inputs a package ‘auto#1.sty’ within a local group, and with normalised catcodes. `\aut@global` is set to `\global` so that `\newif` `\newcommand` and related commands make global definitions.

```
286 \def\@autoload#1{%
287   \begingroup
288   \makeatletter
289   \let\aut@global\global
290   \nfss@catcodes
291   \catcode'\ =10
292   \let\@latex@e@error@gobble
293   \@@input auto#1.sty\relax
294   \endgroup}

295 </autoload>
```

File e

lalloc.dtx

12 Counters

This section deals with counter and other variable allocation.

1 $\langle *2\text{kernel} \rangle$

The following are from plain T_EX:

$\backslash\text{z@}$ A zero dimen or number. It's more efficient to write $\backslash\text{parindent}\text{z@}$ than $\backslash\text{parindent}\text{ 0pt}$.

$\backslash\text{@ne}$ The number 1.

$\backslash\text{m@ne}$ The number -1 .

$\backslash\text{tw@}$ The number 2.

$\backslash\text{sixt@@n}$ The number 16.

$\backslash\text{@m}$ The number 1000.

$\backslash\text{@MM}$ The number 20000.

$\backslash\text{@xxxii}$ The constant 32.

2 $\backslash\text{chardef}\backslash\text{@xxxii}=32$

$\backslash\text{@Mi}$ Constants 1001–1004.

$\backslash\text{@Mii}$ 3 $\backslash\text{mathchardef}\backslash\text{@Mi}=10001$

$\backslash\text{@Miii}$ 4 $\backslash\text{mathchardef}\backslash\text{@Mii}=10002$

$\backslash\text{@Miv}$ 5 $\backslash\text{mathchardef}\backslash\text{@Miii}=10003$

6 $\backslash\text{mathchardef}\backslash\text{@Miv}=10004$

$\backslash\text{@tempcnta}$ Scratch count registers used by L^AT_EX kernel commands.

$\backslash\text{@tempcntb}$ 7 $\backslash\text{newcount}\backslash\text{@tempcnta}$

8 $\backslash\text{newcount}\backslash\text{@tempcntb}$

$\backslash\text{if@tempswa}$ General boolean switch used by L^AT_EX kernel commands.

9 $\backslash\text{newif}\backslash\text{if@tempswa}$

$\backslash\text{@tempdima}$ Scratch dimen registers used by L^AT_EX kernel commands.

$\backslash\text{@tempdimb}$ 10 $\backslash\text{newdimen}\backslash\text{@tempdima}$

$\backslash\text{@tempdimc}$ 11 $\backslash\text{newdimen}\backslash\text{@tempdimb}$

12 $\backslash\text{newdimen}\backslash\text{@tempdimc}$

$\backslash\text{@tempboxa}$ Scratch box register used by L^AT_EX kernel commands.

13 $\backslash\text{newbox}\backslash\text{@tempboxa}$

$\backslash\text{@tempskipa}$ Scratch skip registers used by L^AT_EX kernel commands.

$\backslash\text{@tempskipb}$ 14 $\backslash\text{newskip}\backslash\text{@tempskipa}$

15 $\backslash\text{newskip}\backslash\text{@tempskipb}$

`\@temptokena` Scratch token register used by L^AT_EX kernel commands.
16 `\newtoks\@temptokena`

`\@flushglue` Glue used for `\right-` & `\leftskip` = 0pt plus 1fil
17 `\newskip\@flushglue \@flushglue = 0pt plus 1fil`
18 `\</2ekernel>`

File f

ltnctrl.dtx

13 Program control structure

This section defines a number of control structure macros, such as while-loops and for-loops.

```
1 \<*2kernel>
2 \message{control,}
```

```
\@whilenum TEST \do {BODY}
\@whiledim TEST \do {BODY} : These implement the loop
    while TEST do BODY od
    where TEST is a TeX \ifnum or \ifdim test, respectively.
    They are optimized for the normal case of TEST initially false.
```

```
\@whilesw SWITCH \fi {BODY} : Implements the loop
    while SWITCH do BODY od
    Optimized for normal case of SWITCH initially false.
```

```
\@for NAME := LIST \do {BODY} : Assumes that LIST expands to
A1,A2,
... ,An .
    Executes BODY n times, with NAME = Ai on the i-th
iteration.
    Optimized for the normal case of n = 1. Works for n=0.
```

```
\@tfor NAME := LIST \do {BODY}
    if, before expansion, LIST = T1 ... Tn where each Ti is a
    token or {...}, then executes BODY n times, with NAME = Ti
    on the i-th iteration. Works for n=0.
```

NOTES: 1. These macros use no \@temp sequences.
2. These macros do not work if the body contains anything that looks syntactically to TeX like an improperly balanced \if \else \fi.

```
\@whilenum TEST \do {BODY} ==
BEGIN
    if TEST
    then BODY
        \@iwhilenum{TEST \relax BODY}
END
```

```
\@iwhilenum {TEST BODY} ==
BEGIN
    if TEST
    then BODY
```

```

        \@nextwhile = def(\@iwhilenum)
      else \@nextwhile = def(\@whilenoop)
    fi
    \@nextwhile {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
      \@nextwhile = def(\@iwhilesw)
    else \@nextwhile = def(\@whileswnoop)
    fi
    \@nextwhile {SWITCH BODY} \fi
  END

\@whilenoop
\@whilenum 3 \long\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@whilenum{#1\relax
\@iwhilenum 4 #2\relax}\fi}
5 \long\def\@whilenum#1{\ifnum #1\expandafter\@whilenum
6 \else\expandafter\@gobble\fi{#1}}

\@whiledim
\@iwhiledim 7 \long\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
8 \long\def\@whiledim#1{\ifdim #1\expandafter\@iwhiledim
9 \else\expandafter\@gobble\fi{#1}}

\@whileswnoop
\@whilesw 10 \long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\@iwhilesw 11 \long\def\@iwhilesw#1\fi{#1\expandafter\@iwhilesw
12 \else\@gobbletwo\fi{#1}\fi}

\@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}

\@nnil
13 \def\@nnil{\@nil}

\@empty
14 \def\@empty{}

\@fornoop
15 \def\@fornoop#1\@@#2#3{}

\@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}

\@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}

```

```

\@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}}

\@tfor
25 \def\@tfor#1:={\@tfor#1 }
26 \long\def\@tfor#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}}

\@break@tfor Break out of a \@tfor loop. This should be called inside the scope of an \if. See
\@iffileonpath for an example.
31 \long\def\@break@tfor#1\@@#2#3{\fi\fi}

\@removeelement Removes an element from a comma-separated list and puts it into a control se-
quence, called as \@removeelement{<element>}{<list>}{<cs>}.
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}}

38 </2ekernel>

```

File g

lterror.dtx

14 Error handling

This section defines L^AT_EX's error commands.

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.

```
1 <2ekernel>\expandafter\let\csname ver@autoerr.sty\endcsname\fmtversion
2 <2ekernel | autoload>
```

14.1 General commands

\MessageBreak This command prints a new-line inside a message, followed by a continuation line begun with `\@msg@continuation`. Normally it is defined to be `\relax`, but inside messages, it is let to `\@message@break`.

```
3 \let\MessageBreak\relax
```

\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 }
```

\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 }
18 </2ekernel | autoload>
```

\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.

```
19 <autoload>\def\GenericError{\@autoerr\GenericError}
20 <2ekernel | def>
21 \bgroup
22 \lccode'\@=' \ %
```



```

23 \lccode'\~=' \ %
24 \lccode'\}= ' \ %
25 \lccode'\{=' \ %
26 \lccode'\T=' \T%
27 \lccode'\H=' \H%
28 \catcode'\ =11\relax%
29 \lowercase{%
30 \egroup%

```

Unfortunately T_EX versions older than 3.141 have a bug which means that \sim J does not force a linebreak in `\message` and `\errmessage` commands. So for these old T_EX'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 T_EX! 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 T_EX. See the comments in `ltdircheck.dtx`.

```

31 \dimen@ \ifx\@TeXversion\undefined4\else\@TeXversion\fi\p@%
32 \ifdim\dimen@>3.14\p@%

```

First the 'standard case'.

```

33 \DeclareRobustCommand{\GenericError}[4]{%
34 \begingroup%
35 \immediate\write\@unused{}%
36 \def\MessageBreak{\sim J}%
37 \set@display@protect%
38 \edef%
39 %    %<-----do not delete this space!----->%
40 \@err@
41 {{#4}}%
42 \errhelp
43 %    %<-----do not delete this space!----->%
44 \@err@
45 \let
46 %    %<-----do not delete this space!----->%
47 \@err@
48 \@empty
49 \def\MessageBreak{\sim J#1}%
50 \def~{\errmessage{%
51 #2.\sim J\sim J%
52 #3\sim J%
53 Type H <return> for immediate help%
54 %    %<-----do not delete this space!----->%
55 \@err@
56 }}%
57 ~%
58 \endgroup}%
59 \else%

```

Secondly the version for old T_EX's.

```

60 \DeclareRobustCommand{\GenericError}[4]{%
61 \begingroup%

```

```

62 \immediate\write\@unused{}%
63 \def\MessageBreak{^^J}%
64 \set@display@protect%
65 \edef%
66 %    %<-----do not delete this space!----->%
67 \@err@
68 {{#4}}%
69 \errhelp
70 %    %<-----do not delete this space!----->%
71 \@err@
72 \let
73 %    %<-----do not delete this space!----->%
74 \@err@
75 \errmessage
76 \def\MessageBreak{^^J#1}%
77 \def~{\typeout{! %
78 #2.^^J^^J%
79 #3^^J%
80 Type H <return> for immediate help.}%
81 %    %<-----do not delete this space!----->%
82 \@err@
83 {}}%
84 ~%
85 \endgroup}%
86 \fi}%
87 </2ekernel | def>

```

<pre> \PackageError \PackageWarning \PackageWarningNoLine \PackageInfo \ClassError \ClassWarning \ClassWarningNoLine \ClassInfo </pre>	<pre> These commands are intended for use by package and class writers, to give information to authors. The syntax is: \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}{F00}
\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 F00 on input line 54.

```

```

88 <autoload>\def\PackageError{\@autoerr\PackageError}
89 <*2ekernel | def>
90 \gdef\PackageError#1#2#3{%
91   \GenericError{%
92     (#1)\@spaces\@spaces\@spaces\@spaces
93   }{%
94     Package #1 Error: #2%
95   }{%
96     See the #1 package documentation for explanation.%
97   }{#3}%
98 }
99 </2ekernel | def>

100 <*2ekernel | autoload>
101 \def\PackageWarning#1#2{%
102   \GenericWarning{%
103     (#1)\@spaces\@spaces\@spaces\@spaces
104   }{%
105     Package #1 Warning: #2%
106   }%
107 }
108 \def\PackageWarningNoLine#1#2{%
109   \PackageWarning{#1}{#2\@gobble}%
110 }
111 \def\PackageInfo#1#2{%
112   \GenericInfo{%
113     (#1) \@spaces\@spaces\@spaces
114   }{%
115     Package #1 Info: #2%
116   }%
117 }
118 </2ekernel | autoload>

119 <autoload>\def\ClassError{\@autoerr\ClassError}
120 <*2ekernel | def>
121 \gdef\ClassError#1#2#3{%
122   \GenericError{%
123     (#1) \space\@spaces\@spaces\@spaces
124   }{%
125     Class #1 Error: #2%
126   }{%
127     See the #1 class documentation for explanation.%
128   }{#3}%
129 }
130 </2ekernel | def>

131 <*2ekernel | autoload>
132 \def\ClassWarning#1#2{%
133   \GenericWarning{%
134     (#1) \space\@spaces\@spaces\@spaces
135   }{%
136     Class #1 Warning: #2%
137   }%
138 }
139 \def\ClassWarningNoLine#1#2{%
140   \ClassWarning{#1}{#2\@gobble}%

```

```

141 }
142 \def\ClassInfo#1#2{%
143   \GenericInfo{%
144     (#1) \space\space\@spaces\@spaces
145   }{%
146     Class #1 Info: #2%
147   }%
148 }
149 </2ekernel | autoload>

\@latex@error Errors and other info, for use in the LATEX core.
\@latex@warning 150 <autoload>\def\@latex@error{\@autoerr\@latex@error}
\@latex@warning@no@line 151 <*2ekernel | def>
\@latex@info 152 \gdef\@latex@error#1#2{%
\@latex@info@no@line 153   \GenericError{%
154     \space\space\space\@spaces\@spaces\@spaces
155   }{%
156     LaTeX Error: #1%
157   }{%
158     See the LaTeX manual or LaTeX Companion for explanation.%
159   }{#2}%
160 }
161 </2ekernel | def>
162 <*2ekernel | autoload>
163 \def\@latex@warning#1{%
164   \GenericWarning{%
165     \space\space\space\@spaces\@spaces\@spaces
166   }{%
167     LaTeX Warning: #1%
168   }%
169 }
170 \def\@latex@warning@no@line#1{%
171   \@latex@warning{#1\@gobble}}
172 \def\@latex@info#1{%
173   \GenericInfo{%
174     \@spaces\@spaces\@spaces
175   }{%
176     LaTeX Info: #1%
177   }%
178 }
179 \def\@latex@info@no@line#1{%
180   \@latex@info{#1\@gobble}}

\@font@warning and \@font@info are defined later since they have to be
redefined by the tracefnt package.

\def\@font@warning#1{%
  \GenericWarning{%
    {(font)\@spaces\@spaces}%
    {Font Warning: #1}%
  }
}
\def\@font@info#1{%
  \GenericInfo{%

```

```

        (font)\space\@spaces
    }{%
        Font Info: #1%
    }%
}

\c@errorcontextlines \errorcontextlines as a LATEX counter, so that it may be manipulated with
\setcounter (once it is defined :-)
181 \let\c@errorcontextlines\errorcontextlines
182 \c@errorcontextlines=-1

\on@line The message ‘ on input line n’, if possible.
183 \ifnum\inputlineno=\m@ne
184 \let\on@line\@empty
185 \else
186 \def\on@line{ on input line \the\inputlineno}
187 \fi

\@warning Older LATEX messages. For the moment, these \let to the new message commands.
\@@warning They may be changed later, once only obsolete packages and classes contain them.
\@latexerr

188 \let\@warning\@latex@warning
189 \let\@@warning\@latex@warning@no@line
190 \</2ekernel | autoload>
191 \global\let\@latexerr\@latex@error

\@spaces Four spaces.
192 \<2ekernel | autoload>
193 \def\@spaces{\space\space\space\space}
194 \</2ekernel | autoload>

```

14.2 Specific errors

```

\@eha The more common error help messages.
\@ehb 195 \<2ekernel | def>
\@ehc 196 \gdef\@eha{%
\@ehd 197 Your command was ignored.\MessageBreak
198 Type \space I <command> <return> \space to replace it %
199 with another command,\MessageBreak
200 or \space <return> \space to continue without it.}
201 \gdef\@ehb{%
202 You’ve lost some text. \space \@ehc}
203 \gdef\@ehc{%
204 Try typing \space <return> %
205 \space to proceed.\MessageBreak
206 If that doesn’t work, type \space X <return> \space to quit.}
207 \gdef\@ehd{%
208 You’re in trouble here. \space \@ehc}
209 \</2ekernel | def>

```

As `\latex@error` triggers the autoload, these definitions should not be needed in the autoload format, but just to be safe...

```
210 \*autoload
211 \let\@eha\@empty\let\@ehb\@empty\let\@ehc\@empty\let\@ehd\@empty
212 \</autoload>
```

Here are most of the error message-generating commands of L^AT_EX.

```
\@autoerr  Make this autoload command robust, as it may be read in at unpredictable times.
213 \<autoload>\def\@autoerr{\protect\@autoload{err}\protect}

\@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....
214 \gdef\@notdefinable{%
215 \<autoload> \@latex@error{%
216 \<autoload>   Command \@backslashchar\reserved@a\space
217 \<autoload>   already defined.\MessageBreak
218 \<autoload>   Or name \@backslashchar\@qend... illegal,
219 \<autoload>   see p.192 of the manual}\@eha}
220 \<autoload> \@autoerr\@notdefinable}

\@nolnerr  Generated by \newline and \\ when called in vertical mode.
221 \gdef\@nolnerr{%
222 \<autoload> \@latex@error{There's no line here to end}\@eha}
223 \<autoload> \@autoerr\@nolnerr}

\@nocounterr  Generated by \setcounter, \addtocounter or \newcounter if applied to an undefined counter <cnt>.

\@nocnterr  Obsolete error message generated in LATEX2.09 by \setcounter, \addtocounter or \newcounter for undefined counter. DO NOT use for LATEX2ε it MIGHT vanish! Use \@nocounterr{<cnt>} instead.
224 \gdef\@nocounterr#1{%
225 \<autoload> \@latex@error{No counter '#1' defined}\@eha}
226 \<autoload> \@autoerr\@nocounterr}
227 \gdef\@nocnterr{\@nocounterr?}

\@ctrerr  Called when trying to print the value of a counter numbered by letters that's greater than 26.
228 \gdef\@ctrerr{%
229 \<autoload> \@latex@error{Counter too large}\@ehb}
230 \<autoload> \@autoerr\@ctrerr}

\@nodocument  Error produced if paragraphs are typeset in the preamble.
231 \!def\gdef\@nodocument{%
232 \!def \@latex@error{Missing \protect\begin{document}}\@ehd}

\@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.
233 \gdef\@badend#1{%
```

```

234 \!autoload \latex@error{\protect\begin{\@currenvir}\@currenvline
235 \!autoload \space ended by \protect\end{#1}}\@eha}
236 \!autoload \!autoerr\@badend}

\@badmath Called by \[, \], \( or \) when used in wrong mode.
237 \gdef\@badmath{%
238 \!autoload \latex@error{Bad math environment delimiter}\@eha}
239 \!autoload \!autoerr\@badmath}

\@toodeep Called by a list environment nested more than six levels deep, or an enumerate or
itemize nested more than four levels.
240 \gdef\@toodeep{%
241 \!autoload \latex@error{Too deeply nested}\@ehd}
242 \!autoload \!autoerr\@toodeep}

\@badpoptabs Called by \endtabbing when not enough \poptabs have occurred, or by \poptabs
when too many have occurred.
243 \gdef\@badpoptabs{%
244 \!autoload \latex@error{\protect\pushtabs\space and \protect\poptabs
245 \!autoload \space don't match}\@ehd}
246 \!autoload \!autoerr\@badpoptabs}

\@badtab Called by \>, \+ , \- or \< when stepping to an undefined tab.
247 \gdef\@badtab{%
248 \!autoload \latex@error{Undefined tab position}\@ehd}
249 \!autoload \!autoerr\@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.
250 \gdef\@preamerr#1{%
251 \begingroup
252 \let\protect\relax
253 \!autoload
254 \latex@error{\ifcase #1 Illegal character\or
255 Missing @-exp\or Missing p-arg\fi\space
256 in array arg}\@ehd
257 \!autoload}
258 \!autoload \!autoerr\@preamerr{#1}%
259 \endgroup}

\@badlinearg Occurs in \line and \vector command when a bad slope argument is encoun-
tered.
260 \gdef\@badlinearg{%
261 \!autoload \latex@error{%
262 \!autoload Bad \protect\line\space or \protect\vector
263 \!autoload \space argument}\@ehb}
264 \!autoload \!autoerr\@badlinearg}

\@parmoderr Occurs in a oat environment or a \marginpar when encountered in inner vertical
mode.

```

```

265 \gdef\@parmoderr{%
266 \!autoload\ \@latex@error{Not in outer par mode}\@ehb}
267 \!autoload\ \@autoerr\@parmoderr}

\@fltovf Occurs in oat environment or \marginpar when there are no more free boxes for
storing oats.

268 \gdef\@fltovf{%
269 \!autoload\ \@latex@error{Too many unprocessed floats}\@ehb}
270 \!autoload\ \@autoerr\@fltovf}

\@latexbug Occurs in output routine. This is bad news.

271 \gdef\@latexbug{%
272 \!autoload\ \@latex@error{This may be a LaTeX bug}{Call for help}}
273 \!autoload\ \@autoerr\@latexbug}

\@badcrerr This error was removed and replaced by \@nolnerr.

274 %\def\@badcrerr {\@latex@error{Bad use of \protect}\@ehc}

\@noitemerr \addvspace or \addpenalty was called when not in vmode. Probably caused by
a missing \item.

275 \gdef\@noitemerr{%
276 \!autoload\ \@latex@error{Something's wrong--perhaps a missing %
277 \!autoload\ \protect\item}\@ehc}
278 \!autoload\ \@autoerr\@noitemerr}

\@notprerr A command that can be used only in the preamble appears after the command
\begin{document}.

279 \gdef\@notprerr{%
280 \!autoload\ \@latex@error{Can be used only in preamble}\@eha}
281 \!autoload\ \@autoerr\@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.

282 \gdef\@inmatherr#1{%
283 \relax
284 \ifmmode
285 \!autoload\ \@latex@error{Command \protect#1 invalid in math mode}\@ehc
286 \!autoload\ \@autoerr\@inmatherr#1%
287 \fi}

\@invalidchar An error for use with invalid characters. This is commented out, since we decided
to use chatcode 15 instead.

288 %\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}

As well as the above error commands some error messages are directly coded
to save space. The Messages already present in LATEX2.09 included:
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 `oat` environment or `\marginpar` occurring in inner vertical mode.

File h

ltpar.dtx

15 Paragraphs

This section of the kernel declares the commands used to set `\par` and `\everypar` when ever their function needs to be changed for a long time.

15.1 Implementation

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.

`\@setpar` To permit the proper interaction of these two situations, long-term changes are made by the `\@setpar{<VAL>}` command. It's 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:

1. 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.

2. 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 <*2kernel>
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}
5 </2kernel>
```

`\@restorepar` Restore from a short-term change to `\par`.

```
6 \def\@restorepar{\def\par{\@par}}
```

File i

ltspace.dtx

16 Spacing

This section deals with spacing, and line- and page-breaking.

16.1 User Commands

`\nopagebreak` [$\langle i \rangle$] : $\langle i \rangle = 0, \dots, 4$.
 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` [$\langle i \rangle$] : same as except negatives of its penalty
`\linebreak` [$\langle i \rangle$] : analog of the above
`\nolinebreak` [$\langle i \rangle$] : analog of the above
`\samepage` : inhibits page breaking most places by setting the following penalties to 10000:
`\interlinepenalty`
`\postdisplaypenalty`
`\interdisplaylinepenalty`
`\@beginparpenalty`
`\@endparpenalty`
`\@itempenalty`
`\@secpenalty`
`\interfootnotelinepenalty`
`\` : initially defined to be `\newline`
`\` [$\langle length \rangle$] : initially defined to be `\vspace{\langle length \rangle}\newline`
 Note: `*` adds a `\adjust{\penalty 10000}`
 OBSOLETE COMMANDS (which never made it into the manual):
`\obeycr` : defines `\CR_i == \relax`
`\restorecr` : restores `\CR_i` to its usual meaning.

16.2 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}
```

16.3 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(?) `\unskips` 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 `\unskips`.
- Investigate and rationalise the ‘newline’ code.
- Find better algorithms for all sorts of things or, easier(?), fix TeX itself.

16.4 The code

```

1 <*2ekernel>
2 \message{spacing,}

\pagebreak
\nopagebreak 3 \def\pagebreak{\@testopt{\@no@pgbk-}4}
4 \def\nopagebreak{\@testopt{\@no@pgbk4}

\@no@pgbk
5 \def\@no@pgbk #1[#2]{%
6   \ifvmode
7     \penalty #1\@getpen{#2}%
8   \else
9     \@bsphack
10    \vadjust{\penalty #1\@getpen{#2}}%
11    \@esphack
12  \fi}

\linebreak
\nolinebreak 13 \def\linebreak{\@testopt{\@no@lnbk-}4}
14 \def\nolinebreak{\@testopt{\@no@lnbk4}

\@no@lnbk
15 \def\@no@lnbk #1[#2]{%
16   \ifvmode
17     \@nolnerr
18   \else
19     \@tempskipa\lastskip
20     \unskip
21     \penalty #1\@getpen{#2}%
22     \ifdim\@tempskipa>\z@
23       \hskip\@tempskipa
24       \ignorespaces
25     \fi
26   \fi}

\samepage
27 \def\samepage{\interlinepenalty\@M
28   \postdisplaypenalty\@M

```

```

29 \interdisplaylinepenalty\@M
30 \@beginparpenalty\@M
31 \@endparpenalty\@M
32 \@itempenalty\@M
33 \@secpenalty\@M
34 \interfootnotelinepenalty\@M}

```

`\@` 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 `\@`.

`\@normalcr` The internal definition of the ‘normal’ definition of `\@`.

```

35 \DeclareRobustCommand\@{%
36   \let \reserved@e \relax
37   \let \reserved@f \relax
38   \ifstar{\let \reserved@e \vadjust \let \reserved@f \nobreak
39     \@xnewline}%
40     \@xnewline}
41 \expandafter\let\expandafter\@normalcr
42   \csname\expandafter\@gobble\string\@ \endcsname

```

`\newline` A simple form of the ‘normal’ definition of `\@`.

```

43 \DeclareRobustCommand\newline{\@normalcr\relax}

```

`\@xnewline`

```

44 \def\@xnewline{\@ifnextchar[% ] bracket matching
45   \@newline
46   {\@gnewline\relax}}

```

`\@newline`

```

47 \def\@newline[#1]{\let \reserved@e \vadjust
48   \@gnewline {\vskip #1}}

```

`\@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

```

49 \def\@gnewline #1{%
50   \ifvmode
51     \@nolnerr
52   \else
53     \unskip \reserved@e {\reserved@f#1}\nobreak \hfil \break
54   \fi}

```


`\@getpen`

```
55 \def\@getpen#1{\ifcase #1 \z@ \or \@lowpenalty\or
56     \@medpenalty \or \@highpenalty
57     \else \@M \fi}
```

`\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.

```
58 \def\@nobreakfalse{\global\let\if@nobreak\iffalse}
59 \def\@nobreaktrue {\global\let\if@nobreak\iftrue}
60 \@nobreakfalse
```

`\@savsk` Registers used to save the space factor and last skip.

```
\@savsf 61 \newdimen\@savsk
        62 \newcount\@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
}
```

For the moment we are going to ignore the vertical versions until they are correct.

```
63 \def\@bsphack{%
64   \relax
65   \ifhmode
66     \@savsk\lastskip
67     \@savsf\spacefactor
68   \fi}
```

`\@esphack` Companion to `\@bsphack`.

```
69 \def\@esphack{%
70   \relax
71   \ifhmode
72     \spacefactor\@savsf
73     \ifdim\@savsk>\z@
74       \ignorespaces
75     \fi
76   \fi}
```

`\@Esphack` A variant of `\@esphack` that sets the `@ignore` switch to true (as `\@esphack` used to do previously). This is currently used only for oats and similar environments.

```
77 \def\@Esphack{%
78   \relax
79   \ifhmode
80     \spacefactor\@savsf
81     \ifdim\@savsk>\z@
82       \@ignoretrue
83     \ignorespaces
84   \fi
85   \fi}
```

`\@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
}
```

16.5 Vertical spacing

L^AT_EX supports the plain T_EX commands `\smallskip`, `\medskip` and `\bigskip`. However, it redefines them using `\vspace` instead of `\vskip`.

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.

```

86 \def\@xaddvskip{%
87   \ifdim\lastskip<\@tempskipb
88     \vskip-\lastskip
89     \vskip\@tempskipb
90   \else
91     \ifdim\@tempskipb<\z@
92       \ifdim\lastskip<\z@
93         \else
94           \advance\@tempskipb\lastskip
95           \vskip-\lastskip
96           \vskip \@tempskipb
97         \fi
98       \fi
99   \fi}

```

`\addvspace` Add vertical space taking into account space already added, as described above.

```

100 \def\addvspace#1{%
101   \ifvmode
102     \if@minipage\else
103       \ifdim \lastskip =\z@
104         \vskip #1\relax
105       \else
106         \@tempskipb#1\relax
107         \@xaddvskip
108       \fi
109     \fi
110   \else
111     \@noitemerr
112   \fi}

```

`\addpenalty`

```

113 \def\addpenalty#1{%
114   \ifvmode
115     \if@minipage
116     \else
117       \if@nobreak
118       \else
119         \ifdim\lastskip=\z@
120           \penalty#1\relax
121         \else
122           \@tempskipb\lastskip
123           \vskip -\lastskip
124           \penalty#1%
125           \vskip\@tempskipb
126         \fi
127       \fi
128     \fi
129   \else
130     \@noitemerr
131   \fi}

```

`\vspace` The new code for these commands depends on the following facts:

`\@vspace`

`\@vspacer` File i: `ltspace.dtx` Date: 2004/02/15 Version v1.3a

- 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.

```

132 \DeclareRobustCommand\vspace{\@ifstar\@vspacer\@vspace}
133 \def\@vspace #1{%
134   \ifvmode
135     \vskip #1
136     \vskip\z@skip
137   \else
138     \@bsphack
139     \vadjust{\@restorepar
140               \vskip #1
141               \vskip\z@skip
142             }%
143     \@esphack
144   \fi}

145 \def\@vspacer#1{%
146   \ifvmode
147     \dimen@\prevdepth
148     \hrule \@height\z@
149     \nobreak
150     \vskip #1
151     \vskip\z@skip
152     \prevdepth\dimen@
153   \else
154     \@bsphack
155     \vadjust{\@restorepar
156               \hrule \@height\z@
157               \nobreak
158               \vskip #1
159               \vskip\z@skip}%
160     \@esphack
161   \fi}

\smallskip
\medskip 162 \def\smallskip{\vspace\smallskipamount}
\bigskip 163 \def\medskip{\vspace\medskipamount}
          164 \def\bigskip{\vspace\bigskipamount}

\smallskipamount
\medskipamount 165 \newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\bigskipamount 166 \newskip\medskipamount \medskipamount =6pt plus 2pt minus 2pt
                167 \newskip\bigskipamount \bigskipamount =12pt plus 4pt minus 4pt

```

16.6 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 spacefactor; so use it carefully!

```

168 \DeclareRobustCommand{\nobreakdashes}{%
169   \leavevmode
170   \toks@{}%
171   \def\reserved@a##1{\toks@\expandafter{\the\toks@-}%
172                     \futurelet\@let@token \reserved@b}%
173   \def\reserved@b  {\ifx\@let@token -%
174                   \expandafter\reserved@a
175                   \else
176                   \setbox\z@ \hbox{\the\toks@\nobreak}%
177                   \unhbox\z@
178                   \spacefactor\sfcode‘-
179                   \fi}%
180   \futurelet\@let@token \reserved@b
181 }
```

`\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^AT_EX internal commands.

The braces in the definition of `~` are needed to ensure that a following space is preserved when reading to/from internal files.

We need to keep `\@xobeysp` as it is widely used; so here it is let to the non-robust command `\nobreakspace`.

```

182 \DeclareRobustCommand{\nobreakspace}{%
183   \leavevmode\nobreak\ }
184 \catcode ‘\~=13
185 \def~{\nobreakspace{}}
186 \expandafter\let\expandafter\@xobeysp\csname nobreakspace \endcsname
```

`\,` Used in paragraph mode produces a `\thinspace`. It has the ordinary definition in math mode. Useful for quotes inside quotes, as in ‘`\,`‘Foo’, he said.’’

```

187 \DeclareRobustCommand{\,}{%
188   \relax\ifmmode\mskip\thinmuskip\else\thinspace\fi
189 }
```

`\@` 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.

```

190 \def\@{\spacefactor\@m}
```

`\hspace`

```

191 \DeclareRobustCommand\hspace{\@ifstar\@hspacer\@hspace}
```

```

\@hspace
192 \def\@hspace#1{\hskip #1\relax}

\@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
193 \def\@hspacer#1{\vrule \@width\z@\nobreak
194                 \hskip #1\hskip \z@skip}

\fill
195 \newskip\fill
196 \fill = 0pt plus 1fill

\stretch
197 \def\stretch#1{\z@ \@plus #1fill\relax}

\thinspace
\negthinspace 198 \def\thinspace{\kern .16667em }
\enspace      199 \def\negthinspace{\kern-.16667em }
              200 \def\enspace{\kern.5em }

\enskip
\quad         201 \def\enskip{\hskip.5em\relax}
\qqquad       202 \def\quad{\hskip1em\relax}
              203 \def\qqquad{\hskip2em\relax}

\obeycr       The following definitions will probably get deleted or moved to compatibility mode
\restorecr    soon.
204 {\catcode'\^M=13 \gdef\obeycr{\catcode'\^M13 \def^^M{\\relax}%
205   \gobblecr}%
206 {\catcode'\^M=13 \gdef\gobblecr{\@ifnextchar
207 \gobble\ignorespaces}}
208 \gdef\restorecr{\catcode'\^M5 }}

209 </2kernel>

```

File j

ltlogos.dtx

17 Logos

Various logos are defined here.

`\TeX` The $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ logo, adjusted so that a full stop after the logo counts as ending a sentence.

```
1 < *2ekernel>
2 \def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

`\LaTeX` The $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ logo.

```
3 \DeclareRobustCommand{\LaTeX}{L\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}
```

`\LaTeXe` The $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X} 2_{\varepsilon}$ logo as proposed by A-W designers.

```
13 \DeclareRobustCommand{\LaTeXe}{\mbox{\m@th
14   \if b\expandafter\@car\@series\@nil\boldmath\fi
15   \LaTeX\kern.15em2$_{\textstyle\varepsilon}$}}
16 </2ekernel>
```


File k

ltfiles.dtx

18 File Handling

The following user commands are defined in this part:

<code>\document</code>	<code>(ie \begin{document})</code> Reads in the .AUX files and <code>\catcode</code> 's @ to 12.
<code>\nofiles</code>	Suppresses all file output by setting <code>\@files</code> false.
<code>\includeonly</code>	<code>{\NAME1, ... ,NAMEn}</code> Causes only parts NAME1, ... ,NAMEn to be read by their <code>\include</code> commands. Works by setting <code>partsw</code> true and setting <code>\@partlist</code> to NAME1, ... ,NAMEn.
<code>\include</code>	<code>{\NAME}</code> Does an <code>\input</code> NAME unless <code>\@partsw</code> is true and NAME is not in <code>\@partlist</code> . If <code>\@files</code> is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.
<code>\input</code>	<code>{\NAME}</code> The same as TeX's <code>\input</code> , except it allows optional braces around the file name. In $\text{\LaTeX 2}_{\epsilon}$, it also avoids the primitive 'missing file' error, if the file can not be found.
<code>\IfFileExists</code>	<code>{\NAME}{\<then>}{\<else>}</code> If the file exists on the system, execute <i>then</i> otherwise execute <i>else</i> .
<code>\InputIfFileExists</code>	<code>{\NAME}{\<then>}{\<else>}</code> If the file exists on the system, execute <i>then</i> and input NAME otherwise execute <i>else</i> .

```
1 \*2ekernel | autoload)
2 \message{files,}
```

VARIABLES, SWITCHES AND INTERNAL COMMANDS:

<code>\@mainaux</code>	: Output file number for main .AUX file.
<code>\@partaux</code>	: Output file number for current part's .AUX file.
<code>\@auxout</code>	: Either <code>\@mainout</code> or <code>\@partout</code> , depending on which .AUX file output goes to.
<code>\@input{foo}</code>	: If file foo exists, then <code>\input</code> 's it, otherwise types a warning message.
<code>@files</code>	: Switch – set false if no .AUX, .TOC, .IDX etc files are to be written
<code>@partsw</code>	: Set true by a <code>\includeonly</code> command.
<code>\@partlist</code>	: Set to the argument of the <code>\includeonly</code> command.
<code>\cp@FOO</code>	: The checkpoint for <code>\include</code> 'd file FOO.TEX, written by <code>\wroteckpt</code> at the end of file FOO.AUX

```
\includeonly{FILELIST} ==
BEGIN
```

```

\@partsw := T
\@partlist := FILELIST
END

\include{FILE} ==
BEGIN
\clearpage
if \@filesw = T
then \immediate\write\@mainaux{\string\@input{FILE.AUX}}
fi
if \@partsw = T
then \@tempswa := F
\reserved@a := 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 \@filesw = T
then \immediate\openout\@partaux{FILE.AUX}
\immediate\write\@partaux{\relax}
fi
\@input{FILE.TEX}
\clearpage
\@writeckpt{FILE}
if @filesw then \closeout \@partaux fi
\@auxout := \@mainaux
else \cp@FILE
fi
END

\@writeckpt{FILE} ==
BEGIN
if \@filesw = 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

\@inputcheck Allocate read stream for testing and output stream.
  \@unused 3 \newread\@inputcheck
            4 \newwrite\@unused

  \@mainaux
  \@partaux 5 \newwrite\@mainaux
            6 \newwrite\@partaux

  \if@filesw
  \if@partsw 7 \newif\if@filesw \@fileswtrue
            8 \newif\if@partsw \@partswfalse

\@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).
               9 \newcount\@clubpenalty
               10 \@clubpenalty \@clubpenalty

\document      Cancel the \begingroup from \begin
               11 \def\document{\endgroup
               If some options on \documentclass haven't been used by any package we will now
               give a warning since this is most certainly a misspelling.
               12 \ifx\@unusedoptionlist\@empty\else
               13   \@latex@warning@no@line{Unused global option(s):^^J%
               14     \@spaces[\@unusedoptionlist]]}%
               15 \fi
               16 \@colht\textheight
               17 \@colroom\textheight \vsize\textheight
               18 \@columnwidth\textwidth
               19 \@clubpenalty\clubpenalty
               20 \if@twocolumn
               21   \advance\columnwidth -\columnsep
               22   \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
               23 \fi
               24 \hsize\columnwidth \linewidth\hsize
               25 \begingroup\@floatplacement\@dblfloatplacement
               26   \makeatletter\let\@writefile\@gobbletwo
               27   \global \let \@multiplelabels \relax
               28   \@input{\jobname.aux}%
               29 \endgroup
               30 \if@filesw
               31   \immediate\openout\@mainaux\jobname.aux
               32   \immediate\write\@mainaux{\relax}%
               33 \fi

```

Dateline 1991/03/26: FMi added `\process@table` to support NFSS; This will also work with old fonts if no other style defines `\process@table`. The following line forces the initialization of the math fonts.

```

34 \process@table
35 \let\glb@currsize\@empty %% Force math initialization.

36 \normalsize
37 \everypar{}%
```

So that punctuation in headings is not disturbed by verbatim or other local changes to the space factor codes, save the document default here. This will be locally reset by the output routine. For special cases a class may want to define `\normalsfcodes` directly, in case that definition will be used. (This is an old bug, problem existed in L^AT_EX2.0x and plain T_EX.)

```

38 \ifx\normalsfcodes\@empty
39   \ifnum\sfcode'\.=\@m
40     \let\normalsfcodes\frenchspacing
41   \else
42     \let\normalsfcodes\nonfrenchspacing
43   \fi
44 \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.

```

45 \@noskipsecfalse

46 \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...

```

47 \let\AtBeginDocument\@firstofone
48 \@begindocumenthook
```

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.

```

49 \ifdim\topskip<1sp\global\topskip 1sp\relax\fi
50 \global\@maxdepth\maxdepth
51 \global\let\@begindocumenthook\@undefined
52 \ifx\@listfiles\@undefined
53   \global\let\@filelist\relax
54   \global\let\@addtofilelist\@gobble
55 \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.

```

56 \gdef\do##1{\global\let ##1\@notprerr}%
57 \@preamblecmds
```

The next line saves tokens and also allows `\@nocument` to be used directly to trap preamble errors.

```
58 \global\let \@nocument \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.

```
59 \global\let\do\noexpand
```

Use of `\AtBeginDocument` hook might mean that we are already in horizontal mode, so ignore the space after `\begin{document}`.

```
60 \ignorespaces}
```

```
61 \@onlypreamble\document
```

`\normalsfcodes` The setting of `\@empty` is just a ag. 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.

```
62 \let\normalsfcodes\@empty
```

`\nofiles` Set `\@filesfalse` which suppresses the places where L^AT_EX makes `\immediate` writes. The `\makeindex` and `\makeglossary` are disabled. `\protected@write` is redefined not to write to the file specified, but rather to write a blank line to the log file. This ensures that a *⟨whatsit⟩* node is still created, and so spacing is not affected by the `\nofiles` command; to ensure this more generally, the `\if@nobreak` test is needed.

```
63 \def\nofiles{%
64   \@filesfalse
65   \typeout{No auxiliary output files.^^J}%
66   \long\def\protected@write##1##2##3%
67     {\write\m@ne{}\if@nobreak\ifvmode\nobreak\fi\fi}%
68   \let\makeindex\relax
69   \let\makeglossary\relax}
70 \@onlypreamble\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`.

```
71 \long\def \protected@write#1#2#3{%
72   \begingroup
73   \let\thepage\relax
74   #2%
75   \let\protect\@unexpandable@protect
76   \edef\reserved@a{\write#1{#3}}%
77   \reserved@a
78   \endgroup
79   \if@nobreak\ifvmode\nobreak\fi\fi
80 }
```

```
81 \let\@auxout=\@mainaux
```

`\includeonly`

```
82 \def\includeonly#1{%
83   \@partswtrue
```

```

84 \edef\@partlist{\zap@space#1 \@empty}}
85 \onlypreamble\includeonly

\include In the definition of \include, \def\reserved@b changed to \edef\reserved@b
to be 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 LATEX 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.

86 \def\include#1{\relax
87 \ifnum\@auxout=\@partaux
88 \latexerror{\string\include\space cannot be nested}\@eha
89 \else \@include#1 \fi}

\@include

90 \def\@include#1 {%
91 \clearpage
92 \if@filesw
93 \immediate\write\@mainaux{\string\input{#1.aux}}%
94 \fi
95 \@tempswatrue
96 \if@partsw
97 \@tempswafalse
98 \edef\reserved@b{#1}%
99 \@for\reserved@a:=\@partlist\do
100 {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
101 \fi
102 \if@tempswa
103 \let\@auxout\@partaux
104 \if@filesw
105 \immediate\openout\@partaux #1.aux
106 \immediate\write\@partaux{\relax}%
107 \fi
108 \input@{#1.tex}%
109 \clearpage
110 \@writeckpt{#1}%
111 \if@filesw
112 \immediate\closeout\@partaux
113 \fi
114 \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.

115 \deadcycles\z@
116 \@nameuse{cp@#1}%
117 \fi
118 \let\@auxout\@mainaux}

\@writeckpt

119 \def\@writeckpt#1{%
120 \if@filesw

```

```

121     \immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
122     {\let\@elt\@wckptelt \c1@ckpt}%
123     \immediate\write\@partaux{\@charrb}%
124     \fi}

\@wckptelt

125 \def\@wckptelt#1{%
126     \immediate\write\@partaux{%
127         \string\setcounter{#1}{\the\@nameuse{c@#1}}}}

\@setckpt RmS 93/08/31: introduced \@setckpt

128 \def\@setckpt#1{\global\@namedef{cp@#1}}

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

129 {\catcode' [=1 \catcode' ]=2
130 \catcode' {=11 \catcode' }=11
131 \gdef\@charlb[{
132 \gdef\@charrb[}]
133 ]% }brace matching

```

18.1 Safe Input Macros

```

\IfFileExists

134 \long\def \IfFileExists#1#2#3{%
135     \openin\@inputcheck#1 %
136     \ifeof\@inputcheck
137         \ifx\input@path\@undefined
138             \def\reserved@a{#3}%
139         \else
140             \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
141         \fi
142     \else
143         \closein\@inputcheck
144         \edef\@filef@und{#1 }%
145         \def\reserved@a{#2}%
146     \fi
147     \reserved@a}

\@iffileonpath If the file is not found by \openin, and \input@path is defined, look in all the
directories specified in \input@path.

148 \long\def\@iffileonpath#1{%
149     \let\reserved@a\@secondoftwo
150     \expandafter\@tfor\expandafter\reserved@b\expandafter
151         :\expandafter=\input@path\do{%
152         \openin\@inputcheck\reserved@b#1 %
153         \ifeof\@inputcheck\else
154             \edef\@filef@und{\reserved@b#1 }%
155             \let\reserved@a\@firstoftwo%
156             \closein\@inputcheck
157             \@break@tfor
158         \fi}%
159     \reserved@a}

```

`\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’.

```

160 \long\def \InputIfFileExists#1#2{%
161   \IfFileExists{#1}%
162     {#2\@addtofilelist{#1}\@input \@filef@und}}

```

`\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^AT_EX).

```

163 \def\input{\ifnextchar\bgroup\@input\@input}

```

`\@input` Define `\@input` (i.e., `\input`) in terms of `\InputIfFileExists`.

```

164 \def\@input#1{%
165   \InputIfFileExists{#1}{}%
166   {\filename@parse{#1}%
167     \edef\reserved@a{\noexpand\@missingfileerror
168       {\filename@area\filename@base}%
169       {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
170   \reserved@a}}

```

`\@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 .bbl probably should be listed and thus should be implemented not by `\@input`.

```

171 \def\@input#1{%
172   \IfFileExists{#1}{\@input\@filef@und}{\typeout{No file #1.}}}

```

`\@input@` Version of `\@input` that does add the file to `\@filelist`.

```

173 \def\@input@#1{\InputIfFileExists{#1}{\typeout{No file #1.}}}

```

`\@missingfileerror` This ‘error’ command avoids TeX’s primitive missing file loop.

Missing file error. Prompt for a new filename, offering a default extension.

```

174 \autoload\def\@missingfileerror{\@autoerr\@missingfileerror}
175 \</2ekernel\autoload\
176 \<*2ekernel\autoerr\
177 \gdef\@missingfileerror#1#2{%
178   \typeout{^^J! LaTeX Error: File ‘#1.#2’ not found.^^J^^J%
179   Type X to quit or <RETURN> to proceed,^^J%
180   or enter new name. (Default extension: #2)^^J}%
181   \message{Enter file name: }%
182   {\endlinechar\m@ne
183     \global\read\m@ne to\@gtempa}%
184   \ifx\@gtempa\@empty
185   \else
186     \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@end\fi
187     \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@end\fi
188     \filename@parse\@gtempa
189     \edef\filename@ext{%
190       \ifx\filename@ext\relax#2\else\filename@ext\fi}%
191     \edef\reserved@a{%
192       \noexpand\InputIfFileExists
193       {\filename@area\filename@base.\filename@ext}%

```



```

194      {}%
195      {\noexpand\@missingfileerror
196       {\filename@area\filename@base}{\filename@ext}}}%
197      \reserved@a
198      \fi}
199 \</2ekernel|autoerr)
200 \<*2ekernel|autoload)

```

\@obsoletefile For compatibility with L^AT_EX 2.09 document styles, we distribute files called `article.sty`, `book.sty`, `report.sty`, `slides.sty` and `letter.sty`. These use the command `\@obsoletefile`, which produces a warning message.

```

201 \def\@obsoletefile#1#2{%
202   \@latex@warning@no@line{inputting ‘#1’ instead of obsolete ‘#2’}}
203 \@onlypreamble\@obsoletefile

```

18.2 Listing files

\@filelist A list of files input so far. The initial value of `\@gobble` eats the comma before the first file name.

```

204 \let\@filelist\@gobble

```

\@addtofilelist Add to the list of files input so far. This ‘real’ definition is only used for ‘`cfg`’ files during `intex`. An initial definition of `\@gobble` has already been set.

```

205 %\def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}

```

\listfiles A preamble command to cause `\end{document}` to list files input from the main file.

```

206 \def\listfiles{%
207   \let\listfiles\relax
208   \def\@listfiles##1##2##3##4##5##6##7##8##9\@{%
209     \def\reserved@d{\}%
210     \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
211       \ifx\reserved@c\reserved@d
212         \edef\filename@area{ \filename@area}%
213       \fi}}%
214   \def\@dofilelist{%
215     \typeout{^^J *File List*}%
216     \@for\@currname:=\@filelist\do{%
217       \filename@parse\@currname
218       \edef\reserved@a{%
219         \filename@base.%
220         \ifx\filename@ext\relax tex\else\filename@ext\fi}%
221       \expandafter\let\expandafter\reserved@b
222         \csname ver@\reserved@a\endcsname
223       \expandafter\expandafter\expandafter\@listfiles\expandafter
224         \filename@area\filename@base\////////////////////////////////\@
225       \typeout{%
226         \filename@area\reserved@a
227         \ifx\reserved@b\relax\else\@spaces\reserved@b\fi}}%
228     \typeout{ *****^^J}}

```

The `\@filelist` will be de-activated if `\listfiles` does not appear in the preamble. `\begin{document}` contains code equivalent to the following:

```
\AtBeginDocument{%
  \ifx\@listfiles\undefined
    \let\@filelist\relax
    \let\@addtofilelist\@gobble
  \fi}

229 \@onlypreamble\listfiles

\@dofilelist

230 \let\@dofilelist\relax

231 </2ekernel | autoload>
```

File 1

ltoutenc.dtx

19 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 OLM, 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 l}
```

`\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`.

19.1 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 acky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in OT1 L^AT_EX will still find the encoding specific-definition for OT1 and therefore ignore the new default. Therefore to ensure that in this case the TS1 version is used we have to remove the OT1 declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the \$ sign is a proper glyph in the T1 encoding there is no point removing its definition and forcing L^AT_EX to pick up the TS1 version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., \$ for your dollars. In that case you have to get rid of the T1 declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar} {T1}
\DeclareTextCommandDefault{\textdollar}
{\UseTextSymbol{TS1}\textdollaroldstyle}
```

19.2 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.

19.3 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. Ryćko for use with the Polish version of Computer Modern and Computer Concrete.
package	generates <code>fontenc.sty</code> for selecting encodings.
2kernel	for the kernel commands.
autoload	for the 'autoload' kernel commands.
autoerr	for the <code>autoerr.sty</code> error message autoload file.

19.4 Definitions for the kernel

19.4.1 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 \*2kernel | autoload)
2 \message{font encodings,}
```

Far too many macros in one block here!

```
\DeclareTextCommand      If you say:
\ProvideTextCommand      \DeclareTextCommand{\foo}{T1}...
\DeclareTextSymbol
  \@dec@text@cmd         then \foo is defined to be \T1-cmd \foo \T1\foo, where \T1\foo is one control
\chardef@text@cmd        sequence, not two! We then call \newcommand to define \T1\foo.
  \@changed@cmd          3 \def\DeclareTextCommand{%
  \@changed@x             4   \@dec@text@cmd\newcommand}
\TextSymbolUnavailable   5 \def\ProvideTextCommand{%
  \@inmathwarn            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\@@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     \inmathwarn#1%
28   \else
29     \noexpand#1\expandafter\@gobble
30   \fi}

31 \def\@changed@cmd#1#2{%
32   \ifx\protect\@typeset@protect
33     \inmathwarn#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 \</2ekernel | autoload>

```



```

50 <*2ekernel | autoerr>
51 \gdef\TextSymbolUnavailable#1{%
52   \latex@error{%
53     Command \protect#1 unavailable in encoding \cf@encoding%
54   }\@eha}
55 </2ekernel | autoerr>
56 <autoload>\gdef\TextSymbolUnavailable{\@autoerr\TextSymbolUnavailable}
57 <*2ekernel | autoload>

```

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).

```

58 \def\@inmathwarn#1{%
59   \ifmmode
60     \latex@warning{Command \protect#1 invalid in math mode}%
61   \fi}

```

`\DeclareTextCommandDefault` These define commands with encoding ?.

`\ProvideTextCommandDefault` 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.

```

62 \def\DeclareTextCommandDefault#1{%
63   \DeclareTextCommand#1?}
64 \def\ProvideTextCommandDefault#1{%
65   \ProvideTextCommand#1?}
66 \@onlypreamble\DeclareTextCommandDefault
67 %\@onlypreamble\ProvideTextCommandDefault

```

They require `\? -cmd` to be initialized as `\@changed@cmd`.

```

68 \expandafter\let\csname?-cmd\endcsname\@changed@cmd

```

`\DeclareTextAccent` This is just a disguise for defining a T_EX `\accent` command.

```

69 \def\DeclareTextAccent#1#2#3{%
70   \DeclareTextCommand#1{#2}{\add@accent{#3}}
71 \@onlypreamble\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 reimplementaion and its deficiencies, see pr/3160.

```
72 \def\add@accent#1#2{\hmode\bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
73   \let\hmode@start@before@group\@firstofone
74   \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.

```
75   \global\mathchardef\accent@spacefactor\spacefactor}%
76   \accent#1 #2\egroup\spacefactor\accent@spacefactor}
```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
77 \let\accent@spacefactor\relax
```

```
\hmode\bgroup
```

```
78 \def\hmode\bgroup{\leavevmode\bgroup}
```

<pre>\DeclareTextCompositeCommand \DeclareTextComposite \@text@composite \@text@composite@x \@strip@args</pre>	<p>Another amusing game to play with <code>\expandafter</code>, <code>\csname</code>, and <code>\string</code>. When you say <code>\DeclareTextCompositeCommand{\foo}{T1}{a}{bar}</code>, we look to see if the expansion of <code>T1\foo</code> begins with <code>\@text@composite</code>, and if it doesn't, we redefine <code>T1\foo</code> to be:</p> <pre> #1 -> \@text@composite T1\foo #1\@empty \@text@composite {...}</pre>
--	--

where ... is the previous definition of `T1\foo`. Finally, we define `\\T1\foo-a` to expand to `bar`.

```
79 \def\DeclareTextCompositeCommand#1#2#3#4{%
80   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
81   \expandafter\expandafter\expandafter\ifx
82   \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
83     \edef\reserved@b##1{%
84       \def\expandafter\noexpand
85         \csname#2\string#1\endcsname###1{%
86         \noexpand\@text@composite
87         \expandafter\noexpand\csname#2\string#1\endcsname
88         ###1\noexpand\@empty\noexpand\@text@composite
89         {##1}}}%
90   \expandafter\reserved@b\expandafter{\reserved@a{##1}}}%
91   \fi
92   \expandafter\def\csname\expandafter\string\csname
93     #2\endcsname\string#1-\string#3\endcsname{#4}}
94 \@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 `{...}` 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` 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.

```
95 \def\@text@composite#1#2#3\@text@composite{%
96   \expandafter\@text@composite@x
97   \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.

```
98 \def\@text@composite@x#1{%
99   \ifx#1\relax
100     \expandafter\@secondoftwo
101   \else
102     \expandafter\@firstoftwo
103   \fi
104   #1}
```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```
105 \catcode\z@=11\relax
106 \def\DeclareTextComposite#1#2#3#4{%
107   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
108   \bgroup
109     \lccode\z@#4%
110     \lowercase{%
111       \egroup
112       \reserved@a ^^@}}
```

```

113 \catcode\z@=15\relax
114 \@onlypreamble\DeclareTextComposite

```

`\UseTextAccent` These fragile commands access glyphs from different encodings. They use grotty
`\UseTextSymbol` low-level calls to the font selection scheme for speed, and in order to make sure
`\@use@text@encoding` 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 reimplementaion and its deficiencies, see
pr/3160.

```

115 \def\UseTextAccent#1#2#3{%
116   \hmode@start@before@group
117   {%
    Turn off the group in \UseTextSymbol in case this is used inside the arguments
    of \UseTextAccent.
118     \let\hmode@start@before@group\@firstofone
119     \let\@curr@enc\cf@encoding
120     \@use@text@encoding{#1}%
121     #2{\@use@text@encoding\@curr@enc#3}%
122   }}

```

```

123 \def\UseTextSymbol#1#2{%
124   \hmode@start@before@group
125   {%
126     \def\@wrong@font@char{\MessageBreak
127       for \noexpand\symbol'\string#2'}%
128     \@use@text@encoding{#1}%
129     #2%
130   }%
131 }

```

```

132 \def\@use@text@encoding#1{%
133   \edef\f@encoding{#1}%
134   \xdef\font@name{%
135     \csname\curr@fontshape/\f@size\endcsname}%
136   \pickup@font
137   \font@name
138   \@@enc@update}

```

`\hmode@start@before@group` The `\hmode@start@before@group` starts hmode and should be immediately fol-
lowed 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 conict with the `\accent` primitive.

For a detailed discussion see pr/3160.

```

139 \let\hmode@start@before@group\leavevmode

```

`\DeclareTextSymbolDefault` Some syntactic sugar. Again, these should probably be optimized for speed.

```

\DeclareTextAccentDefault 140 \def\DeclareTextSymbolDefault#1#2{%
141   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
142 \def\DeclareTextAccentDefault#1#2{%
143   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
144 \@onlypreamble\DeclareTextSymbolDefault
145 \@onlypreamble\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.

```
146 \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).

```
147 \expandafter\ifx\csname#2\string#1\endcsname\relax
```

```
148 \else
```

Else: throw away that declaration.

```
149 \global\expandafter\let\csname#2\string#1\endcsname
```

```
150 \@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 on 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:

```
151 \expandafter\expandafter\expandafter
152 \ifx\expandafter\@thirdofthree#1\@undefined
153 \expandafter\gdef\expandafter#1\expandafter
154 {\csname ?-cmd\expandafter\endcsname\expandafter
155 #1\csname?\string#1\endcsname}%
156 \fi
157 \fi
158 }
159 \@onlypreamble\UndeclareTextCommand
```

19.4.2 Hyphenation

`\patterns` We redefine `\patterns` and `\hyphenation` to allow the use of commands declared with `\DeclareText*` to be used inside them.

```
\@patterns 160 %\let\@patterns\patterns
\hyphenation 161 %\let\@hyphenation\hyphenation
\@hyphenation 162 %\def\patterns{%
163 % \bgroup
164 % \let\protect\@empty
165 % \let\@typeset@protect\@empty
166 % \let\@changed@x\@changed@x@mouth
167 % \afterassignment\egroup
168 % \@patterns
169 %}
170 %\def\hyphenation{%
```

```

171 % \bgroup
172 % \let\protect\@empty
173 % \let\@typeset@protect\@empty
174 % \let\@changed@x\@changed@x@mouth
175 % \afterassignment\egroup
176 % \@@hyphenation
177 %}

```

19.4.3 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.

```

178 \def\@tabacckludge#1{\expandafter\@changed@cmd
179                               \csname\string#1\endcsname\relax}
180 \let\a=\@tabacckludge

```

19.4.4 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 T_EX 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:

```

181 \DeclareTextAccentDefault{"}{OT1}
182 \DeclareTextAccentDefault{'}{OT1}
183 \DeclareTextAccentDefault{.}{OT1}
184 \DeclareTextAccentDefault{=}{OT1}
185 \DeclareTextAccentDefault{\H}{OT1}
186 \DeclareTextAccentDefault{\^}{OT1}
187 \DeclareTextAccentDefault{\'}{OT1}
188 \DeclareTextAccentDefault{\b}{OT1}
189 \DeclareTextAccentDefault{\c}{OT1}
190 \DeclareTextAccentDefault{\d}{OT1}
191 \DeclareTextAccentDefault{\r}{OT1}
192 \DeclareTextAccentDefault{\u}{OT1}
193 \DeclareTextAccentDefault{\v}{OT1}

```

```
194 \DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
195 %\DeclareTextSymbolDefault{\AA}{OT1}
```

```
196 \DeclareTextSymbolDefault{\AE}{OT1}
```

```
197 \DeclareTextSymbolDefault{\L}{OT1}
```

```
198 \DeclareTextSymbolDefault{\OE}{OT1}
```

```
199 \DeclareTextSymbolDefault{\O}{OT1}
```

```
200 %\DeclareTextSymbolDefault{\aa}{OT1}
```

```
201 \DeclareTextSymbolDefault{\ae}{OT1}
```

```
202 \DeclareTextSymbolDefault{\i}{OT1}
```

```
203 \DeclareTextSymbolDefault{\j}{OT1}
```

```
204 \DeclareTextSymbolDefault{\ij}{OT1}
```

```
205 \DeclareTextSymbolDefault{\IJ}{OT1}
```

```
206 \DeclareTextSymbolDefault{\l}{OT1}
```

```
207 \DeclareTextSymbolDefault{\oe}{OT1}
```

```
208 \DeclareTextSymbolDefault{\o}{OT1}
```

```
209 \DeclareTextSymbolDefault{\ss}{OT1}
```

```
210 \DeclareTextSymbolDefault{\textdollar}{OT1}
```

```
211 \DeclareTextSymbolDefault{\textendash}{OT1}
```

```
212 \DeclareTextSymbolDefault{\textendash}{OT1}
```

```
213 \DeclareTextSymbolDefault{\textexclamdown}{OT1}
```

```
214 %\DeclareTextSymbolDefault{\texthyphenchar}{OT1}
```

```
215 %\DeclareTextSymbolDefault{\texthyphen}{OT1}
```

```
216 \DeclareTextSymbolDefault{\textquestiondown}{OT1}
```

```
217 \DeclareTextSymbolDefault{\textquotedblleft}{OT1}
```

```
218 \DeclareTextSymbolDefault{\textquotedblright}{OT1}
```

```
219 \DeclareTextSymbolDefault{\textquoteleft}{OT1}
```

```
220 \DeclareTextSymbolDefault{\textquoteright}{OT1}
```

```
221 \DeclareTextSymbolDefault{\textsterling}{OT1}
```

Some symbols from OMS:

```
222 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
```

```
223 \DeclareTextSymbolDefault{\textbackslash}{OMS}
```

```
224 \DeclareTextSymbolDefault{\textbar}{OMS}
```

```
225 \DeclareTextSymbolDefault{\textbardbl}{OMS}
```

```
226 \DeclareTextSymbolDefault{\textbraceleft}{OMS}
```

```
227 \DeclareTextSymbolDefault{\textbraceright}{OMS}
```

```
228 \DeclareTextSymbolDefault{\textbullet}{OMS}
```

```
229 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
```

```
230 \DeclareTextSymbolDefault{\textdagger}{OMS}
```

```
231 \DeclareTextSymbolDefault{\textparagraph}{OMS}
```

```
232 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
```

```
233 \DeclareTextSymbolDefault{\textsection}{OMS}
```

```
234 \DeclareTextAccentDefault{\textcircled}{OMS}
```

Some symbols from OML:

```
235 \DeclareTextSymbolDefault{\textless}{OML}
```

```
236 \DeclareTextSymbolDefault{\textgreater}{OML}
```

```
237 \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.

```

238 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
239 % \expandafter\def\expandafter
240 % \copyright\expandafter{\expandafter{\copyright}}

241 \DeclareTextCommandDefault{\textasciicircum}{\^{}{}}
242 \DeclareTextCommandDefault{\textasciitilde}{\~{}{}}
243 \DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
244 \DeclareTextCommandDefault{\textunderscore}{\%
245 \leavevmode \kern.06em\vbox{\hrule\@width.3em}}

246 \DeclareTextCommandDefault{\textvisiblespace}{\%
247 \mbox{\kern.06em\vrule \@height.3ex}%
248 \vbox{\hrule \@width.3em}%
249 \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).

```

250 \DeclareTextCommandDefault{\textellipsis}{\%
251 \kern\fontdimen3\font
252 \kern\fontdimen3\font
253 \kern\fontdimen3\font}

254 %\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
255 \DeclareTextCommandDefault{\textregistered}{\textcircled{\%
256 \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
257 \DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
258 \DeclareTextCommandDefault{\SS}{SS}

259 \DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
260 \DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}

```

19.4.5 Math material

Some commands can be used in both text and math mode:

```

261 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}
262 \DeclareRobustCommand{\{ }{\ifmmode\lbrace\else\textbraceleft\fi}
263 \DeclareRobustCommand{\} }{\ifmmode\rbrace\else\textbraceright\fi}
264 \DeclareRobustCommand{\P }{\ifmmode\mathparagraph\else\textparagraph\fi}
265 \DeclareRobustCommand{\S }{\ifmmode\mathsection\else\textsection\fi}
266 \DeclareRobustCommand{\dag }{\ifmmode\dagger\else\textdagger\fi}
267 \DeclareRobustCommand{\ddag }{\ifmmode\ddagger\else\textdaggerdbl\fi}

```

For historical reasons `\copyright` needs `{ }` around the definition in maths.

```

268 \DeclareRobustCommand{\_ }{\%
269 \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
270 \DeclareRobustCommand{\copyright }{\%
271 \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
272 \DeclareRobustCommand{\pounds }{\%
273 \ifmmode\mathsterling\else\textsterling\fi}

274 \DeclareRobustCommand{\dots }{\%
275 \ifmmode\mathellipsis\else\textellipsis\fi}

276 \let\ldots\dots
277 \let\ldots\dots

```


19.5 Definitions for the OT1 encoding

The definitions for the ‘T_EX text’ (OT1) encoding.

Declare the encoding.

```
278 \*OT1)
279 \DeclareFontEncoding{OT1}{-}{-}
```

Declare the accents.

```
280 \DeclareTextAccent{"}{OT1}{127}
281 \DeclareTextAccent{'}{OT1}{19}
282 \DeclareTextAccent{.}{OT1}{95}
283 \DeclareTextAccent{=}{OT1}{22}
284 \DeclareTextAccent{^}{OT1}{94}
285 \DeclareTextAccent{'}{OT1}{18}
286 \DeclareTextAccent{~}{OT1}{126}
287 \DeclareTextAccent{H}{OT1}{125}
288 \DeclareTextAccent{u}{OT1}{21}
289 \DeclareTextAccent{v}{OT1}{20}
290 \DeclareTextAccent{r}{OT1}{23}
```

Some accents have to be built by hand: Note that `\oalign` 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.

```
291 \DeclareTextCommand{\b}{OT1}[1]
292   {\hmode\bgroup\o@lign{\relax#1\crr\hidewidth\ltx@sh@ft{-3ex}%
293     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
294 \DeclareTextCommand{\c}{OT1}[1]
295   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
296     \else{\oalign{\unhbox\z@\crr\hidewidth\char24\hidewidth}}\fi}
297 \DeclareTextCommand{\d}{OT1}[1]
298   {\hmode\bgroup
299     \o@lign{\relax#1\crr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
300 \DeclareTextSymbol{\AE}{OT1}{29}
301 \DeclareTextSymbol{\OE}{OT1}{30}
302 \DeclareTextSymbol{\O}{OT1}{31}
303 \DeclareTextSymbol{\ae}{OT1}{26}
304 \DeclareTextSymbol{\i}{OT1}{16}
305 \DeclareTextSymbol{\j}{OT1}{17}
306 \DeclareTextSymbol{\oe}{OT1}{27}
307 \DeclareTextSymbol{\o}{OT1}{28}
308 \DeclareTextSymbol{\ss}{OT1}{25}
309 \DeclareTextSymbol{\textemdash}{OT1}{124}
310 \DeclareTextSymbol{\textendash}{OT1}{123}
```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```
311 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}
312 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
313 \DeclareTextCommand{\textexclamdown}{OT1}{!'}
314 \DeclareTextCommand{\textquestiondown}{OT1}{?' }
315 %\DeclareTextSymbol{\texthyphenchar}{OT1}{'\-}
316 %\DeclareTextSymbol{\texthyphen}{OT1}{'\-}
317 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
```

```

318 \DeclareTextSymbol{\textquotedblright}{OT1}{‘\"}
319 \DeclareTextSymbol{\textquoteleft}{OT1}{‘\’}
320 \DeclareTextSymbol{\textquoteright}{OT1}{‘\’}

```

Some symbols which are faked from others:

```

321 % \DeclareTextCommand{\aa}{OT1}
322 %   {\accent23a}
323 \DeclareTextCommand{\L}{OT1}
324   {\leavevmode\setbox\z@\hbox{L}\hb@xt@\wd\z@{\hss\@xxxii L}}
325 \DeclareTextCommand{\l}{OT1}
326   {\hmode\bgroup\@xxxii l\egroup}
327 % \DeclareTextCommand{\AA}{OT1}
328 %   {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
329 %     \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`.

```

330 \DeclareTextCompositeCommand{\r}{OT1}{A}
331   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
332     \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. Therefor we fake it for the OT1 encoding.

```

333 \DeclareTextCommand{\ij}{OT1}{-%
334   \nobreak\hskip\z@skip i\kern-0.02em j\nobreak\hskip\z@skip}
335 \DeclareTextCommand{\IJ}{OT1}{-%
336   \nobreak\hskip\z@skip I\kern-0.02em J\nobreak\hskip\z@skip}

```

In the OT1 encoding, £ and \$ share a slot.

```

337 \DeclareTextCommand{\textdollar}{OT1}{\hmode\bgroup
338   \ifdim \fontdimen\@ne\font >\z@
339     \slshape
340   \else
341     \upshape
342   \fi
343   \char‘\$’\egroup}
344 \DeclareTextCommand{\textsterling}{OT1}{\hmode\bgroup
345   \ifdim \fontdimen\@ne\font >\z@
346     \itshape
347   \else
348     \fontshape{ui}\selectfont
349   \fi
350   \char‘\$’\egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with i compatible with their use with the T1 encoding; this enables them to become true L^AT_EX internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

351 \DeclareTextComposite{\.}{OT1}{i}{‘\i}
352 \DeclareTextComposite{\.}{OT1}{i}{‘\i}
353 \DeclareTextCompositeCommand{\’}{OT1}{i}{\@tabacckludge‘\i}
354 \DeclareTextCompositeCommand{\’}{OT1}{i}{\@tabacckludge‘\i}
355 \DeclareTextCompositeCommand{\^}{OT1}{i}{‘\^i}
356 \DeclareTextCompositeCommand{\"}{OT1}{i}{“\i}

```

357 $\langle /OT1 \rangle$

19.6 Definitions for the T1 encoding

The definitions for the ‘Extended T_EX text’ (T1) encoding.

Declare the encoding.

```
358  $\langle *T1 \rangle$ 
359 \DeclareFontEncoding{T1}{-}{-}

Declare the accents.

360 \DeclareTextAccent{\`}{T1}{0}
361 \DeclareTextAccent{\'}{T1}{1}
362 \DeclareTextAccent{\~}{T1}{2}
363 \DeclareTextAccent{\`}{T1}{3}
364 \DeclareTextAccent{\"}{T1}{4}
365 \DeclareTextAccent{\H}{T1}{5}
366 \DeclareTextAccent{\r}{T1}{6}
367 \DeclareTextAccent{\v}{T1}{7}
368 \DeclareTextAccent{\u}{T1}{8}
369 \DeclareTextAccent{\=}{T1}{9}
370 \DeclareTextAccent{\.}{T1}{10}
```

Some accents have to be built by hand. Note that `\ooalign` 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.

```
371 \DeclareTextCommand{\b}{T1}[1]
372   {\hmode\bgroup\o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-3ex}%
373     \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
374 \DeclareTextCommand{\c}{T1}[1]
375   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
376     \else{\ooalign{\unhbox\z@\cr\cr
377       \hidewidth\char11\hidewidth}}\fi}
378 \DeclareTextCommand{\d}{T1}[1]
379   {\hmode\bgroup
380     \o@lign{\relax#1\cr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
381 \DeclareTextCommand{\k}{T1}[1]
382   {\hmode\bgroup\ooalign{\null#1\cr\hidewidth\char12}\egroup}
383 \DeclareTextCommand{\textogonekcentered}{T1}[1]
384   {\hmode\bgroup\ooalign{\null#1\cr\hidewidth\char12\hidewidth}\egroup}
```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```
385 \DeclareTextCommand{\textperthousand}{T1}
386   {\%\char 24 } % space or ‘relax as delimiter?’
387 \DeclareTextCommand{\textpertenthousand}{T1}
388   {\%\char 24\char 24 } % space or ‘relax as delimiter?’
```

Declare the text symbols.

```
389 %\DeclareTextSymbol{\AA}{T1}{197}
390 \DeclareTextSymbol{\AE}{T1}{198}
391 \DeclareTextSymbol{\DH}{T1}{208}
392 \DeclareTextSymbol{\DJ}{T1}{208}
393 \DeclareTextSymbol{\L}{T1}{138}
394 \DeclareTextSymbol{\NG}{T1}{141}
```

```

395 \DeclareTextSymbol{\OE}{T1}{215}
396 \DeclareTextSymbol{\O}{T1}{216}
397 \DeclareTextSymbol{\SS}{T1}{223}
398 \DeclareTextSymbol{\TH}{T1}{222}
399 %\DeclareTextSymbol{\aa}{T1}{229}
400 \DeclareTextSymbol{\ae}{T1}{230}
401 \DeclareTextSymbol{\dh}{T1}{240}
402 \DeclareTextSymbol{\dj}{T1}{158}
403 \DeclareTextSymbol{\guillemotleft}{T1}{19}
404 \DeclareTextSymbol{\guillemotright}{T1}{20}
405 \DeclareTextSymbol{\guilsinglleft}{T1}{14}
406 \DeclareTextSymbol{\guilsinglright}{T1}{15}
407 \DeclareTextSymbol{\i}{T1}{25}
408 \DeclareTextSymbol{\j}{T1}{26}
409 \DeclareTextSymbol{\ij}{T1}{188}
410 \DeclareTextSymbol{\IJ}{T1}{156}
411 \DeclareTextSymbol{\l}{T1}{170}
412 \DeclareTextSymbol{\ng}{T1}{173}
413 \DeclareTextSymbol{\oe}{T1}{247}
414 \DeclareTextSymbol{\o}{T1}{248}
415 \DeclareTextSymbol{\quotedblbase}{T1}{18}
416 \DeclareTextSymbol{\quotesinglbase}{T1}{13}
417 \DeclareTextSymbol{\ss}{T1}{255}
418 \DeclareTextSymbol{\textasciicircum}{T1}{`\^}
419 \DeclareTextSymbol{\textasciitilde}{T1}{`\~}
420 \DeclareTextSymbol{\textbackslash}{T1}{`\}
421 \DeclareTextSymbol{\textbar}{T1}{`\|}
422 \DeclareTextSymbol{\textbraceleft}{T1}{`\{ }
423 \DeclareTextSymbol{\textbraceright}{T1}{`\} }
424 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
425 \DeclareTextSymbol{\textdollar}{T1}{`\$}
426 \DeclareTextSymbol{\textemdash}{T1}{22}
427 \DeclareTextSymbol{\textendash}{T1}{21}
428 \DeclareTextSymbol{\textexclamdown}{T1}{189}
429 \DeclareTextSymbol{\textgreater}{T1}{`\>}
430 %\DeclareTextSymbol{\texthyphenchar}{T1}{127}
431 %\DeclareTextSymbol{\texthyphen}{T1}{`\-}
432 \DeclareTextSymbol{\textless}{T1}{`\<}
433 \DeclareTextSymbol{\textquestiondown}{T1}{190}
434 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
435 \DeclareTextSymbol{\textquotedblright}{T1}{17}
436 \DeclareTextSymbol{\textquotedbl}{T1}{`\"}
437 \DeclareTextSymbol{\textquoteleft}{T1}{`\'}
438 \DeclareTextSymbol{\textquoteright}{T1}{`\'}
439 \DeclareTextSymbol{\textsection}{T1}{159}
440 \DeclareTextSymbol{\textsterling}{T1}{191}
441 \DeclareTextSymbol{\textunderscore}{T1}{95}
442 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
443 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

444 \DeclareTextComposite{\.}{T1}{i}{`\i}
445 \DeclareTextComposite{\.}{T1}{i}{`\i}

```

"80 = 128

```

446 \DeclareTextComposite{\u}{T1}{A}{128}
447 \DeclareTextComposite{\k}{T1}{A}{129}
448 \DeclareTextComposite{\'}{T1}{C}{130}
449 \DeclareTextComposite{\v}{T1}{C}{131}
450 \DeclareTextComposite{\v}{T1}{D}{132}
451 \DeclareTextComposite{\v}{T1}{E}{133}
452 \DeclareTextComposite{\k}{T1}{E}{134}
453 \DeclareTextComposite{\u}{T1}{G}{135}
"88 = 136
454 \DeclareTextComposite{\'}{T1}{L}{136}
455 \DeclareTextComposite{\v}{T1}{L}{137}
456 \DeclareTextComposite{\'}{T1}{N}{139}
457 \DeclareTextComposite{\v}{T1}{N}{140}
458 \DeclareTextComposite{\H}{T1}{O}{142}
459 \DeclareTextComposite{\'}{T1}{R}{143}
"90 = 144
460 \DeclareTextComposite{\v}{T1}{R}{144}
461 \DeclareTextComposite{\'}{T1}{S}{145}
462 \DeclareTextComposite{\v}{T1}{S}{146}
463 \DeclareTextComposite{\c}{T1}{S}{147}
464 \DeclareTextComposite{\v}{T1}{T}{148}
465 \DeclareTextComposite{\c}{T1}{T}{149}
466 \DeclareTextComposite{\H}{T1}{U}{150}
467 \DeclareTextComposite{\r}{T1}{U}{151}
"98 = 152
468 \DeclareTextComposite{\"}{T1}{Y}{152}
469 \DeclareTextComposite{\'}{T1}{Z}{153}
470 \DeclareTextComposite{\v}{T1}{Z}{154}
471 \DeclareTextComposite{\.}{T1}{Z}{155}
472 \DeclareTextComposite{\.}{T1}{I}{157}
"A0 = 160
473 \DeclareTextComposite{\u}{T1}{a}{160}
474 \DeclareTextComposite{\k}{T1}{a}{161}
475 \DeclareTextComposite{\'}{T1}{c}{162}
476 \DeclareTextComposite{\v}{T1}{c}{163}
477 \DeclareTextComposite{\v}{T1}{d}{164}
478 \DeclareTextComposite{\v}{T1}{e}{165}
479 \DeclareTextComposite{\k}{T1}{e}{166}
480 \DeclareTextComposite{\u}{T1}{g}{167}
"A8 = 168
481 \DeclareTextComposite{\'}{T1}{l}{168}
482 \DeclareTextComposite{\v}{T1}{l}{169}
483 \DeclareTextComposite{\'}{T1}{n}{171}
484 \DeclareTextComposite{\v}{T1}{n}{172}
485 \DeclareTextComposite{\H}{T1}{o}{174}
486 \DeclareTextComposite{\'}{T1}{r}{175}
"B0 = 176
487 \DeclareTextComposite{\v}{T1}{r}{176}
488 \DeclareTextComposite{\'}{T1}{s}{177}
489 \DeclareTextComposite{\v}{T1}{s}{178}

```

```

490 \DeclareTextComposite{\c}{T1}{s}{179}
491 \DeclareTextComposite{\v}{T1}{t}{180}
492 \DeclareTextComposite{\c}{T1}{t}{181}
493 \DeclareTextComposite{\H}{T1}{u}{182}
494 \DeclareTextComposite{\r}{T1}{u}{183}

"B8 = 184

495 \DeclareTextComposite{\"}{T1}{y}{184}
496 \DeclareTextComposite{\'}{T1}{z}{185}
497 \DeclareTextComposite{\v}{T1}{z}{186}
498 \DeclareTextComposite{\.}{T1}{z}{187}

"C0 = 192

499 \DeclareTextComposite{\'}{T1}{A}{192}
500 \DeclareTextComposite{\'}{T1}{A}{193}
501 \DeclareTextComposite{\^}{T1}{A}{194}
502 \DeclareTextComposite{\~}{T1}{A}{195}
503 \DeclareTextComposite{\"}{T1}{A}{196}
504 \DeclareTextComposite{\r}{T1}{A}{197}
505 \DeclareTextComposite{\c}{T1}{C}{199}

"C8 = 200

506 \DeclareTextComposite{\'}{T1}{E}{200}
507 \DeclareTextComposite{\'}{T1}{E}{201}
508 \DeclareTextComposite{\^}{T1}{E}{202}
509 \DeclareTextComposite{\"}{T1}{E}{203}
510 \DeclareTextComposite{\'}{T1}{I}{204}
511 \DeclareTextComposite{\'}{T1}{I}{205}
512 \DeclareTextComposite{\^}{T1}{I}{206}
513 \DeclareTextComposite{\"}{T1}{I}{207}

"D0 = 208

514 \DeclareTextComposite{\~}{T1}{N}{209}
515 \DeclareTextComposite{\'}{T1}{O}{210}
516 \DeclareTextComposite{\'}{T1}{O}{211}
517 \DeclareTextComposite{\^}{T1}{O}{212}
518 \DeclareTextComposite{\~}{T1}{O}{213}
519 \DeclareTextComposite{\"}{T1}{O}{214}

"D8 = 216

520 \DeclareTextComposite{\'}{T1}{U}{217}
521 \DeclareTextComposite{\'}{T1}{U}{218}
522 \DeclareTextComposite{\^}{T1}{U}{219}
523 \DeclareTextComposite{\"}{T1}{U}{220}
524 \DeclareTextComposite{\'}{T1}{Y}{221}

"E0 = 224

525 \DeclareTextComposite{\'}{T1}{a}{224}
526 \DeclareTextComposite{\'}{T1}{a}{225}
527 \DeclareTextComposite{\^}{T1}{a}{226}
528 \DeclareTextComposite{\~}{T1}{a}{227}
529 \DeclareTextComposite{\"}{T1}{a}{228}
530 \DeclareTextComposite{\r}{T1}{a}{229}
531 \DeclareTextComposite{\c}{T1}{c}{231}

"E8 = 232

532 \DeclareTextComposite{\'}{T1}{e}{232}

```

```

533 \DeclareTextComposite{'}{T1}{e}{233}
534 \DeclareTextComposite{^}{T1}{e}{234}
535 \DeclareTextComposite{"}{T1}{e}{235}
536 \DeclareTextComposite{'}{T1}{i}{236}
537 \DeclareTextComposite{'}{T1}{i}{236}
538 \DeclareTextComposite{'}{T1}{i}{237}
539 \DeclareTextComposite{'}{T1}{i}{237}
540 \DeclareTextComposite{^}{T1}{i}{238}
541 \DeclareTextComposite{^}{T1}{i}{238}
542 \DeclareTextComposite{"}{T1}{i}{239}
543 \DeclareTextComposite{"}{T1}{i}{239}

"FO = 240

544 \DeclareTextComposite{~}{T1}{n}{241}
545 \DeclareTextComposite{'}{T1}{o}{242}
546 \DeclareTextComposite{'}{T1}{o}{243}
547 \DeclareTextComposite{^}{T1}{o}{244}
548 \DeclareTextComposite{~}{T1}{o}{245}
549 \DeclareTextComposite{"}{T1}{o}{246}

"F8 = 248

550 \DeclareTextComposite{'}{T1}{u}{249}
551 \DeclareTextComposite{'}{T1}{u}{250}
552 \DeclareTextComposite{^}{T1}{u}{251}
553 \DeclareTextComposite{"}{T1}{u}{252}
554 \DeclareTextComposite{'}{T1}{y}{253}

555 \DeclareTextCompositeCommand{k}{T1}{o}{\textogonekcentered{o}}
556 \DeclareTextCompositeCommand{k}{T1}{0}{\textogonekcentered{0}}

557 </T1>

```

19.7 Definitions for the OMS encoding

The definitions for the ‘TeX math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard L^AT_EX text symbols.

Declare the encoding.

```

558 <*OMS>
559 \DeclareFontEncoding{OMS}{-}{-}

Declare the symbols.

560 % \changes{v1.99}{2004/02/02}{Added \cs{textbigcircle}}
561 %   Note that slot 13 has in places been named |\Orb|: please root
562 %   out and destroy this impolity wherever you find it!
563 %   \begin{macrocode}
564 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3}      % "03
565 \DeclareTextSymbol{\textbackslash}{OMS}{110}           % "6E
566 \DeclareTextSymbol{\textbar}{OMS}{106}                 % "6A
567 \DeclareTextSymbol{\textbardbl}{OMS}{107}              % "6B
568 \DeclareTextSymbol{\textbraceleft}{OMS}{102}           % "66
569 \DeclareTextSymbol{\textbraceright}{OMS}{103}          % "67
570 \DeclareTextSymbol{\textbullet}{OMS}{15}               % "0F
571 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122}           % "7A
572 \DeclareTextSymbol{\textdagger}{OMS}{121}              % "79
573 \DeclareTextSymbol{\textparagraph}{OMS}{123}           % "7B

```

```

574 \DeclareTextSymbol{\textperiodcentered}{OMS}{1}      % "01
575 \DeclareTextSymbol{\textsection}{OMS}{120}            % "78
576 \DeclareTextSymbol{\textbigcirc}{OMS}{13}             % "0D
577 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
578   \ooalign{%
579     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
580     \char 13 % "0D
581   }%
582 \egroup}
583 \</OMS>

```

19.8 Definitions for the OML encoding

The definitions for the ‘T_EX math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard L^AT_EX text symbols.

Declare the encoding.

```

584 \<*OML>
585 \DeclareFontEncoding{OML}{}{}
586 \DeclareTextSymbol{\textless}{OML}{'\<}
587 \DeclareTextSymbol{\textgreater}{OML}{'\>}
588 \DeclareTextAccent{\t}{OML}{127} % "7F
589 \</OML>

```

19.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘T_EX 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 L^AT_EX 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.

```

590 \<*OT4>
591 \DeclareFontEncoding{OT4}{}{}
592 \DeclareFontSubstitution{OT4}{cmr}{m}{n}
593 \DeclareTextAccent{"}{OT4}{127}
594 \DeclareTextAccent{\'}{OT4}{19}
595 \DeclareTextAccent{\.}{OT4}{95}
596 \DeclareTextAccent{\=}{OT4}{22}
597 \DeclareTextAccent{\~}{OT4}{94}
598 \DeclareTextAccent{\'}{OT4}{18}
599 \DeclareTextAccent{\~}{OT4}{126}
600 \DeclareTextAccent{\H}{OT4}{125}
601 \DeclareTextAccent{\u}{OT4}{21}
602 \DeclareTextAccent{\v}{OT4}{20}
603 \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:

```
604 \DeclareTextCommand{\k}{OT4}[1]{%
605   \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.

606 \DeclareTextCommand{\b}{OT4}[1]
607   {\hmode\bgroup\o@lign{\relax#1\crr\hidewidth\ltx@sh@ft{-3ex}%
608     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
609 \DeclareTextCommand{\c}{OT4}[1]
610   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
611     \else{\oalign{\unhbox\z@\crr\hidewidth\char24\hidewidth}}\fi}
612 \DeclareTextCommand{\d}{OT4}[1]
613   {\hmode\bgroup
614     \o@lign{\relax#1\crr\hidewidth\ltx@sh@ft{-1ex}.\hidewidth}\egroup}
```

Declare the text symbols.

```
615 \DeclareTextSymbol{\AE}{OT4}{29}
616 \DeclareTextSymbol{\OE}{OT4}{30}
617 \DeclareTextSymbol{\O}{OT4}{31}
618 \DeclareTextSymbol{\L}{OT4}{138}
619 \DeclareTextSymbol{\ae}{OT4}{26}
620 \DeclareTextSymbol{\guillemotleft}{OT4}{174}
621 \DeclareTextSymbol{\guillemotright}{OT4}{175}
622 \DeclareTextSymbol{\i}{OT4}{16}
623 \DeclareTextSymbol{\j}{OT4}{17}
624 \DeclareTextSymbol{\l}{OT4}{170}
625 \DeclareTextSymbol{\o}{OT4}{28}
626 \DeclareTextSymbol{\oe}{OT4}{27}
627 \DeclareTextSymbol{\quotedblbase}{OT4}{255}
628 \DeclareTextSymbol{\ss}{OT4}{25}
629 \DeclareTextSymbol{\textendash}{OT4}{124}
630 \DeclareTextSymbol{\textendash}{OT4}{123}
631 \DeclareTextSymbol{\textexclamdown}{OT4}{60}
632 %\DeclareTextSymbol{\textthyphenchar}{OT4}{'\-}
633 %\DeclareTextSymbol{\textthyphen}{OT4}{'\-}
634 \DeclareTextSymbol{\textquestiondown}{OT4}{62}
635 \DeclareTextSymbol{\textquotedblleft}{OT4}{92}
636 \DeclareTextSymbol{\textquotedblright}{OT4}{'\'}
637 \DeclareTextSymbol{\textquoteleft}{OT4}{'\'}
638 \DeclareTextSymbol{\textquoteright}{OT4}{'\'}%
```

Definition for Å as in OT1:

```
639 \DeclareTextCompositeCommand{\r}{OT4}{A}
640   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
641     \rlap{\raise.67\dimen@\hbox{\char23}}A}
```

In the OT4 encoding, £ and \$ share a slot.

```
642 \DeclareTextCommand{\textdollar}{OT4}{\hmode\bgroup
643   \ifdim \fontdimen\@ne\font >\z@
644     \slshape
645   \else
646     \upshape
647   \fi}
```

```

648 \char'\$ \egroup}
649 \DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
650 \ifdim \fontdimen\@ne\font >\z@
651 \itshape
652 \else
653 \fontshape{ui}\selectfont
654 \fi
655 \char'\$ \egroup}

```

Declare the composites.

```

656 \DeclareTextComposite{\k}{OT4}{A}{129}
657 \DeclareTextComposite{\'}{OT4}{C}{130}
658 \DeclareTextComposite{\k}{OT4}{E}{134}
659 \DeclareTextComposite{\'}{OT4}{N}{139}
660 \DeclareTextComposite{\'}{OT4}{S}{145}
661 \DeclareTextComposite{\'}{OT4}{Z}{153}
662 \DeclareTextComposite{\.}{OT4}{Z}{155}
663 \DeclareTextComposite{\k}{OT4}{a}{161}
664 \DeclareTextComposite{\'}{OT4}{c}{162}
665 \DeclareTextComposite{\k}{OT4}{e}{166}
666 \DeclareTextComposite{\'}{OT4}{n}{171}
667 \DeclareTextComposite{\'}{OT4}{s}{177}
668 \DeclareTextComposite{\'}{OT4}{z}{185}
669 \DeclareTextComposite{\.}{OT4}{z}{187}
670 \DeclareTextComposite{\'}{OT4}{0}{211}
671 \DeclareTextComposite{\'}{OT4}{o}{243}
672 \</OT4>

```

19.10 Definitions for the TS1 encoding

```

673 \<*TS1>
674 \DeclareFontEncoding{TS1}{}{}
675 \DeclareFontSubstitution{TS1}{cmr}{m}{n}

```

Some accents have to be built by hand. Note that `\oalign` and `\o@lign` must be inside a group.

```

676 \DeclareTextCommand{\capitalcedilla}{TS1}[1]
677 {\hmode@bgroup
678 \oalign{\null#1\crcr\hidewidth\char11\hidewidth}\egroup}
679 \DeclareTextCommand{\capitalogonek}{TS1}[1]
680 {\hmode@bgroup
681 \oalign{\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

```

682 \DeclareTextAccent{\capitalgrave}{TS1}{0}
683 \DeclareTextAccent{\capitalacute}{TS1}{1}
684 \DeclareTextAccent{\capitalcircumflex}{TS1}{2}

```

```

685 \DeclareTextAccent{\capitaltilde}{TS1}{3}
686 \DeclareTextAccent{\capitaldieresis}{TS1}{4}
687 \DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}
688 \DeclareTextAccent{\capitalring}{TS1}{6}
689 \DeclareTextAccent{\capitalcaron}{TS1}{7}

```

"08 = 8

```

690 \DeclareTextAccent{\capitalbreve}{TS1}{8}
691 \DeclareTextAccent{\capitalmacron}{TS1}{9}
692 \DeclareTextAccent{\capitaldotaccent}{TS1}{10}

```

Tie accents.

The tie accent was borrowed from the `cmmt` font. The `tc` fonts now provide four tie accents, the first two are done in the classical way with assymetric 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.

" =

```

693 \DeclareTextAccent{\t}{TS1}{26}
694 \DeclareTextAccent{\capitaltie}{TS1}{27}
695 \DeclareTextAccent{\newtie}{TS1}{28}
696 \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`.

```

697 \DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
698 \DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}

```

The text companion symbols.

```

699 \DeclareTextSymbol{\textquotestraightbase}{TS1}{13}

```

"10 = 16

```

700 \DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
701 \DeclareTextSymbol{\texttwelveudash}{TS1}{21}
702 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}

```

"18 = 24

```

703 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
704 \DeclareTextSymbol{\textrightarrow}{TS1}{25}

```

"20 = 32

```

705 \DeclareTextSymbol{\textblank}{TS1}{32}
706 \DeclareTextSymbol{\textdollar}{TS1}{36}
707 \DeclareTextSymbol{\textquotesingle}{TS1}{39}

```

"28 = 40

```

708 \DeclareTextSymbol{\textasteriskcentered}{TS1}{42}

```

Note that '054 is a comma and '056 is a full stop: these make numbers using oldstyle digits easier to input.

```

709 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
710 \DeclareTextSymbol{\textfractionsolidus}{TS1}{47}

```

Oldstyle digits.

"30 = 48

```

711 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}

```

```

712 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
713 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
714 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
715 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
716 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
717 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
718 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}
"38 = 56
719 \DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
720 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}
    More text companion symbols.
721 \DeclareTextSymbol{\textlangle}{TS1}{60}
722 \DeclareTextSymbol{\textminus}{TS1}{61}
723 \DeclareTextSymbol{\textrangle}{TS1}{62}
"48 = 72
724 \DeclareTextSymbol{\textmho}{TS1}{77}
    The big circle is here to define the command \textcircled. Formerly it was
    taken from the cmsy font.
725 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
726 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
727     \ooalign{%
728         \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
729         \char 79     % '117 = "4F
730     }%
731 \egroup}
    More text companion symbols.
"50 = 80
732 \DeclareTextSymbol{\textohm}{TS1}{87}
"58 = 88
733 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
734 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
735 \DeclareTextSymbol{\textuparrow}{TS1}{94}
736 \DeclareTextSymbol{\textdownarrow}{TS1}{95}
"60 = 96
737 \DeclareTextSymbol{\textasciigrave}{TS1}{96}
738 \DeclareTextSymbol{\textborn}{TS1}{98}
739 \DeclareTextSymbol{\textdivorced}{TS1}{99}
740 \DeclareTextSymbol{\textdied}{TS1}{100}
"68 = 104
741 \DeclareTextSymbol{\textleaf}{TS1}{108}
742 \DeclareTextSymbol{\textmarried}{TS1}{109}
743 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}
"78 = 120
744 \DeclareTextSymbol{\texttildelow}{TS1}{126}
    This glyph, \textdblhyphenchar is hanging, like the hyphenchar of the ec
    fonts.
745 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}

```

”80 = 128

746 \DeclareTextSymbol{\textasciibreve}{TS1}{128}

747 \DeclareTextSymbol{\textasciicaron}{TS1}{129}

This next glyph is *not* the same as \textquotedbl.

748 \DeclareTextSymbol{\textacutedbl}{TS1}{130}

749 \DeclareTextSymbol{\textgravedbl}{TS1}{131}

750 \DeclareTextSymbol{\textdagger}{TS1}{132}

751 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}

752 \DeclareTextSymbol{\textbardbl}{TS1}{134}

753 \DeclareTextSymbol{\textperthousand}{TS1}{135}

”88 = 136

754 \DeclareTextSymbol{\textbullet}{TS1}{136}

755 \DeclareTextSymbol{\textcelsius}{TS1}{137}

756 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}

757 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}

758 \DeclareTextSymbol{\textflorin}{TS1}{140}

759 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}

760 \DeclareTextSymbol{\textwon}{TS1}{142}

761 \DeclareTextSymbol{\textnaira}{TS1}{143}

”90 = 144

762 \DeclareTextSymbol{\textguarani}{TS1}{144}

763 \DeclareTextSymbol{\textpeso}{TS1}{145}

764 \DeclareTextSymbol{\textlira}{TS1}{146}

765 \DeclareTextSymbol{\textrecipe}{TS1}{147}

766 \DeclareTextSymbol{\textinterrobang}{TS1}{148}

767 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}

768 \DeclareTextSymbol{\textdong}{TS1}{150}

769 \DeclareTextSymbol{\texttrademark}{TS1}{151}

”98 = 152

770 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}

771 \DeclareTextSymbol{\textpilcrow}{TS1}{153}

772 \DeclareTextSymbol{\textbaht}{TS1}{154}

773 \DeclareTextSymbol{\textnumero}{TS1}{155}

This next name may change. For the following sign we know only a german name, which is abzüglich. The meaning is something like “commercial minus”. An ASCII ersatz is ./ (dot slash dot). The temporary English name is \textdiscount.

774 \DeclareTextSymbol{\textdiscount}{TS1}{156}

775 \DeclareTextSymbol{\textestimated}{TS1}{157}

776 \DeclareTextSymbol{\textopenbullet}{TS1}{158}

777 \DeclareTextSymbol{\textservicemark}{TS1}{159}

”A0 = 160

778 \DeclareTextSymbol{\textlquill}{TS1}{160}

779 \DeclareTextSymbol{\textrquill}{TS1}{161}

780 \DeclareTextSymbol{\textcent}{TS1}{162}

781 \DeclareTextSymbol{\textsterling}{TS1}{163}

782 \DeclareTextSymbol{\textcurrency}{TS1}{164}

783 \DeclareTextSymbol{\textyen}{TS1}{165}

784 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}

785 \DeclareTextSymbol{\textsection}{TS1}{167}

```

" A8 = 168
786 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
787 \DeclareTextSymbol{\textcopyright}{TS1}{169}
788 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
789 \DeclareTextSymbol{\textcopyrightleft}{TS1}{171}
790 \DeclareTextSymbol{\textlnot}{TS1}{172}

    The meaning of the circled-P is "sound recording copyright".

791 \DeclareTextSymbol{\textcircledP}{TS1}{173}
792 \DeclareTextSymbol{\textregistered}{TS1}{174}
793 \DeclareTextSymbol{\textasciimacron}{TS1}{175}

" B0 = 176
794 \DeclareTextSymbol{\textdegree}{TS1}{176}
795 \DeclareTextSymbol{\textpm}{TS1}{177}
796 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
797 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
798 \DeclareTextSymbol{\textasciacute}{TS1}{180}
799 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
800 \DeclareTextSymbol{\textparagraph}{TS1}{182}
801 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}

" B8 = 184
802 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
803 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
804 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
805 \DeclareTextSymbol{\textsurd}{TS1}{187}
806 \DeclareTextSymbol{\textonequarter}{TS1}{188}
807 \DeclareTextSymbol{\textonehalf}{TS1}{189}
808 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
809 \DeclareTextSymbol{\texteuro}{TS1}{191}

" E0 = 208
810 \DeclareTextSymbol{\texttimes}{TS1}{214}

" F0 = 240
811 \DeclareTextSymbol{\textdiv}{TS1}{246}
812 \</TS1>

```

20 Package files

This file now also contains some packages that provide access to the more specialised encodings.

20.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding F00, the package looks to see if the encoding F00 has already been declared. If it has not, the file `fooenc.def` is loaded. The default encoding is set to be F00.

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.

813 `{*package}`

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

```
814 \def@update@uclc@with@cyrillic{%
815   \expandafter\def\expandafter\@uclclist\expandafter
816     {\@uclclist
817       \cyra\CYRA\cyrabhch\CYRABHCH\cyrabhchdsc\CYRABHCHDSC\cyrabhdze
818       \CYRABHDZE\cyrabhha\CYRABHHA\cyrae\CYRAE\cyrb\CYRB\cyrbys
819       \CYRBYUS\cyrc\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrdsc
820       \CYRCHRDSC\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
821       \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
822       \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
823       \CYRFITA\cyrg\CYRG\cyrgdsc\CYRGDSC\cyrgdschcrs\CYRGDSCHCRS
824       \cyrgchcrs\CYRGHCRS\cyrgkh\CYRGHK\cyrgup\CYRGUP\cyrh\CYRH
825       \cyrhdsdsc\CYRHDS\cyrhcrs\CYRHHCRS\cyrhkh\CYRHHK\cyrhdsn
826       \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
827       \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
828       \cyrkbeak\CYRKBEAK\cyrkdsdsc\CYRKDSC\cyrkchcrs\CYRKHCRS\cyrkhh
829       \CYRKHK\cyrkvcrs\CYRKVCRS\cyr1\CYRL\cyrldsc\CYRLDSC\cyr1hk
830       \CYRLHK\cyr1je\CYRLJE\cyrm\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMHK
831       \cyrn\CYRN\cyrndsc\CYRNDSC\cyrng\CYRNG\cyrnkh\CYRNHK\cyrnje
832       \CYRNJE\cyrn1hk\CYRN1HK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
833       \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsdsc\CYRRDSC\cyrrhk
834       \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
835       \cyrschwa\CYRSCHWA\cyrsdsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
836       \cyrsftsn\CYRSFTSN\cyrsh\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
837       \cyrt\CYRT\cyrtddsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
838       \cyru\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyr1\CYRY
839       \CYRYA\CYRYA\cyr1at\CYRYAT\cyr1hcrs\CYRYHCRS\cyr1i\CYRYI\cyr1o
840       \CYRYO\cyr1u\CYRYU\cyrz\CYRZ\cyrzdsdsc\CYRZDSC\cyrzh\CYRZH
841       \cyrzhdsc\CYRZHDSC}%
842 \let@update@uclc@with@cyrillic\relax
843 }
```

Here we process each option:

```
844 \DeclareOption*{%
845   \let\encodingdefault\CurrentOption
846   \edef\reserved@f{%
847     \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
848   \reserved@f
849   \InputIfFileExists\reserved@f
850     {}{\PackageError{fontenc}%
851       {Encoding file '\reserved@f' not found.%
852       \MessageBreak
853       You might have misspelt the name of the encoding}%
854     {Necessary code for this encoding was not
855     loaded.\MessageBreak
856     Thus calling the encoding later on will
857     produce further error messages.}}%
858   \let\reserved@f\relax
```

In case the current encoding is one of a list of known cyrillic ones we extend

```

the \@uclclist:
859   \expandafter\in@\expandafter{\CurrentOption}%
860                                   {T2A,T2B,T2C,X2,LCY,OT2}%
861   \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.

862   \expandafter\in@\expandafter\cyrillic\expandafter
863                                   {\@uclclist}%
864   \ifin@
865   \else
866       \update@uclc@with@cyrillic
867   \fi
868 \fi
869 }

870 \ProcessOptions*
871 \fontencoding\encodingdefault\selectfont

```

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

```
872 \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.

```

873 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
874 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
875 \global\let\@ifl@ter@\@ifl@ter
876 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@@}
877 \</package>

```

20.2 The textcomp 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.

```
878 \<TS1sty>
```

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 $TS1 \cap (ISO-Adobe \cup MacRoman)$.
(Except for the missing "currency".)

#2 = #3 + `\textestimated` + `\textcurrency`. Can also be described as $TS1 \cap Adobe-Western-2$. This may be relevant for OpenType fonts, which usually show the Adobe-Western-2 character set.

#1 = TS1 without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

#0 = full TS1

And here a summary to go in the transcript file:

```

879 \PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
880   \space\space 5 = only ISO-Adobe without \string\textcurrency\MessageBreak
881   \space\space 4 = 5 + \string\texteuro\MessageBreak
882   \space\space 3 = 4 + \string\textohm\MessageBreak
883   \space\space 2 = 3 + \noexpand\textestimated+ \string\textcurrency\MessageBreak
884   \space\space 1 = TS1 - \noexpand\textcircled- \string\t\MessageBreak
885   \space\space 0 = TS1 (full)\MessageBreak
886   Font families with sub-encoding setting implement\MessageBreak
887   only a restricted character set as indicated.\MessageBreak
888   Family '?' is the default used for unknown fonts.\MessageBreak
889   See the documentation for details\@gobble}

```

`\DeclareEncodingSubset` An encoding subset to which a font family belongs is declared by `\DeclareEncodingSubset` that take the major encoding as the first argument (e.g., TS1), the family name as the second argument (e.g., cmr), and the subset encoding id as a third, (e.g., 0 for cmr).

The default encoding subset to use when nothing is known about the current font family is named ?.

```

890 \def\DeclareEncodingSubset#1#2#3{%
891   \@ifundefined{#1:#2}%
892     {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
893     {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
894   \@namedef{#1:#2}{#3}}
895 \onlypreamble\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

```
896 \newif\iftc@forced \tc@forcedfalse
```

This is implemented by defining the default subset:

```
897 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
898 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
899 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
900 \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.

```
901 \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.

```
902 \def\tc@errorwarn{\PackageError}
903 \DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
904 \ExecuteOptions{almostfull}
905 \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.

```
906 \iftc@forced
```

If the “force” option was given we always use the default for testing against.

```
907 \def\CheckEncodingSubset#1#2#3#4#5{%
908   \ifnum #4>%
909     0\csname #2:\endcsname
910     \relax
911   \expandafter\@firstoftwo
912   \else
913   \expandafter\@secondoftwo
914   \fi
```

```

915   {#1{#2}}{#3}%
916   #5%
917 }

```

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.

```

918 \else
919 \def\CheckEncodingSubset#1#2#3#4#5{%
920   \ifnum #4>%
921     \expandafter\ifx\csname #2:\f@family\endcsname\relax
922       0\csname #2:\f@family\endcsname
923     \else
924       \csname #2:\f@family\endcsname
925     \fi
926   \relax
927   \expandafter\@firstoftwo
928 \else
929   \expandafter\@secondoftwo
930 \fi
931 {#1{#2}}{#3}%
932 #5%
933 }
934 \fi

```

tc@subst

```

935 \def\tc@subst#1{%
936   \tc@errorwarn{textcomp}% % should be latex error if general
937   {Symbol \string#1 not provided by\MessageBreak
938     font family \f@family\space
939     in TS1 encoding.\MessageBreak Default family used instead}\@eha
940   \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
941 }

```

\textcompsubstdefault

```

942 \def\textcompsubstdefault{cmr}

```

\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.

```

943 % error commands take argument:
944 % #1 symbol to be used
945 \def\tc@error#1{%
946   \PackageError{textcomp}% % should be latex error if general
947   {Accent \string#1 not provided by\MessageBreak
948     font family \f@family\space
949     in TS1 encoding}\@eha
950 }

```

\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 “=”.

```

951 \def\tc@fake@euro#1{%

```

```

952 \leavevmode
953 \PackageInfo{textcomp}{Faking \noexpand#1 for font family
954 \f@family\MessageBreak in TS1 encoding}%
955 \valign{##\cr
956 \vfil\hbox to 0.07em{\dimen@\f@size\p@
957 \math@fontsfalse
958 \fontsize{.7\dimen@}\z@\selectfont=\hss}\vfil\cr%
959 \hbox{C}\crrc
960 }%
961 }

```

\tc@check@symbol These are two abbreviations that we use below to check symbols and accents in
\tc@check@accent TS1. 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.

```

962 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
963 \def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}

```

We start with the commands that are “safe” and which can be unconditionally
set up, first the accents...

```

964 \DeclareTextAccentDefault{\capitalcedilla}{TS1}
965 \DeclareTextAccentDefault{\capitalogonek}{TS1}
966 \DeclareTextAccentDefault{\capitalgrave}{TS1}
967 \DeclareTextAccentDefault{\capitalacute}{TS1}
968 \DeclareTextAccentDefault{\capitalcircumflex}{TS1}
969 \DeclareTextAccentDefault{\capitaltilde}{TS1}
970 \DeclareTextAccentDefault{\capitaldieresis}{TS1}
971 \DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
972 \DeclareTextAccentDefault{\capitalring}{TS1}
973 \DeclareTextAccentDefault{\capitalcaron}{TS1}
974 \DeclareTextAccentDefault{\capitalbreve}{TS1}
975 \DeclareTextAccentDefault{\capitalmacron}{TS1}
976 \DeclareTextAccentDefault{\capitaldotaccent}{TS1}

```

... and then the other glyphs.

```

977 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
978 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
979 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
980 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
981 \DeclareTextSymbolDefault{\texttwelveudash}{TS1}
982 \DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
983 \DeclareTextSymbolDefault{\textdollar}{TS1}
984 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
985 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
986 \DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
987 \DeclareTextSymbolDefault{\textminus}{TS1}
988 \DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
989 \DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
990 \DeclareTextSymbolDefault{\textasciigrave}{TS1}
991 \DeclareTextSymbolDefault{\texttildelow}{TS1}
992 \DeclareTextSymbolDefault{\textasciibreve}{TS1}

```

```

993 \DeclareTextSymbolDefault{\textasciicaron}{TS1}
994 \DeclareTextSymbolDefault{\textgravedbl}{TS1}
995 \DeclareTextSymbolDefault{\textacutedbl}{TS1}
996 \DeclareTextSymbolDefault{\textdagger}{TS1}
997 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
998 \DeclareTextSymbolDefault{\textbardbl}{TS1}
999 \DeclareTextSymbolDefault{\textperthousand}{TS1}
1000 \DeclareTextSymbolDefault{\textbullet}{TS1}
1001 \DeclareTextSymbolDefault{\textcelsius}{TS1}
1002 \DeclareTextSymbolDefault{\textflorin}{TS1}
1003 \DeclareTextSymbolDefault{\texttrademark}{TS1}
1004 \DeclareTextSymbolDefault{\textcent}{TS1}
1005 \DeclareTextSymbolDefault{\textsterling}{TS1}
1006 \DeclareTextSymbolDefault{\textyen}{TS1}
1007 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
1008 \DeclareTextSymbolDefault{\textsection}{TS1}
1009 \DeclareTextSymbolDefault{\textasciidieresis}{TS1}
1010 \DeclareTextSymbolDefault{\textcopyright}{TS1}
1011 \DeclareTextSymbolDefault{\textordfeminine}{TS1}
1012 \DeclareTextSymbolDefault{\textlnot}{TS1}
1013 \DeclareTextSymbolDefault{\textregistered}{TS1}
1014 \DeclareTextSymbolDefault{\textasciimacron}{TS1}
1015 \DeclareTextSymbolDefault{\textdegree}{TS1}
1016 \DeclareTextSymbolDefault{\textpm}{TS1}
1017 \DeclareTextSymbolDefault{\texttwosuperior}{TS1}
1018 \DeclareTextSymbolDefault{\textthreesuperior}{TS1}
1019 \DeclareTextSymbolDefault{\textasciiacute}{TS1}
1020 \DeclareTextSymbolDefault{\textmu}{TS1}
1021 \DeclareTextSymbolDefault{\textparagraph}{TS1}
1022 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
1023 \DeclareTextSymbolDefault{\textonesuperior}{TS1}
1024 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
1025 \DeclareTextSymbolDefault{\textonequarter}{TS1}
1026 \DeclareTextSymbolDefault{\textonehalf}{TS1}
1027 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
1028 \DeclareTextSymbolDefault{\texttimes}{TS1}
1029 \DeclareTextSymbolDefault{\textdiv}{TS1}

```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```

1030 \DeclareTextCommandDefault{\texteuro}
1031   {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}

```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```

1032 \DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}

```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```

1033 \DeclareTextCommandDefault{\textestimated}{\tc@check@symbol3\textestimated}
1034 \DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}

```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```

1035 \DeclareTextCommandDefault{\capitaltie}{\tc@check@accent2\capitaltie}

```

```

1036 \DeclareTextCommandDefault{\newtie}{\tc@check@accent2\newtie}
1037 \DeclareTextCommandDefault{\capitalnewtie}{\tc@check@accent2\capitalnewtie}
1038 \DeclareTextCommandDefault{\textleftarrow}{\tc@check@symbol2\textleftarrow}
1039 \DeclareTextCommandDefault{\textrightarrow}{\tc@check@symbol2\textrightarrow}
1040 \DeclareTextCommandDefault{\textblank}{\tc@check@symbol2\textblank}
1041 \DeclareTextCommandDefault{\textdblhyphen}{\tc@check@symbol2\textdblhyphen}
1042 \DeclareTextCommandDefault{\textzerooldstyle}{\tc@check@symbol2\textzerooldstyle}
1043 \DeclareTextCommandDefault{\textoneoldstyle}{\tc@check@symbol2\textoneoldstyle}
1044 \DeclareTextCommandDefault{\texttwooldstyle}{\tc@check@symbol2\texttwooldstyle}
1045 \DeclareTextCommandDefault{\textthreeoldstyle}{\tc@check@symbol2\textthreeoldstyle}
1046 \DeclareTextCommandDefault{\textfouroldstyle}{\tc@check@symbol2\textfouroldstyle}
1047 \DeclareTextCommandDefault{\textfiveoldstyle}{\tc@check@symbol2\textfiveoldstyle}
1048 \DeclareTextCommandDefault{\textsixoldstyle}{\tc@check@symbol2\textsixoldstyle}
1049 \DeclareTextCommandDefault{\textsevenoldstyle}{\tc@check@symbol2\textsevenoldstyle}
1050 \DeclareTextCommandDefault{\texteightoldstyle}{\tc@check@symbol2\texteightoldstyle}
1051 \DeclareTextCommandDefault{\textnineoldstyle}{\tc@check@symbol2\textnineoldstyle}
1052 \DeclareTextCommandDefault{\textlangle}{\tc@check@symbol2\textlangle}
1053 \DeclareTextCommandDefault{\textrangle}{\tc@check@symbol2\textrangle}
1054 \DeclareTextCommandDefault{\textmho}{\tc@check@symbol2\textmho}
1055 \DeclareTextCommandDefault{\textbigcircle}{\tc@check@symbol2\textbigcircle}
1056 \DeclareTextCommandDefault{\textuparrow}{\tc@check@symbol2\textuparrow}
1057 \DeclareTextCommandDefault{\textdownarrow}{\tc@check@symbol2\textdownarrow}
1058 \DeclareTextCommandDefault{\textborn}{\tc@check@symbol2\textborn}
1059 \DeclareTextCommandDefault{\textdivorced}{\tc@check@symbol2\textdivorced}
1060 \DeclareTextCommandDefault{\textdied}{\tc@check@symbol2\textdied}
1061 \DeclareTextCommandDefault{\textleaf}{\tc@check@symbol2\textleaf}
1062 \DeclareTextCommandDefault{\textmarried}{\tc@check@symbol2\textmarried}
1063 \DeclareTextCommandDefault{\textmusicalnote}{\tc@check@symbol2\textmusicalnote}
1064 \DeclareTextCommandDefault{\textdblhyphenchar}{\tc@check@symbol2\textdblhyphenchar}
1065 \DeclareTextCommandDefault{\textdollaroldstyle}{\tc@check@symbol2\textdollaroldstyle}
1066 \DeclareTextCommandDefault{\textcentoldstyle}{\tc@check@symbol2\textcentoldstyle}
1067 \DeclareTextCommandDefault{\textcolonmonetary}{\tc@check@symbol2\textcolonmonetary}
1068 \DeclareTextCommandDefault{\textwon}{\tc@check@symbol2\textwon}
1069 \DeclareTextCommandDefault{\textnaira}{\tc@check@symbol2\textnaira}
1070 \DeclareTextCommandDefault{\textguarani}{\tc@check@symbol2\textguarani}
1071 \DeclareTextCommandDefault{\textpeso}{\tc@check@symbol2\textpeso}
1072 \DeclareTextCommandDefault{\textlira}{\tc@check@symbol2\textlira}
1073 \DeclareTextCommandDefault{\textrecipe}{\tc@check@symbol2\textrecipe}
1074 \DeclareTextCommandDefault{\textinterrobang}{\tc@check@symbol2\textinterrobang}
1075 \DeclareTextCommandDefault{\textinterrobangdown}{\tc@check@symbol2\textinterrobangdown}
1076 \DeclareTextCommandDefault{\textdong}{\tc@check@symbol2\textdong}
1077 \DeclareTextCommandDefault{\textpertenthousand}{\tc@check@symbol2\textpertenthousand}
1078 \DeclareTextCommandDefault{\textpilcrow}{\tc@check@symbol2\textpilcrow}
1079 \DeclareTextCommandDefault{\textbaht}{\tc@check@symbol2\textbaht}
1080 \DeclareTextCommandDefault{\textnumero}{\tc@check@symbol2\textnumero}
1081 \DeclareTextCommandDefault{\textdiscount}{\tc@check@symbol2\textdiscount}
1082 \DeclareTextCommandDefault{\textopenbullet}{\tc@check@symbol2\textopenbullet}
1083 \DeclareTextCommandDefault{\textservicemark}{\tc@check@symbol2\textservicemark}
1084 \DeclareTextCommandDefault{\textlquill}{\tc@check@symbol2\textlquill}
1085 \DeclareTextCommandDefault{\textrquill}{\tc@check@symbol2\textrquill}
1086 \DeclareTextCommandDefault{\textcopyright}{\tc@check@symbol2\textcopyright}
1087 \DeclareTextCommandDefault{\textcircledP}{\tc@check@symbol2\textcircledP}
1088 \DeclareTextCommandDefault{\textreferencemark}{\tc@check@symbol2\textreferencemark}
1089 \DeclareTextCommandDefault{\textsurd}{\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 the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```
1090 \DeclareTextCommandDefault{\textcircled}
1091 {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}
1092 \DeclareTextCommandDefault{\t}
1093 {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}1\t}
```

Finally input the encoding-specific definitions for TS1 thus making the top-level definitions optimised for this encoding (and not for the default encoding, see section 19.2).

```
1094 \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 (see section 19.1 above). So we better get rid of them:

```
1095 \UndeclareTextCommand{\textsterling}{OT1}
1096 \UndeclareTextCommand{\textdollar} {OT1}
```

Similar declarations should probably be made for other encodings like OT4 if they are in use.

```
1097 %\UndeclareTextCommand{\textsterling}{OT4}
1098 %\UndeclareTextCommand{\textdollar} {OT4}
```

From the T1 encoding there are two candidates for removal: `%0` and `%00` 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 `%0` 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 `%0` and `%00` are not taken from the same physical font) and with PostScript fonts `%0` will come out correctly while `%00` 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`.

```
1099 \UndeclareTextCommand{\textperthousand}{T1}
1100 %\UndeclareTextCommand{\textpertenthousand}{T1}
```

20.2.1 Supporting oldstyle digits

```
1101 \DeclareRobustCommand\oldstylenums[1]{%
1102 \begingroup
1103 \ifmmode
1104 \mathgroup\symletters #1%
1105 \else
1106 \CheckEncodingSubset\@use@text@encoding{TS1}%
1107 {\PackageWarning{textcomp}%
1108 {Oldstyle digits unavailable for
1109 family \f@family.\MessageBreak
1110 Lining digits used instead}}%
1111 \tw@{#1}%
```

```

1112 \fi
1113 \endgroup
1114 }

```

20.2.2 Subset encoding defaults

For many font families commonly used in the T_EX 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:

```
1115 \iftc@forced \else
```

Computer modern based fonts (e.g., CM, CM-Bright, Concrete):

```

1116 \DeclareEncodingSubset{TS1}{cmr}      {0}
1117 \DeclareEncodingSubset{TS1}{cmss}     {0}
1118 \DeclareEncodingSubset{TS1}{cmtt}     {0}
1119 \DeclareEncodingSubset{TS1}{cmvtt}    {0}
1120 \DeclareEncodingSubset{TS1}{cmbr}     {0}
1121 \DeclareEncodingSubset{TS1}{cmtl}     {0}
1122 \DeclareEncodingSubset{TS1}{ccr}      {0}

```

PSNFSS fonts:

```

1123 \DeclareEncodingSubset{TS1}{ptm}      {4}
1124 \DeclareEncodingSubset{TS1}{pcr}      {4}
1125 \DeclareEncodingSubset{TS1}{phv}      {4}
1126 \DeclareEncodingSubset{TS1}{ppl}      {3}
1127 \DeclareEncodingSubset{TS1}{pag}      {4}
1128 \DeclareEncodingSubset{TS1}{pbk}      {4}
1129 \DeclareEncodingSubset{TS1}{pnc}      {4}
1130 \DeclareEncodingSubset{TS1}{pzc}      {4}
1131 \DeclareEncodingSubset{TS1}{bch}      {4}
1132 \DeclareEncodingSubset{TS1}{put}      {5}

```

Other CTAN fonts (probably not complete):

```

1133 \DeclareEncodingSubset{TS1}{uag}      {5}
1134 \DeclareEncodingSubset{TS1}{ugq}      {5}
1135 \DeclareEncodingSubset{TS1}{ul8}      {4}
1136 \DeclareEncodingSubset{TS1}{ul9}      {4} % (LuxiSans, one day)
1137 \DeclareEncodingSubset{TS1}{augie}    {5}
1138 \DeclareEncodingSubset{TS1}{dayrom}   {3}
1139 \DeclareEncodingSubset{TS1}{dayroms}  {3}
1140 \DeclareEncodingSubset{TS1}{pxr}      {0}
1141 \DeclareEncodingSubset{TS1}{pxss}     {0}
1142 \DeclareEncodingSubset{TS1}{pxtt}     {0}
1143 \DeclareEncodingSubset{TS1}{txr}      {0}
1144 \DeclareEncodingSubset{TS1}{txss}     {0}
1145 \DeclareEncodingSubset{TS1}{txtt}     {0}

```

Fourier-GUTenberg:

```

1146 \DeclareEncodingSubset{TS1}{futs}     {4}
1147 \DeclareEncodingSubset{TS1}{futx}     {4}
1148 \DeclareEncodingSubset{TS1}{futj}     {4}

```

Y&Y’s Lucida Bright

```
1149 \DeclareEncodingSubset{TS1}{hlh}      {3}
```



```

1150 \DeclareEncodingSubset{TS1}{hls}    {3}
1151 \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`.

```

1152 \DeclareEncodingSubset{TS1}{hlct}    {5}
1153 \DeclareEncodingSubset{TS1}{hlx}     {5}
1154 \DeclareEncodingSubset{TS1}{hlce}     {5}
1155 \DeclareEncodingSubset{TS1}{hlcn}     {5}
1156 \DeclareEncodingSubset{TS1}{hlcw}     {5}
1157 \DeclareEncodingSubset{TS1}{hlcf}     {5}

```

Other commercial families...

```

1158 \DeclareEncodingSubset{TS1}{pplx}     {3}
1159 \DeclareEncodingSubset{TS1}{pplj}     {3}
1160 \DeclareEncodingSubset{TS1}{ptmx}     {4}
1161 \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.

```

1162 \InputIfFileExists{textcomp.cfg}
1163 {\PackageInfo{textcomp}{Local configuration file used}}{}

1164 \fi
1165 </TS1sty>

```

File m

ltcounts.dtx

21 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 T _E X number (as opposed to <code>\the<counter></code> which expands to the <i>printed</i> representation of <code><counter></code>)
<code>\arabic</code>	<code>\arabic{<counter>}</code> : 1, 2, 3, ...
<code>\roman</code>	<code>\roman{<counter>}</code> : i, ii, iii, ...
<code>\Roman</code>	<code>\Roman{<counter>}</code> : I, II, III, ...
<code>\alph</code>	<code>\alph{<counter>}</code> : a, b, c, ...
<code>\Alph</code>	<code>\Alph{<counter>}</code> : A, B, C, ...
<code>\fnsymbol</code>	<code>\fnsymbol{<counter>}</code> : *, †, ‡, ...

1 (*2ekernel)

21.1 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> <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.

`\setcounter` `\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}}
```

`\addtocounter` `\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}}
```

`\newcounter` `\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}}{}}}
```

`\value` `\value{<ctr>}` produces the value of counter `<ctr>`, for use with a `\setcounter` or `\addtocounter` command.

```

14 \def\value#1{\csname c@#1\endcsname}
```

`\@newctr`

```

15 \def\@newctr#1[#2]{%
16   \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}
```

`\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}
```

`\@stpelt`

```

23 \def\@stpelt#1{\global\csname c@#1\endcsname \z@}
```

```

\cl@ckpt
24 \def\cl@ckpt{\elt{page}}

\@definecounter
25 \def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
26     \setcounter{#1}\z@
27     \global\expandafter\let\csname cl@#1\endcsname\@empty
28     \@addtoreset{#1}{ckpt}%
29     \global\expandafter\let\csname p@#1\endcsname\@empty
30     \expandafter
31     \gdef\csname the#1\expandafter\endcsname\expandafter
32         {\expandafter\@arabic\csname c@#1\endcsname}}

\@addtoreset
33 \def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}

    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.
34 \def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}

\roman Representation of counter as lower-case Roman numerals.
35 \def\roman#1{\expandafter\@roman\csname c@#1\endcsname}

\Roman Representation of counter as upper-case Roman numerals.
36 \def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}

\alph Representation of counter as a lower-case letter: 1 = a, 2 = b, etc.
37 \def\alph#1{\expandafter\@alph\csname c@#1\endcsname}

\Alph Representation of counter as an upper-case letter: 1 = A, 2 = B, etc.
38 \def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}

\fnsymbol Representation of COUNTER as a footnote symbol: 1 = *, 2 = †, etc.
39 \def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}

\@arabic \@arabic\F00counter Representation of \F00counter as arabic numerals.
40 \def\@arabic#1{\number #1} %% changed 29 Apr 86

\@roman \@roman\F00counter Representation of \F00counter as lower-case Roman nu-
merals.
41 \def\@roman#1{\romannumeral #1}

\@Roman \@Roman\F00counter Representation of \F00counter as upper-case Roman nu-
merals.
42 \def\@Roman#1{\expandafter\@slowromancap\romannumeral #1@}

```

`\@slowromancap` Fully expandable macro to change a roman number to uppercase.

```

43 \def\@slowromancap#1{\ifx @#1% then terminate
44     \else
45         \if i#1I\else\if v#1V\else\if x#1X\else\if l#1L\else\if
46             c#1C\else\if d#1D\else \if m#1M\else#1\fi\fi\fi\fi\fi\fi\fi
47         \expandafter\@slowromancap
48     \fi
49 }

```

`\@alph` `\@alph\F00counter` Representation of `\F00counter` as a lower-case letter: 1 = a, 2 = b, etc.

```

50 \def\@alph#1{%
51     \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
52         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
53         y\or z\else\@ctrerr\fi}

```

`\@Alph` `\@Alph\F00counter` Representation of `\F00counter` as an upper-case letter: 1 = A, 2 = B, etc.

```

54 \def\@Alph#1{%
55     \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
56         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
57         Y\or Z\else\@ctrerr\fi}

```

`\@fnsymbol` Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

```

58 \def\@fnsymbol#1{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or
59     \mathsection\or \mathparagraph\or \|\or **\or \dagger\dagger
60     \or \ddagger\ddagger \else\@ctrerr\fi}}
61 \</2ekernel>

```

File n

ltlength.dtx

22 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}}

\setlength
4 \def\setlength#1#2{#1#2\relax}

\addtolength \relax added 24 Mar 86
5 \def\addtolength#1#2{\advance#1 #2\relax}

\settoheight The obvious analogs of \settowidth.
\settodepth 6 \def\@settodim#1#2#3{\setbox\@tempboxa\hbox{{#3}}#2#1\@tempboxa
\settowidth Clear the memory afterwards (which might be a lot).
\@settodim 7 \setbox\@tempboxa\box\voidb@x}
8 \def\settoheight{\@settodim\ht}
9 \def\settodepth {\@settodim\dp}
10 \def\settowidth {\@settodim\wd}

\@settopoint This macro takes the contents of the skip register that is supplied as its argument
and removes the fractional part to make it a whole number of points. This can be
used in class files to avoid values like 345.4666666pt when calculating a dimension.
11 \def\@settopoint#1{\divide#1\p@\multiply#1\p@}
12 </2ekernel>
```

File o

ltfssbas.dtx

This file contains the main implementation of the ‘low level’ font selection commands. See other parts of the L^AT_EX distribution, or *The L^AT_EX Companion* for higher level documentation of the L^AT_EX ‘New’ Font Selection Scheme.

Warning: The macro documentation is still basically the documentation from the first NFSS release and therefore in some cases probably not completely accurate.

23 Autoloading parts of NFSS

This code is set up in a way that some parts of it can be kept separate and will only be loaded if needed.

If we are producing an autload version of L^AT_EX 2_ε then all those parts with `def1` or `def2` docstrip guards will be placed into the autloadable files `autofss1.sty` and `autofss2.sty`.

The ‘2ekernel’ code ensures that a `\usepackage{autofss1}` is essentially ignored if a ‘full’ format is being used that has picture mode already in the format.

Note the `autofss2` loading is currently disabled.

```
1 <2ekernel>\expandafter\let\csname ver@autofss1.sty\endcsname\fmtversion
```

The autload file `autofss2` is a specialty because it contains code which will be completely local, ie loaded every time again.

24 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.

```
2 <*2ekernel | autload>
3 \def\@nomath#1{\relax\ifmmode
4   \font@warning{Command \noexpand#1invalid in math mode}\fi}
5 </2ekernel | autload>
```

`\no@alphabet@error` The macro `\no@alphabet@error` is called whenever the user requests a math *alphabet* that is not available in the current *version*. In math mode an error message is produced otherwise the command keeps silent. The argument is the name of the control sequence that identifies the math *alphabet*. The `\relax` at the beginning is necessary to prevent T_EX from scanning too far in certain situations.

```
6 <*2ekernel | def1>
7 \gdef\no@alphabet@error#1{\relax \ifmmode
8   \@latex@error{Math\space alphabet\space identifier\space
9     \noexpand#1is\space undefined\space in\space math\space
10    version\space ‘\math@version’}%
```

```

11      {Your\space requested\space math\space alphabet\space
12      is\space undefined\space in\space the\space current\space
13      math\space version.^^JCheck\space the\space spelling\space
14      or\space use\space the\space \noexpand\SetMathAlphabet\space
15      command.}
16      \fi}
17 </2ekernel | def1>
18 <*autoload>
19 \gdef\no@alphabet@error{\relax \ifmmode
20   \expandafter\try@sizes\expandafter\no@alphabet@error \fi}
21 </autoload>

\new@mathgroup We also give a new name to \newfam and \fam to avoid verbal confusion (see the
\mathgroup      introduction).2
22 <*2ekernel | autoload>
23 \def\new@mathgroup{\alloc@8\mathgroup\chardef\sixt@@n}
24 \let\mathgroup\fam
25 \let\newfam\new@mathgroup
26 \onlypreamble\new@mathgroup

```

25 Macros for setting up the tables

\DeclareFontShape The macro \DeclareFontShape takes 6 arguments:

```
27 \def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
28   \nfss@catcodes
```

We use \expandafter \endgroup to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

\DeclareFontShape

```

29   \expandafter\endgroup
30   \DeclareFontShape@}
31 \def\DeclareFontShape@#1#2#3#4#5#6{%
32   \expandafter\ifx\csname #1+#2\endcsname\relax
33   \latex@error{Font family ‘#1+#2’ unknown}\@eha
34   \else
35   \expandafter
36   \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
37                                   \csname #5\endcsname}%
38   \def\reserved@a{#6}%
39   \global
40   \expandafter\let\csname#5\endcsname\expandafter\endcsname
41   \ifx\reserved@a\@empty
42   \@empty
43   \else
44   \reserved@a
45   \fi
46 \fi
47 }

```

²For 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.

`\DeclareFixedFont` Define a direct font switch that avoids all overhead.

```

48 \def\DeclareFixedFont#1#2#3#4#5#6{%
49   \begingroup
50     \math@fontsfalse
51     \every@math@size{%
52       \fontsize{#6}\z@
53       \usefont{#2}{#3}{#4}{#5}%
54       \global\expandafter\let\expandafter#1\the\font
55     \endgroup
56   }
57 \</2kernel | autoload>

```

`\do@subst@correction`

```

58 \<*2kernel | autoload>
59 \def\do@subst@correction{%
60   \xdef\subst@correction{%
61     \font@name
62     \global\expandafter\font
63     \csname \curr@fontshape/\f@size\endcsname
64     \noexpand\fontname\font
65     \relax}%

```

Calling `\subst@correction` after the current group means calling it after we have loaded the substitution font which is done inside a group.

```

66   \aftergroup\subst@correction
67 }

```

`\DeclareFontFamily`

```

68 \def\DeclareFontFamily#1#2#3{%

```

If we want fast checking for the encoding scheme we can just check for `\T@. .` being defined.

```

69 % \@tempswafalse
70 % \def\reserved@b{#1}%
71 % \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
72 %   \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
73 % \cdp@list
74 % \if@tempswa
75 % \ifundefined{T@#1}%
76 %   {%
77 %     \latex@error{Encoding scheme ‘#1’ unknown}\@eha
78 %   }%
79 %   {%

```

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`.

```

80   \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`.

```

81     \global
82     \expandafter\let\csname #1+#2\expandafter\endcsname
83         \ifx \reserved@a\@empty
84             \@empty
85         \else \reserved@a
86         \fi
87     }%
88 }

\cdp@list We initialize the code page list to be empty.
89 \let\cdp@list\@empty
90 \@onlypreamble\cdp@list

\cdp@elt
91 \let\cdp@elt\relax
92 \@onlypreamble\cdp@elt

\DeclareFontEncoding
93 \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.
94     \begingroup
95     \nfss@catcodes
96     \expandafter\endgroup
97     \DeclareFontEncoding@}
98 \@onlypreamble\DeclareFontEncoding

99 \def\DeclareFontEncoding@#1#2#3{%
100     \expandafter
101     \ifx\csname T@#1\endcsname\relax
102         \def\cdp@elt{\noexpand\cdp@elt}%
103         \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
104             {\default@family}{\default@series}%
105             {\default@shape}}%

To support encoding dependent commands (like accents) we initialise the com-
mand \<encoding>-cmd to be \@changed@cmd. (See ltoutenc.dtx for details.)

106     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
107     \else
108         \font@info{Redeclaring font encoding #1}%
109     \fi

110     \global\@namedef{T@#1}{#2}%
111     \global\@namedef{M@#1}{\default@M#3}%

Keep a record of the last encoding being declared:
112     \xdef\LastDeclaredEncoding{#1}%
113 }
114 \@onlypreamble\DeclareFontEncoding@

\LastDeclaredEncoding The last encoding being declared by \DeclareFontEncoding.
115 \def\LastDeclaredEncoding{

```

`\DeclareFontSubstitution`

```
116 \def\DeclareFontSubstitution#1#2#3#4{%
117   \expandafter
118   \ifx\csname T@#1\endcsname\relax
119     \@latex@error{Encoding scheme ‘#1’ unknown}\@eha
120   \else
121     \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.
122       \edef\reserved@a{#1}%
123       \toks@{}%
124       \def\cdp@elt##1##2##3##4{%
125         \def\reserved@b{##1}%
126         \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.
127           \addto@hook\toks@{\cdp@elt{##1}{#2}{#3}{#4}}%
128         \else
      If \reserved@a and \reserved@b differ then we simply copy from the old list to
      the new.
129           \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
130         \fi}%
131       \cdp@list
132       \xdef\cdp@list{\the\toks@}%
133     \endgroup
134     \global
135     \@namedef{D@#1}{%
136       \def\default@family{#2}%
137       \def\default@series{#3}%
138       \def\default@shape{#4}%
139     }%
140   \fi
141 }
142 \onlypreamble\DeclareFontSubstitution
```

`\DeclareFontEncodingDefaults`

```
143 \def\DeclareFontEncodingDefaults#1#2{%
144   \ifx\relax#1\else
145     \ifx\default@T\@empty\else
146       \@font@info{Overwriting encoding scheme text defaults}%
147       \fi
148       \gdef\default@T{#1}%
149     \fi
150   \ifx\relax#2\else
151     \ifx\default@M\@empty\else
152       \@font@info{Overwriting encoding scheme math defaults}%
153       \fi
154       \gdef\default@M{#2}%
155     \fi
```

```

156 }
157 \@onlypreamble\DeclareFontEncodingDefaults

\default@T
\default@M 158 \let\default@T\@empty
159 \let\default@M\@empty

\DeclarePreloadSizes

160 \def\DeclarePreloadSizes#1#2#3#4#5{%
161   \ifundefined{T@#1}%
162     {\@latex@error{Encoding scheme ‘#1’ unknown}\@eha}%
163     {%
Don't know at the moment what this group here does!
164   \begingroup
We define a macro \reserved@f3 that grabs the next size and loads the corre-
sponding font. This is done by delimiting \reserved@f's only argument by the
token , (comma).
165   \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.)
166     \if>##1>%
167       \let\reserved@f\relax
168       \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.
169       \xdef\font@name{\csname#1/#2/#3/#4/#1\endcsname}%
170       \pickup@font
Now we forget the name of the font just loaded. More precisely, we set the cor-
responding 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.
171       \global\expandafter\let\font@name\relax
172       \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.
173       \reserved@f}%
We finish with reinserting the list of sizes after the \reserved@f macro and ap-
pending an empty element so that the end of the list is recognized properly.
174       \reserved@f#5,,%
175   \endgroup
176   }%
177 }
178 \@onlypreamble\DeclarePreloadSizes

```

³We cannot use \@tempa since it is needed in \pickup@font.

```

\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.
179 \newif\ifmath@fonts \math@fontstrue

\DeclareMathSizes \DeclareMathSizes takes the text size, math text size, math script size, and math
\DeclareMathSizes* scriptscript size as arguments and defines the right \S@... macro.

180 \def\DeclareMathSizes{%
181   \ifstar{\@DeclareMathSizes\math@fontsfalse}%
182   {\@DeclareMathSizes{}}}%
183 \@onlypreamble\DeclareMathSizes

\@DeclareMathSizes
184 \def\@DeclareMathSizes#1#2#3#4#5{%
185   \@defaultunits\dimen@#2pt\relax\@nnil
186   \if$#3$%
187     \expandafter \let
188     \csname S@\strip@pt\dimen@\endcsname
189     \math@fontsfalse
190   \else
191     \expandafter \gdef
192     \csname S@\strip@pt\dimen@\endcsname
193     {\gdef\tf@size{#3}\gdef\sfontsize{#4}%
194       \gdef\ssf@size{#5}%
195       #1%
196     }%
197   \fi}
198 \@onlypreamble\@DeclareMathSizes

```

26 Selecting a new font

26.1 Macros for the user

`\fontencoding` As we said in the introduction a font is described by four parameters. We first
`\f@encoding` 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.

```

199 \DeclareRobustCommand\fontencoding[1]{%
200   \expandafter\ifx\csname T@#1\endcsname\relax
201     \@latex@error{Encoding scheme ‘#1’ unknown}\@eha
202   \else
203     \edef\f@encoding{#1}%
204     \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` inbetween we can save processing by making sure that `\enc@update` is `\relax`.

```

205     \let\enc@update\relax
206   \else

```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```
207      \let\enc@update\@enc@update
208      \fi
209      \fi
210 }
```

`\@enc@update`

```
211 \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`.

```
212      \expandafter
213      \let
214      \csname\cf@encoding -cmd\endcsname
215      \@changed@cmd
```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```
216      \expandafter
217      \let
218      \csname\f@encoding-cmd\endcsname
219      \@current@cmd
```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
220      \default@T
221      \csname T@\f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
222      \csname D@\f@encoding\endcsname
223      \let\enc@update\relax
224      \let\cf@encoding\f@encoding
225 }
```

`\enc@update` The default action in `\selectfont` is to do nothing.

```
226 \let\enc@update\relax
```

`\fontfamily`

```
\f@family 227 \DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
```

```
\fontseries 228 \DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
```

```
\f@series 229 \DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
```

`\fontshape` 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` comand first.

`\f@shape`

```
230 \def\usefont#1#2#3#4{\fontencoding{#1}\fontfamily{#2}%
231      \fontseries{#3}\fontshape{#4}\selectfont
232      \ignorespaces}
```

`\linespread` The command `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```

233 \DeclareRobustCommand\linespread[1]
234   {\set@fontsize{#1}\f@size\f@baselineskip}

```

`\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.

```

235 \DeclareRobustCommand\fontsize[2]
236   {\set@fontsize\baselinestretch{#1}{#2}}

```

`\f@linespread` This macro holds the current internal value for `\baselinestretch`.

```

237 \let\f@family\@empty
238 \let\f@series\@empty
239 \let\f@shape\@empty
240 \let\f@size\@empty
241 \let\f@baselineskip\@empty
242 \let\f@linespread\@empty

```

`\cf@encoding`

```

243 \let\f@encoding\@empty
244 \let\cf@encoding\@empty

```

`\@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.

```

245 \def\@defaultunits{\afterassignment\remove@to@nnil}

```

`\strip@pt` This macro strips the characters `pt` produced by using `\the` on a `dimen` register.

```

\rem@pt 246 \begingroup
247   \catcode‘P=12
248   \catcode‘T=12
249   \lowercase{
250     \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}
251   \expandafter\endgroup\x
252 \def\strip@pt{\expandafter\rem@pt\the}

```

`\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.

```

253 \DeclareRobustCommand\mathversion[1]

```

```

254      {\@nomath\mathversion
255       \expandafter\ifx\csname mv@#1\endcsname\relax
256       \@latex@error{Math version ‘#1’ is not defined}\@eha\else
257       \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@currsiz` to an invalid value since this will trigger the setup when the formula starts.

```

258      \gdef\glb@currsiz{}%

```

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 to do the setting only if the version really has changed after all. This might be interesting in case of `amstext` and `boldsymbol`.

```

259      \aftergroup\glb@settings
260      \fi}

```

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 \LaTeX 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 $\LaTeX 2_\epsilon$ 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 261 \let\@frozen@everymath\everymath
262 \let\@frozen@everydisplay\everydisplay

\everymath Now we provide now user hooks that will be called in the frozen internals.
\everydisplay 263 \newtoks\everymath
264 \newtoks\everydisplay

\@frozen@everymath Now we define the behaviour of the frozen hooks: first check the math setup then
call the user hook.
265 \@frozen@everymath = {\check@mathfonts
266                      \the\everymath}

\@frozen@everydisplay Ditto for the display hook.
267 \@frozen@everydisplay = {\check@mathfonts
268                          \the\everydisplay}

\curr@math@size This holds locally the current math size.
269 \let\curr@math@size\@empty

```


26.2 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.

```
270 \def\pickup@font{%
271     \expandafter \ifx \font@name \relax
272         \define@newfont
273     \fi}
```

`\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`

```
274 {\catcode'\=12
```

and define `\split@name` so that it will define our private `\f@encoding`, `\f@family`, `\f@series`, `\f@shape`, and `\f@size` macros.

```
275 \gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
276                                     \def\f@family{#2}%
277                                     \def\f@series{#3}%
278                                     \def\f@shape{#4}%
279                                     \def\f@size{#5}}}
```

`\curr@fontshape` Abbreviation which may get removed again for speed.

```
280 \def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
281 \</2ekernel | autoloading)
```

`\define@newfont` Now we can tackle the problem of defining a new font.

```
282 \< *2ekernel | def2 | autoloading)
283 \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.

```
284 \begingroup
285     \let\typeout\@font@info
286     \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`.

```
287     \expandafter\expandafter\expandafter
288         \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.

```

289 % \expandafter\ifx
290 % \csname\curr@fontshape\endcsname \relax
291 % \try@load@fontshape % try always
292 % \fi
293 \expandafter\ifx
294 \csname\curr@fontshape\endcsname \relax
295 \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).

```

296 % \csname\curr@fontshape\endcsname
297 \extract@font\fi

```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```

298 \endgroup}
299 </2ekernel | def2 | autoload>

```

As `autofss2.sty` only makes local definitions it is re-loaded for each font, to save some string memory in the kernel, and to speed up the loading of some packages which may load fonts. The code is actually pre-loaded into the kernel and removed at `\begin{document}`. The `\ifx` test below ensures that if `\usepackage{autofss2}` appears in the preamble, then the code is not removed at this time. Can not use `\AtBeginDocument` here as it is not defined yet! Listing all the commands like this is not ideal as any changes to the `autofss2.sty` need to be reected here, but this seems the most memory efficient mechanism as it avoids the use of an extra `csname` to store the list.

This is currently disabled, so the ‘autofss2’ code remains in the kernel, and `autofss2.sty` is not generated in the current public release.

```

300 <*autoloadxxx>
301 \expandafter\def\expandafter\@begindocumenthook\expandafter{%
302 \expandafter\ifx\csname ver@autofss2.sty\endcsname\relax
303 \gdef\define@newfont{%
304 \begingroup
305 \makeatletter\nfss@catcodes
306 \catcode'\#6\relax
307 \@@input autofss2.sty\relax\define@newfont
308 \endgroup}%
309 \begingroup
310 \def\do##1{\global\let##1\@undefined}%
311 \do\extract@sizefn
312 \do\try@simple@size
313 \do\set@simple@size@args
314 \do\extract@range@fontinfo
315 \do\is@range
316 \do\check@range
317 \do\check@single
318 \do\set@size@func@args
319 \do\set@size@func@args@
320 \do\try@size@range
321 \do\empty@sfcnt

```

```

322 \do\gen@sfcnt
323 \do\genb@sfcnt
324 \do\sub@sfcnt
325 \do\subf@sfcnt
326 \do\fixed@sfcnt
327 \endgroup
328 \fi}
329 </autoloadxxx>
330 <*2ekernel | autoload>
331 \def\try@load@fontshape{%
332 \expandafter
333 \ifx\csname \f@encoding+\f@family\endcsname\relax
334 \@font@info{Try loading font information for
335 \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.

```

336 \global\expandafter\let
337 \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).

```

338 \nfss@catcodes
339 \let\nfss@catcodes\relax

```

For increased portability make the external filename monospace, but look for the (old style) mixed case filename if the first attempt fails.

On any monospace 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.

```

340 \edef\reserved@a{%
341 \lowercase{%
342 \noexpand\InputIfFileExists{\f@encoding\f@family.fd}}}%
343 \reserved@a\relax
344 {\@input@{\f@encoding\f@family.fd}}%
345 \fi}

```

`\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.

```

346 \def\nfss@catcodes{%

```

We start by making `@` a letter and ignoring all blanks and newlines.

```

347 \makeatletter
348 \catcode'\ 9%
349 \catcode'\^I9%
350 \catcode'\^M9%

```

Then we set up `\`, `{`, `}`, `#` and `%` in case an `.fd` file is loaded during a verbatim environment.

```
351 \catcode'\z@
352 \catcode'\{ \one
353 \catcode'\} \tw@
354 \catcode'\#6%
355 \catcode'\^7%
356 \catcode'\%14%
```

The we make sure that the important syntax parts have the right `\catcode`.

```
357 \@makeother\<%
358 \@makeother\>%
359 \@makeother\*%
360 \@makeother\.%
361 \@makeother\-%
362 \@makeother\/%
363 \@makeother\[%
364 \@makeother\]%
365 \@makeother\'%
366 \@makeother\'%
367 \@makeother\"%
368 }
```

`\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!

```
369 \def\DeclareErrorFont#1#2#3#4#5{%
370 \xdef\error@fontshape{%
371 \noexpand\expandafter\noexpand\split@name\noexpand\string
372 \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
373 \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 grap "error default values" as well. However, this probably should go also.

```
374 % \gdef\f@encoding{#1}%
375 \gdef\default@family{#2}%
376 \gdef\default@series{#3}%
377 \gdef\default@shape{#4}%
378 \global\let\f@family\default@family
379 \global\let\f@series\default@series
380 \global\let\f@shape\default@shape
381 \gdef\f@size{#5}%
382 \gdef\f@baselineskip{#5pt}%
383 }
384 \@onlypreamble\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!

```

385 \def\wrong@fontshape{%
386   \csname D@f@encoding\endcsname      % install defaults if in math
   We remember the wanted \curr@fontshape combination which we will need in a
   moment.
387   \edef\reserved@a{\csname\curr@fontshape\endcsname}%
388   \ifx\last@fontshape\reserved@a
389     \errmessage{Corrupted NFSS tables}%
390     \error@fontshape
391   \else

```

Then we warn the user about the mess and set the shape to its default.

```

392   \let\f@shape\default@shape
   If the combination is not known, try the default series.
393   \expandafter\ifx\csname\curr@fontshape\endcsname\relax
394     \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!

```

395   \expandafter
396     \ifx\csname\curr@fontshape\endcsname\relax
397       \let\f@family\default@family
398   \fi \fi
399 \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 T_EX 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.

```

400   \@font@warning{Font shape ‘\expandafter\string\reserved@a’
401     \expandafter\@gobble\string\@undefined\MessageBreak
402     using ‘\curr@fontshape’ instead\@wrong@font@char}%
403   \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.

```

404   \gdef\@defaultsubs{%
405     \@font@warning{Some font shapes were not available, defaults
406       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.

```

407   \global\expandafter\expandafter\expandafter\let
408     \expandafter\reserved@a
409     \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.

```
410 \xdef\font@name{%
411 \csname\curr@fontshape/\f@size\endcsname}%
```

The last thing this macro does is to call `\pickup@font` again to load the font if it is not defined yet. At this point this code will loop endlessly if the defaults are not well defined.

```
412 \pickup@font}
```

`\@wrong@font@char` Normally empty but redefined in `\UseTextSymbol` so that the Font shape undefined message can refer to the symbol causing the problem.

```
413 \let\@wrong@font@char\@empty
```

`\@@defaultsubs` See above.

```
\@defaultsubs 414 \let\@defaultsubs\relax
```

`\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 *T_EXbook*, p. 382).

```
415 \def\strip@prefix#1>{}
```

27 Assigning math fonts to *versions*

`\install@mathalphabet` This is just another name for `\gdef` but we can redefine it if necessary later on.

```
416 \let\install@mathalphabet\gdef
```

`\math@fonts`

```
417 \let\math@fonts\@empty
```

`\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.

```
418 %\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.

```
419 % \begingroup
```

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

```
420 % \escapechar\m@ne
```

```
421 % \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts...

```
422 % \globaldefs\@ne \math@fonts
```

... and close the group to restore `\globaldefs` and `\escapechar`.

```
423 % \endgroup
```

As long as no *size* or *version* change occurs the $\langle\textit{math alphabet identifier}\rangle$ should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new $\langle\textit{math alphabet identifier}\rangle$) to expand into a `\mathgroup` switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro.

The original code for the end of `\select@group` was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the $\langle\textit{math alphabet identifier}\rangle$ and then calling the new definition to switch to the wanted $\langle\textit{math group}\rangle$. Now we define the $\langle\textit{math alphabet identifier}\rangle$ as a call to the `\use@mathgroup` command.

```
424 % \xdef#1{\noexpand\use@mathgroup\noexpand#2%
425 %      {\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.

```
426 % \expandafter\extract@alph@from@version
427 % \csname mv@\math@version\expandafter\endcsname
428 % \expandafter{\number\csname c@mv@\math@version\endcsname}%
429 % #1%
430 % \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.

```
431 %\expandafter #1\fi}
```

`\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.

```
432 \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`.

```
433 \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`.

```
434 \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`.

```
435 \def\reserved@c####1{\gdef#1{##1####1##3}}%
```

Then we execute our auxiliary macro.

```
436 \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.

```
437 \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling `\reserved@c`:

```
438 \reserved@c{%
439 \getanddefine@fonts{#2}##2%
440 \install@mathalphabet#3{%
441 \relax\ifmmode \else \non@alpherr#3\fi
442 \use@mathgroup##1{#2}}}%
```

In addition it defines the alphabet the way it should be used from now on.

```
443 \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
444 \use@mathgroup##1{#2}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
445 \expandafter\reserved@a\reserved@b\@nil
446 }
```

`\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.

```
447 \let\math@bgroup\bgroup
448 \def\math@egroup#1{#1\egroup}
449 </2ekernel | autoload>
```

`\calculate@math@sizes` Here is the default definition for `\calculate@math@sizes` a more elaborate interface is under testing in `mthscale.sty`.

```
450 < *2ekernel | def1>
```



```

451 \gdef\calculate@math@sizes{%
452   \@font@info{Calculating\space math\space sizes\space for\space
453     size\space <\f@size>}%
454   \dimen@\f@size \p@
455   \@tempdimb \defaultscriptratio \dimen@
456   \dimen@ \defaultscriptscriptratio \dimen@
457   \expandafter\xdef\csname S@\f@size\endcsname{%
458     \gdef\noexpand\tf@size{\f@size}%
459     \gdef\noexpand\sfs@size{\strip@pt\@tempdimb}%
460     \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
461     \noexpand\math@fontstrue}}
462 \</2ekernel | def1)
463 (*autoload)
464 \def\calculate@math@sizes{\try@sizes\calculate@math@sizes}
465 \</autoload)

```

```

\defaultscriptratio The default ratio for math sizes is:
\defaultscriptscriptratio 1 to \defaultscriptratio to \defaultscriptscriptratio.
By default this is 1 to .7 to .5.
466 (*2ekernel | autoload)
467 \def\defaultscriptratio{.7}
468 \def\defaultscriptscriptratio{.5}

```

```

\noaccents@ If we don't have a definition for \noaccents@ we provide a dummy.
469 \ifx\noaccents@\@undefined
470   \let\noaccents@\@empty
471 \fi

```

```

\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 LATEX.
472 \</2ekernel | autoload)
473 (*2ekernel | autoerr)
474 \gdef\showhyphens#1{%
475   \setbox0\vbox{%
476     \color@begingroup
477     \everypar{}%
478     \parfillskip\z@skip\hsize\maxdimen
479     \normalfont
480     \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@ \ #1%
481     \color@endgroup}}
482 \</2ekernel | autoerr)
483 \<autoload)\def\showhyphens{\@autoerr\showhyphens}
484 (*2ekernel | autoload)

```

```

\addto@hook We need a macro to add tokens to a hook.
485 \long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}

```

```

\@vpt
486 \def\@vpt{5}

```

```

\@vipt
487 \def\@vipt{6}

```

```

\@viipt
488 \def\@viipt{7}

\@viipt
489 \def\@viipt{8}

\@ixpt
490 \def\@ixpt{9}

\@xpt
491 \def\@xpt{10}

\@xipt
492 \def\@xipt{10.95}

\@xiipt
493 \def\@xiipt{12}

\@xivpt
494 \def\@xivpt{14.4}

\@xvipt
495 \def\@xvipt{17.28}

\@xxpt
496 \def\@xxpt{20.74}

\@xxvpt
497 \def\@xxvpt{24.88}
498 </2ekernel | autoload>

```

File p

ltfsstrc.dtx

28 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 `tracefmt` 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 `tracefmt` 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 `tracefmt` package became active.

pausing Turn all font warnings into errors so that L^AT_EX will stop.

29 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>
```

30 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 %                               Standard LaTeX package (font tracing)]
15 </package>
```

The `debug` module makes use of commands contained in a special package file named `trace.sty`.⁴

```
16 <+debug> \input trace.sty
```

31 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 | autoload>
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 | autoload>
```

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>
```

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{%
```

⁴This package is not in distribution at the moment (and probably doesn't any longer work). Think of this part of the code as being historical artefacts.

```

31      \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
32      {LaTeX Font Info: \space\space\space#1}}%
33  \def\@font@warning#1{%
34      \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
35      {LaTeX Font Warning: #1}}%
36  }
37 \DeclareOption{warningshow}{%
38  \def\@font@info#1{%
39      \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
40      {LaTeX Font Info: \space\space\space#1}}%
41  \def\@font@warning#1{%
42      \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
43      {LaTeX Font Warning: #1}}%
44  }
45 \DeclareOption{infoshow}{%
46  \def\@font@info#1{%
47      \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
48      {LaTeX Font Info: \space\space\space#1}}%
49  \def\@font@warning#1{%
50      \GenericWarning{(Font)\@spaces\@spaces\@spaces\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\@spaces\@spaces\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 \endpackage

```

We also need a default definition inside the kernel:

```

73 \ifx\kernel\autoload
74 \def\@font@info#1{%
75     \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
76     {LaTeX Font Info: \space\space\space#1}}%
77 \def\@font@warning#1{%

```

```

78      \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
79      {LaTeX Font Warning: #1}}%
80 \</2ekernel | autoloading)

```

32 Macros common to fam.tex and tracefnt.sty

In the first versions of `tracefnt.dtx` some macros of `fam.dtx`⁵ 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 `lftss.dtx`.

32.1 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 | autoloading)
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 \</trace)

```

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 \</2ekernel | package | autoloading)

```

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.

`\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`.

⁵This file is currently not distributed in documented form. Its code is part of `lftss.dtx`.

```

96 <*2ekernel | autoload>
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
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 </2ekernel | autoload>

```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```

112 <*2ekernel | package | autoload>
113 \DeclareRobustCommand\selectfont
114   {%
When debug is specified we actually want something like 'undebg'. 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.
115 <+debug> \pushtracing
116 <+debug> \ifnum\tracingfonts<4 \tracingoff
117 <+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.
118   \ifx\f@linespread\baselinestretch \else
119     \set@fontsize\baselinestretch\f@size\f@baselineskip \fi
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 LATEX's \twfbbf, etc. We define \font@name globally,
as always. The reason for this is explained later on.
120   \xdef\font@name{%
121     \csname\curr@fontshape/\f@size\endcsname}%
We call the macro \pickup@font which will load the font if necessary.
122   \pickup@font
Then we select the font.
123   \font@name

```

If `\tracingfonts` is greater than 2 we also show the font switch. We do this before `\glb@settings` is called since this macro might redefine `\font@name`.

```
124 <*trace>
125     \ifnum \tracingfonts>\tw@
126     \font@info{Switching to \font@name}\fi
127 </trace>
```

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.

```
128     \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
129     \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
130 <+debug> \poptracing
131     }
```

`\set@fontsize` The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
132 \def\set@fontsize#1#2#3{%
133     \@defaultunits\@tempdimb#2pt\relax\@nnil
134     \edef\f@size{\strip@pt\@tempdimb}%
135     \@defaultunits\@tempskipa#3pt\relax\@nnil
136     \edef\f@baselineskip{\the\@tempskipa}%
137     \edef\f@linespread{#1}%
```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```
138     \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:

```
139     \def\size@update{%
```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```
140         \baselineskip\f@baselineskip\relax
141         \baselineskip\f@linespread\baselineskip
142         \normalbaselineskip\baselineskip
```

then to set up a new `\strutbox`

```
143         \setbox\strutbox\hbox{%
144             \vrule\@height.7\baselineskip
145             \@depth.3\baselineskip
146             \@width\z@}%
```

We end with a bit of tracing information.

```
147 <*trace>
148     \ifnum \tracingfonts>\tw@
149     \ifx\f@linespread\@empty
150         \let\reserved@a\@empty
151     \else
152         \def\reserved@a{\f@linespread x}%
```



```

153      \fi
154      \@font@info{Changing size to \f@size/\reserved@a
155                  \f@baselineskip}%
156      \aftergroup\type@restoreinfo \fi
157 </trace>

```

When all this is processed `\size@update` redefines itself to `\relax` so that in later calls of `\selectfont` no extra code will be executed.

```

158      \let\size@update\relax}%
159  }

```

Instead of defining this macro internally we might speed things up by placing the code into a separate macro and use `\let`!

`\size@update` Normally this macro does nothing; it will be redefined by `\set@fontsize` to initiate an update.

```

160 \let\size@update\relax

```

`\type@restoreinfo` This macro produces some info when a font size and/or baseline change will get restored.

```

161 <*trace>
162   \def\type@restoreinfo{%
163     \ifx\f@linespread\@empty
164       \let\reserved@a\@empty
165     \else
166       \def\reserved@a{\f@linespread x}%
167     \fi
168     \@font@info{Restoring size to
169                \f@size/\reserved@a\f@baselineskip}}
170 </trace>

```

`\glb@settings` The macro `\glb@settings` globally selects all math fonts for the current size if necessary.

```

\glb@currsiz
171 \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.

```

172   \expandafter\ifx\csname S@\f@size\endcsname\relax
173     \calculate@math@sizes
174   \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.

```

175   \csname S@\f@size\endcsname
176   \ifmath@fonts
177 <*trace>

```

```

178      \ifnum \tracingfonts>\tw@
179      \@font@info{Setting up math fonts for
180                  \f@size/\f@baselineskip}\fi
181 \</trace>

```

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`.

```

182      \begingroup
183      \escapechar\m@ne
184      \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@currsiz` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```

185      \globaldefs\@ne
186      \math@fonts
187      \let \glb@currsiz \f@size
188      \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\@ne` 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.

```

189      \the\every@math@size

```

Otherwise we announce that the math fonts are not set up for this size.

```

190 \< *trace>
191      \else
192      \ifnum \tracingfonts>\tw@
193      \@font@info{No math setup for
194                  \f@size/\f@baselineskip}\fi
195 \</trace>
196      \fi
197 }
198 \</2ekernel | package | autoload>

```

`\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).

```

199 \< *2ekernel | autoload>
200 \def\baselinestretch{1}

```

`\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.

```

201 \newtoks\every@math@size
202 \every@math@size={}
203 \</2ekernel | autoload>

```

32.2 Math fonts setup

32.2.1 Outline of algorithm for math font sizes

\TeX uses the 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

$\$ a=b+c \mathrel{\mathop{\boxtimes}} \small \text{ for all } b \text{ and } c \text{ in } \mathbb{Z} \$$

Here the inner formulae b and $c \text{ in } \mathbb{Z}$ 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:

1. At any point in the document the global variable `\gbl@currsz` 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@currsz`.
 - (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).

32.2.2 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.

```

204 <*2ekernel | package | autoload>
205 \def\check@mathfonts{%
206   \ifx \glb@currsz \f@size
207 <*trace>
208   \ifnum \tracingfonts>\thr@@
209     \@font@info{*** MATH: no change \f@size\space
210       curr/global (\curr@math@size/\glb@currsz)}\fi
211 </trace>
212   \else
213 <*trace>
214   \ifnum \tracingfonts>\thr@@
215     \@font@info{*** MATH: setting up \f@size\space
216       curr/global (\curr@math@size/\glb@currsz)}\fi
217 </trace>
218   \glb@settings
219   \init@restore@glb@settings
220 \fi
221 \let\curr@math@size\f@size
222 \def\init@restore@glb@settings{\aftergroup\restglb@settings}%
223 }
```

`\init@restore@glb@settings` This macros does by default nothing but get redefined inside `\check@mathfonts` to initiate fontsize restoring in nested formulas.

```

224 <-trace>\let\init@restore@glb@settings\relax
225 <*trace>
226 \def\init@restore@glb@settings{%
227   \ifnum \tracingfonts>\thr@@
228   \@font@info{*** MATH: no resetting (not in
229     nested math)}\fi
230 }
231 </trace>
```

`\restglb@settings` This macro will be executed the first time after the current formula.

```

232 \def\restglb@settings{%
233 <*trace>
234   \ifnum \tracingfonts>\thr@@
235     \@font@info{*** MATH: restoring}\fi
236 </trace>
237   \begingroup
238     \let\f@size\curr@math@size
239     \ifx\glb@currsz \f@size
240 <*trace>
241     \ifnum \tracingfonts>\thr@@
242       \@font@info{*** MATH: ... already okay (\f@size)}\fi
243 </trace>
244     \else
245 <*trace>
246     \ifnum \tracingfonts>\thr@@
247       \@font@info{*** MATH: ... to \f@size}\fi
248 </trace>
```

```

249         \glb@settings
250         \fi
251     \endgroup
252 }

```

32.2.3 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.

```

253 \def\use@mathgroup#1#2{\relax\ifmmode
254 (*trace)
255   \ifnum \tracingfonts>\tw@
256     \count@#2\relax
257     \@font@info{Using \noexpand\mathgroup
258               (\the\count@) #2}\fi
259 }/trace)

```

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.

```

260     \math@bgroup
261     \expandafter\ifx\csname M@f@encoding\endcsname#1\else
262       #1\fi
263     \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.

```

264     \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 \TeX macros for placing accents.

`\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`.

```

265 (*trace)
266   \ifx\math@bgroup\bgroup
267     \def\math@egroup#1{#1\egroup

```

```

268     \ifnum \tracingfonts>\tw@
269     \@font@info{Restoring \noexpand\mathgroup
270       (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
271       }\fi}
272   \fi
273 \</trace>

```

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the $\langle\textit{math group number}\rangle$ and the *family/series/shape* name as a control sequence.

```

274 \def\getanddefine@fonts#1#2{%

```

First we turn of tracing when `\tracingfonts` is less than 4.

```

275 <+debug> \pushtracing
276 <+debug> \ifnum\tracingfonts<4 \tracingoff
277 <+debug> \else \tracingon\getanddefine@fonts \fi

278 <*trace>
279 \ifnum \tracingfonts>\tw@
280 \count@#1\relax
281 \@font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
282   \string#2 \tf@size/\sf@size/\ssf@size}\fi
283 \</trace>

```

We append the current `\tf@size` to `#2` to obtain the font name.⁶ Again, `font@name` is defined globally, for the reasons explained in the description of `\wrong@fontshape`.

```

284 \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`.

```

285 \pickup@font \let\textfont@name\font@name

Same game for \scriptfont and \scriptscriptfont:

286 \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
287 \pickup@font \let\scriptfont@name\font@name
288 \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
289 \pickup@font

```

Then we append the new `\textfont...` assignments to the `\math@fonts`.

```

290 \edef\math@fonts{\math@fonts
291   \textfont#1\textfont@name
292   \scriptfont#1\scriptfont@name
293   \scriptscriptfont#1\font@name}%

```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```

294 <+debug> \poptracing
295   }
296 \</2ekernel | package | autoload>

```

⁶One 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.

33 Scaled font extraction

`\ifnot@nil` We begin with a simple auxiliary macro. It checks whether its argument is the token `\@nil`. If so, it expands to `\@gobble` which discards the following argument, otherwise it expands to `\@firstofone` which reproduces its argument.

```
297 \*2kernel | autoload)
298 \def\ifnot@nil#1{\def\reserved@a{#1}%
299 \ifx\reserved@a\@nnil \expandafter\@gobble
300 \else \expandafter\@firstofone\fi}
```

`\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`.

```
301 \def\remove@to@nnil#1\@nnil{}
302 \def\remove@angles#1>{\set@simple@size@args}
303 \def\remove@star#1*{#1}
304 \*2kernel | autoload)
```

`\extract@sizefn` This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```
305 \*2kernel | def2 | autoload)
306 \def\extract@sizefn#1*#2\@nnil{%
307 \if>#2>\set@size@funct@args#1\@nnil
308 \let\sizefn@info\@empty
309 \else\expandafter\set@size@funct@args\remove@star#2\@nnil
310 \def\sizefn@info{#1}\fi
311 }
```

`\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 correspondign 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 avour. 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 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<12*#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.

312 % % this could be replaced by \try@size@range making the subst slower!

313 \def\try@simple@size{%

\reserved@a is made an abbreviation for the head of the definition of the macro \extract@fontinfo.

314 \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.

315 \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%

316 \ifnot@nil{##2}%


```

317         {\set@simple@size@args##2<##3\@nnil
318         \execute@size@function\sizefn@info
319         }}%

```

Now we call `\extract@fontinfo`. Note the `<\@nnil` tokens at the end.

```

320     \expandafter\expandafter
321     \expandafter\extract@fontinfo\expandafter\font@info
322     \expandafter<\f@size>\@nnil<\@nnil
323 }

```

`\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,

```

324 \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.

```

325         \if<#1<{%
326         \expandafter\remove@angles
327         \else
328         \extract@sizefn#1*\@nnil
329         \expandafter\remove@to@nnil
330         \fi}

```

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 `<\@nnil->`. It looks for font ranges with font size functions. It's 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.

```

331 \def\extract@rangefontinfo#1<#2>{%
332     \is@range#2->\@nnil#2>}

```

`\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 `\@nnil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```

333 \def\is@range#1-#2\@nil{%
334   \if>#2\expandafter\check@single\else
335     \expandafter\check@range\fi}

\check@range \check@range takes lower bound as parameter #1, upper bound as #2, size func-
tion 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.
336 \def\check@range#1-#2>#3<#4\@nnil{%
337   \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.
338     \def\reserved@f{\extract@rangefontinfo<#4\@nnil}%
    We have to make sure that both boundaries are present, if not we have to set them.
    Here we check the upper bound. If \upper@bound is zero after the assignment we
    set it to \maxdimen (upper open range). We need to use a <dimen> register for
    the scan since we may have a decimal number as the boundary.
339     \upper@bound0#2\p@
340     \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.
341     \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.
342     \lower@bound0#1\p@
343     \ifdim \f@size \p@<\lower@bound
344     \else
    If both tests are passed we can try executing the size function.
345     \set@simple@size@args#3<#4\@nnil
346     \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.
347     \ifx\external@font\@empty
348     \else
349     \let\reserved@f\@empty
350     \fi
351     \fi
352     \fi
353     \reserved@f}}
354 </2ekernel | def2 | autoloading>

\lower@bound We use two dimen registers \lower@bound and \upper@bound to store the lower
\upper@bound and upper endpoints of the range we found.
355 <*2ekernel | autoloading>
356 \newdimen\lower@bound
357 \newdimen\upper@bound
358 </2ekernel | autoloading>

```

`\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).

```
359 <*2kernel | def2 | autoload>
360 \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.

```
361 \def\reserved@f{\extract@rangefontinfo<#3\@nnil}%
```

Now we check the the size against `\f@size`. If it is not equal `\f@size` it is not good and we have to recurse.

```
362 \ifdim \f@size \p@=#1\p@
```

Otherwise if this test is passed we can try executing the size function.

```
363 \set@simple@size@args#2<#3\@nnil
364 \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.

```
365 \ifx\external@font\@empty
366 \else
367 \let\reserved@f\@empty
368 \fi
369 \fi
370 \reserved@f}
```

`\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`.

```
371 \def\set@size@funct@args{\@ifnextchar[%
372 \set@size@funct@args@{\set@size@funct@args@[]}}
373 \def\set@size@funct@args@[#1]#2\@nil{%
374 \def\mandatory@arg{#2}%
375 \def\optional@arg{#1}}
376 </2kernel | def2 | autoload>
```

`\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 <...>.

```
377 <*2kernel | autoload>
378 \def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
379 \@onlypreamble\DeclareSizeFunction
380 </2kernel | autoload>
```

`\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`).

```
381 <*2kernel | package | autoload>
382 \def\execute@size@function#1{%    %% could be added to autoload as well
383 <*trace>
384 \@ifundefined{s@fct@#1}%
```

```

385         {\errmessage{Undefined font size function #1}%
386         \s@fct@}%
387         {\csname s@fct@#1\endcsname}%
388 \</trace>
389 \<-trace>         \csname s@fct@#1\endcsname
390 }
391 \</2ekernel | package | autoload>

```

\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.

```

392 \<*2ekernel | def2 | autoload>
393 \def\try@size@range{%
394     \expandafter\extract@rangefontinfo\font@info <-*\@nil>\@nnil
395 }
396 \</2ekernel | def2 | autoload>

```

\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.

```

397 \<*2ekernel | def1>
398 \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.

```

399     \@tempdimb \maxdimen
400     \let \best@size \@empty

```

Now we loop over the specification

```

401     \expandafter \try@simples \font@info <\number\@M>\@nil<\@nnil
402 }
403 \</2ekernel | def1>
404 \<*autoload>
405 \def\try@size@substitution{\try@simples\try@size@substitution}
406 \</autoload>

```

\font@submax 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{coument}**. The macro **\fontsubfuzz** contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```

407 \<*2ekernel | autoload>
408 \def\font@submax{0pt}
409 \def\fontsubfuzz{.4pt}
410 \</2ekernel | autoload>
411 \<+package>\def\fontsubfuzz{0pt}

```

\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.

```

412 <*2ekernel | def1>
413 \gdef\try@singles#1<#2>{%
414   \tryif@simple#2->\tryif@simple}
415 </2ekernel | def1>
416 <*autoload>
417 \def\try@singles{\@autoload{fss1}}
418 </autoload>

```

`\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.

```

419 <*2ekernel | def1>
420 \gdef\tryif@simple#1-#2\tryif@simple{%

```

Most common case for `\reserved@f` first:

```

421   \let \reserved@f \try@singles
422   \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.

```

423     \dimen@ #1\p@
424     \ifdim \dimen@<\@M\p@

```

If `\dimen@` is `\@M\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.

```

425     \ifdim \f@size\p@<\dimen@
426       \@tempdimc \dimen@
427       \advance\@tempdimc -\f@size\p@
428     \else
429       \@tempdimc \f@size\p@
430       \advance\@tempdimc -\dimen@
431     \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`.

```

432     \ifdim \@tempdimc<\@tempdimb
433       \@tempdimb \@tempdimc
434       \def \best@size{#1}%
435     \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`.

```

436     \else

```

`\subst@size` 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.

```

437 % \subst@size          %% coded inline
438 % \def\subst@size{%

```

```

439 \ifx \external@font\@empty
440 \ifx \best@size\@empty
441 \else
442 \ifdim \@tempdimb>\font@submax \relax
443 \xdef \font@submax {\the\@tempdimb}%
444 \fi
445 \let \f@user@size \f@size
446 \let \f@size \best@size
447 \ifdim \@tempdimb>\fontsubfuzz\relax
448 \@font@warning{Font\space shape\space
449 '\curr@fontshape'\space in\space size\space
450 <\f@user@size>\space not\space available\MessageBreak
451 size\space <\f@size>\space substituted}%
452 \fi
453 \try@simple@size
454 \do@subst@correction
455 \fi
456 \fi
457 % %}

```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

```

458 \let \reserved@f \remove@to@nnil
459 \fi
460 \fi

```

If it's a range iterate also.

```

461 \reserved@f}
462 </2ekernel | def1>

```

33.1 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.

```

463 <*2ekernel | autoload>
464 \DeclareSizeFunction{}\@empty@sfcnt\@font@warning}
465 \DeclareSizeFunction{s}\@empty@sfcnt\@font@info}
466 </2ekernel | autoload>
467 <*2ekernel | def2 | autoload>
468 \def\@empty@sfcnt#1{%
469 \tempdimb \f@size\p@
470 \ifx\optional@arg\@empty
471 \else
472 \tempdimb \optional@arg\@tempdimb
473 #1{Font\space shape\space '\curr@fontshape'\space
474 will\space be\MessageBreak
475 scaled\space to\space size\space \the\@tempdimb}%
476 \fi
477 \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}
478 </2ekernel | def2 | autoload>

```

`\sfct@gen` This size function generates the external name from the mandatory argument and
`\sfct@sgen` 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.

```
479 <*2kernel | autoloading>
480 \DeclareSizeFunction{gen}{\gen@sfcnt\font@warning}
481 \DeclareSizeFunction{sgen}{\gen@sfcnt\font@info}
482 </2kernel | autoloading>

483 <*2kernel | def2 | autoloading>
484 \def\gen@sfcnt{%
485     \edef\mandatory@arg{\mandatory@arg\f@size}%
486     \empty@sfcnt}
487 </2kernel | def2 | autoloading>
```

`\sfct@genb` This size function is similar to `gen`, but for fonts where the size is encoded in
`\sfct@sgenb` the font name in centipoints, 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.

```
488 <*2kernel | autoloading>
489 \DeclareSizeFunction{genb}{\genb@sfcnt\font@warning}
490 \DeclareSizeFunction{sgenb}{\genb@sfcnt\font@info}
491 </2kernel | autoloading>

492 <*2kernel | def2 | autoloading>
493 \def\genb@sfcnt{%
494     \edef\mandatory@arg{\mandatory@arg\expandafter\genb@x\f@size..\@}%
495     \empty@sfcnt}
496 </2kernel | def2 | autoloading>
```

`\genb@x` The auxiliary macros `\genb@x` and `\genb@y` are used to convert the `\f@size` into
`\genb@y` centipoints.

```
497 <*2kernel | def2 | autoloading>
498 \def\genb@x#1.#2.#3\@{\two@digits{#1}\genb@y#200\@}
499 \def\genb@y#1#2#3\@{\#1#2}
500 </2kernel | def2 | autoloading>
```

`\sfct@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.

```
501 <*2kernel | autoloading>
502 \DeclareSizeFunction{sub}{\sub@sfcnt\font@warning}
503 \DeclareSizeFunction{ssub}{\sub@sfcnt\font@info}
504 </2kernel | autoloading>

505 <*2kernel | def2 | autoloading>
506 \def\sub@sfcnt#1{%
507     \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.

```
508     \begingroup
```

```

509     \expandafter\split@name\mandatory@arg/\@nil
510     \try@load@fontshape
511 \endgroup

```

Then we record the current `\f@size` since it may get clobbered.

```

512     \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`.

```

513     \expandafter
514     \ifx\csname\mandatory@arg\endcsname\relax
515         \errmessage{No\space declaration\space for\space
516                 shape\space \mandatory@arg}%
517     \error@fontshape
518 \else

```

Otherwise we warn the user about the substitution taking place.

```

519     #1{Font\space shape\space '\curr@fontshape'\space in\space
520         size\space <\f@size>\space not\space available\MessageBreak
521         Font\space shape\space '\mandatory@arg'\space tried\space
522         instead}%
523     \expandafter\split@name\mandatory@arg/\@nil
524 \fi

```

Then we restart the font specification scan by calling `\get@external@font`.

```

525     \edef\f@size{\f@user@size}%
526     \get@external@font

```

Finally `\do@subst@correction` is called to get the font name right.

```

527     \do@subst@correction
528 }
529 </2ekernel | def2 | autoloading>

```

\sfct@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.

```

530 <*2ekernel | autoloading>
531 \DeclareSizeFunction{subf}{\subf@sfcnt\@font@warning}
532 \DeclareSizeFunction{ssubf}{\subf@sfcnt\@font@info}
533 </2ekernel | autoloading>
534 <*2ekernel | def2 | autoloading>
535 \def\subf@sfcnt#1{%
536     #1{Font\space shape\space '\curr@fontshape'\space in\space
537         size\space \f@size\space not\space available\MessageBreak
538         external\space font\space '\mandatory@arg'\space used}%
539     \empty@sfcnt#1%
540 }
541 </2ekernel | def2 | autoloading>

```

\sfct@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.

```

542 <*2ekernel | autoloading>

```



```

543 \DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
544 \DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}
545 </2ekernel | autoload>

546 <*2ekernel | def2 | autoload>
547 \def\fixed@sfcnt#1{%
548   \ifx\optional@arg\@empty
549     \let\external@font\mandatory@arg
550   \else
551     \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
552   \fi
553   #1{External\space font\space '\external@font'\space loaded\space
554     for\space size\MessageBreak
555     <\f@size>}}%
556 }
557 </2ekernel | def2 | autoload>

```

File q

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.

34 Compatibility code for NFSS release 1

There have been a couple of commands which became obsolete with NFSS2. In the past they have been still part of the kernel code to make it possible to process old packages using those commands but since they take up valuable space we decided to remove them and instead auto-load their definitions if they are actually encountered in some file.

Thus the following code doesn’t really belong to this file but I put it here for the moment until finally a documented version of `ltfss.dtx` is available.

[auto-loading not activated]

<code>\new@fontshape</code> <code>\subst@fontshape</code> <code>\extra@def</code> <code>\default@mextra</code> <code>\define@mathalphabet</code> <code>\define@mathgroup</code>	These macros are the interfaces in NFSS1 which shouldn’t be used any longer. We all define them to call the macro <code>\scan@fontshape</code> which is an internal macro that loads the real definitions and then to execute themselves again. Once this auto-loading has happened they have the definition shown below and thus execute their real code directly. 1 <code>*autoload</code> 2 <code>\def\new@fontshape{\scan@fontshape\new@fontshape}</code> 3 <code>\def\subst@fontshape{\scan@fontshape\subst@fontshape}</code> 4 <code>\def\extra@def{\scan@fontshape\extra@def}</code> 5 <code>\def\default@mextra{\scan@fontshape\default@mextra}</code> 6 <code>\def\define@mathalphabet{\scan@fontshape\define@mathalphabet}</code> 7 <code>\def\define@mathgroup{\scan@fontshape\define@mathgroup}</code> <code>\scan@fontshape</code>	Here is the kernel definition for <code>\scan@fontshape</code> which loads the actual definitions from the file <code>nfsscmp.def</code> . 8 <code>\def\scan@fontshape{\input{nfsscmp.def}}</code>
--	--	---

The following definitions are now placed into the auto-load file.

Since we don’t know when this file will be read in we need to provide ourselves with standard `\catcode` settings. This is done by placing all definitions in a group and calling `\nfss@catcodes`. But this macro will also disable spaces which isn’t very appropriate for the following code because it contains a lot of helper messages. Therefore we change this back.

```
9 \begingroup
10 \nfss@catcodes
11 \catcode'\ =10\relax
12 \</autoload>
13 \*compat
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
14 \gdef\new@fontshape#1#2#3#4{%
15     \warn@rel@i\new@fontshape\DeclareFontShape
16     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
17     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}
18 \@onlypreamble\new@fontshape
```

`\warn@rel@i` The warning message used above.

```
19 \gdef\warn@rel@i#1#2{%
20     \@font@warning{*** NFSS release 1 command
21                     \noexpand#1found\MessageBreak
22     *** Update by using release 2 command
23     \string#2.\MessageBreak
24     *** Recovery is probably possible}%
25 }
26 \@onlypreamble\warn@rel@i
```

`\scan@fontshape` This will scan the old font shape definition syntax.

```
27 \gdef\scan@fontshape{%
28     \let\reserved@f\@empty
29     \let\reserved@e\@empty %           holds last info
30     \scan@@fontshape
31 }
32 \@onlypreamble\scan@fontshape
```

`\scan@@fontshape`

```
33 \gdef\scan@@fontshape#1>#2#3<{%
34     \ifx\@nil#1%
35         \edef\reserved@f{\reserved@f\reserved@e}%
36     \else
37         \def\reserved@b{#1}%           nick names
38         \def\reserved@c{#3}%
39         \in@{ at}{#3}%
40         \ifin@
41             \in@{pt}{#3}% not a proof but a good chance
42         \ifin@
```

We grap also everything after pt and discard it if people have forgotten to place a percent sign there.

```
43     \def\reserved@a##1 at##2pt##3\@nil{%
44         \def\reserved@b{##2}%
45         \def\reserved@c{##1}%
46     }%
47     \reserved@a#3\@nil
48     \fi
49 \fi
50 \ifnum 0<0#2
51     \edef\reserved@d{subf*\reserved@c}%
52     \ifcase #2\or
53     \or
54     \else
55         \errmessage{*** What's this? NFSS release 0? ***}%
56     \fi
```

```

57 \else
58 \edef\reserved@d{\#2\reserved@c}%
59 \fi
60 \ifx\reserved@d\reserved@e
61 \edef\reserved@f{\reserved@f<\reserved@b>}%
62 \else
63 \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
64 \let\reserved@e\reserved@d
65 \fi
66 \expandafter\scan@@fontshape
67 \fi
68 }
69 \@onlypreamble\scan@@fontshape

\subst@fontshape This is now also handled by the extend syntax of \DeclareFontShape.
70 \gdef\subst@fontshape#1#2#3#4#5#6{%
71 \warn@rel@i\subst@fontshape\DeclareFontShape
72 \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}
73 \@onlypreamble\subst@fontshape

\extra@def This was replaced by \DeclareFontFamily.
74 \gdef\extra@def#1#2#3{%
75 \warn@rel@i\extra@def\DeclareFontFamily
76 \DeclareFontFamily{U}{#1}{}}
77 }
78 \@onlypreamble\extra@def

\default@mextra The new name is \DeclareFontEncodingDefaults but in this case we don't feel
comfortable with this either.
79 \gdef\default@mextra{%
80 \warn@rel@i\default@mextra\DeclareFontEncodingDefaults
We pick up the argument to \default@mextra implicitly as the second argument
of \DeclareFontEncodingDefaults.
81 \DeclareFontEncodingDefaults\relax
82 }
83 \@onlypreamble\default@mextra

\preload@sizes The new interface is \DeclarePreloadSizes.
84 \gdef\preload@sizes{%
85 \warn@rel@i\preload@sizes\DeclarePreloadSizes
86 \DeclarePreloadSizes U%
87 }
88 \@onlypreamble\preload@sizes

\err@rel@i This macro is used in cases where emulation with NFSS2 features is not really
possible.
89 \gdef\err@rel@i#1#2{%
90 \@latex@error{*** NFSS release 1 command \noexpand#1found%
91 ^^J*** Recovery not possible. Use \string#2}%
92 {The new release of NFSS doesn't support the
93 \noexpand#1command^^Jany longer.
94 Please upgrade your file to the syntax of NFSS
95 release 2^^Jusing the \noexpand#2command.}%

```

Let's die.

```

96 \batchmode\input.\relax
97 }
98 \@onlypreamble\err@rel@i

\newmathalphabet \newmathalphabet is the old form.
\newmathalphabet@@ 99 \gdef\newmathalphabet{%
\newmathalphabet@@@ 100 \if@no@font@opt
101 \latex@error{*** NFSS release 1 command
102 \noexpand\newmathalphabet found%
103 ^^J \space*** Automatic recovery not possible.%
104 ^^J \space*** TYPE H for Help%
105 }%
106 {Please look at the file usrguide.tex for hints on
107 how to resolve this problem.}%
108 \else
109 \warn@rel@i\newmathalphabet\DeclareMathAlphabet
110 \fi
111 \@ifstar\newmathalphabet@@@
112 \newmathalphabet@@}
113 \gdef\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{-}{-}{-}}
114 \gdef\newmathalphabet@@@#1#2#3#4{%
115 \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}
116 \@onlypreamble\newmathalphabet
117 \@onlypreamble\newmathalphabet@@
118 \@onlypreamble\newmathalphabet@@@

\if@no@font@opt
\@no@font@optfalse 119 \global\let\if@no@font@opt\iftrue
120 \gdef\@no@font@optfalse{\let\if@no@font@opt\iffalse}

\define@mathalphabet This is a case where dying is best.
121 \gdef\define@mathalphabet{%
122 \err@rel@i\define@mathalphabet\DeclareMathAlphabet
123 }
124 \@onlypreamble\define@mathalphabet

\define@mathgroup And here is another one
125 \gdef\define@mathgroup{%
126 \err@rel@i\define@mathgroup\DeclareSymbolFont
127 }
128 \@onlypreamble\define@mathgroup
129 </compat>

\addtoversion \addtoversion is the old form.
130 \def\addtoversion#1#2{%
131 \warn@rel@i\addtoversion\SetMathAlphabet
132 \SetMathAlphabet#2{#1}{U}}
133 \@onlypreamble\addtoversion

```

That finishes the definitions for the old interfaces — but first we better finish the group.

```
134 <*autoload>
135 \endgroup
136 </autoload>
```

File r

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.

35 Interface Commands

`\in@` `\@in` is a utility macro with two arguments. It determines whether its first argument occurs in its second (after expanding it) and sets the switch `\if@in` accordingly.

```
1 <*2ekernel | autoloading>
2 \def\in@#1#2{%
3   \def\in@@#1#1##2##3\in@@{%
4     \ifx\in@@#2\in@false\else\in@true\fi}%
5   \in@@#2#1\in@\in@@}
6 \newif\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

```
\version@elt<version1>\version@elt<version2>...
\version@elt<versionn>
```

- the list of all math alphabet identifiers. Here every entry has the form:

```
\group@elt<math group number>
{\{<default family>\}{<default series>\}{<default shape>\}}.
```

- 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 *font info* is either `\reserved@e` (if the combination is not defined yet) or

$$\{\langle family \rangle\}\{\langle series \rangle\}\{\langle shape \rangle\}$$

`\version@list` We initialize the version list to be empty.

```
7 \let\version@list=\@empty
8 \@onlypreamble\version@list
```

`\version@elt`

```
9 \let\version@elt\relax
10 \@onlypreamble\version@elt
```

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```
11 %\def\new@mathversion#1{%
```

The first thing this macro does is to check if the version identifier is already present in `\version@list`. We enclose `\version@list` in braces since it might be empty (if no *version* is defined yet). But this means that we need a suitable number of `\expandafter` primitives.

```
12 % \expandafter\in@\expandafter#1\expandafter{\version@list}%
13 % \ifin@
```

If so it prints an error message. The `\next` macro is used to get rid of the four characters `\mv@` that would otherwise appear at the begin of the version name in the error message.

```
14 % \latexerror{Math version
15 % \expandafter\@gobblefour\string#1'
16 % already defined}\@eha
```

Otherwise we have a new version, and we can proceed with entering it into the tables. We add it to `\version@list`. This is very easy: we define `\version@elt` (which is the delimiter in `\version@list`) to protect itself and the following token from being expanded and simply redefine `\version@list`.

```
17 % \else
18 % \global\expandafter\newcount\csname c@\expandafter
19 % \gobble\string#1\endcsname
20 % \global\csname c@\expandafter
21 % \gobble\string#1\endcsname\@ne
22 % \def\version@elt{\noexpand\version@elt\noexpand}%
23 % \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.

```
24 % \def\reserved@c{\noexpand\reserved@c\noexpand}%
25 % \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).

```
26 % \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.

```
27 %          \edef##1{\expandafter\remove@nil##1%
28 %              \reserved@c
29 %              #1%
30 %              \reserved@e
31 %              \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.

```
32 %      \alpha@list
33 %      \fi}
```

`\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`.

```
34 \let\alpha@list\@empty
35 \@onlypreamble\alpha@list
```

`\alpha@elt`

```
36 \let\alpha@elt\relax
37 \@onlypreamble\alpha@elt
```

`\newgroup` Start the group (fam) allocation at 0. (Doesn't belong here.)

```
38 \count18=-1
```

`\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.

```
39 \def\select@group#1#2#3#4{%
40   \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
41   {%
42     \ifmmode
43       \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
44         \begingroup
45           \escapechar\m@ne
46           \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
47           \globaldefs\@ne \math@fonts
48         \endgroup
49         \init@restore@version
50         \xdef#1{\noexpand\use@mathgroup\noexpand#2%
51             {\number\csname c@mv@\math@version\endcsname}}%
```

```

52   \global\advance\csname c@mv@\math@version\endcsname\@ne
53   \else
54     \let#1\relax
55     \@latex@error{Too many math alphabets used in
56               version \math@version}%
57     \@eha
58   \fi
59 \else \expandafter\@non@alpherr\fi
60 #1{#4}%
61 }%
62 }
63 \@onlypreamble\restore@mathversion

```

`\init@restore@version`

```

64 \def\init@restore@version{%
65   \global\let\init@restore@version\relax
66   \xdef\restore@mathversion
67     {\expandafter\@noexpand\csname mv@\math@version\endcsname
68       \global\csname c@mv@\math@version\endcsname
69       \number\csname c@mv@\math@version\endcsname\relax}%
70   \aftergroup\dorestore@version
71 }
72 \@onlypreamble\init@restore@version

```

`\non@alpherr`

```

73 </2ekernel | autoloading>
74 <*2ekernel | autoerr>
75 \gdef\non@alpherr#1{\@latex@error{%
76   \string#1allowed only in math mode}\@ehd}
77 </2ekernel | autoerr>
78 <autoloading>\def\non@alpherr{\@autoerr\non@alpherr}
79 <*2ekernel | autoloading>

```

The command here will have a space at the end of its name, so we make sure not to insert an extra one.

`\dorestore@version`

```

80 \def\dorestore@version
81 {\ifmmode
82   \aftergroup\dorestore@version
83 \else
84   \gdef\init@restore@version{%
85     \global\let\init@restore@version\relax
86     \xdef\restore@mathversion
87       {\expandafter\@noexpand\csname mv@\math@version\endcsname
88         \global\csname c@mv@\math@version\endcsname
89         \number\csname c@mv@\math@version\endcsname\relax}%
90     \aftergroup\dorestore@version
91   }%
92   \begingroup
93     \let\getanddefine@fonts\@gobbletwo
94     \restore@mathversion
95   \endgroup
96   \fi}%

```

```

97 \@onlypreamble\dorestore@version

\document@select@group We surround \select@group with braces so that functions using it can be used
                        directly after _ or ^.
98 \def\document@select@group#1#2#3#4{%
99   \ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
100  {%
101   \ifmmode
102     \ifnum\csname c@mv@\math@version\endcsname<\sist@n
103       \begingroup
104         \escapechar\m@ne
105         \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
106         \globaldefs\@ne \math@fonts
107       \endgroup
108       \expandafter\extract@alph@from@version
109       \csname mv@\math@version\expandafter\endcsname
110       \expandafter{\number\csname
111         c@mv@\math@version\endcsname}%
112       #1%
113       \global\advance\csname c@mv@\math@version\endcsname\@ne
114     \else
115       \let#1\relax
116       \@latex@error{Too many math alphabets used
117         in version \math@version}%
118       \@eha
119     \fi
120   \else \expandafter\non@alpherr\fi
121   #1{#4}%
122  }%
123 }

\process@table

124 \def\process@table{%
125   \def\cdp@elt##1##2##3##4{%
126     \@font@info{Checking defaults for
127       ##1/##2/##3/##4}%
128     \expandafter
129     \ifx\csname##1/##2/##3/##4\endcsname\relax

Grouping is important for two reasons, first \cdp@elt will get redefined if
\Declare... functions are executed within the external .fd file and secondly
\try@load@fontshape changes a lot of catcodes without surrounding itself with
a group.

130     \begingroup
131     \def\f@encoding{##1}\def\f@family{##2}%
132     \try@load@fontshape
133     \endgroup
134   \fi
135   \expandafter
136   \ifx\csname##1/##2/##3/##4\endcsname\relax
137     \@latex@error{This NFSS system isn't set up properly}%
138     {For encoding scheme ##1 the defaults
139       ##2/##3/##4 do not form a valid font shape}%
140   \else

```

```

141         \@font@info{... okay}%
142     \fi}%
143 \cdp@list
    Now we make sure that \error@fontshape is okay.
144 \begingroup
145     \escapechar\m@ne
146     \error@fontshape
147     \expandafter\ifx\csname \curr@fontshape\endcsname\relax
148         \begingroup
149             \try@load@fontshape
150         \endgroup
151     \fi
152     \expandafter\ifx\csname \curr@fontshape\endcsname\relax
153         \@latex@error{This NFSS system isn't set up properly}%
154         {The system maintainer forgot to specify a suitable
155         substitution
156         font shape using the \noexpand\DeclareErrorFont
157         command}%
158     \fi
159 \endgroup
    Set \select@group to its meaning used within the document body.
160 \let\select@group\document@select@group
    Install the default font attributes they are currently pointing to error font shape.
    Don't use \reset@font since that would trigger \selectfont.
161 \fontencoding{\encodingdefault}%
162 \fontfamily{\familydefault}%
163 \fontseries{\seriesdefault}%
164 \fontshape{\shapedefault}%
    kill all macros not longer needed. we need to add many more!!!!
165 \everyjob{}%
166 }
167 \@onlypreamble\process@table

168 %\@onlypreamble\set@mathradical

\DeclareMathVersion
169 \def\DeclareMathVersion#1{%
170     \expandafter\new@mathversion\csname mv@#1\endcsname}
171 \@onlypreamble\DeclareMathVersion

\new@mathversion
172 \def\new@mathversion#1{%
173     \expandafter\in@\expandafter#1\expandafter{\version@list}%
174     \ifin@
175         \@font@info{Redeclaring math version
176             '\expandafter@gobblefour\string#1'}%
177     \else
178         \global\expandafter\newcount\csname c@\expandafter
179             \expandafter@gobble\string#1\endcsname
180         \def\version@elt{\noexpand\version@elt\noexpand}%
181         \edef\version@list{\version@list\version@elt#1}%
182     \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.

```
183 \toks@{}%
184 \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.

```
185 \def\group@elt##1##2{%
186     \advance\count@\@ne
187     \addto@hook\toks@{\getanddefine@fonts##1##2}%
188     }%
189 \group@list
```

We set the counter for this math version to the number of math groups found in `\group@list`.

```
190 \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.

```
191 \def\alpha@elt##1##2##3{%
192     \ifx##2\no@alphabet@error
193     \toks@\expandafter{\the\toks@\install@mathalphabet##1%
194         {no@alphabet@error##1}}%
195     \else
196     \toks@\expandafter{\the\toks@\install@mathalphabet##1%
197         {\select@group##1##2##3}}%
198     \fi
199     }%
200 \alpha@list
```

Finally we define the math version to expand to the contents of `\toks@`.

```
201 \xdef#1{\the\toks@}%
202 }
203 \@onlypreamble\new@mathversion
```

`\DeclareSymbolFont`

```
204 \def\DeclareSymbolFont#1#2#3#4#5{%
205     \@tempswafalse
206     \edef\reserved@b{#2}%
207     \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
208         \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
209     \cdp@list
210     \if@tempswa
211     \@ifundefined{sym#1}{%
212         \expandafter\new@mathgroup\csname sym#1\endcsname
213         \expandafter\new@symbolfont\csname sym#1\endcsname
214             {#2}{#3}{#4}{#5}}%
215     {%
216         \@font@info{Redeclaring symbol font '#1'}%
```

Update the group list.

```
217     \def\group@elt##1##2{%
218         \noexpand\group@elt\noexpand##1%
```

```

219         \expandafter\ifx\csname sym#1\endcsname##1%
220         \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
221         \else
222             \noexpand##2%
223         \fi}%
224     \xdef\group@list{\group@list}%
    Update the version list.
225     \def\version@elt##1{%
226         \expandafter
227         \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\expandafter
228             \endcsname \csname sym#1\endcsname
229     }%
230     \version@list
231 }%
232 \else
233     \@latex@error{Encoding scheme ‘#2’ unknown}\@eha
234 \fi
235 }
236 \@onlypreamble\DeclareSymbolFont

```

\group@list

```

237 \let\group@list\@empty
238 \@onlypreamble\group@list

```

\group@elt

```

239 \let\group@elt\relax
240 \@onlypreamble\group@elt

```

\new@symbolfont

```

241 \def\new@symbolfont#1#2#3#4#5{%
242     \toks@\expandafter{\group@list}%
243     \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
244         \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
245     \def\version@elt##1{\the\toks@\expandafter{##1}%
246         \edef##1{\the\toks@\noexpand\getanddefine@fonts
247             #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
248         \global\advance\csname c@\expandafter
249             \gobble\string##1\endcsname\@ne
250         }%
251     \version@list
252 }
253 \@onlypreamble\new@symbolfont

```

\SetSymbolFont

```

254 \def\SetSymbolFont#1#2#3#4#5#6{%
255     \@tempswafalse
256     \edef\reserved@b{#3}%
257     \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
258         \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
259     \cdp@list
260     \if@tempswa
261         \expandafter\SetSymbolFont@

```

```

262 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
263 \endcsname \csname sym#1\endcsname
264 \else
265 \@latex@error{Encoding scheme ‘#3’ unknown}\@eha
266 \fi
267 }
268 \@onlypreamble\SetSymbolFont

```

\SetSymbolFont@

```

269 \def\SetSymbolFont@#1#2#3{%
270 \expandafter\in@\expandafter#1\expandafter{\version@list}%
271 \ifin@
272 \expandafter\in@\expandafter#3\expandafter{\group@list}%
273 \ifin@
274 \begingroup
275 \expandafter\get@cdp\string#2\@nil\reserved@a
276 \toks@{}%
277 \def\install@mathalphabet##1##2{%
278 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
279 }%
280 \def\getanddefine@fonts##1##2{%
281 \ifnum##1=#3%
282 \addto@hook\toks@{\getanddefine@fonts#3#2}%
283 \expandafter\get@cdp\string##2\@nil\reserved@b
284 \ifx\reserved@a\reserved@b\else
285 \@font@warning{Encoding ‘\reserved@b’ has changed
286 to ‘\reserved@a’ for symbol font\MessageBreak
287 ‘\expandafter\@gobblefour\string#3’ in the
288 math version ‘\expandafter
289 \@gobblefour\string#1’}%
290 \fi
291 \@font@info{%
292 Overwriting symbol font
293 ‘\expandafter\@gobblefour\string#3’ in
294 version ‘\expandafter
295 \@gobblefour\string#1’\MessageBreak
296 \spaces \expandafter\@gobble\string##2 -->
297 \expandafter\@gobble\string#2}%
298 \else
299 \addto@hook\toks@{\getanddefine@fonts##1##2}%
300 \fi}%
301 #1%
302 \xdef#1{\the\toks@}%
303 \endgroup
304 \else
305 \@latex@error{Symbol font ‘\expandafter\@gobblefour\string#3’
306 not defined}\@eha
307 \fi
308 \else
309 \@latex@error{Math version ‘\expandafter\@gobblefour\string#1’
310 is not
311 defined}{You probably misspelled the name of the math
312 version.^^JOr you have to specify an additional package.}%
313 \fi

```

```

314 }
315 \@onlypreamble\SetSymbolFont@

\get@cdp
316 \def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
317 \@onlypreamble\get@cdp

\DeclareMathAlphabet

318 \def\DeclareMathAlphabet#1#2#3#4#5{%
319   \@tempswafalse
320   \edef\reserved@b{#2}%
321   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
322     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
323   \cdp@list
324   \if@tempswa
325     \expandafter\ifx
326       \csname\expandafter\@gobble\string#1\endcsname
327       \relax
328       \new@mathalphabet#1{#2}{#3}{#4}{#5}%
329     \else
      Check if it is already a math alphabet.
330       \edef\reserved@a{\noexpand\in@{\string\select@group}%
331         {\expandafter\meaning\csname \expandafter
332           \@gobble\string#1\space\endcsname}}%
333       \reserved@a
334       \ifin@
335         \@font@info{Redeclaring math alphabet \string#1}%
336         \def\version@elt##1{%
337           \expandafter\SetMathAlphabet@\expandafter
338             ##1\csname#2/#3/#4/#5\expandafter\endcsname
339           \csname M@#2\expandafter\endcsname
340           \csname \expandafter\@gobble\string#1\space\endcsname#1}%
341         \version@list
342       \else
      Check if it is a math alphabet defined via \DeclareSymbolFontAlphabet.
343       \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
344         {\expandafter\meaning\csname \expandafter
345           \@gobble\string#1\space\endcsname}}%
346       \reserved@a
347       \ifin@
      In that case overwriting is simple since there is nothing inserted in the math
      version macros.
348       \@font@info{Redeclaring math alphabet \string#1}%
349       \new@mathalphabet#1{#2}{#3}{#4}{#5}%
      Otherwise panic.
350       \else
351       \@latex@error{Command '\string#1' already defined}\@eha
352       \fi
353     \fi
354   \fi

```



```

355 \else
356 \latex@error{Encoding scheme ‘#2’ unknown}\@eha
357 \fi
358 }
359 \@onlypreamble\DeclareMathAlphabet

```

`\new@mathalphabet`

```

360 \def\new@mathalphabet#1#2#3#4#5{%
361   \toks@{\expandafter{\alpha@list}}%
362   \edef#1{\expandafter\noexpand\csname \expandafter
363     \@gobble\string#1\space\endcsname
364     \if/#5/%
365       \noexpand\no@alphabet@error
366       \noexpand\no@alphabet@error
367     \else
368       \expandafter\noexpand\csname M@#2\endcsname
369       \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
370     \fi
371   }%
372   \toks2\expandafter{#1}%
373   \edef\alpha@list{\the\toks@ \noexpand\alpha@elt\the\toks2}%
374   \def\version@elt##1{\toks@{\expandafter{##1}}%
375     \edef##1{\the\toks@ \install@mathalphabet
376       \expandafter\noexpand
377       \csname \expandafter\@gobble
378         \string#1\space\endcsname
379       {\if/#5/%
380         \noexpand\no@alphabet@error
381         \noexpand#1%
382       \else
383         \noexpand\select@group\the\toks2
384       \fi}}%
385     }%
386   \version@list
387   \expandafter\edef\csname \expandafter\@gobble
388     \string#1\space\endcsname{\if/#5/%
389     \noexpand\no@alphabet@error
390     \noexpand#1%
391   \else
392     \noexpand\select@group\the\toks2
393   \fi}%
394   \edef#1{\noexpand\protect
395     \expandafter\noexpand\csname \expandafter
396     \@gobble\string#1\space\endcsname}%
397 }
398 \@onlypreamble\new@mathalphabet

```

`\SetMathAlphabet`

```

399 \def\SetMathAlphabet#1#2#3#4#5#6{%
400   \@tempswafalse
401   \edef\reserved@b{#3}%
402   \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
403     \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
404   \cdp@list

```

```

405 \if@tempswa
406 \expandafter\SetMathAlphabet@
407 \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
408 \endcsname \csname M@#3\expandafter\endcsname
409 \csname \expandafter\@gobble\string#1\space\endcsname#1%
410 \else
411 \@latexerror{Encoding scheme ‘#3’ unknown}\@eha
412 \fi
413 }
414 \@onlypreamble\SetMathAlphabet

```

`\SetMathAlphabet@`

```

415 \def\SetMathAlphabet@#1#2#3#4#5{%
416 \expandafter\in@\expandafter#1\expandafter{\version@list}%
417 \ifin@
418 \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
419 \ifin@
420 \begingroup
421 \toks@{}%
422 \def\getanddefine@fonts##1##2{%
423 \addto@hook\toks@{\getanddefine@fonts##1##2}%
424 }%
425 \def\reserved@c##1##2##3##4{% % for message below
426 \expandafter\@gobble\string##4}%
427 \def\install@mathalphabet##1##2{%
428 \ifx##1#4%
429 \addto@hook\toks@
430 {\install@mathalphabet#4{\select@group#4#3#2}}%
431 \@font@info{Overwriting math alphabet
432 ‘\string#5’ in version ‘\expandafter
433 \@gobblefour\string#1’\MessageBreak
434 \@spaces \reserved@c##2 -->
435 \expandafter\@gobble\string#2}%
436 \else
437 \addto@hook\toks@{\install@mathalphabet##1{##2}}%
438 \fi
439 }%
440 #1%
441 \xdef#1{\the\toks@}%
442 \endgroup
443 \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.

```

444 \edef\reserved@a{%
445 \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
446 \reserved@a
447 \ifin@
448 \def\reserved@b##1\use@mathgroup##2##3{%
449 \def\reserved@b{##3}\def\reserved@c{##2}}%
450 \expandafter\reserved@b#4%
451 \begingroup
452 \def\install@mathalphabet##1##2{%

```

```

453         \addto@hook\toks@{\install@mathalphabet##1{##2}}%
454     }%
455     \def\getanddefine@fonts##1##2{%
456         \addto@hook\toks@{\getanddefine@fonts##1##2}%
457         \ifnum##1=\reserved@b
458             \expandafter
459             \addto@hook\expandafter\toks@
460             \expandafter{\expandafter\install@mathalphabet
461             \expandafter#4\expandafter
462                 {\expandafter\select@group\expandafter
463                 #4\reserved@c##2}}}%
464         \fi
465     }%
466     \def\version@elt##1{%
467         \toks@{}%
468         ##1%
469         \xdef##1{\the\toks@}%
470     }%
471     \version@list
472 \endgroup

```

Put it into the `\alpha@list` with default ‘error’

```

473     \expandafter\gdef\expandafter\alpha@list\expandafter
474     {\alpha@list
475         \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
476     \gdef#4{\no@alphabet@error #5}% fake things :-}

```

Then call the internal setting routine again:

```

477     \SetMathAlphabet@{#1}{#2}{#3}#4#5%
478     \else
479         \@latex@error{Command ‘\string#5’ not defined as a
480             math alphabet}%
481         {Use \noexpand\DeclareMathAlphabet to define it.}%
482     \fi
483 \fi
484 \else
485     \@latex@error{Math version ‘\expandafter\@gobblefour\string#1’
486         is not
487         defined}{You probably misspelled the name of the math
488         version.^^JOr you have to specify an additional package.}%
489 \fi
490 }
491 \@onlypreamble\SetMathAlphabet@

```

`\DeclareMathAlphabet` could do with more checks like allowing single number in #4 lowercase in #4 etc

```

492 \def\DeclareMathAccent#1#2#3#4{%
493     \expandafter\in@\csname sym#3\expandafter\endcsname
494     \expandafter{\group@list}%
495     \ifin@
496         \begingroup
497         \count\z@=#4\relax
498         \count\tw@\count\z@
499         \divide\count\z@\sixt@@n
500         \count@\count\z@
501         \multiply\count@\sixt@@n

```

```

502 \advance\count\tw@-\count@
503 \if\relax\noexpand#1% is command?
504 \edef\reserved@a{\noexpand\in@{\string\mathaccent}{\meaning#1}}%
505 \reserved@a
506 \ifin@
507 \expandafter\set@mathaccent
508 \csname sym#3\endcsname#1#2%
509 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
510 \@font@info{Redefining math accent \string#1}%
511 \else
512 \expandafter\ifx
513 \csname\expandafter@gobble\string#1\endcsname
514 \relax
515 \expandafter\set@mathaccent
516 \csname sym#3\endcsname#1#2%
517 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
518 \else
519 \@latex@error{Command '\string#1' already defined}\@eha
520 \fi
521 \fi
522 \else
523 \@latex@error{Not a command name: '\noexpand#1'}\@eha
524 \fi
525 \endgroup
526 \else
527 \@latex@error{Symbol font '#3' is not defined}\@eha
528 \fi
529 }
530 \@onlypreamble\DeclareMathAccent

```

\set@mathaccent

```

531 \def\set@mathaccent#1#2#3#4{%
532 \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}}
533 \@onlypreamble\set@mathaccent

```

\DeclareMathSymbol

```

534 \def\DeclareMathSymbol#1#2#3#4{%
535 \expandafter\in@\csname sym#3\endcsname\expandafter\endcsname
536 \expandafter{\group@list}%
537 \ifin@
538 \begingroup
539 \count\z@=#4\relax
540 \count\tw@\count\z@
541 \divide\count\z@\sift@n
542 \count@\count\z@
543 \multiply\count@\sift@n
544 \advance\count\tw@-\count@
545 \if\relax\noexpand#1% is command?
546 \edef\reserved@a{\noexpand\in@{\string\mathchar}{\meaning#1}}%
547 \reserved@a
548 \ifin@
549 \expandafter\set@mathsymbol
550 \csname sym#3\endcsname#1#2%
551 {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%

```

```

552         \@font@info{Redefining math symbol \string#1}%
553     \else
554         \expandafter\ifx
555         \csname\expandafter\@gobble\string#1\endcsname
556         \relax
557         \expandafter\set@mathsymbol
558         \csname sym#3\endcsname#1#2%
559         {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
560     \else
561         \@latex@error{Command '\string#1' already defined}\@eha
562     \fi
563 \fi
564 \else
565     \expandafter\set@mathchar
566     \csname sym#3\endcsname#1#2
567     {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
568 \fi
569 \endgroup
570 \else
571     \@latex@error{Symbol font '#3' is not defined}\@eha
572 \fi
573 }
574 \@onlypreamble\DeclareMathSymbol

```

\set@mathchar

```

575 \def\set@mathchar#1#2#3#4{%
576     \global\mathcode'#2="\mathchar@type#3\hexnumber@#1#4\relax}
577 \@onlypreamble\set@mathchar

```

\set@mathsymbol

```

578 \def\set@mathsymbol#1#2#3#4{%
579     \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
580 \@onlypreamble\set@mathsymbol

581 %\def\mathsymbol#1#2#3{%
582 %    \@tempcnta=#3\relax
583 %    \@tempcntb\@tempcnta
584 %    \divide\@tempcnta\sixt@@n
585 %    \count@\@tempcnta
586 %    \multiply\count@\sixt@@n
587 %    \advance\@tempcntb-\count@
588 %    \mathchar"\mathchar@type#1\hexnumber@#2%
589 %        \hexnumber@\@tempcnta\hexnumber@\@tempcntb\relax}
590 %
591 %\def\DeclareMathAlphabetCharacter#1#2#3{%
592 %    \DeclareMathSymbol{#1}7{#2}{#3}

```

\DeclareMathDelimiter

```

593 \def\DeclareMathDelimiter#1{%
594     \if\relax\noexpand#1%
595         \expandafter\@DeclareMathDelimiter
596     \else
597         \expandafter\@xxDeclareMathDelimiter
598     \fi

```

```

599   #1}
600 \onlypreamble\DeclareMathDelimiter

\@xxDeclareMathDelimiter This macro checks if the second arg is a “math type” such as \mathopen. The
undocumented original code didn’t use math types when the delimiter was a sin-
gle letter. For this reason the coding is a bit strange as it tries to support the
undocumented syntax for compatibility reasons.
601 \def\@xxDeclareMathDelimiter#1#2#3#4{%
7 is the default value returned in the case that \mathchar@type is passed some-
thing 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!
602   \begingroup
603   \let\mathalpha\mathord
604   \ifnum7=\mathchar@type{#2}%
605   \endgroup
If this branch is taken we have old syntax (5 arguments).
606   \expandafter\@firstofone
607   \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.
608   \endgroup
609   \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.
610   \expandafter\@firstoftwo
611   \fi
612   {\@xDeclareMathDelimiter#1}{#2}{#3}{#4}}
613 \onlypreamble\@xxDeclareMathDelimiter

\@DeclareMathDelimiter
614 \def\@DeclareMathDelimiter#1#2#3#4#5#6{%
615   \expandafter\in@\csname sym#3\expandafter\endcsname
616   \expandafter{\group@list}%
617   \ifin@
618   \expandafter\in@\csname sym#5\expandafter\endcsname
619   \expandafter{\group@list}%
620   \ifin@
621   \begingroup
622   \count\z@=#4\relax
623   \count\tw@\count\z@
624   \divide\count\z@\sist@on
625   \count@\count\z@
626   \multiply\count@\sist@on
627   \advance\count\tw@-\count@
628   \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
629   %
630   \count\z@=#6\relax
631   \count\tw@\count\z@

```

```

632     \divide\count\z@\sist@@n
633     \count@\count\z@
634     \multiply\count@\sist@@n
635     \advance\count\tw@-\count@
636     \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
637     %
638     \edef\reserved@a{\noexpand\in@{\string\delimiter}{\meaning#1}}%
639     \reserved@a
640     \ifin@
641         \expandafter\set@mathdelimiter
642         \csname sym#3\expandafter\endcsname
643         \csname sym#5\endcsname#1#2%
644         \reserved@c\reserved@d
645         \@font@info{Redeclaring math delimiter \string#1}%
646     \else
647         \expandafter\ifx
648         \csname\expandafter\@gobble\string#1\endcsname
649         \relax
650         \expandafter\set@mathdelimiter
651         \csname sym#3\expandafter\endcsname
652         \csname sym#5\endcsname#1#2%
653         \reserved@c\reserved@d
654     \else
655         \@latex@error{Command ‘\string#1’ already defined}\@eha
656     \fi
657 \fi
658 \endgroup
659 \else
660     \@latex@error{Symbol font ‘#5’ is not defined}\@eha
661 \fi
662 \else
663     \@latex@error{Symbol font ‘#3’ is not defined}\@eha
664 \fi
665 }
666 \@onlypreamble\@DeclareMathDelimiter

```

\@xDeclareMathDelimiter

```

667 \def\@xDeclareMathDelimiter#1#2#3#4#5{%
668     \expandafter\in@\csname sym#2\expandafter\endcsname
669     \expandafter{\group@list}%
670     \ifin@
671         \expandafter\in@\csname sym#4\expandafter\endcsname
672         \expandafter{\group@list}%
673     \ifin@
674         \begingroup
675         \count\z@=#3\relax
676         \count\tw@\count\z@
677         \divide\count\z@\sist@@n
678         \count@\count\z@
679         \multiply\count@\sist@@n
680         \advance\count\tw@-\count@
681         \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
682     %
683     \count\z@=#5\relax

```

```

684         \count\tw@\count\z@
685         \divide\count\z@\sist@@n
686         \count@\count\z@
687         \multiply\count@\sist@@n
688         \advance\count\tw@-\count@
689         \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
690         \expandafter\set@@mathdelimiter
691         \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
692         \reserved@c\reserved@d
693     \endgroup
694 \else
695     \latex@error{Symbol font ‘#4’ is not defined}\@eha
696 \fi
697 \else
698     \latex@error{Symbol font ‘#2’ is not defined}\@eha
699 \fi
700 }
701 \@onlypreamble\@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)

```

702 \def\set@mathdelimiter#1#2#3#4#5#6{%
703     \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
704             \hexnumber@#2#6 }}
705 \@onlypreamble\set@mathdelimiter

```

`\set@@mathdelimiter`

```

706 \def\set@@mathdelimiter#1#2#3#4#5{%
707     \global\delcode‘#3’=\hexnumber@#1#4\hexnumber@#2#5\relax}
708 \@onlypreamble\set@@mathdelimiter

```

`\DeclareMathRadical`

```

709 \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!

```

710     \expandafter\ifx
711         \csname\expandafter\@gobble\string#1\endcsname
712         \relax
713         \let#1\radical
714     \fi
715     \edef\reserved@a{\noexpand\in@{\string\radical}{\meaning#1}}%
716     \reserved@a
717     \ifin@
718         \expandafter\in@\csname sym#2\expandafter\endcsname
719         \expandafter{\group@list}%
720     \ifin@
721         \expandafter\in@\csname sym#4\expandafter\endcsname
722         \expandafter{\group@list}%
723     \ifin@
724         \begingroup
725         \count\z@=#3\relax

```



```

726      \count\tw@\count\z@
727      \divide\count\z@\sist@@n
728      \count@\count\z@
729      \multiply\count@\sist@@n
730      \advance\count\tw@-\count@
731      \edef\reserved@c{%
732        \hexnumber@\count\z@}\hexnumber@\count\tw@}}%
733      \count\z@=#5\relax
734      \count\tw@\count\z@
735      \divide\count\z@\sist@@n
736      \count@\count\z@
737      \multiply\count@\sist@@n
738      \advance\count\tw@-\count@
739      \edef\reserved@d{%
740        \hexnumber@\count\z@}\hexnumber@\count\tw@}}%
  Coded inline instead of using \set@mathradical
741 %      \expandafter\set@mathradical
742 %      \csname sym#2\expandafter\endcsname
743 %      \csname sym#4\endcsname#1%
744 %      \reserved@c\reserved@d
745      \xdef#1{\radical"\expandafter\hexnumber@
746        \csname sym#2\endcsname\reserved@c
747        \expandafter\hexnumber@
748        \csname sym#4\endcsname\reserved@d
749        \relax}%
750      \endgroup
751      \else
752      \@latex@error{Symbol font ‘#4’ is not defined}\@eha
753      \fi
754      \else
755      \@latex@error{Symbol font ‘#2’ is not defined}\@eha
756      \fi
757      \else
758      \@latex@error{Command ‘\string#1’ already defined}\@eha
759      \fi
760 }
761 \@onlypreamble\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

```

762 \let\mathalpha\relax

```

\mathchar@type

```

763 \def\mathchar@type#1{%
764   \ifodd 2#11 #1\else           % is this non-negative number?
765   \ifx#1\mathord 0\else
766   \ifx#1\mathop 1\else
767   \ifx#1\mathbin 2\else
768   \ifx#1\mathrel 3\else

```

```

769         \ifx#1\mathopen 4\else
770         \ifx#1\mathclose 5\else
771         \ifx#1\mathpunct 6\else
772             7%                % anything else is variable ord
773         \fi
774     \fi
775 \fi
776 \fi
777 \fi
778 \fi
779 \fi
780 \fi}
781 \@onlypreamble\mathchar@type

```

\DeclareSymbolFontAlphabet

```

782 \def\DeclareSymbolFontAlphabet#1#2{%
783     \expandafter\DeclareSymbolFontAlphabet@
784     \csname \expandafter\@gobble\string#1\space\endcsname{#2}#1}
785 \@onlypreamble\DeclareSymbolFontAlphabet

```

\DeclareSymbolFontAlphabet@

```

786 \def\DeclareSymbolFontAlphabet@#1#2#3{%
    We use the switch \if@tempswa to decide if we can declare this symbol font
    alphabet.
787     \@tempswatrue
    First check if #2 is known to be a symbol font
788     \expandafter\in@\csname sym#2\expandafter\endcsname
789     \expandafter{\group@list}%
790     \ifin@
    Check if #1 is defined as a math alphabet defined via \DeclareMathAlphabet:
791     \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
792     \ifin@
    If so remove it from the \alpha@list and from all math version macros.
793     \@font@info{Redefining math alphabet \string#3}%
794     \toks@{}%
795     \def\alpha@elt##1##2##3{%
796         \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
797     \alpha@list
798     \xdef\alpha@list{\the\toks@}%
    Now we loop over all versions and remove the math alphabet:
799     \def\version@elt##1{%
800         \begingroup
801         \toks@{}%
802         \def\getanddefine@fonts####1####2{%
803             \addto@hook\toks@{\getanddefine@fonts####1####2}}%
804         \def\install@mathalphabet####1####2{%
805             \ifx####1#1\else
806                 \addto@hook\toks@{\install@mathalphabet
807                     ####1{####2}}\fi}%
808         ##1%

```

```

809         \xdef##1{\the\toks@}%
810     \endgroup
811 }%
812 \version@list
813 \else
    If #3 is not defined as a math alphabet check if it is defined at all:
814     \expandafter\ifx
815     \csname\expandafter\@gobble\string#1\space\endcsname
816     \relax
    If it is undefined, fine otherwise check if it is a math alphabet defined via
    \DeclareSymbolFontAlphabet:
817     \else
818     \edef\reserved@a{%
819         \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
820     \reserved@a
821     \ifin@
822     \@font@info{Redeclaring math alphabet \string#3}%
823     \else
    Since the command #3 is defined to be something which is not a math alphabet
    we have to skip redefining it.
824     \@tempwafalse
825     \@latex@error{Command '\string#3' already defined}\@eha
826     \fi
827 \fi
828 \fi
829 \else
    Since the symbol font is not known we better skip defining this alphabet.
830     \@tempwafalse
831     \@latex@error{Unknown symbol font '#2'}\@eha
832     \fi
833     \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.
834     \def\group@elt##1##2{%
835         \expandafter\ifx\csname sym#2\endcsname##1%
836         \expandafter\reserved@a\string##2\@nil
837         \fi}%
838     \def\reserved@a##1##2/##3\@nil{%
839         \def\reserved@a{##2}}%
840     \group@list
841     \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
842     \edef#1{\the\toks@
843         \noexpand\use@mathgroup
844         \expandafter\noexpand\csname M@\reserved@a\endcsname

```

```

845         \csname sym#2\endcsname}%
846     \def#3{\protect#1}%
847     \fi
848 }
849 \@onlypreamble\DeclareSymbolFontAlphabet@
850 </2kernel | autoload>

```

File s

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.

36 NFSS Initialisation

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`.

36.1 Providing math *versions*

L^AT_EX provides two *versions*. We call them *normal* and *bold*, respectively.

```
1 \DeclareMathVersion{normal}
2 \DeclareMathVersion{bold}
```

Now we define the standard font change commands. We don't allow the use of `\rmfamily` etc. in math mode.

First the changes to another *family*:

```
3 \DeclareRobustCommand\rmfamily
4     {\not@math@alphabet\rmfamily\mathrm
5      \fontfamily\rmdefault\selectfont}
6 \DeclareRobustCommand\sffamily
7     {\not@math@alphabet\sffamily\mathsf
8      \fontfamily\sfddefault\selectfont}
9 \DeclareRobustCommand\ttfamily
10    {\not@math@alphabet\ttfamily\mathtt
11     \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
12 \DeclareRobustCommand\bfseries
13     {\not@math@alphabet\bfseries\mathbf
14      \fontseries\bfdefault\selectfont}
15 \DeclareRobustCommand\mdseries
16     {\not@math@alphabet\mdseries\relax
17      \fontseries\mddefault\selectfont}
18 \DeclareRobustCommand\upshape
19     {\not@math@alphabet\upshape\relax
20      \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
21 \DeclareRobustCommand\slshape
22     {\not@math@alphabet\slshape\relax
23      \fontshape\sldefault\selectfont}
24 \DeclareRobustCommand\scshape
25     {\not@math@alphabet\scshape\relax
26      \fontshape\scdefault\selectfont}
27 \DeclareRobustCommand\itshape
28     {\not@math@alphabet\itshape\mathit}
```

```
29 \fontshape\itdefault\selectfont}
```

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`.

```
30 \DeclareRobustCommand\em
31 {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
32 \upshape \else \itshape \fi}
```

`\not@math@alphabet` This function generates an error message when it is called in math mode. The same function should be defined in `newlfont.sty`.

```
33 \def\not@math@alphabet#1#2{%
34 \relax
35 \ifmmode
36 \@latex@error{Command \noexpand#1invalid in math mode}%
37 {%
38 Please
39 \ifx#2\relax
40 define a new math alphabet^^J%
41 if you want to use a special font in math mode%
42 \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 :-).

```
43 use the math alphabet \noexpand#2instead of
44 the #1command%
45 \fi
46 .
47 }%
48 \fi}
```

Finally we provide two abbreviations to switch to the *L^AT_EX* versions.

```
49 \def\boldmath{\@nomath\boldmath
50 \mathversion{bold}}
51 \def\unboldmath{\@nomath\unboldmath
52 \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 `\glb@settings`.

```
53 \def\math@version{normal}
```

36.2 Miscellaneous

`\newfont` We start by defining a few macros that are part of standard *L^AT_EX*'s user interface.
`\symbol` The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

```
54 \def\newfont#1#2{\ifdefinable#1{\font#1=#2\relax}}
55 \def\symbols#1{\char #1\relax}
```

`\@setfontsize` This abbreviation is used by *L^AT_EX*'s user level size changing commands, such as
`\@setsize` `\large`.

```
56 \def\@setfontsize#1#2#3{\@nomath#1%
```

For the benefit of people relying on keeping the name of the current font command saved in `\@currsiz` we define it. To ensure that `\@setfontsize` keeps being robust we omit this assignment during times where `\protect` differs from `\@typeset@protect`.

```
57 \ifx\protect\@typeset@protect
58 \let\@currsiz#1%
59 \fi
60 \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define `\@setsize` the 209 command

```
61 \*compat
62 \def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
63 \*compat}
```

`\oldstylenums` 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.

```
64 \def\oldstylenums#1{%
65 \begingroup
```

Provide spacing using the interword space of the current font.

```
66 \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).

```
67 \usefont{OML}{\rmdefault}{\f@series}{it}%
68 \mathgroup\symletters #1%
69 \endgroup
70 }
```

`\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`.

```
71 \def\hexnumber@#1{\ifcase\number#1
72 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
73 9\or A\or B\or C\or D\or E\or F\fi}
```

`\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.

```
74 \def\nfss@text#1{{\mbox{#1}}}
```

`\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`.

```

75 %\DeclareRobustCommand\copyright
76 %      {\ooalign{\hfil
77 %        \raise.07ex\hbox{\mdseries\upshape c}\hfil\crr
78 %        \mathhexbox20D}}}}

\normalfont The macro \reset@font is used in LATEX to switch to a standard font, in order to
\reset@font initialize the current font in situations where typesetting is done in a new visual
\p@reset@font context (e.g. in a footnote). We define it here to allow the test for the new LATEX
version above but nevertheless are able to run all kind of mixtures.

The user interface name for \reset@font is \normalfont:

79 \DeclareRobustCommand\normalfont
80      {\usefont\encodingdefault
81      \familydefault
82      \seriesdefault
83      \shapedefault
84      \relax}
85 \let\reset@font\normalfont

We left out the special LATEX 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.

86 \def\not@base#1{\@latex@error
87   {Command \noexpand#1not provided in base LaTeX2e}%
88   {Load the latexsym or the amsfonts package to
89   define this symbol}}
90 \def\mho{\not@base\mho}
91 \def\Join{\not@base\Join}
92 \def\Box{\not@base\Box}
93 \def\Diamond{\not@base\Diamond}
94 \def\leadsto{\not@base\leadsto}
95 \def\sqsubset{\not@base\sqsubset}
96 \def\sqsupset{\not@base\sqsupset}
97 \def\lhd{\not@base\lhd}
98 \def\unlhd{\not@base\unlhd}
99 \def\rhd{\not@base\rhd}
100 \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.

101 \DeclareErrorFont{OT1}{cmr}{m}{n}{10} %% don't modify this setting
102                                     %% overwrite it in fontdef.cfg
103                                     %% if necessary

We now load the customizable parts of NFSS.

104 \ifnum\inputlineno=\m@ne
Still using TEX2. need a configuration file to avoid setting the 8bit characters.
105 \InputIfFileExists{fonttext.cfg}
106      {\typeout{=====^~J%

```



```

107         ^^J%
108         Local config file fonttext.cfg used^^J%
109         ^^J%
110         =====}%
111     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
112 }
113 {\typeout{!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
114     !^^J%
115     ! You MUST use a fonttext.cfg file!^^J%
116     ! As you are still using TeX2!!!!^^J%
117     !^^J%
118     ! See the documentation file tex2.txt^^J%
119     !^^J%
120     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!}%
121 \batchmode \@end}
122 \else

```

With TeX3 can use the standard ltx file if no configuration file exists.

```

123 \InputIfFileExists{fonttext.cfg}
124 {\typeout{=====^^J%
125     ^^J%
126     Local config file fonttext.cfg used^^J%
127     ^^J%
128     =====}%
129     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
130 }
131 {\input{fonttext.ltx}}
132 \fi
133 \let\@addtofilelist\@gobble

```

Ditto for math although I don't think that we will get a lot of customisation :-)

```

134 \InputIfFileExists{fontmath.cfg}
135 {\typeout{=====^^J%
136     ^^J%
137     Local config file fontmath.cfg used^^J%
138     ^^J%
139     =====}%
140     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
141 }
142 {\input{fontmath.ltx}}
143 \let\@addtofilelist\@gobble

```

Then we preload several fonts. This file might be customized *without* changing the behavior of the format (i.e. necessary font definitions will be loaded at runtime if they are not preloaded). This is done in the file `preload.ltx`.

```

144 \InputIfFileExists{preload.cfg}
145 {\typeout{=====^^J%
146     ^^J%
147     Local config file preload.cfg used^^J%
148     ^^J%
149     =====}%
150     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
151 }
152 {\input{preload.ltx}}

```

```

153 \let\@addtofilelist\@gobble

\@acci We also save the values of some accents in \@acci, \@accii and \@acciii so they
\@accii can be restored by a minipage inside a tabbing environment.
\@acciii 154 \let\@acci\' \let\@accii\' \let\@acciii\=

\cal Here were the two old alphabet identifiers.
\mit

```

File t

fontdef.dtx

37 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.

38 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 \LaTeX 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 \LaTeX versions. Thus, before sending in a bug report please try your test file with a \LaTeX 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 \LaTeX installations behave in the same way.

T1	Cork \TeX text encoding
OT1	old \TeX text encoding
U	unknown encoding
OML	old \TeX math letters encoding
OMS	old \TeX math symbols encoding
OMX	old \TeX math extension symbols encoding

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 T_EX 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 T_EX 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 L^AT_EX 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.

39 The docstrip modules

The following modules are used to direct `docstrip` in generating external files:

<code>driver</code>	produce a documentation driver file
<code>text</code>	produce the file <code>fonttext.ltx</code>
<code>math</code>	produce the file <code>fontmath.ltx</code>
<code>cfgtext</code>	produce a dummy <code>fonttext.cfg</code> file
<code>cfgmath</code>	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}}
```

40 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.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 \GetFileInfo{fontdef.dtx}
4 \begin{document}
5   \DocInput{fontdef.dtx}
6 \end{document}
7 </driver>
```

41 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}
```

41.1 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 `ltoutenc.dtx`.

By convention, text encoding specific declarations, including the declaration `\DeclareFontEncoding`, 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 {t1enc.def}
12 \input {ot1enc.def}      % <- should come after T1 for speed
13 \input {omsenc.def}
```

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.

```
14 \fontencoding{OT1}
```

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!

```
15 \DeclareFontEncodingDefaults{}{}
```

Then we define the default substitution for every encoding. This release of $\text{\LaTeX 2}_{\epsilon}$ 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.

```
16 \DeclareFontSubstitution{T1}{cmr}{m}{n}
17 \DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

For every encoding declaration, $\text{\LaTeX 2}_{\epsilon}$ 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. $\text{\LaTeX 2}_{\epsilon}$ 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 $\text{\LaTeX 2}_{\epsilon}$ 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.

```
18 \begingroup
19 \nfss@catcodes
20 \input {t1cmr.fd}
21 \input {ot1cmr.fd}
22 \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 \TeX installation, while the amount of main memory is not a limiting factor at most installations.)

```
23 \begingroup
24 \nfss@catcodes
25 \input {ot1cmss.fd}
26 \input {ot1cmtt.fd}
27 \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.

```
28 \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.

41.2 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.

```
\rmdefault The following three definitions set up the meaning for \rmfamily, \sffamily, and
\sffdefault \ttfamily.
\ttdefault 29 \newcommand\rmdefault{cmr}
30 \newcommand\sffdefault{cmss}
31 \newcommand\ttdefault{cmtt}
```

```

\bfdefault Series changing commands are inuenced by the following hooks.
\mddefault 32 \newcommand\bfdefault{bx}
           33 \newcommand\mddefault{m}

\itdefault Shape changing commands use the following hooks.
\sldefault 34 \newcommand\itdefault{it}
\scdefault 35 \newcommand\sldefault{sl}
\updefault 36 \newcommand\scdefault{sc}
           37 \newcommand\updefault{n}

\encodingdefault Finally we have the hooks that describe the behaviour of the \normalfont com-
\familydefault mand. To stay portable, the definition of \encodingdefault should not be
\seriesdefault changed and should match the setting above for \fontencoding. All other values
\shapedefault can be set according to your taste.
           38 \newcommand\encodingdefault{OT1}
           39 \newcommand\familydefault{\rmdefault}
           40 \newcommand\seriesdefault{\mddefault}
           41 \newcommand\shapedefault{\updefault}

           This finishes the low-level setup in fonttext.ltx.
           42 </text>

```

42 The fontmath.ltx file

The identification is done earlier on with a `\ProvidesFile` declaration.

```

43 <*math>
44 \typeout{=== Don't modify this file, use a .cfg file instead ===^^J}

```

42.1 The font encodings used

```

45 \DeclareFontEncoding{OML}{-}{-}
46 \DeclareFontEncoding{OMS}{-}{-}
47 \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- \LaTeX . 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.

```

48 \DeclareFontEncoding{U}{-}{\noaccents@}

```

The encodings for math are next:

```

49 \DeclareFontSubstitution{OML}{cmm}{m}{it}
50 \DeclareFontSubstitution{OMS}{cmsy}{m}{n}
51 \DeclareFontSubstitution{OMX}{cmex}{m}{n}
52 \DeclareFontSubstitution{U}{cmr}{m}{n}

53 \begingroup
54 \nfss@catcodes
55 \input {omlcmm.fd}
56 \input {omscmsy.fd}
57 \input {omxcmex.fd}
58 \input {ucmr.fd}

```

59 \endgroup

42.1.1 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.

```
60 \DeclareSymbolFont{operators}    {OT1}{cmr}{m}{n}
61 \DeclareSymbolFont{letters}      {OML}{cmm}{m}{it}
62 \DeclareSymbolFont{symbols}      {OMS}{cmsy}{m}{n}
63 \DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}

64 \SetSymbolFont{operators}{bold}{OT1}{cmr}{bx}{n}
65 \SetSymbolFont{letters}{bold}{OML}{cmm}{b}{it}
66 \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.

```
67 \DeclareSymbolFontAlphabet{\mathrm}    {operators}
68 \DeclareSymbolFontAlphabet{\mathnormal}{letters}
69 \DeclareSymbolFontAlphabet{\mathcal}    {symbols}
70 \DeclareMathAlphabet          {\mathbf}{OT1}{cmr}{bx}{n}
71 \DeclareMathAlphabet          {\mathsf}{OT1}{cmss}{m}{n}
72 \DeclareMathAlphabet          {\mathit}{OT1}{cmr}{m}{it}
73 \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.

```
74 \SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
75 \SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}
```

42.2 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).

```
76 \DeclareMathSizes{5}{5}{5}{5}
77 \DeclareMathSizes{6}{6}{5}{5}
78 \DeclareMathSizes{7}{7}{5}{5}
79 \DeclareMathSizes{8}{8}{6}{5}
80 \DeclareMathSizes{9}{9}{6}{5}
81 \DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
82 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
83 \DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
```



```

84 \DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
85 \DeclareMathSizes{\@xvipt}{\@xvipt}{\@xipt}{\@xpt}
86 \DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xipt}
87 \DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xvipt}

```

42.3 The math symbol assignments

We start by setting up math codes for most of the characters typed in directly from the keyboard. Most of them are normally already setup up in the same way by Init_{TeX} . However, we repeat them here to have a complete setup which can be exchanged with another if desired.

42.3.1 The letters

```

88 \DeclareMathSymbol{a}{\mathalpha}{letters}{‘a}
89 \DeclareMathSymbol{b}{\mathalpha}{letters}{‘b}
90 \DeclareMathSymbol{c}{\mathalpha}{letters}{‘c}
91 \DeclareMathSymbol{d}{\mathalpha}{letters}{‘d}
92 \DeclareMathSymbol{e}{\mathalpha}{letters}{‘e}
93 \DeclareMathSymbol{f}{\mathalpha}{letters}{‘f}
94 \DeclareMathSymbol{g}{\mathalpha}{letters}{‘g}
95 \DeclareMathSymbol{h}{\mathalpha}{letters}{‘h}
96 \DeclareMathSymbol{i}{\mathalpha}{letters}{‘i}
97 \DeclareMathSymbol{j}{\mathalpha}{letters}{‘j}
98 \DeclareMathSymbol{k}{\mathalpha}{letters}{‘k}
99 \DeclareMathSymbol{l}{\mathalpha}{letters}{‘l}
100 \DeclareMathSymbol{m}{\mathalpha}{letters}{‘m}
101 \DeclareMathSymbol{n}{\mathalpha}{letters}{‘n}
102 \DeclareMathSymbol{o}{\mathalpha}{letters}{‘o}
103 \DeclareMathSymbol{p}{\mathalpha}{letters}{‘p}
104 \DeclareMathSymbol{q}{\mathalpha}{letters}{‘q}
105 \DeclareMathSymbol{r}{\mathalpha}{letters}{‘r}
106 \DeclareMathSymbol{s}{\mathalpha}{letters}{‘s}
107 \DeclareMathSymbol{t}{\mathalpha}{letters}{‘t}
108 \DeclareMathSymbol{u}{\mathalpha}{letters}{‘u}
109 \DeclareMathSymbol{v}{\mathalpha}{letters}{‘v}
110 \DeclareMathSymbol{w}{\mathalpha}{letters}{‘w}
111 \DeclareMathSymbol{x}{\mathalpha}{letters}{‘x}
112 \DeclareMathSymbol{y}{\mathalpha}{letters}{‘y}
113 \DeclareMathSymbol{z}{\mathalpha}{letters}{‘z}

114 \DeclareMathSymbol{A}{\mathalpha}{letters}{‘A}
115 \DeclareMathSymbol{B}{\mathalpha}{letters}{‘B}
116 \DeclareMathSymbol{C}{\mathalpha}{letters}{‘C}
117 \DeclareMathSymbol{D}{\mathalpha}{letters}{‘D}
118 \DeclareMathSymbol{E}{\mathalpha}{letters}{‘E}
119 \DeclareMathSymbol{F}{\mathalpha}{letters}{‘F}
120 \DeclareMathSymbol{G}{\mathalpha}{letters}{‘G}
121 \DeclareMathSymbol{H}{\mathalpha}{letters}{‘H}
122 \DeclareMathSymbol{I}{\mathalpha}{letters}{‘I}
123 \DeclareMathSymbol{J}{\mathalpha}{letters}{‘J}
124 \DeclareMathSymbol{K}{\mathalpha}{letters}{‘K}
125 \DeclareMathSymbol{L}{\mathalpha}{letters}{‘L}
126 \DeclareMathSymbol{M}{\mathalpha}{letters}{‘M}

```

```

127 \DeclareMathSymbol{N}{\mathalpha}{letters}{'N}
128 \DeclareMathSymbol{O}{\mathalpha}{letters}{'O}
129 \DeclareMathSymbol{P}{\mathalpha}{letters}{'P}
130 \DeclareMathSymbol{Q}{\mathalpha}{letters}{'Q}
131 \DeclareMathSymbol{R}{\mathalpha}{letters}{'R}
132 \DeclareMathSymbol{S}{\mathalpha}{letters}{'S}
133 \DeclareMathSymbol{T}{\mathalpha}{letters}{'T}
134 \DeclareMathSymbol{U}{\mathalpha}{letters}{'U}
135 \DeclareMathSymbol{V}{\mathalpha}{letters}{'V}
136 \DeclareMathSymbol{W}{\mathalpha}{letters}{'W}
137 \DeclareMathSymbol{X}{\mathalpha}{letters}{'X}
138 \DeclareMathSymbol{Y}{\mathalpha}{letters}{'Y}
139 \DeclareMathSymbol{Z}{\mathalpha}{letters}{'Z}

```

42.3.2 The digits

```

140 \DeclareMathSymbol{0}{\mathalpha}{operators}{'0}
141 \DeclareMathSymbol{1}{\mathalpha}{operators}{'1}
142 \DeclareMathSymbol{2}{\mathalpha}{operators}{'2}
143 \DeclareMathSymbol{3}{\mathalpha}{operators}{'3}
144 \DeclareMathSymbol{4}{\mathalpha}{operators}{'4}
145 \DeclareMathSymbol{5}{\mathalpha}{operators}{'5}
146 \DeclareMathSymbol{6}{\mathalpha}{operators}{'6}
147 \DeclareMathSymbol{7}{\mathalpha}{operators}{'7}
148 \DeclareMathSymbol{8}{\mathalpha}{operators}{'8}
149 \DeclareMathSymbol{9}{\mathalpha}{operators}{'9}

```

42.3.3 Punctuation, brace, etc. keys

```

150 \DeclareMathSymbol{!}{\mathclose}{operators}{"21}
151 \DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
152 \DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
153 \DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
154 \DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
155 \DeclareMathSymbol{.}{\mathord}{letters}{"3A}
156 \DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
157 \DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
158 \DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
159 \DeclareMathSymbol{?}{\mathclose}{operators}{"3F}

```

The following symbols are defined as delimiters below which automatically defines them as math symbols.

```

160 %\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
161 %\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
162 %\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
163 %\DeclareMathSymbol{[}{\mathopen}{operators}{"5B}
164 %\DeclareMathSymbol{]}{\mathclose}{operators}{"5D}
165 %\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}
166 %\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
167 %\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}

```

Should all of the following being activated by default? Probably not.

```

168 %\DeclareMathSymbol{\}{\mathopen}{symbols}{"66}
169 %\DeclareMathSymbol{\'}{\mathclose}{symbols}{"67}
170 %\DeclareMathSymbol{\}{\mathord}{symbols}{"6E} % \backslash
171 \mathcode'\ =8000 % \space
172 \mathcode'\ ' =8000 % ^\prime

```

```
173 \mathcode'\_="8000 % \_
```

42.3.4 Delimitercodes for characters

[to be completed]

Finally, `InitEX` sets all `\delcode` values to -1, except `\delcode'.`=0

```
174 \DeclareMathDelimiter{\}\mathopen\operators{"28}{largesymbols}{"00}
175 \DeclareMathDelimiter{\}\mathclose\operators{"29}{largesymbols}{"01}
176 \DeclareMathDelimiter{\}\mathopen\operators{"5B}{largesymbols}{"02}
177 \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.

```
178 \DeclareMathDelimiter{<}\mathopen\symbols{"68}{largesymbols}{"0A}
179 \DeclareMathDelimiter{>}\mathclose\symbols{"69}{largesymbols}{"0B}
180 \DeclareMathSymbol{<}\mathrel\letters{"3C}
181 \DeclareMathSymbol{>}\mathrel\letters{"3E}
```

And here is another case where the non-delimiter version produces a glyph different from the delimiter version.

```
182 \DeclareMathDelimiter{/}\mathord\operators{"2F}{largesymbols}{"0E}
183 \DeclareMathSymbol{/}\mathord\letters{"3D}

184 \DeclareMathDelimiter{|}\mathord\symbols{"6A}{largesymbols}{"0C}

185 \expandafter\DeclareMathDelimiter\@backslashchar
186 \mathord\symbols{"6E}{largesymbols}{"0F}
```

N.B. { and } should NOT get delcodes; otherwise parameter grouping fails!

42.4 Symbols accessed via control sequences

42.4.1 Greek letters

```
187 \DeclareMathSymbol{\alpha}\mathord\letters{"0B}
188 \DeclareMathSymbol{\beta}\mathord\letters{"0C}
189 \DeclareMathSymbol{\gamma}\mathord\letters{"0D}
190 \DeclareMathSymbol{\delta}\mathord\letters{"0E}
191 \DeclareMathSymbol{\epsilon}\mathord\letters{"0F}
192 \DeclareMathSymbol{\zeta}\mathord\letters{"10}
193 \DeclareMathSymbol{\eta}\mathord\letters{"11}
194 \DeclareMathSymbol{\theta}\mathord\letters{"12}
195 \DeclareMathSymbol{\iota}\mathord\letters{"13}
196 \DeclareMathSymbol{\kappa}\mathord\letters{"14}
197 \DeclareMathSymbol{\lambda}\mathord\letters{"15}
198 \DeclareMathSymbol{\mu}\mathord\letters{"16}
199 \DeclareMathSymbol{\nu}\mathord\letters{"17}
200 \DeclareMathSymbol{\xi}\mathord\letters{"18}
201 \DeclareMathSymbol{\pi}\mathord\letters{"19}
202 \DeclareMathSymbol{\rho}\mathord\letters{"1A}
203 \DeclareMathSymbol{\sigma}\mathord\letters{"1B}
204 \DeclareMathSymbol{\tau}\mathord\letters{"1C}
205 \DeclareMathSymbol{\upsilon}\mathord\letters{"1D}
206 \DeclareMathSymbol{\phi}\mathord\letters{"1E}
```

```

207 \DeclareMathSymbol{\chi}{\mathord}{letters}{1F}
208 \DeclareMathSymbol{\psi}{\mathord}{letters}{20}
209 \DeclareMathSymbol{\omega}{\mathord}{letters}{21}
210 \DeclareMathSymbol{\varepsilon}{\mathord}{letters}{22}
211 \DeclareMathSymbol{\vartheta}{\mathord}{letters}{23}
212 \DeclareMathSymbol{\varpi}{\mathord}{letters}{24}
213 \DeclareMathSymbol{\varrho}{\mathord}{letters}{25}
214 \DeclareMathSymbol{\varsigma}{\mathord}{letters}{26}
215 \DeclareMathSymbol{\varphi}{\mathord}{letters}{27}
216 \DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{00}
217 \DeclareMathSymbol{\Delta}{\mathalpha}{operators}{01}
218 \DeclareMathSymbol{\Theta}{\mathalpha}{operators}{02}
219 \DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{03}
220 \DeclareMathSymbol{\Xi}{\mathalpha}{operators}{04}
221 \DeclareMathSymbol{\Pi}{\mathalpha}{operators}{05}
222 \DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{06}
223 \DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{07}
224 \DeclareMathSymbol{\Phi}{\mathalpha}{operators}{08}
225 \DeclareMathSymbol{\Psi}{\mathalpha}{operators}{09}
226 \DeclareMathSymbol{\Omega}{\mathalpha}{operators}{0A}

```

42.4.2 Ordinary symbols

```

227 \DeclareMathSymbol{\aleph}{\mathord}{symbols}{40}
228 \def\hbar{{\mathchar'26\mkern-9mu\h}}
229 \DeclareMathSymbol{\imath}{\mathord}{letters}{7B}
230 \DeclareMathSymbol{\jmath}{\mathord}{letters}{7C}
231 \DeclareMathSymbol{\ell}{\mathord}{letters}{60}
232 \DeclareMathSymbol{\wp}{\mathord}{letters}{7D}
233 \DeclareMathSymbol{\Re}{\mathord}{symbols}{3C}
234 \DeclareMathSymbol{\Im}{\mathord}{symbols}{3D}
235 \DeclareMathSymbol{\partial}{\mathord}{letters}{40}
236 \DeclareMathSymbol{\infty}{\mathord}{symbols}{31}
237 \DeclareMathSymbol{\prime}{\mathord}{symbols}{30}
238 \DeclareMathSymbol{\emptyset}{\mathord}{symbols}{3B}
239 \DeclareMathSymbol{\nabla}{\mathord}{symbols}{72}
240 \def\surd{{\mathchar"1270}}
241 \DeclareMathSymbol{\top}{\mathord}{symbols}{3E}
242 \DeclareMathSymbol{\bot}{\mathord}{symbols}{3F}
243 \def\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crrc
244 \not\mathrel{\mkern14mu}\crrc
245 \noalign{\nointerlineskip}
246 \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crrc}}}}
247 \DeclareMathSymbol{\triangle}{\mathord}{symbols}{34}
248 \DeclareMathSymbol{\forall}{\mathord}{symbols}{38}
249 \DeclareMathSymbol{\exists}{\mathord}{symbols}{39}
250 \DeclareMathSymbol{\neg}{\mathord}{symbols}{3A}
251 \let\not=\neg
252 \DeclareMathSymbol{\flat}{\mathord}{letters}{5B}
253 \DeclareMathSymbol{\natural}{\mathord}{letters}{5C}
254 \DeclareMathSymbol{\sharp}{\mathord}{letters}{5D}
255 \DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{7C}
256 \DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{7D}
257 \DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{7E}
258 \DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{7F}

```

42.4.3 Large Operators

```

259 \DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}
260 \DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{"57}
261 \DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{"56}
262 \DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{"55}
263 \DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{"54}
264 \DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{"53}
265 \DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
266 \def\int{\intop\nolimits}
267 \DeclareMathSymbol{\prod}{\mathop}{largesymbols}{"51}
268 \DeclareMathSymbol{\sum}{\mathop}{largesymbols}{"50}
269 \DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{"4E}
270 \DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{"4C}
271 \DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{"4A}
272 \DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{"48}
273 \def\oint{\ointop\nolimits}
274 \DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{"46}
275 \DeclareMathSymbol{\smallint}{\mathop}{symbols}{"73}

```

42.4.4 Binary symbols

```

276 \DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{"2F}
277 \DeclareMathSymbol{\triangleright}{\mathbin}{letters}{"2E}
278 \DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{"34}
279 \DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{"35}
280 \let \varbigtriangledown \bigtriangledown
281 \let \varbigtriangleup \bigtriangleup

```

These last two synonyms are needed because the `stamyr` package redefines them as Operators.

```

282 \DeclareMathSymbol{\wedge}{\mathbin}{symbols}{"5E}
283 \let\land=\wedge
284 \DeclareMathSymbol{\vee}{\mathbin}{symbols}{"5F}
285 \let\lor=\vee
286 \DeclareMathSymbol{\cap}{\mathbin}{symbols}{"5C}
287 \DeclareMathSymbol{\cup}{\mathbin}{symbols}{"5B}
288 \DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{"7A}
289 \DeclareMathSymbol{\dagger}{\mathbin}{symbols}{"79}
290 \DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{"75}
291 \DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{"74}
292 \DeclareMathSymbol{\uplus}{\mathbin}{symbols}{"5D}
293 \DeclareMathSymbol{\amalg}{\mathbin}{symbols}{"71}
294 \DeclareMathSymbol{\diamond}{\mathbin}{symbols}{"05}
295 \DeclareMathSymbol{\bullet}{\mathbin}{symbols}{"0F}
296 \DeclareMathSymbol{\wr}{\mathbin}{symbols}{"6F}
297 \DeclareMathSymbol{\div}{\mathbin}{symbols}{"04}
298 \DeclareMathSymbol{\odot}{\mathbin}{symbols}{"0C}
299 \DeclareMathSymbol{\oslash}{\mathbin}{symbols}{"0B}
300 \DeclareMathSymbol{\otimes}{\mathbin}{symbols}{"0A}
301 \DeclareMathSymbol{\ominus}{\mathbin}{symbols}{"09}
302 \DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
303 \DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
304 \DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
305 \DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
306 \DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}

```

```

307 \DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
308 \DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
309 \DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
310 \DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
311 \DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}

```

42.4.5 Relations

```

312 \DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
313 \DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
314 \DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
315 \DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
316 \DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
317 \DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
318 \DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
319 \DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
320 \DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
321 \DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{"2D}
322 \DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
323 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
324 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
325 \DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
326 \def\neq{\not=} \let\ne=\neq
327 \DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
328 \let\le=\leq
329 \DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}
330 \let\ge=\geq
331 \DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
332 \DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
333 \DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
334 \DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
335 \DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
336 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"1B}
337 \DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
338 \DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
339 \DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
340 \DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
341 \DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}
342 \let\owns=\ni
343 \DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
344 \DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
345 \DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
346 \DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"24}
347 \DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"20}
348 \let\gets=\Leftarrow
349 \DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}
350 \let\to=\rightarrow
351 \DeclareMathSymbol{\mapsto}{\mathrel}{symbols}{"37}
352 \def\mapsto{\mapstochar\rightarrow}
353 \DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
354 \DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
355 \DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
356 \DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
357 \DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
358 \DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}

```

```

359 \DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
360 \DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
361 \DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
362 \DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
363 \DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{"2B}

```

Here cometh much proigate 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.

```

364 \DeclareRobustCommand
365   \cong{\mathrel{\mathpalette\@vereq\sim}} % congruence sign
366 \def\@vereq#1#2{\lower.5\p@\vbox{\lineskiplimit\maxdimen\lineskip-.5\p@
367   \ialign{$\m@th#1\hfil#\hfil$\crrc#2\crrc=\crrc}}}
368 \DeclareRobustCommand
369   \notin{\mathrel{\m@th\mathpalette\c@ncel\in}}
370 \def\c@ncel#1#2{\m@th\oalign{$\hfil#1\mkern1mu\hfil$\crrc$#1#2$}}
371 \DeclareRobustCommand
372   \rightleftharpoons{\mathrel{\mathpalette\rlh@{}}}
373 \def\rlh@#1{\vcenter{\m@th\hbox{\oalign{\raise2pt
374   \hbox{$$#1\rightharpoonup$}\crrc
375   $$#1\leftharpoondown$}}}}
376 \DeclareRobustCommand
377   \doteq{\buildrel\textstyle.\over=}

```

42.4.6 Arrows

```

378 \DeclareRobustCommand
379   \joinrel{\mathrel{\mkern-3mu}}
380 \DeclareRobustCommand
381   \relbar{\mathrel{\smash-}} % \smash, because -
382                               % 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).

```

383 \DeclareRobustCommand
384   \Relbar{\mathrel{=}}
385 \DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
386 \def\hookrightarrow{\lhook\joinrel\rightarrow}
387 \DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
388 \def\hookleftarrow{\leftarrow\joinrel\rhook}
389 \DeclareRobustCommand
390   \bowtie{\mathrel{\triangleright\joinrel\mathrel{\triangleleft}}}
391 \DeclareRobustCommand
392   \models{\mathrel{|\}\joinrel\Relbar}
393 \DeclareRobustCommand
394   \Longrightarrow{\Relbar\joinrel\rightarrow}

```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make then robust.

```

395 \DeclareRobustCommand\longrightarrow
396     {\relbar\joinrel\rightarrow}
397 \DeclareRobustCommand\longleftarrow
398     {\leftarrow\joinrel\relbar}
399 \DeclareRobustCommand
400     \Longleftarrow{\Leftarrow\joinrel\Relbar}
401 \DeclareRobustCommand
402     \longmapsto{\mapstochar\longrightarrow}
403 \DeclareRobustCommand
404     \longlefttrightarrow{\leftarrow\joinrel\rightarrow}
405 \DeclareRobustCommand
406     \Longlefttrightarrow{\Leftarrow\joinrel\Rightarrow}
407 \DeclareRobustCommand
408     \iff{\;\Longlefttrightarrow\;}

```

42.4.7 Punctuation symbols

```

409 \DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
410 \DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
411 \DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}

    This is commented out, since \ldots is now defined in ltoutenc.dtx.
412 %\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
413 %\DeclareRobustCommand\ldots
414 %    {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots\,$}\fi}
415 \DeclareRobustCommand
416     \cdots{\mathinner{\cdotp\cdotp\cdotp}}
417 \DeclareRobustCommand
418     \vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
419         \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
420 \DeclareRobustCommand
421     \ddots{\mathinner{\mkern1mu\raise7\p@
422         \vbox{\kern7\p@\hbox{.}\mkern2mu
423             \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}}

```

42.4.8 Math accents

```

424 \DeclareMathAccent{\acute}{\mathalpha}{operators}{"13}
425 \DeclareMathAccent{\grave}{\mathalpha}{operators}{"12}
426 \DeclareMathAccent{\ddot}{\mathalpha}{operators}{"7F}
427 \DeclareMathAccent{\tilde}{\mathalpha}{operators}{"7E}
428 \DeclareMathAccent{\bar}{\mathalpha}{operators}{"16}
429 \DeclareMathAccent{\breve}{\mathalpha}{operators}{"15}
430 \DeclareMathAccent{\check}{\mathalpha}{operators}{"14}
431 \DeclareMathAccent{\hat}{\mathalpha}{operators}{"5E}
432 \DeclareMathAccent{\vec}{\mathord}{letters}{"7E}
433 \DeclareMathAccent{\dot}{\mathalpha}{operators}{"5F}
434 \DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{"65}
435 \DeclareMathAccent{\widehat}{\mathord}{largesymbols}{"62}

```

For some reason plain T_EX 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:

```

436 \DeclareMathAccent{\mathring}{\mathalpha}{operators}{"17}

```

42.4.9 Radicals


```
437 \DeclareMathRadical{\sqrtsign}{symbols}{"70}{largesymbols}{"70}
```

42.4.10 Over and under something, etc

```
438 \def\overrightarrow#1{\vbox{\m@th\ialign{##\crrc
439   \rightarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}
440   $\hfil\displaystyle{#1}\hfil$\crrc}}}%
441 \def\overleftarrow#1{\vbox{\m@th\ialign{##\crrc
442   \leftarrowfill\crrc\noalign{\kern-\p@\nointerlineskip}%
443   $\hfil\displaystyle{#1}\hfil$\crrc}}}%
444 \def\overbrace#1{\mathop{\vbox{\m@th\ialign{##\crrc\noalign{\kern3\p@}%
445   \downbracefill\crrc\noalign{\kern3\p@\nointerlineskip}%
446   $\hfil\displaystyle{#1}\hfil$\crrc}}}\limits}%
447 \def\underbrace#1{\mathop{\vtop{\m@th\ialign{##\crrc
448   $\hfil\displaystyle{#1}\hfil$\crrc
449   \noalign{\kern3\p@\nointerlineskip}%
450   \upbracefill\crrc\noalign{\kern3\p@}}}\limits}}
```

(quite a waste of tokens, IMHO — Frank)

```
451 \def\skew#1#2#3{\muskip\z@#1mu\divide\muskip\z@ \tw@ \mkern\muskip\z@
452   #2{\mkern-\muskip\z@#3}\mkern\muskip\z@}\mkern-\muskip\z@}{}}
453 \def\rightarrowfill{$\m@th\smash-\mkern-7mu%
454   \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
455   \mkern-7mu\mathord\rightarrow$}%
456 \def\leftarrowfill{$\m@th\mathord\leftarrow\mkern-7mu%
457   \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
458   \mkern-7mu\smash-$}%
459 \DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{"7A}
460 \DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{"7B}
461 \DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{"7C}
462 \DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{"7D}
463 \def\downbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
464   \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
465   \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}%
466 \def\upbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
467   \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
468   \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}
```

42.4.11 Delimiters

```
469 \DeclareMathDelimiter{\lmoustache} % top from (, bottom from )
470   {\mathopen}{largesymbols}{"7A}{largesymbols}{"40}
471 \DeclareMathDelimiter{\rmoustache} % top from ), bottom from (
472   {\mathclose}{largesymbols}{"7B}{largesymbols}{"41}
473 \DeclareMathDelimiter{\arrowvert} % arrow without arrowheads
474   {\mathord}{symbols}{"6A}{largesymbols}{"3C}
475 \DeclareMathDelimiter{\Arrowvert} % double arrow without arrowheads
476   {\mathord}{symbols}{"6B}{largesymbols}{"3D}
477 \DeclareMathDelimiter{\Vert}
478   {\mathord}{symbols}{"6B}{largesymbols}{"0D}
479 \let\lvert=\Vert
480 \DeclareMathDelimiter{\vert}
481   {\mathord}{symbols}{"6A}{largesymbols}{"0C}
482 \DeclareMathDelimiter{\uparrow}
483   {\mathrel}{symbols}{"22}{largesymbols}{"78}
484 \DeclareMathDelimiter{\downarrow}
```

```

485 {\mathrel}{symbols}{"23}{largesymbols}{"79}
486 \DeclareMathDelimiter{\updownarrow}
487 {\mathrel}{symbols}{"6C}{largesymbols}{"3F}
488 \DeclareMathDelimiter{\Uparrow}
489 {\mathrel}{symbols}{"2A}{largesymbols}{"7E}
490 \DeclareMathDelimiter{\Downarrow}
491 {\mathrel}{symbols}{"2B}{largesymbols}{"7F}
492 \DeclareMathDelimiter{\Updownarrow}
493 {\mathrel}{symbols}{"6D}{largesymbols}{"77}
494 \DeclareMathDelimiter{\backslash} % for double coset G\backslash H
495 {\mathord}{symbols}{"6E}{largesymbols}{"0F}
496 \DeclareMathDelimiter{\rangle}
497 {\mathclose}{symbols}{"69}{largesymbols}{"0B}
498 \DeclareMathDelimiter{\langle}
499 {\mathopen}{symbols}{"68}{largesymbols}{"0A}
500 \DeclareMathDelimiter{\rbrace}
501 {\mathclose}{symbols}{"67}{largesymbols}{"09}
502 \DeclareMathDelimiter{\lbrace}
503 {\mathopen}{symbols}{"66}{largesymbols}{"08}
504 \DeclareMathDelimiter{\rceil}
505 {\mathclose}{symbols}{"65}{largesymbols}{"07}
506 \DeclareMathDelimiter{\lceil}
507 {\mathopen}{symbols}{"64}{largesymbols}{"06}
508 \DeclareMathDelimiter{\rfloor}
509 {\mathclose}{symbols}{"63}{largesymbols}{"05}
510 \DeclareMathDelimiter{\lfloor}
511 {\mathopen}{symbols}{"62}{largesymbols}{"04}

```

`\lgroup` There are three plain TeX delimiters which are not fully supported by NFSS,
`\rgroup` since they partly point into a bold cmr font. Allocating a full symbol font, just
`\bracevert` 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.)

```

512 \DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
513 {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
514 \DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
515 {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
516 \DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
517 {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}

```

42.5 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 518 \DeclareMathSymbol{\mathparagraph}{\mathord}{symbols}{"7B}
\mathdollar 519 \DeclareMathSymbol{\mathsection}{\mathord}{symbols}{"78}
\mathsterling 520 \DeclareMathSymbol{\mathdollar}{\mathord}{operators}{"24}
\mathunderscore

```

```

521 \def\mathsterling{\mathit{\mathchar"7024}}
522 \def\mathunderscore{\kern.06em\vbox{\hrule\@width.3em}}

\mathellipsis This is plain TEX's \ldots.
523 \def\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%
```

42.6 Other special functions and parameters

42.6.1 Biggggg

```

524 \def\big#1{{\hbox{$\left#1\ vbox to8.5\p@\right.\n@space$}}}
525 \def\Big#1{{\hbox{$\left#1\ vbox to11.5\p@\right.\n@space$}}}
526 \def\bigg#1{{\hbox{$\left#1\ vbox to14.5\p@\right.\n@space$}}}
527 \def\Bigg#1{{\hbox{$\left#1\ vbox to17.5\p@\right.\n@space$}}}
528 \def\n@space{\null\delimiterspace\z@\m@th}
```

42.6.2 The log-like functions

`\operator@font` The `\operator@font` determines the symbol font used for log-like functions.

```

529 \def\operator@font{\mathgroup\symoperators}
```

42.6.3 Parameters

```

530 \thinmuskip=3mu
531 \medmuskip=4mu plus 2mu minus 4mu
532 \thickmuskip=5mu plus 5mu
```

This finishes the low-level setup in `fontmath.ltx`.

```

533 </math>
```

43 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.

```

534 <*cfgtext | cfgmath | cfgprel>
535 %%
536 %%
537 %%
538 %% Load the standard setup:
539 %%
540 <+cfgtext>\input{fonttext.ltx}
541 <+cfgmath>\input{fontmath.ltx}
542 <+cfgprel>\input{preload.ltx}
543 %%
544 %% Small changes could go here; see documentation in cfgguide.tex for
545 %% allowed modifications.
546 %%
547 %% In particular it is not allowed to misuse this configuration file
548 %% to modify internal LaTeX commands!
549 %%
550 %% If you use this file as the basis for configuration please change
551 %% the \ProvidesFile lines to clearly identify your modification, e.g.,
552 %%
553 <+cfgtext>%% \ProvidesFile{fonttext.cfg}[2001/06/01
554 <+cfgmath>%% \ProvidesFile{fonttext.cfg}[2001/06/01
```

```

555 <+cfgprel>%%    \ProvidesFile{preload.cfg}[2001/06/01
556 %%              Customised local font setup]
557 %%
558 %%
559 </cfgtext | cfgmath | cfgprel>

```

File u

preload.dtx

44 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by $\text{\LaTeX 2}_{\varepsilon}$). 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>lfonts.tex</code>
preload.ltx	The standard selection of preloads
cmpreloa.xpt	preload of CM fonts for 10pt document size
cmpreloa.xip	preload of CM fonts for 11pt document size
cmpreloa.xii	preload of CM fonts for 12pt document size
dcpreloa.xpt	preload of DC fonts for 10pt size
dcpreloa.xip	preload of DC fonts for 11pt size
dcpreloa.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.

45 Customization

You can customize the preloaded fonts in your $\text{\LaTeX 2}_{\varepsilon}$ 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** $\text{\LaTeX 2}_{\varepsilon}$ 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 \LaTeX 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.

46 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
xipt	produce 12pt preloads
ori	produce preloads similar to old <code>lfonts.tex</code>
tex	produce <code>preload.ltx</code>

A typical DOCSTRIP command file would then have entries like:

```
\generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

47 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.

```
1 <*driver>
2 \documentclass{ltxdoc}
3 %\OnlyDescription % comment out for implementation details
4 \begin{document}
5   \DocInput{preload.dtx}
6 \end{document}
7 </driver>
```

48 The code

We begin by loading the math extension font (`cmex10`) and the L^AT_EX line and circle fonts. It is necessary to do this explicitly since these are used by `lplain.tex` and `latex.tex`. 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 =line10\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 <+xipt & cm> \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
30 <+xipt & 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 <*xipt>
57 \DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,12}
58 \DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,12}
59 </xipt>
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 v

ltfntcmd.dtx

Abstract

The commands defined in this file `ltfntcmd` are part of the kernel code for \LaTeX 2 ϵ /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.

49 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 \TeX system and for several reasons it is better to avoid them on the user level whenever possible. In \LaTeX 3 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`.⁷ The default definition for this list is produced by the following.

```
\newcommand \nocorrlist {,.}
```

⁷Any 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>\textrm{..}</code>	<code>\rmfamily</code>	Typeset argument in roman family
<code>\textsf{..}</code>	<code>\sffamily</code>	Typeset argument in sans serif family
<code>\texttt{..}</code>	<code>\ttfamily</code>	Typeset argument in typewriter 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 SMALL CAPS 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.

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
\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 `\/` 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 `\/` in `amstex.sty` is used as this version can cause space to be removed immediately before the `\/`.

50 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 automatically 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 optimises 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 \<*2kernel>
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 }

```

`\textrm` Now we define the `\text{family}` commands in terms of the above; `\texttt` does not look very nice!

```

\textrm 15 \DeclareTextFontCommand{\textrm}{\rmfamily}
\textsf 16 \DeclareTextFontCommand{\textsf}{\sffamily}
\textnormal 17 \DeclareTextFontCommand{\texttt}{\ttfamily}
18 \DeclareTextFontCommand{\textnormal}{\normalfont}

```

`\textbf` For the series attribute:

```

\textmd 19 \DeclareTextFontCommand{\textbf}{\bfseries}
20 \DeclareTextFontCommand{\textmd}{\mdseries}

```

`\textit` And for the shapes:

```

\textsl 21 \DeclareTextFontCommand{\textit}{\itshape}
\textsc 22 \DeclareTextFontCommand{\textsl}{\slshape}
\textup 23 \DeclareTextFontCommand{\textsc}{\scshape}
24 \DeclareTextFontCommand{\textup}{\upshape}

```

`\emph` Finally we have the `\em` font change declaration of L^AT_EX. The corresponding definition with argument is

```

25 \DeclareTextFontCommand{\emph}{\em}

```

`\nocorr` This is just a label, so it does nothing; it should also be unexpandable.

```

26 \let \nocorr \relax

```

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

```

\check@icr 27 \let \check@icl \@empty
28 \let \check@icr \@empty

```

`\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).

`\check@nocorr@` Is this the correct action in the ‘empty’ case? It is efficient but typographically it is, strictly, incorrect!

```

29 \def \text@command #1{%
30   \def \reserved@a {#1}%
31   \ifx \reserved@a \@empty
32     \let \check@icl \@empty
33     \let \check@icr \@empty
34   \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.

```

35 % \def \reserved@b { }%
36 % \ifx \reserved@a \reserved@b
37 \ifx \reserved@a \space
38 \let \check@ic1 \@empty
39 \let \check@icr \@empty
40 \else
41 \check@nocorr@ #1\@nocorr\@nil
42 \fi
43 \fi
44 }
45 \def \check@nocorr@ #1#2\@nocorr#3\@nil {%

```

The two checks are initialised here to their values in the normal case.

```

46 \let \check@ic1 \maybe@ic
47 \def \check@icr {\ifvmode \else \aftergroup \maybe@ic \fi}%
48 \def \reserved@a {\@nocorr}%
49 \def \reserved@b {#1}%
50 \def \reserved@c {#3}%
51 \ifx \reserved@a \reserved@b
52 \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.

```

53 \let \check@ic1 \@empty
54 \else

```

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

```

55 \let \check@ic1 \@empty
56 \let \check@icr \@empty
57 \fi
58 \else
59 \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.

```

60 \else

```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```

61 \let \check@icr \@empty
62 \fi
63 \fi
64 }

```

`\maybe@ic` These macros implement the italic correction.

```

\maybe@ic@ 65 \def \maybe@ic {\futurelet\@let@token\maybe@ic@}
66 \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?

```

67 \ifdim \fontdimen\@ne\font>\z@

```

```

68 \else
69 \@tempswatrue

```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```

70 \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
71 \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.

```

72 \do \t@st@ic

```

Frank thinks that the next bit it 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`.

```

73 \if@tempswa \sw@slant \fi
74 \fi
75 }

```

`\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.

```

76 \def \t@st@ic {%
77 \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
78 \ifx\reserved@b\@let@token

```

If they are the same we record the fact and jump out of the loop.

```

79 \@tempswafalse
80 \@break@tfor
81 \fi
82 }

```

`\sw@slant` The definition of the mysterious `\sw@slant` command is as follows.
`\fix@penalty` 83 \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.

```

84 \ifdim \lastskip=\z@
85 \fix@penalty
86 \else
87 \skip@ \lastskip
88 \unskip
89 \fix@penalty
90 \hskip \skip@
91 \fi
92 }

```

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?

```

93 \def \fix@penalty {%
94   \ifnum \lastpenalty=\z@
95     \@@italiccorr
96   \else
97     \count@ \lastpenalty
98     \unpenalty
99     \@@italiccorr
100    \penalty \count@
101  \fi
102 }
```

`\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.

```

103 \def \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.

```

104 \ifx \nfss@text\@undefined
105   \def \nfss@text {\leavevmode\hbox}
106 \fi
```

`\DeclareOldFontCommand` This is the function used to create declarative font-changing commands that can also be used to change alphabets in math-mode.

Usage: `\DeclareOldFontCommand \fn{<font-change decls>} <math-alphabet>`

Here `\fn` is the font-declaration command being defined, `<font-change decls>` is the declaration it will expand to in text-mode, and `<math-alphabet>` is the (single) math alphabet specifier which is to be used in math-mode.

It does not care whether the command being defined already exists but it does give a warning if it redefines anything.

Here are some typical examples of its use in conjunction with more basic NFSS2 font commands.

```

\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
```

```

107 \def \DeclareOldFontCommand #1#2#3{%
108   \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
109 }
```

`\@fontswitch` These two commands actually do the necessary tests and declarative font- or
`\@@math@egroup` alphabet-changing.

```
\@@math@egroup 110 \def \@fontswitch #1#2{%
111   \ifmmode
112     \let \math@bgroup \relax
113     \def \math@egroup {\let \math@bgroup \@math@bgroup
114                       \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).

```
115     #2\relax
116   \else
117     #1%
118   \fi
119 }
120 \let \@math@bgroup \math@bgroup
121 \let \@math@egroup \math@egroup
```

These commands are available only in the preamble.

```
122 \@onlypreamble \DeclareTextFontCommand
123 \@onlypreamble \DeclareOldFontCommand
```

51 Initialization

`\normalsize` This is defined to produce an error.

```
124 \def\normalsize{%
125   \@latex@error {The font size command \protect\normalsize\space
126                 is not defined:\MessageBreak
127                 there is probably something wrong with
128                 the class file}\@eha
129 }
130 </2ekernel>
```


File w

ltpageno.dtx

52 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 pagenumber 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 <*2kernel>
2 \message{page nos.,}

3 \countdef\c@page=0 \c@page=1
4 \def\cl@page{}
5 \def\pagenumbering#1{%
6   \global\c@page \one \gdef\thepage{\csname @#1\endcsname
7     \c@page}}
8 </2kernel>
```

File x

ltxref.dtx

53 Cross Referencing

The user writes `\label{foo}` to define the following cross-references:

`\ref{foo}`: value of most recently incremented referencable counter. in the current environment. (Chapter, section, theorem and enumeration counters counters are referencable, footnote counters are not.)

`\pageref{foo}`: page number at which `\label{foo}` command appeared. where foo can be any string of characters not containing ‘\’, ‘{’ or ‘}’.

Note: The scope of the `\label` command is delimited by environments, so `\begin{theorem} \label{foo} ... \end{theorem} \label{bar}` defines `\ref{foo}` to be the theorem number and `\ref{bar}` to be the current section number.

Note: `\label` does the right thing in terms of spacing – i.e., leaving a space on both sides of it is equivalent to leaving a space on either side.

53.1 Cross Referencing

```
1 (*2ekernel)
2 \message{x-ref,}
```

This is implemented as follows. A referencable counter CNT is incremented by the command `\refstepcounter{CNT}`, which sets `\@currentlabel == {CNT}{eval(\p@cnt\theCNT)}`. The command `\label{FOO}` then writes the following on file `\@auxout`:

```
\newlabel{FOO}{{eval(\@currentlabel)}{eval(\thepage)}}
```

```
\ref{FOO} ==
BEGIN
  if \r@foo undefined
  then  @refundefined := G T
        ??
        Warning: 'reference foo on page ... undefined'
  else  \@car \eval(\r@FOO)\@nil
  fi
END
```

```
\pageref{foo} =
BEGIN
  if \r@foo undefined
  then  @refundefined := G T
        ??
        Warning: 'reference foo on page ... undefined'
  else  \@cdr \eval(\r@FOO)\@nil
  fi
END
```

`\G@refundefinedtrue` This does not save on name-space (since `\G@refundefinedfalse` was never needed) but it does make the implmentation 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* correspnd to an `\if` command, and there is no matching `...false`. It would be more natural to call the command `\G@refundefined` (as inspection of the change log will reveal) but unfortunately such a change would break any package that had defined a `\ref`-like command that mimicked the definition of `\ref`, calling `\G@refundefinedtrue`. Inspection of the T_EX 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

```

`\ref` Referencing a `\label`. RmS 91/10/25: added a few extra `\reset@font`, as suggested by Bernd Raichle
`\pageref`
`\@setref` RmS 92/08/14: made `\ref` and `\pageref` robust
RmS 93/09/08: Added setting of `refundefined` switch.

```

10 \def\@setref#1#2#3{%
11   \ifx#1\relax
12     \protect\G@refundefinedtrue
13     \nfss@text{\reset@font\bfseries ??}%
14     \@latex@warning{Reference ‘#3’ on page \thepage \space
15                   undefined}%
16   \else
17     \expandafter#2#1\null
18   \fi}
19 \def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
20 \def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
21               \@secondoftwo{#1}}

```

`\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 `\bibcite`. Note that this macro does it's work inside a group. That way the local assignments it needs to do don't clutter the save stack. This prevents large documents with many labels to run out of save stack.

```

22 \def\@newl@bel#1#2#3{%
23   \@ifundefined{#1@#2}%
24     \relax
25     {\gdef \@multiplelabels {%
26       \@latex@warning@no@line{There were multiply-defined labels}}%
27       \@latex@warning@no@line{Label ‘#2’ multiply defined}}%
28   \global\@namedef{#1@#2}{#3}}

```

```

29 \def\newlabel{\@newl@bel r}
30 \@onlypreamble\@newl@bel

\if@multiplelabels This is redefined to produce a warning if at least one label is defined more than
\@multiplelabels once. It is executed by the \enddocument command.
31 \let \@multiplelabels \relax

\label The commands \label and \refstepcounter have been changed to allow
\refstepcounter \protect'ed commands to work properly. For example,

\def\thechapter{\protect\foo{\arabic{chapter}.\roman{section}}}}

will cause a \label{bar} command to define \ref{bar} to expand to something
like \foo{4.d}. Change made 20 Jul 88.

32 \def\label#1{\@bsphack
33   \protected@write\@auxout{}%
34     {\string\newlabel{#1}{\@currentlabel}\thepage}}%
35   \@esphack}

36 \def\refstepcounter#1{\stepcounter{#1}%
37   \protected@edef\@currentlabel
38     {\csname p@#1\endcsname\csname the#1\endcsname}%
39 }

\@currentlabel For \label commands that come before any environment

40 \def\@currentlabel{}

41 </2ekernel>

```

53.2 An extension of counter referencing

At the moment a reference to a counter `foo` will generate the equivalent of `\p@foo\thefoo` although not quite in this form. For some applications it would be nice if one could have `\thefoo` being an argument to `\p@foo` to be able to put material before and after the number generated by `\thefoo`. This can be easily achieved with a small change to one of the kernel commands as follows:

```

\def\refstepcounter#1{\stepcounter{#1}%
  \protected@edef\@currentlabel
    {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
}

```

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 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.

We have refrained from making this change in the kernel code although for existing documents it would be 100% backward compatible. The reason being

that any class or package making use of this functionality would then horribly fail with older L^AT_EX installations.

Instead we suggest that people who are interested in using this functionality in a document class or package add the redefinition to the class file. To ensure that this redefinition is properly applied they might want to test for the original definition first, e.g.

```
\CheckCommand*\refstepcounter[1]{\stepcounter{#1}%
  \protected@edef\@currentlabel
    {\csname p@#1\endcsname\csname the#1\endcsname}%
}
\renewcommand*\refstepcounter[1]{\stepcounter{#1}%
  \protected@edef\@currentlabel
    {\csname p@#1\expandafter\endcsname\csname the#1\endcsname}%
}
```

File y

ltmiscen.dtx

54 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the ‘ushing’ environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1 \*2ekernel
2 \message{environments,}
```

54.1 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 T_EX82.

`\enddocument` is the user’s command for ending the manuscript file.

`\stop` is a panic button — to end T_EX in the middle.

```
\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 @tempwa := true          fi
      END
      \bibcite{LABEL}{VAL} == null
      BEGIN
        \reserved@a == VAL
        if def(\reserved@a) = def(\g@LABEL)
        else @tempwa := true          fi
    fi
  fi
END
```

```

                                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

\@currentvir The name of the current environment.  Initialized to document to so that
\end{document} works correctly.
3 \def\@currentvir{document}

\if@ignore
\@ignoretrue 4 \def\@ignorefalse{\global\let\if@ignore\iffalse}
\@ignorefalse 5 \def\@ignoretrue {\global\let\if@ignore\iftrue}
6 \@ignorefalse

\ignorespacesafterend

7 \let\ignorespacesafterend\@ignoretrue

\enddocument

8 \def\enddocument{%
The \end{document} hook is executed first.  If necessary it can contain a
\clearpage to output dangling oats 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 en-
vironments 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...

9 \let\AtEndDocument\@firstofone
10 \@enddocumenthook
11 \@checkend{document}%
12 \clearpage
13 \begingroup
14 \if@filesw
15 \immediate\closeout\@mainaux
16 \let\@setckpt\@gobbletwo
17 \let\@newl@bel\@testdef
The previous line is equiv to setting

\def\newlabel{\@testdef r}%
\def\bibcite{\@testdef b}%

```

```

18      \@tempswafalse
19      \makeatletter \input\jobname.aux
20      \fi

```

```

21      \@dofilelist

```

First we check for font size substitution bigger than `\fontsubfuzz`. The `\relax` is necessary because this is a macro not a register.

```

22      \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.

```

23      \@font@warning{Size substitutions with differences\MessageBreak
24                      up to \font@submax\space have occurred.\gobbletwo}%
25      \fi

```

The macro `\@defaultsubs` is initially `\relax` but gets redefined to produce a warning if there have been some default font substitutions.

```

26      \@defaultsubs

```

The macro `\@refundefined` is initially `\relax` but gets redefined to produce a warning if there are undefined refs.

```

27      \@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 y), we suppress this message.

```

28      \if@filesw
29      \ifx \@multiplelabels \relax
30      \if@tempswa
31      \@latex@warning@no@line{Label(s) may have changed.
32      Rerun to get cross-references right}%
33      \fi
34      \else
35      \@multiplelabels
36      \fi
37      \fi
38      \endgroup
39      \deadcycles\z@\@end}

```

```

\@testdef

```

```

40 \def\@testdef #1#2#3{%
41   \def\reserved@a{#3}\expandafter \ifx \csname #1@#2\endcsname
42   \reserved@a \else \@tempswatrue \fi}

```

```

\@writefile

```

```

43 \long\def\@writefile#1#2{%
44   \@ifundefined{tf@#1}\relax
45   {\@temptokena{#2}%
46    \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
47   }%
48 }

```


\stop

```
49 \def\stop{\clearpage\deadcycles\z@\let\par\@par\@end}

50 \everypar{\@nodocument} %% To get an error if text appears before the
51 \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
```

```

\begin
52 \def\begin#1{%
53   \@ifundefined{#1}%
54   {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
55   {\def\reserved@a{\def\@currenvir{#1}%
56     \edef\@currencline{\on@line}%
57     \csname #1\endcsname}}%
58   \@ignorefalse
59   \begingroup\@endpfalse\reserved@a}

\end
60 \def\end#1{%
61   \csname end#1\endcsname\@checkend{#1}%
62   \expandafter\endgroup\if@endpe\@doendpe\fi
63   \if@ignore\@ignorefalse\ignorespaces\fi}

\@checkend
64 \def\@checkend#1{\def\reserved@a{#1}\ifx
65   \reserved@a\@currenvir \else\@badend{#1}\fi}

\@currencline We do need a default value for \@currencline 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.
66 \let\@currencline\@empty

```

54.2 Center, Flushright, Flushleft

```

67 \message{center,}

\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 := ushglue`. Every environment, like the list environments, that set `\rightskip` to its 'normal' value set it to `\@rightskip`

```

\@centercr
68 \def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
69     \par\@ifstar{\nobreak\@xcentercr}\@xcentercr}

\@xcentercr
70 \def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
71     [\@icentercr\ignorespaces}

\@icentercr
72 \def\@icentercr[#1]{\vskip #1\ignorespaces}

center We use \relax to prevent \item scanning too far.
73 \def\center{\trivlist \centering\item\relax}
74 \def\endcenter{\endtrivlist}

\centering
75 \def\centering{%
76     \let\\\@centercr
77     \rightskip\@flushglue\leftskip\@flushglue
78     \parindent\z@\parfillskip\z@skip}

\@rightskip
79 \newskip\@rightskip \@rightskip \z@skip

flushleft We use \relax to prevent \item scanning too far.
80 \def\flushleft{\trivlist \raggedright\item\relax}
81 \def\endflushleft{\endtrivlist}

\raggedright
82 \def\raggedright{%
83     \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
84     \leftskip\z@skip
85     \parindent\z@}

flushright We use \relax to prevent \item scanning too far.
86 \def\flushright{\trivlist \raggedleft\item\relax}
87 \def\endflushright{\endtrivlist}

\raggedleft
88 \def\raggedleft{%
89     \let\\\@centercr
90     \rightskip\z@skip\leftskip\@flushglue
91     \parindent\z@\parfillskip\z@skip}

92 \message{verbatim,}

```

54.3 Verbatim

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carriage return (or sequence of consecutive carriage returns), and interprets *every* character literally. I.e., all special characters `\`, `{`, `$`, etc. are `\catcode'd` to 'other'.

The command `\verb` produces in-line verbatim text, where the argument is delimited by any pair of characters. E.g., `\verb #...#` takes '...' as its argument, and sets it verbatim in `\ttfamily` font.

The *-variants of these commands are the same, except that spaces print as the T_EXbook's space character instead of as blank spaces.

`\@vobeyspaces`

```
93 {\catcode'\ =\active%
94 \gdef\@vobeyspaces{\catcode'\ \active\let \xobeysp}}
```

`\@xobeysp`

`\@xverbatim`

```
\@sxverbatim 95 \begingroup \catcode '|=0 \catcode '['= 1
96 \catcode']=2 \catcode '{=12 \catcode '\}=12
97 \catcode'\}=12 \gdef\@xverbatim#1\end{verbatim}[#1\end[verbatim]]
98 \gdef\@sxverbatim#1\end{verbatim*}[#1\end[verbatim*]]
99 \endgroup
```

`\@verbatim` Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

```
100 \def\@verbatim{\trivlist \item\relax
101 \if@minipage\else\vskip\parskip\fi
102 \leftskip\@totalleftmargin\rightskip\z@skip
103 \parindent\z@\parfillskip\@flushglue\parskip\z@skip
```

Added `\@@par` to clear possible `\parshape` definition from a surrounding list (the verbatim guru says).

```
104 \@@par
105 \@tempswafalse
106 \def\par{%
107 \if@tempswa
```

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```
108 \leavevmode \null \@@par\penalty\interlinepenalty
109 \else
110 \@tempswatrue
111 \ifhmode\@@par\penalty\interlinepenalty\fi
112 \fi}%
```

To allow customization we hide the font used in a separate macro.

```
113 \let\do\@makeother \dospecials
114 \obeylines \verbatim@font \@noligs
115 \hyphenchar\font\m@ne
```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty`!

```

116 \everypar \expandafter{\the\everypar \unpenalty}%
117 }

\verbatim (RmS 93/09/19) Protected against 'missing item' error message triggered by
\endverbatim empty verbatim environment.
118 \def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
119 \def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}

\verbatim@font Macro to select the font used for verbatim typesetting. It also does other work if
necessary for the font used.
120 \def\verbatim@font{\normalfont\ttfamily}

verbatim*
121 \@namedef{verbatim*}{\@verbatim\@sxverbatim}
122 \expandafter\let\csname endverbatim*\endcsname =\endverbatim

\@makeother
123 \def\@makeother#1{\catcode'#112\relax}

\verb@balance@group
124 \let\verb@balance@group\@empty

\verb@egroup
125 \def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}

\verb@eol@error
126 \begingroup
127 \obeylines%
128 \gdef\verb@eol@error{\obeylines%
129 \def^^M{\verb@egroup\@latex@error{%
130 \noexpand\verb ended by end of line}\@ehc}}%
131 \endgroup

\verb Typesetting a small piece verbatim.
132 \def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
133 \bgroup
134 \verb@eol@error \let\do\@makeother \dospecials
135 \verbatim@font\@noligs
136 \@ifstar\@sverb\@verb}

\@sverb Definitions of \@sverb and \@verb changed so \verb+ foo+ does not lose lead-
ing blanks when it comes at the beginning of a line. Change made 24 May 89.
Suggested by Frank Mittelbach and Rainer Schöpf.
137 \def\@sverb#1{%
138 \catcode'#1\active
139 \lccode'\~'#1%
140 \gdef\verb@balance@group{\verb@egroup
141 \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
142 \aftergroup\verb@balance@group
143 \lowercase{\let~\verb@egroup}}%

```

```

\@verb
144 \def\@verb{\@vobeyspaces \frenchspacing \@sverb}

\verbatim@nolig@list
145 \def\verbatim@nolig@list{\do\'\do\<\do\>\do\,\do\' \do\ -}

\do@noligs
146 \def\do@noligs#1{%
147   \catcode'#1\active
148   \begingroup
149     \lccode'\~'#1\relax
150     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char'#1}}

\@noligs To stay compatible with packages that use \@noligs we keep it.
151 \def\@noligs{\let\do\do@noligs \verbatim@nolig@list}

152 \</2kernel>

```

File z

ltmath.dtx

55 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,}
```

55.1 Math commands based on plain T_EX

55.1.1 The log-like functions

```
\log The standard operators:
3 \def\log{\mathop{\operator@font log}\nolimits}
4 \def\lg{\mathop{\operator@font lg}\nolimits}
5 \def\ln{\mathop{\operator@font ln}\nolimits}
6 \def\lim{\mathop{\operator@font lim}\nolimits}
7 \def\limsup{\mathop{\operator@font lim}\nolimits,\sup}
8 \def\liminf{\mathop{\operator@font lim}\nolimits,\inf}
9 \def\sin{\mathop{\operator@font sin}\nolimits}
10 \def\arcsin{\mathop{\operator@font arcsin}\nolimits}
11 \def\sinh{\mathop{\operator@font sinh}\nolimits}
12 \def\cos{\mathop{\operator@font cos}\nolimits}
13 \def\arccos{\mathop{\operator@font arccos}\nolimits}
14 \def\cosh{\mathop{\operator@font cosh}\nolimits}
15 \def\tan{\mathop{\operator@font tan}\nolimits}
16 \def\arctan{\mathop{\operator@font arctan}\nolimits}
17 \def\tanh{\mathop{\operator@font tanh}\nolimits}
18 \def\cot{\mathop{\operator@font cot}\nolimits}
19 \def\coth{\mathop{\operator@font coth}\nolimits}
20 \def\sec{\mathop{\operator@font sec}\nolimits}
21 \def\csc{\mathop{\operator@font csc}\nolimits}
22 \def\max{\mathop{\operator@font max}\nolimits}
23 \def\min{\mathop{\operator@font min}\nolimits}
24 \def\sup{\mathop{\operator@font sup}\nolimits}
25 \def\inf{\mathop{\operator@font inf}\nolimits}
26 \def\arg{\mathop{\operator@font arg}\nolimits}
27 \def\ker{\mathop{\operator@font ker}\nolimits}
28 \def\dim{\mathop{\operator@font dim}\nolimits}
29 \def\hom{\mathop{\operator@font hom}\nolimits}
30 \def\det{\mathop{\operator@font det}\nolimits}
31 \def\exp{\mathop{\operator@font exp}\nolimits}
32 \def\Pr{\mathop{\operator@font Pr}\nolimits}
33 \def\gcd{\mathop{\operator@font gcd}\nolimits}
34 \def\deg{\mathop{\operator@font deg}\nolimits}

\bmod And some operators have to be done by hand:
35 \def\bmod{%
36   \nonscript\mskip-\medmuskip\mkern5mu%
```

```

37 \mathbin{\operatorname@font mod}\penalty900\mkern5mu%
38 \nonscript\mskip-\medmuskip}

```

`\pmod`

```

39 \def\pmod#1{%
40 \allowbreak\mkern18mu({\operatorname@font mod}\,\,\,#1)}

```

55.1.2 Biggggg

`\big` Variants on `\big` and friends for use with delimiters:

```

41 \def\bigl{\mathopen\big}
42 \def\bigm{\mathrel\big}
43 \def\bigl{\mathclose\big}
44 \def\Bigl{\mathopen\Big}
45 \def\Bigm{\mathrel\Big}
46 \def\Bigr{\mathclose\Big}
47 \def\biggl{\mathopen\bigg}
48 \def\biggm{\mathrel\bigg}
49 \def\biggr{\mathclose\bigg}
50 \def\Biggl{\mathopen\Bigg}
51 \def\Biggm{\mathrel\Bigg}
52 \def\Biggr{\mathclose\Bigg}

```

55.1.3 The UNSORTED Rest

The other math commands are lifted from plain T_EX.

`\jot`

```

53 \newdimen\jot
54 \jot=3pt

```

`\interdisplaylinepenalty`

```

55 \newcount\interdisplaylinepenalty
56 \interdisplaylinepenalty=100

```

`\choose`

```

57 \def\choose{\atopwithdelims()}

```

`\brack`

```

58 \def\brack{\atopwithdelims[]}

```

`\brace`

```

59 \def\brace{\atopwithdelims\{ \}}

```

`\mathpalette`

```

60 \def\mathpalette#1#2{%
61 \mathchoice
62 {\#1\displaystyle{#2}}%
63 {\#1\textstyle{#2}}%
64 {\#1\scriptstyle{#2}}%
65 {\#1\scriptscriptstyle{#2}}}

```



```

\root
\rootbox 66 \newbox\rootbox
\root 67 \def\root#1\of{%
68 \setbox\rootbox\hbox{\m@th\scriptscriptstyle{#1}}}%
69 \mathpalette\root@t}

70 \def\root#1#2{%
71 \setbox\z@\hbox{\m@th#1\sqrt{sign{#2}}}%
72 \dimen@ht\z@ \advance\dimen@-\dp\z@
73 \mkern5mu\raise.6\dimen@\copy\rootbox
74 \mkern-10mu\box\z@}

\phantom
\hphantom 75 \newif\ifv@
\vphantom 76 \newif\ifh@

77 \def\vphantom{\v@true\h@false\ph@nt}
78 \def\hphantom{\v@false\h@true\ph@nt}
79 \def\phantom{\v@true\h@true\ph@nt}

80 \def\ph@nt{%
81 \ifmmode
82 \expandafter\mathpalette\expandafter\mathph@nt
83 \else
84 \expandafter\makeph@nt
85 \fi}

86 \def\makeph@nt#1{%
87 \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}

88 \def\mathph@nt#1#2{%
89 \setbox\z@\hbox{\m@th#1{#2}}\finph@nt}

90 \def\finph@nt{%
91 \setbox\tw@\null
92 \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
93 \ifh@ \wd\tw@\wd\z@\fi \box\tw@}

\mathstrut
94 \def\mathstrut{\vphantom{}}

\smash
95 \def\smash{%
96 \relax % \relax, in case this comes first in \halign
97 \ifmmode
98 \expandafter\mathpalette\expandafter\mathsm@sh
99 \else
100 \expandafter\makesm@sh
101 \fi}

102 \def\makesm@sh#1{%
103 \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
104 \def\mathsm@sh#1#2{%
105 \setbox\z@\hbox{\m@th#1{#2}}\finsm@sh}
106 \def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}

```

```

\buildrel
107 \def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}

\cases
108 \def\cases#1{\left\{\,\,\vcenter{\normalbaselines\m@th
109 \ialign{\$##\hfil\$&\quad{##}\hfil\crrc#1\crrc}}\right.}

\matrix
110 \def\matrix#1{\null\,\vcenter{\normalbaselines\m@th
111 \ialign{\hfil\$##\hfil&\quad\hfil\$##\hfil\crrc
112 \mathstrut\crrc\noalign{\kern-\baselineskip}
113 #1\crrc\mathstrut\crrc\noalign{\kern-\baselineskip}}}\,,}

\pmatrix
114 \def\pmatrix#1{\left(\matrix{#1}\right)}

\bordermatrix
115 \def\bordermatrix#1{\begingroup \m@th
116 \@tempdima 8.75\p@
117 \setbox\z@\vbox{%
118 \def\cr{\crrc\noalign{\kern2\p@\global\let\cr\endline}}%
119 \ialign{\$##\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil\$##\hfil
120 &\quad\hfil\$##\hfil\crrc
121 \omit\strut\hfil\crrc\noalign{\kern-\baselineskip}%
122 #1\crrc\omit\strut\cr}}%
123 \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
124 \setbox\tw@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
125 \setbox\tw@\hbox{\$ \kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
126 \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
127 \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,,\right)$}%
128 \null\;\vbox{\kern\ht\@ne\box\tw@}\endgroup}

\openup
129 \def\openup{\afterassignment\@openup\dimen@}
130 \def\@openup{\advance\lineskip\dimen@
131 \advance\baselineskip\dimen@
132 \advance\lineskiplimit\dimen@}

\displaylines
133 \newif\ifdt@p
134 \def\displ@y{\global\dt@ptrue\openup\jot\m@th
135 \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
136 \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
137 \else \penalty\interdisplaylinepenalty \fi}}}
138 \def\@lign{\tabskip\z@skip\everycr{}} % restore inside \displ@y
139 \def\displaylines#1{\displ@y \tabskip\z@skip
140 \halign{\hb@xt@\displaywidth{\$ \@lign\hfil\displaystyle##\hfil\$}\crrc
141 #1\crrc}}

```

```

\sp
\sb 142 \let\sp=~
    143 \let\sb=_

\>
\; 144 %\def\,{\mskip\thinmuskip}      % already defined in ltspac
\! 145 \def\>{\mskip\medmuskip}
    146 \def\;{\mskip\thickmuskip}
    147 \def\!{\mskip-\thinmuskip}

\*
    148 \def\*{\discretionary{\thinspace\the\textfont2\char2}{\}}{}

\: Nickname for the medium space since \> is not available inside tabbing.
    149 \let\:=\>

\active@math@prime This is the definition of the active math prime.
    150 \def\active@math@prime{^{\bgroup\prim@s}}

\prime@s
    151 {\catcode'\='=\active \global\let'\active@math@prime}
    152 \def\prim@s{%
    153   \prime\futurelet\@let@token\pr@m@s}
    154 \def\pr@m@s{%
    155   \ifx'\@let@token
    156     \expandafter\pr@@@s
    157   \else
    158     \ifx^\@let@token
    159       \expandafter\expandafter\expandafter\pr@@@t
    160     \else
    161       \egroup
    162     \fi
    163   \fi}
    164 \def\pr@@@s#1{\prim@s}
    165 \def\pr@@@t#1#2{#2\egroup}

    166 {\catcode'\_=\active \gdef\_{} } % _ in math is
    167                                     % either subscript or \_

```

55.2 Math Environments

```

\(  

\(  

    168 \def\({\relax\ifmmode\@badmath\else$\fi}  

    169 \def\){\relax\ifmmode\ifinner$\else\@badmath\fi\else \@badmath\fi}

```

`\[` Produces `$$...$$` with checks that `\[` isn't used in math mode, and that `\]` is only used in math mode begun with `\[`.

```

170 \def\[{%
171   \relax\ifmmode
172     \@badmath
173   \else
174     \ifvmode
175       \nointerlineskip
176       \makebox[.6\linewidth]{}%
177     \fi
178     $$$%$$ BRACE MATCH HACK
179   \fi
180 }

181 \def\]{%
182   \relax\ifmmode
183     \ifinner
184       \@badmath
185     \else
186       $$$%$$ BRACE MATCH HACK
187     \fi
188   \else
189     \@badmath
190   \fi
191   \ignorespaces
192 }

```

`math` Disguises for `\(...\)` and `\[...\]`.

```

displaymath 193 \let\math=(
194 \let\endmath=\)

195 \def\displaymath{\[
196 \def\enddisplaymath{\]\@ignoretrue}

```

`equation` Numbered equations, using the counter `\c@equation`. *Note:* The document style must define `\theequation` etc., and do the appropriate `\@addtoreset`. It should also redefine `\@eqnnum` if another format for the equation number is desired other than the standard (...), or to move the equation numbers to the ushleft. (See comment on the `\def` of `\@eqnnum`.)

```

\c@equation 197 \@definecounter{equation}
198 \def\equation{$$\refstepcounter{equation}}
199 \def\endequation{\eqno \hbox{\@eqnnum}$$\@ignoretrue}

```

`\@eqnnum` Produces the equation number for equation and eqnarray environments. The following definition is for ushright numbers; for ushleft numbers, see `leqno.clo`. The equation number is set in black roman type even if an eqnarray environment appears in an italic environment.

```

200 \def\@eqnnum{\normalfont \normalcolor (\theequation)}

```

`\stackrel` A disguise for plain T_EX's `\stackrel`.

```

201 \def\stackrel#1#2{\mathrel{\mathop{#2}\limits^{#1}}}

```

```

\frac A disguise for plain TEX's \over.
202 \def\frac#1#2{{\begingroup#1\endgroup\over#2}}

\sqrt Add an optional argument to plain's \sqrt to give the nth root of an expression
\@sqrt  $\sqrt[n]{e}$ .
203 \DeclareRobustCommand\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
204 \def\@sqrt[#1]{\root #1\of}

eqnarray Here's the eqnarray environment: Default is for left-hand side of equations to be
\@eqcnt ushright. To make them ushleft, \let\@eqnrel = \hfil.
\@eqpen 205 \newcount\@eqcnt
\if@eqnsw 206 \newcount\@eqpen
\@eqnrel 207 \newif\if@eqnsw\@eqnswtrue
208 \newskip\@centering
209 \@centering = 0pt plus 1000pt

To get a proper \@currentlabel we have to redefine it for the whole display. Note
that we can't use \refstepcounter as this results in \@currentlabel getting
restored at the wrong and thus always writing the first label to the .aux file.

210 \def\eqnarray{%
211   \stepcounter{equation}%
212   \def\@currentlabel{\p@equation\theequation}%
213   \global\@eqnswtrue
214   \m@th
215   \global\@eqcnt\z@
216   \tabskip\@centering
217   \let\\\@eqnrcr
218   $$$\everycr{}\halign to\displaywidth\bgroup
219     \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnrel
220     &\global\@eqcnt\@ne\hskip \tw@arraycolsep \hfil${##}$\hfil
221     &\global\@eqcnt\tw@ \hskip \tw@arraycolsep
222     $\displaystyle{##}$\hfil\tabskip\@centering
223     &\global\@eqcnt\thr@@ \hbext@z@\bgroup\hss##\egroup
224     \tabskip\z@skip
225   \cr
226 }

227 \def\endeqnarray{%
228   \@@eqnrcr
229   \egroup
230   \global\advance\c@equation\m@ne
231   $$$\ignoretrue
232 }

233 \let\@eqnrel=\relax

\nonumber Switches off equation numbering.
234 \def\nonumber{\global\@eqnswfalse}

\@eqnrcr
\@xeqnrcr 235 \def\@eqnrcr{%
\@yeqnrcr 236   {\ifnum0='}\fi
237   \@ifstar{%

```

```

238     \global\@eqpen\@M\@yeqncr
239   }{%
240     \global\@eqpen\interdisplaylinepenalty \@yeqncr
241   }%
242 }

243 \def\@yeqncr{\@testopt\@xeqncr\z@skip}

244 \def\@xeqncr[#1]{%
245   \ifnum0='{ \fi}%
246   \@eqncr
247   \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
248 }

\@eqncr

249 \def\@eqncr{\let\reserved@a\relax
250   \ifcase\@eqcnt \def\reserved@a{& & }\or \def\reserved@a{& & }%
251   \or \def\reserved@a{& }\else
252     \let\reserved@a\empty
253     \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
254   \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
255   \global\@eqnswtrue\global\@eqcnt\z@\cr}

eqnarray* Here's the eqnarray* environment:
\@seqncr 256 \let\@seqncr=\@eqncr

257 \@namedef{eqnarray*}{\def\@eqncr{\nonumber\@seqncr}\eqnarray}
258 \@namedef{endeqnarray*}{\nonumber\endeqnarray}

\lefteqn \lefteqn{FORMULA} typesets FORMULA in display math style ushleft in a box of
width zero.
259 \def\lefteqn#1{\rlap{$\displaystyle #1$}}

\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.
260 \DeclareRobustCommand{\ensuremath}{%
261   \ifmmode
262     \expandafter\@firstofone
263   \else
264     \expandafter\@ensuredmath
265   \fi}

\@ensuredmath The \relax stops \ensuremath{} starting display math.
266 \long\def\@ensuredmath#1{$\relax#1$}

267 </2ekernel>

```

55.3 External options to the standard document classes

55.3.1 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.

```
268 \*leqno>
269 \renewcommand\@eqnnum{\hb@xt@.01\p@{}%
270 \rlap{\normalfont\normalcolor
271 \hskip -\displaywidth(\theequation)}}
272 \*leqno>
```

55.3.2 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.

```
273 \*eqn >
274 \newdimen\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.

```
275 \AtEndOfClass{\mathindent\leftmargini}
```

`\[` Begin display math;

```
276 \renewcommand\[{\relax
277 \ifmode\@badmath
278 \else
279 \begin{trivlist}%
280 \@beginparpenalty\predisplaypenalty
281 \@endparpenalty\postdisplaypenalty
282 \item[]\leavevmode
283 \hb@xt@\linewidth\bgroup $\m@th\displaystyle %$
284 \hskip\mathindent\bgroup
285 \fi}
```

`\]` end display math;

```
286 \renewcommand\]{\relax
287 \ifmode
288 \egroup $\hfil% $
289 \egroup
290 \end{trivlist}%
291 \else \@badmath
292 \fi}
```

`equation` The equation environment

```
293 \renewenvironment{equation}%
294 {\@beginparpenalty\predisplaypenalty
295 \@endparpenalty\postdisplaypenalty
```

```

296 \refstepcounter{equation}%
297 \trivlist \item[]\leavevmode
298 \hb@xt@\linewidth\bgroup $\m@th% $
299 \displaystyle
300 \hskip\mathindent}%
301 {\$ \hfil % $
302 \displaywidth\linewidth\hbox{\@eqnnum}%
303 \egroup
304 \endtrivlist}

```

eqnarray The eqnarray environment

```

305 \renewenvironment{eqnarray}{%
306 \stepcounter{equation}%
307 \def\@currentlabel{\p@equation\theequation}%
308 \global\@eqnswtrue\m@th
309 \global\@eqcnt\z@
310 \tabskip\mathindent
311 \let\@=\@eqnocr
312 \setlength\abovedisplayskip{\topsep}%
313 \ifvmode
314 \addtolength\abovedisplayskip{\partopsep}%
315 \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.

```

316 \addtolength\abovedisplayskip{\parskip}%
317 \setlength\belowdisplayskip{\abovedisplayskip}%
318 \setlength\belowdisplayshortskip{\abovedisplayskip}%
319 \setlength\abovedisplayshortskip{\abovedisplayskip}%
320 $$\everycr{}\halign to\linewidth% $$
321 \bgroup
322 \hskip\@centering
323 $\displaystyle\tabskip\z@skip{##}$\@eqnsele%
324 \global\@eqcnt\@ne \hskip \tw@arraycolsep \hfil${##}$\hfil%
325 \global\@eqcnt\tw@ \hskip \tw@arraycolsep
326 $\displaystyle{##}$\hfil \tabskip\@centering%
327 \global\@eqcnt\thr@@
328 \hb@xt@\z@\bgroup\hss##\egroup\tabskip\z@skip\cr}%
329 {\@eqnocr
330 \egroup
331 \global\advance\c@equation\m@ne$$$ $$
332 \@ignoretrue
333 }
334 \end{eqnarray}

```


File A

ltlists.dtx

56 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 usright, `\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 exability 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.

56.1 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`.

`\boxlabels` 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.

`@nopalist` 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.

`@nmbrlist` 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^AT_EX 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.

56.2 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.

56.3 Penalties

`\@beginparpenalty`: put at the beginning of a list

`\@endparpenalty`: put at end of list

`\@itempenalty`: put between items.

56.4 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 pushed 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.

56.5 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.

```
\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
    \@trivlist
    \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 \@donoparitem
      \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

\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

```

```

\leftmargin
\rightmargin 9 \newdimen\leftmargin
\listparindent 10 \newdimen\rightmargin
\itemindent 11 \newdimen\listparindent
\labelwidth 12 \newdimen\itemindent
\labelsep 13 \newdimen\labelwidth
\@totalleftmargin 14 \newdimen\labelsep
15 \newdimen\linewidth
16 \newdimen\@totalleftmargin \@totalleftmargin=\z@

\leftmargini
\leftmarginii 17 \newdimen\leftmargini
\leftmarginiii 18 \newdimen\leftmarginii
\leftmarginiv 19 \newdimen\leftmarginiii
\leftmarginv 20 \newdimen\leftmarginiv
\leftmarginvi 21 \newdimen\leftmarginv
22 \newdimen\leftmarginvi

\@listdepth
\@itempenalty 23 \newcount\@listdepth \@listdepth=0
\@beginparpenalty 24 \newcount\@itempenalty
\@endparpenalty 25 \newcount\@beginparpenalty
26 \newcount\@endparpenalty

\@labels
27 \newbox\@labels

\if@inlabel
\@inlabelfalse 28 \newif\if@inlabel \@inlabelfalse
\@inlabeltrue
\if@newlist
\@newlistfalse 29 \newif\if@newlist \@newlistfalse
\@newlisttrue
\if@noparitem
\@noparitemfalse 30 \newif\if@noparitem \@noparitemfalse
\@noparitemtrue
\if@nopalist
\@nopalistfalse 31 \newif\if@nopalist \@nopalistfalse
\@nopalisttrue
\if@noitemarg
\@noitemargfalse 32 \newif\if@noitemarg \@noitemargfalse
\@noitemargtrue
\if@newlist
\@newlistfalse 33 \newif\if@nmbrlist \@nmbrlistfalse
\@newlisttrue
\list
34 \def\list#1#2{%
35 \ifnum \@listdepth >5\relax
36 \@toodeep
37 \else
38 \global\advance\@listdepth\@ne

```



```

39 \fi
40 \rightmargin\z@
41 \listparindent\z@
42 \itemindent\z@
43 \csname @list\romannumeral\the\@listdepth\endcsname
44 \def\@itemlabel{#1}%
45 \let\makelabel\@mklab
46 \@nmbrrlistfalse
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 \@ne \@totalleftmargin \linewidth
55 \ignorespaces}

```

\par@deathcycles

```

56 \newcount\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 ii 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     \@nparitemtrue
67     \@nparlisttrue
68   \else
69     \if@newlist \@noitemerr \fi
70     \@nparlistfalse
71     \@topsep \@topsepadd
72   \fi
73   \advance\@topsep \parskip
74   \leftskip \z@skip
75   \rightskip \@rightskip
76   \parfillskip \@flushglue
77   \par@deathcycles \z@
78   \@setpar{\if@newlist
79             \advance\par@deathcycles \@ne
80             \ifnum \par@deathcycles >\@m
81               \@noitemerr
82               {\@par}%
83             \fi
84           \else
85             {\@par}%

```

```

86         \fi}%
87     \global \@newlisttrue
88     \@outerparskip \parskip}

```

`\trivlist`

```

89 \def\trivlist{%
90     \parsep\parskip
91     \@nmbrlistfalse
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}}

```

`\endlist`

```

98 \def\endlist{%
99     \global\advance\@listdepth\m@ne
100     \endtrivlist}

```

The definition of `\trivlist` used to be in `ltspace.dtx` so that other commands could be 'let to it'. They now use `\def`.

`\endtrivlist`

```

101 \def\endtrivlist{%
102     \if@inlabel
103         \leavevmode
104         \global \@inlabelfalse
105     \fi
106     \if@newlist
107         \noitemerr
108         \global \@newlistfalse
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@nparlist \else
115         \ifdim\lastskip >\z@
116             \@tempskipa\lastskip \vskip -\lastskip
117             \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
118             \vskip\@tempskipa
119         \fi
120         \@endparrn
121     \fi
122 }

```

```

\@endparenv  To suppress the paragraph indentation in text immediately following a paragraph-
\@doendpe    making environment, \everypar is changed to remove the space, and \par
              is redefined to restore \everypar. Instead of redefining \par and \everpar,
              \@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 \def\@doendpe{\@endpetrue
126   \def\par{\@restorepar\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 environ-
              ment(23 Oct 86).

127   {\setbox\z@\lastbox}\everypar}\@endpefalse}}

\if@endpe
\@endpefalse 128 \newif\if@endpe
\@endpetrue  129 \@endpefalse

\@mklab
130 \def\@mklab#1{\hfil #1}

\item
131 \def\item{%
132   \@inmatherr\item
133   \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}

\@donoparitem
134 \def\@donoparitem{%
135   \@noparitemfalse
136   \global\setbox\@labels\hbox{\hskip -\leftmargin
137                                   \unhbox\@labels
138                                   \hskip \leftmargin}%
139   \if@minipage\else
140     \@tempskipa\lastskip
141     \vskip -\lastskip
142     \advance\@tempskipa\@outerparskip
143     \advance\@tempskipa -\parskip
144     \vskip\@tempskipa
145   \fi}

\@item
146 \def\@item[#1]{%
147   \if@noparitem
148     \@donoparitem
149   \else
150     \if@inlabel
151       \indent \par
152     \fi

```

```

153 \ifhmode
154   \unskip\unskip \par
155 \fi
156 \if@newlist
157   \if@nbreak
158     \@nitem
159   \else
160     \addpenalty\@beginparpenalty
161     \addvspace\@topsep
162     \addvspace{-\parskip}%
163   \fi
164 \else
165   \addpenalty\@itempenalty
166   \addvspace\itemsep
167 \fi
168 \global\@inlabeltrue
169 \fi
170 \everypar{%
171   \@minipagefalse
172   \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. `nobreakfalse`, etc etc.

```

173 \if@inlabel
174   \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.

```

175   {\setbox\z@\lastbox
176     \ifvoid\z@
177       \kern-\itemindent
178     \fi}%
179   \box\@labels
180   \penalty\z@
181 \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 `\@nbreak` when it is true is now essential since now it is sometimes set locally.

```

182 \if@nbreak
183   \@nbreakfalse
184   \clubpenalty \@M
185 \else
186   \clubpenalty \@clubpenalty
187   \everypar{}%
188 \fi}%

```

```

189 \if@noitemarg
190 \@noitemargfalse
191 \if@nmbrlist

192 \refstepcounter\@listctr
193 \fi
194 \fi

We use \sbox to support colour commands.
195 \sbox\@tempboxa{\makelabel{#1}}%
196 \global\setbox\@labels\hbox{%
197 \unhbox\@labels
198 \hskip \itemindent
199 \hskip -\labelwidth
200 \hskip -\labelsep
201 \ifdim \wd\@tempboxa >\labelwidth
202 \box\@tempboxa

203 \else
204 \hbox to\labelwidth {\unhbox\@tempboxa}%
205 \fi
206 \hskip \labelsep}%
207 \ignorespaces}

```

`\makelabel`

```

208 \def\makelabel#1{%
209 \latexerror{Lonely \string\item--perhaps a missing
210 list environment}\@ehc}

```

`\@nbitem`

```

211 \def\@nbitem{%
212 \@tempskipa\@outerparskip
213 \advance\@tempskipa -\parskip
214 \addvspace\@tempskipa}

```

`\usecounter`

```

215 \def\usecounter#1{\@nmbrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}

```

56.6 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{\theenumi}
\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 `\bullet`. `\@itemspacing` and `\@itemdepth` are the analogs of `\@enumspacing` and `\@enumdepth`.

```

\enumerate ==
  BEGIN
    if \@enumdepth > 3
      then errormessage: "Too deeply nested".
    else \@enumdepth :=L \@enumdepth + 1
      \@enumctr :=L eval(enum@\romannumeral\the\@enumdepth)
      \list{\label{\@enumctr}}
        {\usecounter{\@enumctr}
          \makelabel{LABEL} == \hss \llap{LABEL}}
    fi
  END

\endenumerate == \endlist

\@enumdepth
216 \newcount\@enumdepth \@enumdepth = 0

\c@enumi
\c@enumii 217 \@definecounter{enumi}
\c@enumii 218 \@definecounter{enumii}
\c@enumiv 219 \@definecounter{enumiii}
220 \@definecounter{enumiv}

enumerate
221 \def\enumerate{%
222   \ifnum \@enumdepth >\thr@@\toodeep\else
223     \advance\@enumdepth\@ne
224     \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
225     \expandafter
226     \list
227       \csname label\@enumctr\endcsname
228       {\usecounter{\@enumctr}\def\makelabel##1{\hss\llap{##1}}}%
229   \fi}
230 \let\endenumerate=\endlist

\itemize ==
  BEGIN
    if \@itemdepth > 3
      then errormessage: 'Too deeply nested'.
    else \@itemdepth :=L \@itemdepth + 1
      \@itemitem ==
eval(labelitem\romannumeral\the\@itemdepth)
      \list{\@nameuse{\@itemitem}}
        {\makelabel{LABEL} == \hss \llap{LABEL}}
    fi

```

```

END

\enditemize == \endlist

\@itemdepth
231 \newcount\@itemdepth \@itemdepth = 0

itemize
232 \def\itemize{%
233   \ifnum \@itemdepth >\thr@@\toodeep\else
234     \advance\@itemdepth\@ne
235     \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%

236     \expandafter
237     \list
238       \csname\@itemitem\endcsname
239       {\def\makelabel##1{\hss\llap{##1}}}%
240   \fi}

241 \let\enditemize =\endlist
242 \</2kernel>

```

File B

ltboxes.dtx

57 L^AT_EX Box commands

<code>\makebox</code>	<code>\makebox[⟨wid⟩][⟨pos⟩]{⟨obj⟩}</code> Puts <code>⟨obj⟩</code> in an <code>\hbox</code> of width <code>⟨wid⟩</code> , positioned by <code>⟨pos⟩</code> . The possible <code>⟨pos⟩</code> are: s stretched, l ushleft, r ushright, c (default) centred. If <code>⟨wid⟩</code> is missing, then <code>⟨pos⟩</code> is also missing and <code>⟨obj⟩</code> is put in an <code>\hbox</code> of its natural width. <code>\makebox(⟨x⟩,⟨y⟩)[⟨pos⟩]{⟨obj⟩}</code> Puts <code>⟨obj⟩</code> in an <code>\hbox</code> of width <code>x*\unitlength</code> and height <code>y*\unitlength</code> . <code>⟨pos⟩</code> arguments are s, l, r or c (default) for stretched, ushleft, ushright 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 <code>\makebox</code> a [b] aligns on the <i>bottom</i> of the text as documented. If you want to align on the <i>baseline</i> use <code>\makebox(,) [b]{\raisebox{0pt}{\height}[0pt]{xyz}}</code> or <code>\makebox(,) [b]{\smash{xyz}}</code>
<code>\mbox</code>	<code>\mbox{⟨obj⟩}</code> The same as <code>\makebox{⟨obj⟩}</code> , but is more efficient as no checking for optional arguments is done.
<code>\newsavebox</code>	<code>\newsavebox{⟨cmd⟩}</code> : If <code>⟨cmd⟩</code> is undefined, then defines it to be a T _E X box register.
<code>\savebox</code>	<code>\savebox{⟨cmd⟩} ...</code> : <code>⟨cmd⟩</code> is defined to be a T _E X box register, and the '...' are any <code>\makebox</code> arguments. It is like <code>\makebox</code> , except it doesn't produce text but saves the value in <code>\box ⟨cmd⟩</code> .
<code>\sbox</code>	<code>\sbox{⟨cmd⟩}{⟨obj⟩}</code> is an efficient abbreviation for <code>\savebox{⟨cmd⟩}{⟨obj⟩}</code> .
<code>\lrbox</code>	<code>\begin{lrbox}{⟨cmd⟩}⟨text⟩\end{lrbox}</code> is equivalent to <code>\sbox{⟨cmd⟩}{⟨text⟩}</code> except that any white space at the beginning and end of <code>⟨text⟩</code> is ignored.
<code>\framebox</code>	<code>\framebox ...</code> : like <code>\makebox</code> , except it puts a 'frame' around the box. The frame is made of lines of thickness <code>\fboxrule</code> , separated by space <code>\fboxsep</code> from the text – except for <code>\framebox(X,Y) ...</code> , where the thickness of the lines is as for the picture environment, and there is no separation added.
<code>\fbox</code>	<code>\fbox{⟨obj⟩}</code> is an abbreviation for <code>\framebox{⟨obj⟩}</code> .
<code>\parbox</code>	<code>\parbox[⟨pos⟩][⟨height⟩][⟨inner-pos⟩]{⟨width⟩}{⟨text⟩}</code> : Makes a box with <code>\hsize ⟨width⟩</code> , positioned by <code>⟨pos⟩</code> as follows: c : <code>\vcenter</code> (placed in <code>\$. . . \$</code> if not in math mode) b : <code>\vbox</code> t : <code>\vtop</code> default value is c. Sets <code>\hsize := ⟨width⟩</code> and calls <code>\@parboxrestore</code> , which does the following: Restores the original definitions of:


```

\par
\\
\--
\'
\'
\=
Resets the following parameters:
\parindent      = 0pt
\parskip        = 0pt
\linewidth      = \hsize
\@totalleftmargin = 0pt
\leftskip       = 0pt
\rightskip      = 0pt
\@rightskip     = 0pt
\parfillskip    = 0pt plus 1fil
\lineskip       = \normallineskip
\baselineskip   = \normalbaselineskip
Calls \sloppy
Note: \@arrayparboxrestore same as \@parboxrestore but it doesn't re-
store \\.
minipage      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[\langle raised \rangle]{\langle width \rangle}{\langle height \rangle} : Makes a \langle width \rangle * \langle height \rangle rule, raised
\langle raised \rangle.
\underline \underline{\langle text \rangle} : Makes an underlined hbox with \langle text \rangle in it.
\raisebox \raisebox{\langle distance \rangle}[\langle height \rangle][\langle depth \rangle]{\langle box \rangle} :
Raises \langle box \rangle up by \langle distance \rangle length (down if \langle distance \rangle negative). Makes TEX
think that the new box extends \langle height \rangle above the line and \langle depth \rangle below, for a
total vertical length of \langle height \rangle + \langle depth \rangle. Default values of \langle height \rangle & \langle depth \rangle =
actual height and depth of box in new position.
1 \*2kernel
2 \message{boxes,}

\makebox \makebox User level command just looks for optional [ or (.
3 \def\makebox{%
4   \leavevmode
5   \@ifnextchar(%)

```

```

6      \@makepicbox
7      {\@ifnextchar[\@makebox\mbox]}

\mbox The basic horizontal box command for LATEX.
8 \long\def\mbox#1{\leavevmode\hbox{#1}}

\@makebox Look for a possible second optional argument (defaults to c).
9 \def\@makebox[#1]{%
10   \@ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}

\@begin@tempboxa Helper macro for supporting \height, \width etc. Grab #1 into \@tempboxa and
measure it.
11 \long\def\@begin@tempboxa#1#2{%
12   \begingroup
13   \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
14   \def\width{\wd\@tempboxa}%
15   \def\height{\ht\@tempboxa}%
16   \def\depth{\dp\@tempboxa}%
17   \let\totalheight\@ovri
18   \totalheight\height
19   \advance\totalheight\depth}

\@end@tempboxa End the group started by \@begin@tempboxa, so that the scope of \height only
includes the 'length' argument to the user-command.
20 \let\@end@tempboxa\endgroup

\bm@c Set up spacing.
\bm@l 21 \def\bm@c{\hss\unhbox\@tempboxa\hss}
\bm@r 22 \def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\bm@s 23 \def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\bm@t 24 \def\bm@s{\unhbox\@tempboxa}
\bm@b
\@imakebox Internal form of \makebox.
25 \long\def\@imakebox[#1][#2]#3{%
26   \@begin@tempboxa\hbox{#3}%
27   \setlength\@tempdima{#1}%      support calc
28   \hb@xt@\@tempdima{\csname bm@#2\endcsname}%
29   \@end@tempboxa}

\@makepicbox Picture mode form of \makebox.
30 \def\@makepicbox(#1,#2){%
31   \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2)[]}}

\@imakepicbox picture mode version
32 \long\def\@imakepicbox(#1,#2)[#3]#4{%
33   \vbox to#2\unitlength
34   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
35    \let\mb@t\vss
36    \@tfor\reserved@a :=#3\do{%
37      \if s\reserved@a
38        \let\mb@l\relax\let\mb@r\relax
39      \else

```

```

40      \expandafter\let\csname mb@\reserved@a\endcsname\relax
41      \fi}%
42      \mb@t
43      \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
44      \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_EXBook. The kern is removed in compatibility mode.

```

45      \kern\z@}}

```

`\set@color` This macro is initially a no-op, but the colour package will redefine it to insert a `\special`.

```

46 \let\set@color\relax

```

`\color@begingroup` These macros are initially a no-op, but the colour package will redefine them to be

`\color@endgroup` `\begingroup`, `\endgroup`, `\begingroup\set@color`,

`\color@setgroup` `\hbox\bgroup\color@begingroup`, `\color@endgroup\egroup`. and *(set to main document colour)* respectively.

`\normalcolor`

`\color@hbox` 47 `\let\color@begingroup\relax`

`\color@vbox` 48 `\let\color@endgroup\relax`

`\color@endbox` 49 `\let\color@setgroup\relax`

50 `\let\normalcolor\relax`

51 `\let\color@hbox\relax`

52 `\let\color@vbox\relax`

53 `\let\color@endbox\relax`

`\newsavebox` Allocate a new ‘savebox’.

```

54 \def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}

```

`\savebox` Save #1 in a box register.

```

55 \def\savebox#1{%
56   \@ifnextchar(%)
57     {\@savepicbox#1}{\@ifnextchar [{\@savebox#1}{\sbox#1}}

```

`\sbox` Save #1 in a box register.

```

58 \long\def\sbox#1#2{\setbox#1\hbox{%
59   \color@setgroup#2\color@endgroup}}

```

`\@savebox` Look for second optional argument.

```

60 \def\@savebox#1[#2]{%
61   \@ifnextchar [{\@isavebox#1[#2]}{\@isavebox#1[#2][c]}

```

`\@isavebox`

```

62 \long\def\@isavebox#1[#2][#3]#4{%
63   \sbox#1{\@imakebox[#2][#3]{#4}}

```

`\@savepicbox` Picture mode version of `\savebox`.

```

64 \def\@savepicbox#1(#2,#3){%
65   \@ifnextchar [%]
66     {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3) []}

```

```

\@isavepicbox Picture mode version of \savebox.
67 \long\def\@isavepicbox#1(#2,#3)[#4]#5{%
68   \sbox#1{\@imakepicbox(#2,#3)[#4]{#5}}}}

\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.
69 \def\lrbox#1{%
70   \edef\reserved@a{%
71     \endgroup
72     \setbox#1\hbox{%
73       \begingroup\aftergroup}%
74     \def\noexpand\@currenvir{\@currenvir}%
75     \def\noexpand\@currencline{\on@line}}%
76   \reserved@a
77   \@endpefalse
78   \color@setgroup
79   \ignorespaces}

\endlrbox End the lrbox environment.
80 \def\endlrbox{\unskip\color@endgroup}

\usebox unchanged
81 \def\usebox#1{\leavevmode\copy #1\relax}

\frame The following definition of \frame was written by Pavel Curtis (Extra space
removed 14 Jan 88) RmS 92/08/24: Replaced occurrence of \@halfwidth by
\@wholewidth
82 \long\def\frame#1{%
83   \leavevmode
84   \hbox{%
85     \hskip-\@wholewidth
86     \vbox{%
87       \vskip-\@wholewidth
88       \hrule \@height\@wholewidth
89       \hbox{%
90         \vrule\@width\@wholewidth
91         #1%
92         \vrule\@width\@wholewidth}%
93       \hrule\@height\@wholewidth
94       \vskip-\@wholewidth}%
95     \hskip-\@wholewidth}}

\fbxrule user level parameters,
\fbxsep 96 \newdimen\fbxrule
97 \newdimen\fbxsep

\fbx Abbreviated framed box command.
98 \long\def\fbx#1{%
99   \leavevmode
100   \setbox\@tempboxa\hbox{%
101     \color@begingroup

```

```

102     \kern\fboxsep{#1}\kern\fboxsep
103     \color@endgroup}%
104     \@frameb@x\relax}

\framebox    Framed version of \makebox.
105 \def\framebox{%
106     \@ifnextchar(%)
107     \@framepicbox{\@ifnextchar[\@framebox\fbox]}

\@framebox    Deal with optional arguments.
108 \def\@framebox[#1]{%
109     \@ifnextchar[%
110     {\@ifframebox[#1]}%
111     {\@ifframebox[#1][c]}}

\ifframebox    The handling the optional arguments.  In order to set the whole box, including
the frame to the specified dimension, we first determine that dimension from the
natural size of the text, #3.  calculated width.
112 \long\def\@ifframebox[#1][#2]#3{%
113     \leavevmode
114     \@begin@tempboxa\hbox{#3}%
115     \setlength\@tempdima{#1}%
116     \setbox\@tempboxa\hb@xt@\@tempdima
117         {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
118     \@frameb@x{\kern-\fboxrule}%
119     \@end@tempboxa}

\@frameb@x    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.
120 \def\@frameb@x#1{%
121     \@tempdima\fboxrule
122     \advance\@tempdima\fboxsep
123     \advance\@tempdima\dp\@tempboxa
124     \hbox{%
125         \lower\@tempdima\hbox{%
126             \vbox{%
127                 \hrule\@height\fboxrule
128                 \hbox{%
129                     \vrule\@width\fboxrule
130                     #1%
131                     \vbox{%
132                         \vskip\fboxsep
133                         \box\@tempboxa
134                         \vskip\fboxsep}%
135                     #1%
136                     \vrule\@width\fboxrule}%
137                 \hrule\@height\fboxrule}%
138                 }%
139     }%
140 }

\@framepicbox    Picture mode version.
141 \def\@framepicbox(#1,#2){%
142     \@ifnextchar[{\@ifframepicbox(#1,#2)}{\@ifframepicbox(#1,#2)[]}]

```

`\@iframepicbox` Picture mode version.

```
143 \long\def\@iframepicbox(#1,#2)[#3]#4{%  
144   \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

`\parbox` The main vertical-box command for L^AT_EX.

```
145 \def\parbox{%  
146   \@ifnextchar[%]  
147     \@iparbox  
148     {\@iiiparbox c\relax[s]}}
```

`\@iparbox` Optional argument handling.

```
149 \def\@iparbox[#1]{%  
150   \@ifnextchar[%]  
151     {\@iiiparbox{#1}}%  
152     {\@iiiparbox{#1}\relax[s]}}
```

`\@iiiparbox` Optional argument handling.

```
153 \def\@iiiparbox#1[#2]{%  
154   \@ifnextchar[%]  
155     {\@iiiparbox{#1}{#2}}%  
156     {\@iiiparbox{#1}{#2}[#1]}}
```

`\@iiiparbox` The internal version of `\parbox`.

```
\@parboxto 157 \let\@parboxto\@empty  
158 \long\def\@iiiparbox#1#2[#3]#4#5{%  
159   \leavevmode  
160   \@pboxswfalse  
161   \setlength\@tempdima{#4}%  
162   \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\@@par}%  
163   \ifx\relax#2\else  
164     \setlength\@tempdimb{#2}%  
165     \def\@parboxto{to\@tempdimb}%  
166     \fi  
167     \if#1b\vbox  
168       \else\if #1t\vtop  
169         \else\ifmmode\vcenter  
170           \else\@pboxswtrue $\vcenter  
171           \fi\fi\fi  
172     \@parboxto{\let\hss\vss\let\unhbox\unvbox  
173               \csname bm@#3\endcsname}%  
174     \if@pboxsw \m@th$\fi  
175     \@end@tempboxa}
```

`\@arrayparboxrestore` Restore various paragraph parameters.

The rationale for allowing two normally global ags to be set locally here was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these ags 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 ags should be treated as in `\set@nobeak`; otherwise this command will be redundant.

```

176 \def\@arrayparboxrestore{%
177   \let\if@nobreak\iffalse
178   \let\if@noskipsec\iffalse
179   \let\par\@par
180   \let\-\@dischyph

```

Redefined accents to allow changes in font encoding

```

181   \let'\@acci\let'\@accii\let\=\@acciii
182   \parindent\z@ \parskip\z@skip
183   \everypar{%
184     \linewidth\hsize
185     \@totalleftmargin\z@
186     \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
187     \parfillskip\@flushglue \lineskip\normallineskip
188     \baselineskip\normalbaselineskip
189     \sloppy}

```

\parboxrestore Restore various paragraph parameters, and also \.

```

190 \def\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}

```

\if@minipage Switch that is true at the start of a minipage.

```

191 \def\@minipagefalse{\global\let\if@minipage\iffalse}
192 \def\@minipagetrue {\global\let\if@minipage\iftrue}
193 \@minipagefalse

```

\minipage Essentially an environment form of \parbox.

```

194 \def\minipage{%
195   \@ifnextchar[%]
196     \@iminipage
197     {\@iiiminipage c\relax[s]}}

```

\@iminipage Optional argument handling.

```

198 \def\@iminipage[#1]{%
199   \@ifnextchar[%]
200     {\@iiiminipage{#1}}%
201     {\@iiiiminipage{#1}\relax[s]}}

```

\@iiiminipage Optional argument handling.

```

202 \def\@iiiminipage#1[#2]{%
203   \@ifnextchar[%]
204     {\@iiiiminipage{#1}{#2}}%
205     {\@iiiiminipage{#1}{#2}[#1]}}

```

\@iiiiminipage Internal form of minipage.

```

206 \def\@iiiiminipage#1#2[#3]#4{%
207   \leavevmode
208   \@pboxswfalse
209   \setlength\@tempdima{#4}%
210   \def\@mpargs{{#1}{#2}[#3]{#4}}%
211   \setbox\@tempboxa\vbox\bgroup
212     \color@begingroup
213       \hsize\@tempdima
214       \textwidth\hsize \columnwidth\hsize

```

```

215     \@parboxrestore
216     \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
217     \let\@footnotetext\@mpfootnotetext
218     \let\@listdepth\@mplistdepth \@mplistdepth\z@
219     \@minipagerestore
220     \@setminipage}

\@minipagerestore Hook so that other styles can reset other commands in a minipage.
221 \let\@minipagerestore=\relax

\endminipage
222 \def\endminipage{%
223     \par
224     \unskip
225     \ifvoid\@mpfootins\else
226         \vskip\skip\@mpfootins
227         \normalcolor
228         \footnoterule
229         \unvbox\@mpfootins
230     \fi
231     \@minipagefalse    %% added 24 May 89
232     \color@endgroup
233     \egroup
234     \expandafter\@iiparbox\@mpargs{\unvbox\@tempboxa}}

\@mplistdepth Versions of \@listdepth and \footins local to minipage.
\@mpfootins 235 \newcount\@mplistdepth
236 \newinsert\@mpfootins

\@mpfootnotetext Minipage version of \@footnotetext.
                Final \strut added 27 Mar 89, on suggestion by Don Hosek
237 \long\def\@mpfootnotetext#1{%
238     \global\setbox\@mpfootins\vbox{%
239         \unvbox\@mpfootins
240         \reset@font\footnotesize
241         \hsize\columnwidth
242         \@parboxrestore
243         \protected@edef\@currentlabel
244             {\csname p@mpfootnote\endcsname\@thefnmark}%
245         \color@begingroup
246             \@makefntext{%
247                 \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
248             \color@endgroup}}

249 \newif\if@pboxsw

\@rule Draw a rule of the specified size.
250 \def\@rule{\@ifnextchar[\@rule{\@rule[\z@]}}

\@rule Internal form of \@rule.
251 \def\@rule[#1]#2#3{%
252     \leavevmode
253     \hbox{%

```



```

254     \setlength\@tempdima{#1}%
255     \setlength\@tempdimb{#2}%
256     \setlength\@tempdimc{#3}%
257     \advance\@tempdimc\@tempdima
258     \vrule\@width\@tempdimb\@height\@tempdimc\@depth-\@tempdima}}

@@underline Saved primitive \underline.
259 \let\@@underline\underline

\underline LATEX version works outside math.
260 \def\underline#1{%
261     \relax
262     \ifmmode\@@underline{#1}%
263     \else $\@@underline{\hbox{#1}}\m@th$\relax\fi}

\raisebox Raise a box, and change its vertical dimensions.
264 \def\raisebox#1{%
265     \leavevmode
266     \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]]}

\@rsbox Optional argument handling.
267 \def\@rsbox#1[#2]{%
268     \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}

\@argsbox ...

\@irsbox Internal version of \raisebox (less than two optional args).
269 \long\def\@irsbox#1[#2]#3{%
270     \@begin@tempboxa\hbox{#3}%
271     \setlength\@tempdima{#1}%
272     \ifx\@#2\\\else\setlength\@tempdimb{#2}\fi
273     \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
274     \ifx\@#2\\\else\ht\@tempboxa\@tempdimb\fi
275     \box\@tempboxa
276     \@end@tempboxa}

\@iirsbox Internal version of \raisebox (two optional args).
277 \long\def\@iirsbox#1[#2][#3]#4{%
278     \@begin@tempboxa\hbox{#4}%
279     \setlength\@tempdima{#1}%
280     \setlength\@tempdimb{#2}%
281     \setlength\dimen@{#3}%
282     \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
283     \ht\@tempboxa\@tempdimb
284     \dp\@tempboxa\dimen@
285     \box\@tempboxa
286     \@end@tempboxa}

\@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.

```
287 \def\@finalstrut#1{%
288   \unskip\ifhmode\nobreak\fi\vrule\@width\z@\@height\z@\@depth\dp#1}
```

57.1 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 289 \def\@@line{\hb@xt@\hsize}
\centerline 290 \def\leftline#1{\@@line{#1\hss}}
           \@@line 291 \def\rightline#1{\@@line{\hss#1}}
           292 \def\centerline#1{\@@line{\hss#1\hss}}

\rlap      These macros place text to the left or right of the current reference point without
\llap      taking up space.
           293 \def\rlap#1{\hb@xt@\z@{#1\hss}}
           294 \def\llap#1{\hb@xt@\z@{\hss#1}}

           295 </2ekernel>
```

File C

lftab.dtx

58 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.

58.1 tabbing

`\dimen{\@firsttab + i}` = distance of tab stop `i` from left margin
0 <= `i` <= 15 (?).

`\dimen\@firsttab` is initialized to `\@totalleftmargin`, 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 := \@totalleftmargin + \linewidth
      \hb@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
  \@hightab := \@nxttabmar :=G \@firsttab
  \@tabpush :=G 0
  \dimen\@firsttab := \@totalleftmargin
  @rjfield :=G F
  \trivlist \item\relax
  if @minipage = F then \vskip \parskip fi
  \box\@tabfbox = \rlap{\indent\the\everypar}
    % note: \the\everypar sets @inlabel :=G F
  \@itemfudge == BEGIN \box\@tabfbox END
  \@startline
  \ignorespaces
  END

\@endtabbing ==
BEGIN
  \@stopline
  if \@tabpush > 0 then error message: "unmatched \poptabs" fi
  \endtrivlist
  END

\@rtab ==
BEGIN
  \@stopfield
  \@addfield
  if \@curtab < \@hightab
    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 untab"      fi
    else error message "Left tab in middle of line"
    fi
END

\@tabplus ==
BEGIN
  if \@nxxtabmar < \@hightab
    then \@nxxtabmar :=G \@nxxtabmar+1
    else error message "Undefined tab"
    fi
END

\@tabminus ==
BEGIN
  if \@nxxtabmar > \@firsttab
    then \@nxxtabmar :=G \@nxxtabmar-1
    else error message "Too many untab"
    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
\beginingroup
\@contfield
END

\poptabs ==
BEGIN
\@stopfield
if \@tabpush > 0
then \endgroup
\@tabpush :=G \@tabpush - 1
else error message: "Too many \poptabs"
fi
\@contfield
END

```

\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.

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 <2ekernel>\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion
```

```

\@firsttab
\@maxtab 2 <*2ekernel | autoload>
3 \newdimen\@gtempa
4 \chardef\@firsttab=\the\allocationnumber
5 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
6 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
7 \newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
8 \newdimen\@gtempa
9 \chardef\@maxtab=\the\allocationnumber
10 \dimen\@firsttab=0pt

```

```

\@nxttabmar
\@curtabmar 11 \newcount\@nxttabmar
\@curtab 12 \newcount\@curtabmar
\@hightab 13 \newcount\@curtab
\@tabpush 14 \newcount\@hightab
15 \newcount\@tabpush

\@curline
\@curfield 16 \newbox\@curline
\@tabfbox 17 \newbox\@curfield
18 \newbox\@tabfbox

19 </2ekernel | autoloading>
20 <*2ekernel | def>

\if@rjfield
21 \newif\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 aw which is not going to be corrected now.
22 \gdef\@startline{%
23 \ifnum \@nxttabmar >\@hightab
24 \@badtab
25 \global\@nxttabmar \@hightab
26 \fi
27 \global\@curtabmar \@nxttabmar
28 \global\@curtab \@curtabmar
29 \global\setbox\@curline \hbox {}%
30 \@startfield
31 \strut}

\@stopline
32 \gdef\@stopline{%
33 \unskip
34 \@stopfield
35 \if@rjfield
36 \global\@rjfieldfalse
37 \@tempdima\@totalleftmargin
38 \advance\@tempdima\linewidth
39 \hb@xt@\@tempdima{%
40 \itemfudge\hskip\dimen\@curtabmar
41 \box\@curline
42 \hfil
43 \box\@curfield}%
44 \else
45 \@addfield
46 \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}%
47 \fi}

\@startfield
48 \gdef\@startfield{%
49 \global\setbox\@curfield\hbox\bgroup\color@begingroup}

```



```

\@stopfield
50 \gdef\@stopfield{%
51   \color@endgroup\egroup}

\@contfield
52 \gdef\@contfield{%
53   \global\setbox\@curfield\hbox\bgroup\color@begingroup
54   \unhbox\@curfield}

\@addfield
55 \gdef\@addfield{\global\setbox\@curline\hbox{\unhbox
56   \@curline\unhbox\@curfield}}

\@ifatmargin
57 \gdef\@ifatmargin{\ifdim \wd\@curline =\z@}

\@tabcr
58 \gdef\@tabcr{\@stopline \@ifstar{\penalty \@M \@xtabcr}\@xtabcr}

\@xtabcr
59 \gdef\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}

\@itabcr
60 \gdef\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
61 \gdef\kill{\@stopfield\@startline\ignorespaces}

\tabbing We use \relax to prevent \item from scanning too far.
62 \gdef\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\@=\@settab
63   \let\+\@tabplus\let\-\@tabminus\let\'\@tabrj\let\'\@tablab
64   \let\@=\@tabcr
65   \@hightab\@firsttab
66   \global\@nxttabmar\@firsttab
67   \dimen\@firsttab\@totalleftmargin
68   \global\@tabpush\z@ \global\@rjfieldfalse
69   \trivlist \item\relax
70   \if@minipage\else\vskip\parskip\fi
71   \setbox\@tabfbox\hbox{%
72     \rlap{\hskip\@totalleftmargin\indent\the\everypar}}%
73   \def\@itemfudge{\box\@tabfbox}%
74   \@startline\ignorespaces}

\endtabbing
75 \gdef\endtabbing{%
76   \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}

\@rtab Omitted \global added to \@rtab 17 Jun 86
77 \gdef\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
78   \global\advance\@curtab \@ne \else\@badtab\fi
79   \@tempdima\dimen\@curtab
80   \advance\@tempdima -\dimen\@curtabmar
81   \advance\@tempdima -\wd\@curline
82   \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
83   \@startfield\ignorespaces}

```

```

\@settab
84 \gdef\@settab{\@stopfield\@addfield
85   \ifnum \@curtab <\@maxtab
86     \ifnum\@curtab =\@hightab
87       \advance\@hightab \@ne
88     \fi
89     \global\advance\@curtab \@ne
90   \else
91     \latexerror{Tab overflow}\@ehd
92   \fi
93   \dimen\@curtab \dimen\@curtabmar
94   \advance\dimen\@curtab \wd\@curline
95   \@startfield
96   \ignorespaces}

\@ltab
97 \gdef\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
98   \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
99   \@badtab\fi\else
100   \latexerror{\string<\space in mid line}\@ehd\fi\ignorespaces}

\@tabplus
101 \gdef\@tabplus{%
102   \ifnum\@nxttabmar<\@hightab
103     \global\advance\@nxttabmar\@ne
104   \else
105     \@badtab
106   \fi
107   \ignorespaces}

\@tabminus
108 \gdef\@tabminus{%
109   \ifnum\@nxttabmar>\@firsttab
110     \global\advance\@nxttabmar\m@ne
111   \else
112     \@badtab
113   \fi
114   \ignorespaces}

\@tabrj
115 \gdef\@tabrj{%
116   \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}

\@tablab \setbox\@curline made \global in \@tablab. 17 Jun 86
117 \gdef\@tablab{%
118   \@stopfield
119   \global\setbox\@curline\hbox{%
120     \box\@curline
121     \hskip-\wd\@curfield \hskip-\tabbingsep
122     \box\@curfield
123     \hskip\tabbingsep}%
124   \@startfield
125   \ignorespaces}

```

`\pushtabs`

```
126 \gdef\pushtabs{%
127   \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
128     \@contfield}
```

`\poptabs` It is, in some sense, an error if, after the endgroup, the current tab setting is higher than the new value of `\@hightab` (which is a local variable). That this is allowed is a fundamental design aw which is not going to be corrected now.

```
129 \gdef\poptabs{\@stopfield\@addfield
130   \ifnum \@tabpush >\z@
131     \endgroup
132     \global\advance\@tabpush \m@ne
133     \ifnum \@curtab >\@hightab
134       \global \@curtab \@hightab
135       \@badtab
136     \fi
137   \else
138     \@badpoptabs
139   \fi
140   \@contfield}

141 </2ekernel | def>
```

`\tabbingsep`

```
142 <*2ekernel | autoload>
143 \newdimen\tabbingsep
144 </2ekernel | autoload>
```

`\tabbing`

```
145 <*autoload>
146 \def\tabbing{\@autoload{tabg}\tabbing}
147 </autoload>
```

58.2 array and tabular environments

ARRAY PARMETERS:

`\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 \crrc } END

\tabular ==
  BEGIN
    \@halignto == NULL
    \@tabular
  END
```

```

\tabular*{WIDTH} ==
  BEGIN
    \@halignto == to WIDTH
    \@tabular
  END

\@tabular ==
  BEGIN
    \leavevmode
    \hbox { $
      \@acol == \@tabacol
      \@classz == \@tabclassz
      \@classiv == \@tabclassiv
      \\ == \@tabularcr
    \@tabarray
  END

\endtabular == BEGIN \crrc}} $} 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.
148 \<*2kernel | autoload)

\extracolsep

149 \def\extracolsep#1{\tabskip #1\relax}

\array

150 \def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
151 \let\@classiv\@arrayclassiv
152 \let\\ \@arraycr \let\@halignto\@empty\@tabarray}

\endarray
\endtabular 153 \def\endarray{\crcr\egroup\egroup}
\endtabular* 154 \def\endtabular{\crcr\egroup\egroup $\egroup}
155 \expandafter \let \csname endtabular*\endcsname = \endtabular

\tabular

156 \def\tabular{\let\@halignto\@empty\@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.
157 \namedef{tabular*}#1{%
158 \setlength\dimen@{#1}%
159 \edef\@halignto{to\the\dimen@}\@tabular}

\@tabular

160 \def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
161 \let\@classz\@tabclassz
162 \let\@classiv\@tabclassiv \let\\ \@tabularcr \@tabarray}

\@tabarray RmS 91/11/04 added \m@th.
163 \def\@tabarray{\m@th@ifnextchar[\@array{\@array[c]}}

RmS 1993/11/03 changed \halign to \ialign and removed superuous
\tabskip assignment

```

`\@array`

```

164 \def\@array[#1]#2{%
165   \if #1t\top \else \if#1b\vbox \else \vcenter \fi\fi

166   \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.

```

167   \setbox\@arstrutbox\hbox{%
168     \vrule \@height\arraystretch\ht\strutbox
169           \@depth\arraystretch \dp\strutbox
170           \@width\z@}%
171   \@mkpream{#2}%
172   \edef\@preamble{%
173     \ialign \noexpand\@halignto
174       \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%

```

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.

```

175   \let\@startpbox\@startpbox \let\@endpbox\@endpbox
176   \let\tabularnewline\\
177   \let\par\@empty
178   \let\@sharp##%
179   \set@typeset@protect
180   \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).

```

181   \ifhmode \@preamerr\z@ \@@par\fi
182   \@preamble}

```

`\@arraycr` Array version of `\\`.

```

183 \def\@arraycr{%
184   ${\ifnum0=}\fi\@ifstar\@xarraycr\@arraycr}

```

`\@arraycr`

```

185 \def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0='{ \fi}$\}\cr}}

```

`\@argarraycr`

```

186 \def\@argarraycr[#1]{%
187   \ifnum0='{ \fi}$\}\ifdim #1>\z@ \@xargarraycr{#1}\else
188     \@yargarraycr{#1}\fi}

```

```

\tabularnewline Tabular version of \\.
189 \let\tabularnewline\relax

\@tabularcr
190 \def\@tabularcr{%
191   {\ifnum0='}\fi\@ifstar\@tabularcr\@xtabularcr}

\@xtabularcr
192 \def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0='{ \fi}\cr}}

\@argtabularcr
193 \def\@argtabularcr[#1]{%
194   \ifnum0='{ \fi}%
195   \ifdim #1>\z@
196     \unskip\@xargarraycr{#1}%
197   \else
198     \@yargarraycr{#1}%
199   \fi}

\@xargarraycr
200 \def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
201   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}

\@yargarraycr
202 \def\@yargarraycr#1{\cr\noalign{\vskip #1}}

\multicolumn \multicolumn{NUMBER}{FORMAT}{ITEM} ==
BEGIN
\multispan{NUMBER}
\begingroup
\@addamp == null
\@mkpream{FORMAT}
\@sharp == ITEM
\protect == \relax
\@startpbox == \@@startpbox
\@endpbox == \@@endpbox
\@arstrut
\@preamble
\endgroup
END

```

The command `\def\@addamp{}` 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.


```

203 \long\def\multicolumn#1#2#3{\multispan{#1}\begingroup
204   \mkpream{#2}%
205   \def\@sharp{#3}\set@typeset@protect
206   \let\@startpbox\@startpbox\let\@endpbox\@endpbox
207   \@arstrut \preamble\hbox{}\endgroup\ignorespaces}

```

Codes for classes and character numbers of array, tabular and multicolumn arguments.

Character	Class	Number
c	0	0
l	0	1
r	0	2
l	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 \@firstamp = true then \@firstamp := false
                                     else &                               fi
END

```

```

\@mkpream TOKENLIST ==
BEGIN
  \@firstamp      := 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 ->                               % l
    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$\relax\@sharp$
    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 \@arrayclassiv 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 \@expast S *0x\@0 END

```

```

\@xexpast S1 *{N}{S2} S3 \@@ ==
BEGIN
  \reserved@a := S1
  \@tempcnta := N
  if \@tempcnta > 0
    then while \@tempcnta > 0 do \reserved@a := \reserved@a S2
      \@tempcnta := \@tempcnta - 1 od
    \reserved@b == \@xexpast
  else \reserved@b == \@xexnoop
  fi
  \expandafter \reserved@b \reserved@a S3 \@@
END

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

\@xexpast
209 \def\@xexpast#1{\@xexpast #1*0x\@{ }

\@xexpast
210 \def\@xexpast#1*#2#3#4\@{ %
211   \edef\reserved@a{#1}%
212   \@tempcnta#2\relax
213   \ifnum\@tempcnta>z@
214     \@whilenum\@tempcnta>z@\do
215       {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
216     \let\reserved@b\@xexpast
217   \else
218     \let\reserved@b\@xexnoop
219   \fi
220   \expandafter\reserved@b\reserved@a #4\@{ }

\if@firstamp
  \@addamp 221 \newif\if@firstamp
222 \def\@addamp{%
223   \if@firstamp
224     \@firstampfalse
225   \else
226     \edef\@preamble{\@preamble &}%
227   \fi}

\@arrayacol
  \@tabacol 228 \def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
  \@ampacol 229 \def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\@acolampacol 230 \def\@ampacol{\@addamp \@acol}
231 \def\@acolampacol{\@acol\@addamp\@acol}

\@mkpream
232 \def\@mkpream#1{\@firstamptrue\@lastchclass6
233   \let\@preamble\@empty
234   \let\protect\@unexpandable@protect

```

```

235 \let\@sharp\relax
236 \let\@startpbox\relax\let\@endpbox\relax
237 \@expast{#1}%
238 \expandafter\@tfor \expandafter
239   \@nextchar \expandafter:\expandafter=\reserved@a\do
240   {\@testpach\@nextchar
241     \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
242       \or \@classiv \or \@classv \fi\@lastchclass\@chclass}%
243 \ifcase \@lastchclass \@acol
244   \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}

```

\@arrayclassz

```

245 \def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
246   \or \or \@addamp \or
247   \@acolampacol \or \@firstampfalse \@acol \fi
248 \edef\@preamble{\@preamble
249   \ifcase \@chnum
250     \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
251     \or \hfil$\relax\@sharp$\fi}}

```

\@tabclassz RmS 91/08/14 inserted extra braces around entry for NFSS

```

252 \def\@tabclassz{%
253   \ifcase\@lastchclass
254     \@acolampacol
255   \or
256     \@ampacol
257   \or
258   \or
259   \or
260     \@addamp
261   \or
262     \@acolampacol
263   \or
264     \@firstampfalse\@acol
265   \fi
266 \edef\@preamble{%
267   \@preamble{%
268     \ifcase\@chnum
269       \hfil\ignorespaces\@sharp\unskip\hfil
270     \or
271       \hskip1sp\ignorespaces\@sharp\unskip\hfil
272     \or
273       \hfil\hskip1sp\ignorespaces\@sharp\unskip
274     \fi}}}

```

\@classi

```

275 \def\@classi{%
276   \ifcase\@lastchclass
277     \@acol\@arrayrule
278   \or
279     \addtopreamble{\hskip \doublerulesep}\@arrayrule
280   \or
281   \or

```

```

282 \or
283 \arrayrule
284 \or
285 \@acol\arrayrule
286 \or
287 \arrayrule
288 \fi}

\@classii
289 \def\@classii{%
290 \ifcase\@lastchclass
291 \or
292 \addtopreamble{\hskip .5\arrayrulewidth}%
293 \fi}

\@classiii
294 \def\@classiii{\ifcase \@lastchclass \@acolampacol \or
295 \addamp\@acol \or
296 \or \or \addamp \or
297 \@acolampacol \or \ampacol \fi}

\@tabclassiv
298 \def\@tabclassiv{\@addtopreamble\@nextchar}

\@arrayclassiv
299 \def\@arrayclassiv{\@addtopreamble{$\@nextchar$}}

\@classv
300 \def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
301 \@sharp\@endpbox}}

\@addtopreamble
302 \def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}

\@chclass
\@lastchclass 303 \newcount\@chclass
\@chnum 304 \newcount\@lastchclass
305 \newcount\@chnum

\arraycolsep
\tabcolsep 306 \newdimen\arraycolsep
\arrayrulewidth 307 \newdimen\tabcolsep
\doublerulesep 308 \newdimen\arrayrulewidth
309 \newdimen\doublerulesep

\arraystretch
310 \def\arraystretch{1} % Default value.

\@arstrutbox
\@arstrut 311 \newbox\@arstrutbox

```

```

312 \def\@arstrut{%
313   \relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}

\@arrayrule
314 \def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
315   \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}

\@testpatch
316 \def\@testpatch#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
317   \ifnum \@lastchclass=3 5 \else
318     \z@ \if #1c\@chnum \z@ \else
319       \if #1l\@chnum \@ne \else
320       \if #1r\@chnum \tw@ \else
321       \@chclass \if #1|\@ne \else
322       \if #1@\tw@ \else
323       \if #1p3 \else \z@ \@preamerr 0\fi
324   \fi \fi \fi \fi \fi \fi
325 \fi}

\hline
326 \def\hline{%
327   \noalign{\ifnum0='}\fi\hrule \@height \arrayrulewidth \futurelet
328   \reserved@a\@xhline}

\@xhline
329 \def\@xhline{\ifx\reserved@a\hline
330   \vskip\doublerulesep
331   Measure from the middle of the rules.
332   \vskip-\arrayrulewidth
333   \fi
334   \ifnum0='{ \fi}}

\vline
335 \def\vline{\vrule \@width \arrayrulewidth}

\cline
\@cline The old LATEX 2.09 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 ltplain.dtx.
These counters are no longer declared.

\newcount\@cla
\newcount\@clb
336 \def\cline#1{\@cline#1\@nil}
337 \omit

Use the counter from \multispan.
338 \@multicnt#1%
339 \advance\@multispan\m@ne
340 \ifnum\@multicnt=\@ne\@firstofone{&\omit}\fi

```

```

341 \multicnt#2%
342 \advance\multicnt-#1%
343 \advance\multispan\@ne
    The original had \unskip at this point, but how could a skip get here ???
344 \leaders\hrule\@height\arrayrulewidth\hfill
345 \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.
346 \noalign{\vskip-\arrayrulewidth}}

\mscount The \mscount counter is no longer declared, saving a csname and a register. It is
          declared in compatibility mode.

\multispan Modify \multispan slightly from its plain TEX definition to allow more efficient
\@multispan code sharing with \multicolumn. Also share a count register with \multiput.
    \sp@n 347 \def\multispan{\omit\@multispan}
          348 \def\@multispan#1{%
          349 \multicnt#1\relax
          350 \loop\ifnum\multicnt>\@ne \sp@n\repeat}
          351 \def\sp@n{\span\omit\advance\multicnt\m@ne}

\@startpbox Helper macros for ‘p’ columns.
\@endpbox   \@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)
352 \def\@startpbox#1{\vtop\bgroup \setlength\hsize{#1}\@arrayparboxrestore}
353 \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.

\@@startpbox
\@@endpbox 354 \let\@@startpbox=\@startpbox
           355 \let\@@endpbox=\@endpbox

           356 </2ekernel | autoload>

```


File D

ltpictur.dtx

59 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.

<code>\unitlength</code>	= value of dimension argument
<code>\@wholewidth</code>	= current line width
<code>\@halfwidth</code>	= half of current line width
<code>\@linefnt</code>	= font for drawing lines
<code>\@circlefnt</code>	= 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.

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 | autoload>
3 \newdimen\@wholewidth
4 \newdimen\@halfwidth

```

```
\unitlength
```

```
5 \newdimen\unitlength \unitlength =1pt
```

```
\@picbox
```

```

\@picht 6 \newbox\@picbox
7 \newdimen\@picht
8 </2ekernel | autoload>

```

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

```
\pictur@ #1 should be a ( (eating any white space before the bracket),
```

```

9 <*2ekernel | def>
10 \long\gdef\picture#1{\pictur@#1}
11 \gdef\pictur@(#1){%
12 \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}

```

```

13 </2ekernel | def>
14 <*autoload>
15 \def\pictur@{\@autoload{pict}}
16 \def\picture{\pictur@\picture}
17 </autoload>

\@picture
18 <*2ekernel | def>
19 \gdef\@picture(#1,#2)(#3,#4){%
20   \@picht#2\unitlength
21   \setbox\@picbox\hb@xt@#1\unitlength\bgroup
22     \hskip -#3\unitlength
23     \lower #4\unitlength\hbox\bgroup
24     \ignorespaces}

\endpicture
25 \gdef\endpicture{%
26   \egroup\hss\egroup
27   \ht\@picbox\@picht\dp\@picbox\z@
28   \mbox{\box\@picbox}}

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).

29 \long\gdef\put(#1,#2)#3{%
30   \@killglue\raise#2\unitlength
31   \hb@xt@\z@{\kern#1\unitlength #3\hss}%
32   \ignorespaces}

\multiput #3 had better be a (.
33 \gdef\multiput(#1,#2)#3{%
34   \@xdim #1\unitlength
35   \@ydim #2\unitlength
36   \@multiput{ }

\multiput
37 \long\gdef\@multiput(#1,#2)#3#4{%
38   \@killglue\@multicnt #3\relax
39   \@whilenum \@multicnt >\z@\do
40     {\raise\@ydim\hb@xt@\z@{\kern\@xdim #4\hss}%
41     \advance\@multicnt@m@ne
42     \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
43   \ignorespaces}

\@killglue
44 \gdef\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}
45 </2ekernel | def>

\thinlines
\thicklines 46 <*2ekernel | def>
47 \gdef\thinlines{\let\@linefnt\tenln \let\@circlefnt\tencirc
48   \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
49 \gdef\thicklines{\let\@linefnt\tenlnw \let\@circlefnt\tencircw

```

```

50 \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
51 \</2ekernel | def>
52 \*autoload>
53 \def\thinlines{\pictur@\thinlines}
54 \def\thicklines{\pictur@\thicklines}
55 \</autoload>

\linethickness

56 \*2ekernel | def>
57 \gdef\linethickness#1{\@wholewidth #1\relax \@halfwidth .5\@wholewidth}
58 \</2ekernel | def>
59 \*autoload>
60 \def\linethickness{\pictur@\linethickness}
61 \</autoload>

\ishortstack

62 \*2ekernel | def>
63 \gdef\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}

\@ishortstack

64 \gdef\@shortstack[#1]{%
65 \leavevmode
66 \vbox\bgroup
67 \baselineskip-\p@\lineskip 3\p@
68 \let\mb@l\hss\let\mb@r\hss
69 \expandafter\let\csname mb@#1\endcsname\relax
70 \let\\\@stackcr
71 \@ishortstack}

\@ishortstack

72 \gdef\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\cr}\egroup}

\@stackcr
\@ixstackcr
73 \gdef\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
74 \gdef\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}

\@istackcr

75 \gdef\@istackcr[#1]{\cr\noalign{\vskip #1}\ignorespaces}

\line(X,Y){LEN} ==
BEGIN
\@xarg := X
\@yarg := Y
\@linelen := LEN * \unitlength
if \@xarg = 0
then \@vline
else if \@yarg = 0
then \@hline
else \@sline
if

```

```

END

\@sline ==
BEGIN
  if \@xarg < 0
    then @negarg := T
         \@xarg := -\@xarg
         \@yyarg := -\@yyarg
    else @negarg := F
         \@yyarg := \@yyarg
    fi
  \@tempcnta := |\@yyarg|
  if \@tempcnta > 6
    then error: 'LATEX ERROR: Illegal \line or \vector argument.'
         \@tempcnta := 0
    fi
  \box\@linechar := \hbox{\@linefont \@getlinechar(\@xarg,\@yyarg)
}
    if \@yarg > 0 then \@upordown = \raise
                        \@clnht := 0
    else \@upordown = \lower
         \@clnht := height of \box\@linechar
    fi
  \@clnwd := width of \box\@linechar
  if @negarg
    then \hskip - width of \box\@linechar
         \reserved@a == \hskip - 2* width of box \@linechar
    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
  \@tempcnta := \@tempdima / width of \box\@linechar
  \@tempdima := (\@tempcnta * 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
    \@tempcnta := 8*X - 9
    if Y > 0
        then \@tempcnta := \@tempcnta + Y
        else \@tempcnta := \@tempcnta - Y + 64
    fi
    \char\@tempcnta
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
        fi
    fi
END

\@hvector ==
BEGIN
    \@hline
    {\@linefont 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
           {\@linefnt
            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

\if@negarg

76 \newif\if@negarg

\line

```
77 \gdef\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
78 \@linelen #3\unitlength
79 \ifdim\@linelen<\z@ \badlinearg\else
80   \ifnum\@xarg =\z@ \vline
81   \else \ifnum\@yarg =\z@ \hline \else \sline\fi
82   \fi
83 \fi}
```

\@sline

```
84 \gdef\@sline{%
85   \ifnum\@xarg<\z@ \negargtrue \@xarg -\@xarg \@yyarg -\@yarg
86   \else \negargfalse \@yyarg \@yarg \fi
87 \ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
88 \ifnum \@tempcnta>6 \badlinearg\@tempcnta\z@ \fi
89 \ifnum\@xarg>6 \badlinearg\@xarg \one \fi
90 \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.

```
91 \ifdim\wd\@linechar=\z@
92   \setbox\@linechar\hbox{.}%
93   \badlinearg
94 \fi
95 \ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
96   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
97 \@clnwd \wd\@linechar
98 \if@negarg
99   \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
100 \else
101   \let\reserved@a\relax
102 \fi
103 \@whiledim \@clnwd <\@linelen \do
104   {\@upordown\@clnht\copy\@linechar
105     \reserved@a
106     \advance\@clnht \ht\@linechar
107     \advance\@clnwd \wd\@linechar}%
108 \advance\@clnht -\ht\@linechar
109 \advance\@clnwd -\wd\@linechar
110 \@tempdima\@linelen\advance\@tempdima -\@clnwd
111 \@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
112 \if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
113 \multiply\@tempdima \@m
114 \@tempcnta \@tempdima
115 \@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
116 \@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
117 \divide\@tempdima \@m
118 \advance\@clnht \@tempdima
```



```

119 \ifdim \@linelen <\wd\@linechar
120   \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).
121   \ifdim \@linelen = \z@
122   \else
123     \@picture@warn
124   \fi
125   \else\@upordown\@clnht\copy\@linechar\fi}

\@hline
126 \gdef\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
127 \vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
128 \ifnum \@xarg <\z@ \hskip -\@linelen \fi}

\getlinechar
129 \gdef\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
130 \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
131 \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
132 \char\@tempcnta}

\vector
133 \gdef\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
134 \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
135 \ifnum\@tempcnta<5\relax
136 \@linelen #3\unitlength
137 \ifdim\@linelen<\z@\@badlinearg\else
138   \ifnum\@xarg =\z@ \@vvector
139   \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
140   \fi
141 \fi
142 \else\@badlinearg\fi}

\@hvector
143 \gdef\@hvector{\@hline\hb@xt@\z@\@linefnt
144 \ifnum \@xarg <\z@ \getlarrow(1,0)\hss\else
145   \hss\getrarrow(1,0)\fi}}

\@vvector
146 \gdef\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}

\@svector
147 \gdef\@svector{\@sline
148 \@tempcnta\@yarg \ifnum\@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
149 \ifnum\@tempcnta <5%
150   \hskip -\wd\@linechar
151   \@upordown\@clnht \hbox{\@linefnt \if@negarg
152   \getlarrow(\@xarg,\@yyarg)\else \getrarrow(\@xarg,\@yyarg)\fi}%
153   \else\@badlinearg\fi}

```

\@getlarrow

```

154 \gdef\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta 27 % '33
155   \else
156   \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n
157   \advance\@tempcnta -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
158   \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
159   \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
160   \fi\fi\char\@tempcnta}

```

\@getrarrow

```

161 \gdef\@getrarrow(#1,#2){\@tempcntb #2\relax
162 \ifnum\@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
163 \ifcase \@tempcntb\relax \@tempcnta 45 % '55
164 \or
165 \ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
166 24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
167 \else\@tempcnta 58 \fi\fi\or
168 \ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
169 24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
170 \@tempcnta #1\relax\multiply\@tempcnta
171 \sixt@@n \advance\@tempcnta -\tw@ \else
172 \@tempcnta #1\relax\multiply\@tempcnta
173 \sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
174 \char\@tempcnta}

```

\@vline

```

175 \gdef\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}

```

\@upline

```

176 \gdef\@upline{%
177   \hb@xt@\z@\{ \hskip -\@halfwidth \vrule \@width \@wholewidth
178     \@height \@linelen \@depth \z@\hss\}

```

\@downline

```

179 \gdef\@downline{%
180   \hb@xt@\z@\{ \hskip -\@halfwidth \vrule \@width \@wholewidth
181     \@height \z@ \@depth \@linelen \hss\}

```

\@upvector

```

182 \gdef\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefont\char 54}% % '66
183   \raise \@linelen \hb@xt@\z@\{ \lower \ht\@tempboxa\box\@tempboxa\hss\}

```

\@downvector

```

184 \gdef\@downvector{\@downline\lower \@linelen
185   \hb@xt@\z@\{ \@linefont\char 63 % '77
186   \hss\}

```

```

\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 < \@dashcnt
    do \copy\@dashbox
      \@tempcnta := \@tempcnta + 1
    od
  }
  \@tempcnta := 0
  \put(0,Y){\hskip \@dashdim
    while \@tempcnta < \@dashcnt
    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

\@makepicbox(X,Y)
END

\dashbox

187 \gdef\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
188 \lineskip \z@skip
189 \@dashdim #2\unitlength
190 \@dashcnt \@dashdim \advance\@dashcnt 200
191 \@dashdim #1\unitlength\divide\@dashcnt \@dashdim
192 \ifodd\@dashcnt\@dashdim \z@
193 \advance\@dashcnt \@one \divide\@dashcnt \tw@
194 \else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
195 \advance\@dashcnt \m@ne
196 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
197 \@width \@dashdim}\put(0,0){\copy\@dashbox}%
198 \put(0,#3){\copy\@dashbox}%
199 \put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
200 \put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
201 \multiply\@dashdim \thr@@
202 \fi

```

```

203 \setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
204 \@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
205 \put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
206 \do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
207 \put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
208 \do{\copy\@dashbox\advance\@tempcnta \@ne }}%
209 \@dashdim #3\unitlength
210 \@dashcnt \@dashdim \advance\@dashcnt 200
211 \@dashdim #1\unitlength\divide\@dashcnt \@dashdim
212 \ifodd\@dashcnt \@dashdim \z@
213 \advance\@dashcnt \@ne \divide\@dashcnt \tw@
214 \else
215 \divide\@dashdim \tw@ \divide\@dashcnt \tw@
216 \advance\@dashcnt \m@ne
217 \setbox\@dashbox\hbox{\hskip -\@halfwidth
218 \vrule \@width \@wholewidth
219 \@height \@dashdim}\put(0,0){\copy\@dashbox}%
220 \put(#2,0){\copy\@dashbox}%
221 \put(0,#3){\lower\@dashdim\copy\@dashbox}%
222 \put(#2,#3){\lower\@dashdim\copy\@dashbox}%
223 \multiply\@dashdim \thr@@
224 \fi
225 \setbox\@dashbox\hbox{\vrule \@width \@wholewidth
226 \@height #1\unitlength}\@tempcnta\z@
227 \put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
228 \do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }}%
229 \vskip\@dashdim}\@tempcnta\z@
230 \put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
231 \do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }}%
232 \vskip\@dashdim}}\@makepicbox(#2,#3)}

```

CIRCLES AND OVALS

USER COMMANDS:

`\circle{D}` : Produces the circle with the diameter as close as possible to $D * \text{\unitlength}$. `\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 $X * \text{\unitlength}$ and height $Y * \text{\unitlength}$. The reference point is the center.

`\oval(X,Y)[POS]` : Same as `\oval(X,Y)` except it draws only the half or quadrant of the oval indicated by POS. E.G., `\oval(X,Y)[t]` draws just the top half and `\oval(X,Y)[br]` draws just the bottom right quadrant. In all cases, the reference point is the same as the unqualified `\oval(X,Y)` command.

`\@ovvert {DELTA1} {DELTA2}` : Makes a vbox containing either the left side or the right side of the oval being constructed. The baseline

will coincide with the outside bottom edge of the oval; the left side of the box will coincide with the left edge of the vertical rule. The width of the box will be `\@tempdima`.

DELTA1 and DELTA2 are added to the character number in `\@tempcnta` to get the characters for the top and bottom quarter circle pieces.

`\@ovhorz` : Makes an hbox containing the straight rule for either the top or the bottom of the oval being constructed. The baseline will coincide with bottom edge of the rule; the left side of the box will coincide with the left side of the oval. The width of the box will be `\@ovxx`.

`\@getcirc {DIAM}` : Sets `\@tempcnta` to the character number of the top-right quarter circle with the largest diameter less than or equal to DIAM. Sets `\@tempboxa` to an hbox containing that character. Sets `\@tempdima` to `\wd \@tempboxa`, which is the distance from the circle's left outside edge to its right inside edge. (These characters are like those described in the TeXbook, pp. 389-90.)

```
\@getcirc {DIAM} ==
BEGIN
  \@tempcnta      := integer coercion of (DIAM + 2pt)
                                     + 2pt added 1 Nov 88
  \@tempcnta      := \@tempcnta / integer coercion of 4pt
  if \@tempcnta > 10
    then \@tempcnta := 10 fi
  if \@tempcnta > 0
    then \@tempcnta := \@tempcnta-1
    else LaTeX Warning: Oval too small.
  fi
  \@tempcnta      := 4 * \@tempcnta
  \@tempboxa      := \hbox{\@circlefnt \char \@tempcnta}
  \@tempdima      := \wd \@tempboxa
END
```

```
\@put{X}{Y}{OBJ} ==
BEGIN
  \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
END
```

```
\@oval(X,Y)[POS] ==
BEGIN
  \begingroup
    \boxmaxdepth := \maxdimen
    @ovt := @ovb := @ovl := @ovr := true
    for all E in POS
```

```

do @ovE := false od
\@ovxx      := X * \unitlength
\@ovyy      := Y * \unitlength
\@tempdimb := min(\@ovxx,\@ovyy)
\@getcirc{\@tempdimb-2pt} %% "-2pt" added 7 Dec 89
\@ovro      := \ht \@tempboxa
\@ovri      := \dp \@tempboxa
\@ovdx      := \@ovxx - \@tempdima
\@ovdy      := \@ovyy - \@tempdima
\@ovdx      := \@ovdx/2
\@ovdy      := \@ovdy/2
\@circlefnt
\@tempboxa :=
\hbox{
  if @ovr
  then \@ovvert{3}{2} \kern -\@tempdima
  fi
  if @ovl
  then \kern \@ovxx \@ovvert{0}{1} \kern
- \@tempdima
\kern -\@ovxx
  fi
  if @ovt
  then \@ovhorz \kern -\@ovxx
  fi
  if @ovb
  then \raise \@ovyy \@ovhorz
  fi
}
\@ovdx := \@ovdx + \@ovro
\@ovdy := \@ovdy + \@ovro
\ht\@tempboxa := \dp\@tempboxa := 0
\@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
\endgroup
END

\@ovvert {DELTA1} {DELTA2} ==
BEGIN
  \vbox to \@ovyy {
    if @ovb
    then \@tempcntb := \@tempcnta + DELTA1
      \kern -\@ovro
      \hbox { \char \@tempcntb }
      \nointerlineskip
    else \kern \@ovri \kern \@ovdy
    fi
    \leaders \vrule width \@wholewidth \vfil
    \nointerlineskip
    if @ovt
    then \@tempcntb := \@tempcnta + DELTA2

```

```

\hbox { \char \@tempcntb }
else \kern \@ovdy \kern \@ovro
fi
}

END

\@ovhorz ==
BEGIN
\hb@xt@ \@ovxx{
\kern \@ovro
if @ovr
then
else \kern \@ovdx
fi
\leaders \hrule height \@wholewidth \hfil
if @ovl
then
else \kern \@ovdx
fi
\kern \@ovri
}

END

\circle{DIAM} ==
BEGIN
\begingroup
\boxmaxdepth := maxdimen
\@tempdimb := DIAM *\unitlength
if \@tempdimb > 15.5pt
then \@getcirc{\@tempdimb}
\@ovro := \ht \@tempboxa
\@tempboxa := \hbox{
\@circlefnt
\@tempcnta := \@tempcnta + 2
\char \@tempcnta
\@tempcnta := \@tempcnta - 1
\char \@tempcnta
\kern -2\@tempdima
\@tempcnta := \@tempcnta + 2
\raise \@tempdima \hbox { \char \@tempcnta }
\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

\if@ovt If producing the Top Bottom Left or Right of an oval.
\if@ovb 233 \newif\if@ovt
\if@ovl 234 \newif\if@ovb
\if@ovr 235 \newif\if@ovl
236 \newif\if@ovr

237 </2ekernel | def>
238 <*2ekernel | autoload>

\@ovxx
\@ovyy 239 \newdimen\@ovxx
\@ovdx 240 \newdimen\@ovyy
\@ovdy 241 \newdimen\@ovdx
\@ovro 242 \newdimen\@ovdy
\@ovri 243 \newdimen\@ovro
244 \newdimen\@ovri

245 </2ekernel | autoload>

  \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.

246 <*2ekernel | def>

\@getcirc

247 \gdef\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
248 \@tempcnta\@tempdima
249 \@tempdima 4\p@ \divide\@tempcnta\@tempdima
250 \ifnum \@tempcnta >10\relax
251 \@picture@warn
252 \@tempcnta 10\relax
253 \fi
254 \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne

Warn if requirements for oval or circle can't be met.

255 \else \@picture@warn \fi
256 \multiply\@tempcnta 4\relax
257 \setbox \@tempboxa \hbox{\@circlefnt
258 \char \@tempcnta}\@tempdima \wd \@tempboxa}

```

`\@picture@warn` Generic warning for lines, vectors (used in `\@sline`) and oval or circle (used in `\@getcirc`) are not available at right size.

```
259 \def\@picture@warn{\@latex@warning{%
260     \string\oval, \string\circle, or \string\line\space
261     size unavailable}}
```

`\@put`

```
262 \gdef\@put#1#2#3{\raise #2\hb@xt@#3{\hskip #1#3\hss}}
```

`\oval`

```
263 \gdef\oval(#1,#2){\ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2) []}}
```

`\@oval`

```
264 \gdef\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
265   \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
266   \@tfor\reserved@a :=#3\do{\csname @ov\reserved@a false\endcsname}%
267   \@ovxx
268   #1\unitlength \@ovyy #2\unitlength
269   \@tempdima \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
270   \advance \@tempdima -2\p@
271   \@getcirc \@tempdima
272   \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
273   \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
274   \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
275   \circlefth \setbox\@tempboxa
276   \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
277   \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
278   \if@ovt \@ovhorz \kern -\@ovxx \fi
279   \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
280   \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
281   \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
282   \endgroup}
```

`\@ovvert`

```
283 \gdef\@ovvert#1#2{\vbox to\@ovyy{%
284   \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
285   \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
286   \else \kern \@ovri \kern \@ovdy \fi
287   \leaders\vrule \width \@wholewidth\vfil \nointerlineskip
288   \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
289   \hbox{\char \@tempcntb}%
290   \else \kern \@ovdy \kern \@ovro \fi}}
```

`\@ovhorz`

```
291 \gdef\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
292   \if@ovr \else \kern \@ovdx \fi
293   \leaders\hrule \height \@wholewidth \hfil
294   \if@ovl \else \kern \@ovdx \fi
295   \kern \@ovri}}
```

`\circle`

```
296 \gdef\circle{\@inmatherr\circle\@ifstar\@dot\@circle}
```

```

\@circle
297 \gdef\@circle#1{%
298   \beginngroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
299   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
300     \ovro\ht\@tempboxa
301     \setbox\@tempboxa\hbox{\@circlefnt
302       \advance\@tempcnta\tw@ \char \@tempcnta
303       \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
304       \advance\@tempcnta\tw@
305       \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
306       \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
307       \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
308     \else \@circ\@tempdimb{96}\fi\endgroup}

\@dot Internal form of \circle*.
309 \gdef\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}

\@circ
310 \gdef\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
311   \@tempcnta\@tempdima \@tempdima \p@
312   \divide\@tempcnta\@tempdima
313   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
314   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
315   \advance\@tempcnta #2\relax
316   \@circlefnt \char\@tempcnta}

317 </2ekernel | def>
318 <*2ekernel | autload>

\@xarg Counters used for manipulating the ‘slope’ arguments.
\@yarg 319 \newcount\@xarg
\@yyarg 320 \newcount\@yarg
321 \newcount\@yyarg

\@multicnt Counter used in \multitup, and also \multicolumn.
322 \newcount\@multicnt

\@xdim Length registers.
\@ydim 323 \newdimen\@xdim
324 \newdimen\@ydim

\@linechar Box for holding a line segment character, for sloping lines.
325 \newbox\@linechar

\@linelen Length of the line currently being built.
326 \newdimen\@linelen

\@clnwd Height and width of current line segment.
\@clnht 327 \newdimen\@clnwd
328 \newdimen\@clnht

```

```

\@dashdim \dashbox internal registers.
\@dashbox 329 \newdimen\@dashdim
\@dashcnt 330 \newbox\@dashbox
331 \newcount\@dashcnt

Initialization: "\thinlines"
332 \let\@linefnt\tenln
333 \let\@circlefnt\tencirc
334 \@wholewidth\fontdimen8\tenln
335 \@halfwidth .5\@wholewidth
336 \kernel | autoload)

```

59.1 Curves

The new `\qbezier` command, based on the old `\bezier` defined in `bezier.sty`.

```

\qbezier[N] == \bezier{N}

\bezier{N}(AX,AY)(BX,BY)(CX,CY) ==
BEGIN
  IF N = 0
  THEN \@xdima := |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

```

`\qbeziermax` The maximum number of points to plot.

```

337 (*2ekernel | def)
338 (def)\ifx\qbeziermax\@undefined
339 \gdef\qbeziermax{500}
340 (def)\fi

```

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 (.

```

341 \newcommand\qbezier[2][0]{\bezier{#1}#2}

```

`\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 (.

```

342 \gdef\bezier#1)#2(#3)#4({\@bezier#1)(#3){}

```

`\@bezier`

```

343 \gdef\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
344   \ifnum #1=\z@
345     \@ovxx #4\unitlength
346     \advance\@ovxx -#2\unitlength
347     \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
348     \@ovdx #6\unitlength
349     \advance\@ovdx -#4\unitlength
350     \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
351     \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
352     \@ovyy #5\unitlength
353     \advance\@ovyy -#3\unitlength
354     \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
355     \@ovdy #7\unitlength
356     \advance\@ovdy -#5\unitlength
357     \ifdim \@ovdy<\z@ \@ovdy -\@ovdy \fi
358     \ifdim \@ovyy<\@ovdy \@ovyy \@ovdy \fi
359     \@multicnt
360     \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
361     \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
362     \ifnum \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax \fi
363   \else \@multicnt#1\relax \fi
364   \@tempcnta\@multicnt \advance\@tempcnta\@ne
365   \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
366   \multiply\@ovdx \tw@
367   \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
368   \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
369   \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength

```

```

370      \multiply\@ovdy \tw@
371 \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
372      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt

373 \setbox\@tempboxa\hbox{%
374      \hskip -\@halfwidth
375      \vrule \@height\@halfwidth
376      \@depth \@halfwidth
377      \@width \@wholewidth}%
378 \put(#2,#3){%
379      \count@\z@
380      \@whilenum{\count@<\@tempcnta}\do
381      {\@xdim\count@\@ovxx
382      \advance\@xdim\@ovdx
383      \divide\@xdim\@multicnt
384      \multiply\@xdim\count@
385      \@ydim\count@\@ovyy
386      \advance\@ydim\@ovdy
387      \divide\@ydim\@multicnt
388      \multiply\@ydim\count@
389      \raise \@ydim
390      \hb@xt@\z@{\kern\@xdim
391      \unhcopy\@tempboxa\hss}%
392      \advance\count@\@ne}}
393 </2kernel | def>

```

File E

ltthm.dtx

60 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 $\langle name \rangle$ to be just as one would expect a theorem environment to be, except that it prints $\langle text \rangle$ instead of “Theorem”.

If $\langle oldname \rangle$ is given, then environments $\langle name \rangle$ and $\langle oldname \rangle$ use the same counter, so using a $\langle name \rangle$ environment advances the number of the next $\langle name \rangle$ environment, and vice-versa.

If $\langle counter \rangle$ is given, then environment $\langle name \rangle$ is numbered within $\langle counter \rangle$.

E.g., if $\langle counter \rangle = \text{subsection}$, then the first $\langle name \rangle$ in subsection 7.2 is numbered $\langle text \rangle$ 7.2.1.

The way $\langle name \rangle$ environments are numbered can be changed by redefining $\backslash the\langle name \rangle$.

DOCUMENT STYLE PARAMETERS

$\backslash @thmcounter\{COUNTER\}$: A command such that

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

defines $\backslash theCOUNTER$ to produce a number for a theorem environment.

The default is:

```
BEGIN \noexpand\arabic{COUNTER} END
```

$\backslash @thmcountersep$: 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, $\backslash @thmcountersep$ should be $\backslash def$ 'ed to '-'. Its default is ' '.

$\backslash @begintheorem\{NAME\}\{NUMBER\}$: A command that begins a theorem

environment for a 'theorem' named 'NAME NUMBER' – e.g., $\backslash @begintheorem\{Lemma\}\{3.7\}$ starts Lemma 3.7.

$\backslash @opargbegintheorem\{NAME\}\{NUMBER\}\{OPARG\}$:

A command that begins a theorem

environment for a 'theorem' named 'NAME NUMBER' with optional

argument OPARG – e.g., $\backslash @begintheorem\{Lemma\}\{3.7\}\{Jones\}$ starts 'Lemma 3.7 (Jones)':.

$\backslash @endtheorem$: A command that ends a theorem environment.

$\backslash newtheorem\{NAME\}\{TEXT\}[COUNTER] ==$

```

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 nonexistant
  then ERROR
  else
    if \NAME is definable
    then BEGIN
      \theNAME == \theOLDNAME
      \NAME == \@thm{OLDNAME}{TEXT}
      \endNAME == \@endtheorem
      END
    else error
    fi
  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

`\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 \LaTeX book indicates that `\newtheorem` may be used anywhere in the document...

```
1 (*2ekernel)
2 \def\newtheorem#1{%
3   \@ifnextchar[{\@othm{#1}}{\@nthm{#1}}}
```

`\@nthm`

```
4 \def\@nthm#1#2{%
5   \@ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

`\@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}}}
```

`\@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}}}
```

`\@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}}}
```

`\@thm`

```
26 \def\@thm#1#2{%
27   \refstepcounter{#1}%
28   \@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}
```

`\@xthm`

`\@ythm`

```
29 \def\@xthm#1#2{%
30   \@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
31 \def\@ythm#1#2[#3]{%
32   \@opargbegintheorem{#2}{\csname the#1\endcsname}{#3}\ignorespaces}
```

Default values

```

\@thmcounter
\@thmcountersep 33 \def\@thmcounter#1{\noexpand\arabic{#1}}
                 34 \def\@thmcountersep{.}

\@begintheorem Providing theorem defaults.
\@opargbegintheorem 35 \def\@begintheorem#1#2{\trivlist
\@endtheorem        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 \kernel

```

File F

ltsect.dtx

61 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,}
```

61.1 The Title

```
\title The user defines the title and author by the declarations \title{<name>},
\author \author{<name>}
\date Similarly the date is declared with \date{<date>}.
\thanks Inside these, the \thanks{<footnote text>} command may be used to make
\and acknowledgements, notice of address, etc. in a footnote. If there are multiple
\maketitle authors, they have to be separated with the \and command.
And finally, the \maketitle command produces the actual title, using the
information previously saved with the other commands.

\title \title for use in \maketitle. If not given \maketitle will produce an error
\@title message.
3 \def\title#1{\gdef\@title{#1}}
4 \def\@title{\@latex@error{No \noexpand\title given}\@ehc}

\author \author for use in \maketitle. If not given \maketitle will produce a warning
\@author message.
5 \def\author#1{\gdef\@author{#1}}
6 \def\@author{\@latex@warning{no@line{No \noexpand\author given}}

\date \date for use in \maketitle. If not given \maketitle will produce \today as the
\@date default.
7 \def\date#1{\gdef\@date{#1}}
8 \gdef\@date{\today}

\thanks
9 \def\thanks#1{\footnotemark
10 \protected@xdef\@thanks{\@thanks
11 \protect\footnotetext[\the\c@footnote]{#1}}%
12 }

\@thanks
13 \let\@thanks\empty
```

```

\and
14 \def\and{%                % \begin{tabular}
15   \end{tabular}%
16   \hskip 1em \@plus.17fil%
17   \begin{tabular}[t]{c}%   % \end{tabular}
18 \message{sectioning,}

```

61.2 Sectioning

```

\@secpenalty
19 \newcount\@secpenalty
20 \@secpenalty = -300

\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.
21 \newif\if@noskipsec \@noskipsectrue

\@startsection The \@startsection{<name>}{<level>}{<indent>}{<beforeskip>}
{<afterskip>}{<style>}*[<altheading> ]{<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=1, section = 2, etc.

indent: Indentation of heading from left margin

beforeskip: Absolute value = skip to leave above the heading. If negative, then paragraph indent of text following heading is suppressed.

afterskip: if positive, then skip to leave below heading, else negative of skip to leave to right of run-in heading.

style: Commands to set style. Since June 1996 release the *last* command in this argument may be a command such as `\MakeUppercase` or `\fbox` that takes an argument. The section heading will be supplied as the argument to this command. So setting #6 to, say, `\bfseries\MakeUppercase` would produce bold, uppercase headings.

If ‘*’ is missing, then increment the counter. If it is present, then there should be no `[<altheading>]` argument. The command uses the counter ‘secnumdepth’. It contains a pointer to the highest section level that is to be numbered.

Warning: The `\@startsection` command should be at the same or higher grouping level as the text that follows it. For example, you should *not* do something like

```

\def\foo{ \begingroup ...
          \paragraph{...}
          \endgroup}

```

```

Pseudocode for the \@startsection command
\@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

22 \def\@startsection#1#2#3#4#5#6{%
23   \if@noskipsec \leavevmode \fi
24   \par
25   \@tempskipa #4\relax
26   \@afterindenttrue
27   \ifdim \@tempskipa <\z@
28     \@tempskipa -\@tempskipa \@afterindentfalse
29   \fi
30   \if@nobreak
31     \everypar{}%
32   \else
33     \addpenalty\@secpenalty\addvspace\@tempskipa
34   \fi
35   \@ifstar
36     {\@ssect{#3}{#4}{#5}{#6}}%
37     {\@dblarg{\@sect{#1}{#2}{#3}{#4}{#5}{#6}}}

```

\@sect Pseudocode for the \@sect command

```

\@sect{NAME}{LEVEL}{INDENT}{BEFORESKIP}{AFTERSKIP}{STYLE}[ARG1]{ARG2}
==
BEGIN
  IF LEVEL > \c@secnumdepth
    THEN \@svsec :=L null
    ELSE \refstepcounter{NAME}
          \@svsec :=L BEGIN \@seccntformat{#1}\relax END

```

```

FI
IF AFTERSKIP > 0
  THEN \beginngroup
    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

38 \def\@sect#1#2#3#4#5#6[#7]#8{%
39   \ifnum #2>\c@secnumdepth
40     \let\@svsec\@empty
41   \else
42     \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.

43   \protected@edef\@svsec{\@seccntformat{#1}\relax}%
44   \fi
45   \@tempskipa #5\relax
46   \ifdim \@tempskipa>\z@
47     \beginngroup

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 #6.

48     #6{%
49       \@hangfrom{\hskip #3\relax\@svsec}%
50       \interlinepenalty \@M #8\@par}%
51   \endgroup
52   \csname #1mark\endcsname{#7}%
53   \addcontentsline{toc}{#1}{%
54     \ifnum #2>\c@secnumdepth \else
55       \protect\numberline{\csname the#1\endcsname}%

```

```

56     \fi
57     #7}%
58 \else
\relax added 2 May 90
59     \def\@svsechd{%
60         #6{\hskip #3\relax
61         \@svsec #8}%
62         \csname #1mark\endcsname{#7}%
63         \addcontentsline{toc}{#1}{%
64             \ifnum #2>\c@secnumdepth \else
65                 \protect\numberline{\csname the#1\endcsname}%
66             \fi
67             #7}}%
68 \fi
69 \@xsect{#5}}

\@xsect Pseudocode for the \@xsect command
\@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 :=G 10000
                \hskip -\parindent
                \begingroup
                \@svsechd
                \endgroup
                \unskip
                \hskip -AFTERSKIP \relax
                %% relax added 14 Jan 91
            ELSE \clubpenalty :=G \@clubpenalty
                \everypar := NULL
            FI
        }
    FI

    FI

END

70 \def\@xsect#1{%
71     \@tempskipa #1\relax
72     \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.

73     \par \nobreak
74     \vskip \@tempskipa

```

```

75   \@afterheading
76   \else

77   \@nobreakfalse
78   \global\@noskipsectrue
79   \everypar{%
80     \if@noskipsec
81       \global\@noskipsecfalse
82       {\setbox\z@\lastbox}%
83       \clubpenalty\@M
84       \begingroup \@svsechd \endgroup
85       \unskip
86       \@tempskipa #1\relax
87       \hskip -\@tempskipa
88     \else
89       \clubpenalty \@clubpenalty
90       \everypar{}%
91     \fi}%
92   \fi
93   \ignorespaces}

```

`\@secCNTformat` This command formats the section number including the space following it.

```

94 \def\@secCNTformat#1{\csname the#1\endcsname\quad}

```

Pseudocode for the `\@ssect` command

```

\@ssect{INDENT}{BEFORESKIP}{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

```

Pseudocode for the `\@afterheading` command

```

\@afterheading ==
BEGIN
  @nobreak :=G true
  \everypar := BEGIN IF @nobreak = T
    THEN @nobreak :=G false
    \clubpenalty :=G 10000
    IF @afterindent = F
      THEN remove \lastbox
    FI
    ELSE \clubpenalty :=G \@clubpenalty
    \everypar := NULL
  END IF
END

```



```

                                FI
                                END
                                END

\@ssect
95 \def\@ssect#1#2#3#4#5{%
96   \@tempskipa #3\relax
97   \ifdim \@tempskipa>\z@
98     \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.
99     #4{%
100       \@hangfrom{\hskip #1}%
101       \interlinepenalty \@M #5\@@par}%
102     \endgroup
103   \else
104     \def\@svsechd{#4{\hskip #1\relax #5}}%
105   \fi
106   \@xsect{#3}}

\if@afterindent
\@afterindenttrue 107 \newif\if@afterindent \@afterindenttrue

\@afterheading This hook is used in setting up custom-built headings in classes.dtx.
108 \def\@afterheading{%
109   \@nbreaktrue
110   \everypar{%
111     \if@nbreak
112       \@nbreakfalse
113       \clubpenalty \@M
114       \if@afterindent \else
115         {\setbox\z@\lastbox}%
116       \fi
117     \else
118       \clubpenalty \@clubpenalty
119       \everypar{}%
120     \fi}}

\@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.
121 \def\@hangfrom#1{\setbox\@tempboxa\hbox{#1}}%
122   \hangindent \wd\@tempboxa\noindent\box\@tempboxa}

\c@secnumdepth
\c@tocdepth 123 \newcount\c@secnumdepth
124 \newcount\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 \langlestarcmd\rangle \langleunstarcmd\rangle }`
2. `\def\langlestarcmd\rangle[#1]#2{ ... } % Command to define \chapter[...]{...}`
3. `\def\langleunstarcmd\rangle#1{ ... } % Command to define \chapter*{...}`

```
125 \def\secdef#1#2{\ifstar{#2}{\@dblarg{#1}}}
```

61.2.1 Initializations

```
\sectionmark
\subsectionmark 126 \let\sectionmark\@gobble
\subsubsectionmark 127 \let\subsectionmark\@gobble
\paragraphmark 128 \let\subsubsectionmark\@gobble
\subparagraphmark 129 \let\paragraphmark\@gobble
130 \let\subparagraphmark\@gobble

131 \message{contents,}
```

61.3 Table of Contents etc.

61.3.1 Convention

`\tf@⟨foo⟩` = file number for output for table foo. The file is opened only if `@filesw = true`.

61.3.2 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.

```
\@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

```

132 \def\@starttoc#1{%
133   \begingroup
134   \makeatletter
135   \@input{\jobname.#1}%
136   \if@files
137     \expandafter\newwrite\csname tf@#1\endcsname
138     \immediate\openout \csname tf@#1\endcsname \jobname.#1\relax
139   \fi
140   \@nobreakfalse
141   \endgroup}

```

\addcontentsline The `\addcontentsline{<table>}{<type>}{<entry>}` command allows the user to add his/her own entry to a table of contents, etc. The command adds the entry `\contentsline{<type>}{<entry>}{<page>}` to the `.<table>` 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`.

```

142 \def\addcontentsline#1#2#3{%
143   \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}}

```

\addtocontents The `\addtocontents{<table>}{<text>}` command adds `<text>` to the `.<table>` file, with no page number.

```

144 \long\def\addtocontents#1#2{%
145   \protected@write\@auxout
146     {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
147     {\string\@writefile{#1}{#2}}}

```

\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.

```

148 \def\contentsline#1{\csname l@#1\endcsname}

```

`\@dottedtocline{<level>}{<indent>}{<numwidth>}{<title>}{<page>}`: Macro to produce a table of contents line with the following parameters:

level If `<level>` > `\c@tocdepth`, then no line produced.

indent Total indentation from the left margin.

numwidth Width of box for number if the `<title>` 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`

```
149 \def\@dottedtocline#1#2#3#4#5{%
150   \ifnum #1>\c@tocdepth \else
151     \vskip \z@ \@plus.2\p@
152     {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
153     \parindent #2\relax\@afterindenttrue
154     \interlinepenalty\@M
155     \leavevmode
156     \@tempdima #3\relax
157     \advance\leftskip \@tempdima \null\nobreak\hskip -\leftskip
158     {#4}\nobreak
159     \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.

```
160       \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
161       mu$}\hfill
162       \nobreak
163       \hb@xt@\@pnumwidth{\hfil\normalfont \normalcolor #5}%
164       \par}%
165 \fi}
```

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{<number>}`: For use in a `\contentsline` command. It puts `<number>` ushleft in a box of width `\@tempdima` (Before 25 Jan 88 change, it also added `\@tempdima` to the hanging indentation.)

```
166 \def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
167 \</2kernel>
```

File G

ltoat.dtx

62 Floats

The different types of oats are identified by a $\langle type \rangle$ name, which is the name of the counter for that kind of oat. For example, figures are of type ‘figure’ and tables are of type ‘table’. Each $\langle type \rangle$ has associated a positive $\langle type\ number \rangle$, which is a power of two. E.g., figures might have type number 1, tables type number 2, programs type number 4, etc.

The locations where a oat can go are specified by a $\langle placement\ specifier \rangle$, 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 oat page

In addition, in conjunction with these, you can use ‘!’ which means that the current values of the oat positioning parameters are ignored for this oat. (Has no effect on ‘p’, oat page positioning.) For example, ‘pht’ specifies that the oat can appear in any of three locations: page, here or top.

62.1 Floating Environments

```
1 \*2ekernel
2 \message{floats,}
```

Where oats may appear on a page, and how many may appear there are specified by the following oat placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

<code>\c@topnumber</code>	: Number of oats allowed at the top of a column.
<code>\topfraction</code>	: Fraction of column that can be devoted to oats.
<code>\c@dbltopnumber, \dbltopfraction</code>	: Same as above, but for double-column oats.
<code>\c@bottomnumber, \bottomfraction</code>	: Same as above for bottom of page.
<code>\c@totalnumber</code>	: Number of oats allowed in a single column, including in-text oats.
<code>\textfraction</code>	: Minimum fraction of column that must contain text.
<code>\floatpagefraction</code>	: Minimum fraction of page that must be taken up by oat page.
<code>\dblfloatpagefraction</code>	: Same as above, for double-column oats.

The document style must define the following.

`\fps@TYPE` : The default placement specifier for oats of type TYPE.

`\ftype@TYPE` : The type number for oats of type TYPE.

`\ext@TYPE` : The file extension indicating the file on which the contents list for oat 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 oat environment for a
single-column oat 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
 `\@captype ==L TYPE`
 `\@dblflset`
 `\@fps ==L PLACEMENT`
 `\@onelevel@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 oats'
 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{\@capttype}
\@dblarg{\caption{\@capttype}}
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.
\@floatpenalty
    3 \newcount\@floatpenalty

\caption This is set to be an error message outside a oat since no captype is defined there;
this may need to be changed by some classes.
    4 \def\caption{%
    5   \ifx\@captype\@undefined
    6     \latex@error{\noexpand\caption outside float}\@ehd
    7     \expandafter\@gobble
    8   \else
    9     \refstepcounter\@captype
   10     \expandafter\@firstofone
   11   \fi
   12   {\@dblarg{\caption\@captype}}%
   13 }

\@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 oat mechanism, like minipage, sets the ag @minipage true before exe-
cuting the user-supplied text. Many LATEX constructs test for this ag and do not
add vertical space when it is true. The intention is that this emulates TEX's 'top
of page' behaviour. The ag 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 ag is already false
then the \caption was not the first entry in the oat, and so some other para-
graph 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

\@float
\@dblflset
    26 \def\@float#1{%
    27   \ifnextchar[%
    28     {\@xfloat{#1}}%
    29     {\edef\reserved@a{\noexpand\@xfloat{#1}[\csname fps@#1\endcsname]}}%
    30     \reserved@a}}

```


\@dblfloat

```
31 \def\@dblfloat{%
32   \if@twocolumn\let\reserved@a\@dbflt\else\let\reserved@a\@float\fi
33   \reserved@a}
```

\fps@dbl Note that all double floats have default fps ‘tp’.

\@setfps This sets the fps, dealing with error conditions by adding the default.

\@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 \def\@xfloat #1[#2]{%
35   \nodocument
36   \def \@cuptype {#1}%
37   \def \@fps {#2}%
38   \@onelevel@sanitize \@fps
39   \def \reserved@b {!}%
40   \ifx \reserved@b \@fps
41     \fpsadddefault
42   \else
43     \ifx \@fps \@empty
44       \fpsadddefault
45     \fi
46   \fi
47   \ifhmode
48     \@bsphack
49     \@floatpenalty -\@Mii
50   \else
51     \@floatpenalty-\@Miii
52   \fi
53   \ifinner
54     \@parmoderr\@floatpenalty\z@
55   \else
56     \@next\@currbox\@freelist
57     {%
58       \@tempcnta \sixt@@n
59       \expandafter \@tfor \expandafter \reserved@a
60       \expandafter :\expandafter =\@fps
61       \do
62       {%
63         \if \reserved@a h%
64           \ifodd \@tempcnta
65         \else
66           \advance \@tempcnta \@ne
67         \fi
68       \fi
69       \if \reserved@a t%
70         \@setfpsbit \tw@
71       \fi
72       \if \reserved@a b%
73         \@setfpsbit 4%
```

```

74         \fi
75         \if \reserved@a p%
76             \setfpsbit 8%
77         \fi
78         \if \reserved@a !%
79             \ifnum \@tempcnta>15
80                 \advance\@tempcnta -\sixt@@n\relax
81             \fi
82         \fi
83     }%
84     \@tempcntb \csname ftype@\@capttype \endcsname
85     \multiply \@tempcntb \@xxxii
86     \advance \@tempcnta \@tempcntb
87     \global \count\@currbox \@tempcnta
88 }%
89 \fltovf
90 \fi

```

The remainder sets up the box in which the oat 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 oat.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column oats.

```

91 \global \setbox\@currbox
92 \color@vbox
93 \normalcolor
94 \vbox \bgroup
95 \hsize\columnwidth
96 \@parboxrestore
97 \@floatboxreset
98 }

```

`\@floatboxreset` The rationale for allowing these normally global ags to be set locally here, via `\@parboxrestore`, was stated originally by Donald Arseneau and extended by Chris Rowley. It is because these ags are only set globally to true by section commands, and these should never appear within marginals or oats 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 ags should be treated as in `\set@nobreak`; otherwise this command will be redundant.

```

99 \def \@floatboxreset {%
100     \reset@font
101     \normalsize
102     \setminipage
103 }

\@setnobreak

104 \def \@setnobreak{%
105     \if@nobreak
106         \let\outer@nobreak\@nobreaktrue
107         \@nobreakfalse
108     \fi
109 }

```

```

\@setminipage
110 \def \@setminipage{%
111   \@minipagetrue
112   \everypar{\@minipagefalse\everypar{}}%
113 }

\end@float
114 \def\end@float{%
115   \endfloatbox
116   \ifnum\@floatpenalty <\z@
      We make sure that we never exceed \textheight, otherwise oat will never get
      typeset (91/03/15 FMi).
117     \@largefloatcheck
118     \@cons\@currlist\@currbox
119     \ifnum\@floatpenalty <-\@Mii
120       \penalty -\@Miv
      Saving and restoring \prevdepth added 26 May 87 to prevent extra vertical space
      when used in vertical mode.
121       \@tempdima\prevdepth
122       \vbox{%
123         \prevdepth\@tempdima
124         \penalty\@floatpenalty

125     \else
126       \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
127     \fi
128     \fi
129 }

\end@dblfloat
130 \def\end@dblfloat{%
131   \if@twocolumn
132     \endfloatbox
133     \ifnum\@floatpenalty <\z@
      We make sure that we never exceed \textheight, otherwise oat will never get
      typeset (91/03/15 FMi).
134       \@largefloatcheck
135       \@cons\@dbldeferlist\@currbox
136       \fi
      RmS 92/03/18 changed \@esphack to \@Esphack.
137       \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
138     \else
139       \end@float
140     \fi
141 }

\@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.
142 \def \@endfloatbox{%
143   \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87

```

```

144     \@minipagefalse
145     \outer@nobreak
146     \egroup                %% end of vbox
147     \color@endbox
148 }
149 %
150 % \begin{macro}{\outer@nobreak}
151 % \changes{v1.0h}{1994/05/20}{Macro added: default is to do nothing.}
152 %     \begin{macrocode}
153 \let\outer@nobreak\@empty

```

\@largefloatcheck This calculates by how much a float is oversized for the page and prints this in a warning message.

```

154 \def \@largefloatcheck{%
155     \ifdim \ht\@currbox>\textheight
156         \@tempdima -\textheight
157         \advance \@tempdima \ht\@currbox

158     \@latex@warning {Float too large for page by \the\@tempdima}%
159     \ht\@currbox \textheight
160     \fi
161 }

```

\@dbflt

```

\@xdblfloat 162 \def \@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
163 \def \@xdblfloat#1[#2]{%
164     \@xfloat{#1}[#2]\hsize\textwidth\linewidth\textwidth}

```

Moved to ltoutput 93/12/16

```

165 %\newcount\c@topnumber
166 %\newcount\c@dbltopnumber
167 %\newcount\c@bottomnumber
168 %\newcount\c@totalnumber

    An analysis of \@floatplacement:
    This should be called whenever \@colht has been set.

169 \def \@floatplacement{\global\@topnum\c@topnumber
170     % Textpage bit, global:
171     \global\@toproom \topfraction\@colht
172     \global\@botnum \c@bottomnumber
173     \global\@botroom \bottomfraction\@colht
174     \global\@colnum \c@totalnumber
175     % Floatpage bit, local:
176     \@fpmin \floatpagefraction\@colht}

```

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

```

177 \def \@dblfloatplacement {%
    Textpage bit: global, but need not be.
178     \global \@dbltopnum \c@dbltopnumber
179     \global \@dbltoproom \dbltopfraction\@colht

```

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

```

180 \textmin \@colht
181 \advance \@textmin -\dbltoproom
Floatpage bit: must be local.
182 \fpmmin \dblfloatpagefraction\textheight
183 \fptop \dblfpptop
184 \fpsep \dblfpsep
185 \fpbot \dblfpbot
186 }

```

MARGINAL NOTES:

Marginal notes use the same mechanism as oats to communicate with the `\output` routine. Marginal notes are distinguished from oats 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., `\@leftmargin skip ==`
`\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 oats'
            \@currbox, \@marbox := \@tempboxa    %%use \def
        fi
    fi
    if optional argument
    then %% \@xmpar ==
        \@savemarbox\@marbox{LTEXT}
        \@savemarbox\@currbox{RTEXT}
    else %% \@ympar ==
        \@savemarbox\@marbox{RTEXT}
        \box\@currbox :=G \box\@marbox
    fi
    \@xympar
END

\reversemarginpar == BEGIN \@mparbottom    :=G 0
                    @reversemargin :=G true
                    END

\normalmarginpar == BEGIN \@mparbottom    :=G 0
                    @reversemargin :=G false
                    END

\marginpar
187 \def\marginpar{%
188   \ifhmode
189     \@bsphack
190     \@floatpenalty -\@Mii
191   \else
192     \@floatpenalty-\@Miii
193   \fi
194   \ifinner
195     \@parmoderr
196     \@floatpenalty\z@
197   \else
198     \@next\@currbox\@freelist{}{}%
199     \@next\@marbox\@freelist{\global\count\@marbox\m@ne}%
200     {\@floatpenalty\z@
201       \fltovf\def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}%
202   \fi
203   \@ifnextchar [\@xmpar\@ympar}

\@xmpar
204 \long\def\@xmpar[#1]#2{%
205   \@savemarbox\@marbox{#1}%
206   \@savemarbox\@currbox{#2}%
207   \@xympar}

```

\@ympar

```
208 \long\def \@ympar#1{%
209   \@savemarbox \@marbox{#1}%
210   \global\setbox \@currbox\copy \@marbox
211   \@xympar}
```

\@savemarbox

```
212 \long\def \@savemarbox #1#2{%
213   \global\setbox #1%
214   \color@vbox
215   \vtop{%
216     \hsize\marginparwidth
217     \@parboxrestore
218     \@marginparreset
219     #2%
220     \@minipagefalse
221     \outer@nobreak
222   }%
223   \color@endbox
224 }
```

\@marginparreset The rationale for allowing these normally global ags to be set locally here, via \@parboxrestore was stated originally by Donald Arsenau and extended by Chris Rowley. It is because these ags are only set globally to true by section commands, and these should never appear within marginals or oats 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 ags should be treated as in \set@nobreak; otherwise this command will be redundant.

```
225 \def \@marginparreset {%
226   \reset@font
227   \normalsize
228 %   \let@if@nobreak\iffalse
229 %   \let@if@noskipsec\iffalse
230 %   \@setnobreak
231   \@setminipage
232 }
```

\@xympar

Setting the box here is done only because the code uses \end@float; it will be empty and gets discarded.

```
233 \def \@xympar{%
234   \ifnum \@floatpenalty < \z@ \@cons \@currlist \@marbox \fi
235   \setbox \@tempboxa
236   \color@vbox
237   \vbox \bgroup
238   \end@float
239   \@ignorefalse
240   \@esphack
241 }
```

\reversemarginpar

```
\normalmarginpar 242 \def \reversemarginpar {\global \@parbottom \z@ \@reversemargintrue}
243 \def \normalmarginpar {\global \@parbottom \z@ \@reversemarginfalse}
```

62.2 Footnotes

`\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 to 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
`\@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$ 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 \@x@sf := \the\spacefactor FI
  \@makefnmark % put number in main text
  IF hmode THEN \spacefactor := \@x@sf FI
END

```

```

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

```

```

\footnotetext[ NUM ] ==
BEGIN begingroup counter \@mpfn :=L NUM
  \protect == \noexpand
  \@thefnmark :=G eval (\thempfn)
endgroup
\@footnotetext
END

```

`\footins` L^AT_EX does use the same insert for footnotes as PLAIN.

245 \newinsert\footins

L^AT_EX leaves these initializations for the `\footins` insert.

```
246 \skip\footins=\bigskipamount % space added when footnote is present
247 \count\footins=1000 % footnote magnification factor (1 to 1)
248 \dimen\footins=8in % maximum footnotes per page
```

`\footnoterule` L^AT_EX keeps PLAIN T_EX's `\footnoterule` as the default.

```
249 \def\footnoterule{\kern-3\p@
250   \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high
```

`\thefootnote`

```
251 \@definecounter{footnote}
252 \def\thefootnote{\@arabic\c@footnote}
```

`\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.

```
253 \@definecounter{mpfootnote}
254 \def\thempfootnote{{\itshape\@alph\c@mpfootnote}}
```

`\@makefnmark` Default definition.

```
255 %\def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
256 \def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!!

```
257 \DeclareRobustCommand*\textsuperscript[1]{%
258   \@textsuperscript{\selectfont#1}}
```

`\@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.

```
259 \def\@textsuperscript#1{%
260   {\m@th\ensuremath{\^{\mbox{\fontsize\sfontsize\z@#1}}}}}
```

`\footnotesep`

```
261 \newdimen\footnotesep
```

`\footnote`

```
262 \def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
263   \protected@xdef\@thefnmark{\thempfn}%
264   \@footnotemark\@footnotetext}}
```

`\@xfootnote`

```
265 \def\@xfootnote[#1]{%
266   \begingroup
267     \csname c@\@mpfn\endcsname #1\relax
268     \unrestored@protected@xdef\@thefnmark{\thempfn}%
269   \endgroup
270   \@footnotemark\@footnotetext}
```

```

\@footnotetext
271 \long\def\@footnotetext#1{\insert\footins{%
272   \reset@font\footnotesize
273   \interlinepenalty\interfootnotelinepenalty
274   \splittopskip\footnotesep
275   \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
276   \hsize\columnwidth \@parboxrestore
277   \protected@edef\@currentlabel{%
278     \csname p@footnote\endcsname\@thefnmark
279   }%
280   \color@begingroup
281     \makefnmark{%
282       \rule{z@footnotesep}{\ignorespaces#1\@finalstrut\strutbox}%
283     \color@endgroup}}%

\footnotemark
284 \def\footnotemark{%
285   \@ifnextchar[\@xfootnotemark
286     {\stepcounter{footnote}%
287       \protected@xdef\@thefnmark{\thefootnote}%
288       \@footnotemark}}

\@xfootnotemark
289 \def\@xfootnotemark[#1]{%
290   \begingroup
291     \c@footnote #1\relax
292     \unrestored@protected@xdef\@thefnmark{\thefootnote}%
293   \endgroup
294   \@footnotemark}

\@footnotemark
295 \def\@footnotemark{%
296   \leavevmode
297   \ifhmode\edef\@xsf{\the\spacefactor}\nobreak\fi
298   \@makefnmark
299   \ifhmode\spacefactor\@xsf\fi
300   \relax}

\footnotetext
301 \def\footnotetext{%
302   \@ifnextchar [\@xfootnotenext
303     {\protected@xdef\@thefnmark{\thempfn}%
304     \@footnotetext}}

\@xfootnotenext
305 \def\@xfootnotenext[#1]{%
306   \begingroup
307     \csname c@\mpfn\endcsname #1\relax
308     \unrestored@protected@xdef\@thefnmark{\thempfn}%
309   \endgroup
310   \@footnotetext}

```

```

\thempfn
\@mpfn 311 \def\@mpfn{footnote}
        312 \def\thempfn{\thefootnote}
        313 \</2ekernel>

```

File H

ltidxglo.dtx

63 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.
\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.
1 <*2ekernel>
2 \message{index,}

\makeindex
3 \def\makeindex{%
4   \newwrite\@indexfile

```

```

5 \immediate\openout\@indexfile=\jobname.idx
6 \def\index{\@bsphack\beginngroup
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

```

\@wrindex

```

13 \def\@wrindex#1{%
14     \protected@write\@indexfile{%
15         {\string\indexentry{#1}{\thepage}}}%
16 \endgroup
17 \@esphack}

```

\index

```

18 \def\index{\@bsphack\beginngroup \@sanitize\@index}

```

\@index

```

19 \def\@index#1{\endgroup\@esphack}

```

\makeglossary

```

20 \def\makeglossary{%
21     \newwrite\@glossaryfile
22     \immediate\openout\@glossaryfile=\jobname.glo
23     \def\glossary{\@bsphack\beginngroup
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

```

\@wrglossary

```

30 \def\@wrglossary#1{%
31     \protected@write\@glossaryfile{%
32         {\string\glossaryentry{#1}{\thepage}}}%
33 \endgroup
34 \@esphack}

```

\glossary

```

35 \def\glossary{\@bsphack\beginngroup\@sanitize\@index}
36 \</2kernel>

```

File I

ltbibl.dtx

64 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.

<code>\bibliography</code>	<code>\bibliography{⟨file1,file2, ...,filen⟩}</code> : specifies the bibdata files. Writes a <code>\bibdata</code> entry on the <code>.aux</code> file and tries to read in <code>mainfile.bbl</code> .
<code>\bibliographystyle</code>	<code>\bibliographystyle{⟨style⟩}</code> : Writes a <code>\bibstyle</code> entry on the <code>.aux</code> file.
<code>thebibliography</code>	The <code>thebibliography</code> environment is a list environment. To save the use of an extra counter, it should use <code>enumiv</code> as the item counter. Instead of using <code>\item</code> , items in the bibliography are produced by the following commands: <code>\bibitem{⟨name⟩}</code> : Produces a numbered entry cited as <code>⟨name⟩</code> . <code>\bibitem[⟨label⟩]{⟨name⟩}</code> : Produces an entry labeled by <code>⟨Label⟩</code> and cited by <code>⟨name⟩</code> .

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.

<code>\cite</code>	Entries are cited by the command <code>\cite{⟨name⟩}</code> .
<code>\nocite</code>	<code>\nocite{⟨citations⟩}</code> puts information on the <code>.aux</code> file that causes <code>BIBTEX</code> to include the <code>{⟨citations⟩}</code> list in the bibliography, but puts nothing in the text. <code>\nocite{*}</code> is special: it tells <code>BIBTEX</code> to put the whole of a collection of references into the bibliography.

```
1 ⟨*2ekernel⟩
2 \message{bibliography,}
```

PARAMETERS

<code>@cite</code>	: A macro such that <code>\@cite{LABEL1,LABEL2}{NOTE}</code> produces the output for a <code>\cite[NOTE]{FOO1,FOO2}</code>
command,	where entry <code>FOOi</code> is defined by <code>\bibitem[LABELi]{FOOi}</code> . The switch <code>@tempswa</code> is true if the optional <code>NOTE</code>
argument	is present. The default definition is : <pre>\@cite{LABELS}{NOTE} == BEGIN [LABELS IF @tempswa = T THEN , NOTE FI] END</pre>

`\@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`.

`\bibitem`

```
3 \def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

`\@lbibitem`

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

`\@bibitem`

```
8 \def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
9     {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

`\bibcite`

```
10 \def\bibcite{\@newl@bel b}
```

`\citation`

```
11 \let\citation\@gobble
```

`\cite`

```
12 \DeclareRobustCommand\cite{%
13   \@ifnextchar [{\@tempswatrue\@citex}{\@tempswafalse\@citex[]}]}
```

`\@citex` `\penalty\@m` added to definition of `\@citex` to allow a line break after the ‘,’ in citations like [Jones80,Smith77] (Added 23 Oct 86)
 space added after the ‘,’ (21 Nov 87)

```
14 \def\@citex[#1]#2{\leavevmode
15   \let\@citea\@empty
16   \@cite{\@for\@citeb:=#2\do
17     {\@citea\def\@citea{\penalty\@m }%
18     \edef\@citeb{\expandafter\@firstofone\@citeb\@empty}}%
19   \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
```

Using `\hbox` instead of `\mbox` is fine because of the `\leavevmode` above. In fact the use of a box around the citation contents is more than questionable in my view (FMi), but within 2e I have to keep that for compatibility reasons as it would probably change too many existing documents. Its main reason is to avoid hyphenation of labels such as [FOOB89] into [FOO- B89] so in certain styles it makes sense; but, for example, in author year citations it becomes more than questionable.

So Chris added yet another hook here, as suggested by, at least, Donald Arseneau. Note that this one is inside the first argument of the `\@cite` hook. This decouples the top-level typesetting of the citation from the details of the other business conducted here. All this really needs a complete rethink to get the right modularity.

```

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

```

`\bibdata`

```

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

```

`\bibliography`

```

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

```

`\bibliographystyle`

```

32 \def\bibliographystyle#1{%
33   \ifx\@begindocumenthook\@undefined\else
34     \expandafter\AtBeginDocument
35   \fi
36   {\if@files
37     \immediate\write\@auxout{\string\bibstyle{#1}}%
38   \fi}}

```

`\nocite` (Added 14 Jun 85)

This puts information on the `.aux` file that causes BibTeX to include the citation list in the bibliography, but puts nothing in the text.

RmS 93/08/06: Made loop for `\nocite` like that for `\citex`, to get rid of leading spaces.

```

39 \def\nocite#1{\@bsphack

```

With the implementation designed already in L^AT_EX 2.09 the `\nocite` command will not work before `\begin{document}` since it tries to write to the `.aux` file which is not open before that point. As a result the “reference” will appear on the terminal and nothing else will happen.

This would be easy to fix, but then a document using the fix will silently fail on an older release of L^AT_EX, missing all citations done with `\nocite`. Thus we do only generate an error message and leave the fix for a L^AT_EX 2_ε successor.

```

40   \ifx\@onlypreamble\document

```

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

```

41     \@for\@citeb:=#1\do{%
42       \edef\@citeb{\expandafter\@firstofone\@citeb}%
43       \if@files\immediate\write\@auxout{\string\citation{\@citeb}}\fi
44       \ifundefined{b@\@citeb}{\G@refundefinedtrue

```

```

45      \latex@warning{Citation ‘\@citeb’ undefined}}{}%
46  \else

```

But before `\begin{document}` we raise an error message:

```

47  \latex@error{Cannot be used in preamble}\@eha

```

Without the compatibility problems we could fix the problem as follows:

```

48  % \AtBeginDocument{\nocite{#1}}
49  \fi
50  \esphack}

```

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).

```

51 \expandafter\let\csname b@*\endcsname\@empty

```

64.1 Default definitions

This hook determines the ‘relative formatting’ of the two logical parts of a citation with comment.

`\@cite`

```

52 \def\@cite#1#2{[#1\if@tempswa , #2\fi]}

```

`\@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.

```

53 \let\@cite@ofmt\hbox

```

`\@biblabel`

```

54 \def\@biblabel#1{[#1]}
55 \</2kernel>

```

File J

ltpage.dtx

65 Page styles and related commands

65.1 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.

65.2 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.)

65.3 marking conventions

L^AT_EX extends T_EX'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 T_EX's `\botmark`.

`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like T_EX'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 `\mkboth` command, which is `\let` by the `pagestyle` command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

```
1 \<2kernel>
```

`\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}}}
```

`\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}}

```

`\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}

```

`\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}

```

`\@leftmark` 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

```

`\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`

```

18 \def\markboth#1#2{%
19   \begingroup
20     \let\label\relax \let\index\relax \let\glossary\relax
21     \unrestored@protected@xdef\@themark {{#1}{#2}}%
22     \@temptokena \expandafter{\@themark}%
23     \mark{\the\@temptokena}%
24   \endgroup
25   \if@nobreak\ifvmode\nobreak\fi\fi}
26 \def\markright#1{%
27   \begingroup
28     \let\label\relax \let\index\relax \let\glossary\relax

```

Protection is handled inside `\@markright`.

```

29     \expandafter\@markright\@themark {#1}%
30     \@temptokena \expandafter{\@themark}%
31     \mark{\the\@temptokena}%
32   \endgroup
33   \if@nobreak\ifvmode\nobreak\fi\fi}

```

`\@markright`

```

\leftmark 34 \def\@markright#1#2#3{\@temptokena {#1}%
\rightmark 35   \unrestored@protected@xdef\@themark{{\the\@temptokena}{#3}}}
36 \def\leftmark{\expandafter\@leftmark\botmark\@empty\@empty}
37 \def\rightmark{\expandafter\@rightmark\firstmark\@empty\@empty}

```

`\@themark` Initialise L^AT_EX's marks without setting a T_EX mark (*whatsit*).

```

38 \def\@themark{{}{}}

```

`\mark` Test versions of L^AT_EX 2_ε initialised T_EX's `\mark` system at this point, but this was removed before the first release.

`\AtBeginDocument{\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`.)

```

39 \def\raggedbottom{%
40   \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}

```

`\flushbottom` `\flushbottom`: Inverse of `\raggedbottom` — makes all pages the same height.

```

41 \def\flushbottom{%
42   \let\@textbottom\relax \let\@texttop\relax}

```

`\sloppy` `\sloppy` will never (well, hardly ever) produce overfull boxes, but may produce underfull ones. (14 June 85)

```

43 \def\sloppy{%
44   \tolerance 9999%
45   \emergencystretch 3em%
46   \hfuzz .5\p@
47   \vfuzz\hfuzz}

```

`sloppypar` A `sloppypar` environment is equivalent to `{\par \sloppy ... \par}`.

```

48 \def\sloppypar{\par\sloppy}
49 \def\endsloppypar{\par}

```

`\fussy` Resets T_EX's parameters to their normal finicky values.

```

50 \def\fussy{%
51   \emergencystretch\z@
52   \tolerance 200%
53   \hfuzz .1\p@
54   \vfuzz\hfuzz}

```

`\overfullrule` L^AT_EX default is no overfull box rule. Changed by document class option.

```

55 \overfullrule 0pt

```

56 `</2ekernel>`

File K

ltoutput.dtx

66 Output Routine

66.1 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
5 <*2ekernel | autoload>
6 \message{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 oats.

`\textfloatsep` : Space between last top oat or first bottom oat and the text.

`\topfigrule` : Command to place rule (or whatever) between oats at top of page and text. Executed in inner vertical mode right before the `\textfloatsep` skip separating the oats from the text. Must occupy zero vertical space. (See `\footnoterule`.)

`\botfigrule` : Same as `\topfigrule`, but put after the `\textfloatsep` skip separating text from the oats at bottom of page.

`\intextsep` : Space left on top and bottom of an in-text oat.

`\dblfloatsep` : Space between double-column oats.

`\dbltextfloatsep` : Space between top double-column oats and text.

`\dblfigrule` : Similar to `\topfigrule`, but for double-column oats.

`\@fptop` : Glue to go at top of oat column – must be 0pt + stretch

`\@fpsep` : Glue to go between oats in a oat column.

`\@fpbot` : Glue to go at bottom of oat column – must be 0pt + stretch

`\@dblfpptop`, `\@dblfpsep`, `\@dblfpbot` : Analogous for double-column oat 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 oats on page.

Note that some are set globally and some locally:

```

\@topnum :=G Maximum number of oats allowed on the top of a
column.
\@toproom :=G Maximum amount of top of column devoted to oats—
excluding \textfloatsep separation below the oats
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 oats allowed in a column,
including in-text oats.

`\@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
oats.

`\@fpmin` :=L Minimum height of oats in a oat column.

The macro `\@dblfloatplacement` sets the following parameters.

`\@dbltopnum` :=G Maximum number of double-column oats allowed
at
the top of a two-column page.

`\@dbltoproom` :=G Maximum height of double-column oats allowed at
top of two-column page.

`\@fpmin` :=L Minimum height of oats in a oat 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 oats
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
oats committed to the top and bottom of the current
column.

`\@textfloatsheight` : The total height of in-text oats 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{} \penalty -10001</code>)
-10002	oat insertion, called from horizontal mode
-10003	oat insertion, called from vertical mode.
-10004	oat insertion.

Note: A `oat` 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 `oat` 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 `oat` column composed of `oats` from `\FLIST` (if nonempty) with the following parameters:

```
\@colht : height of box
\@fpmin : minimum height of oats in the box
\@fpsep : interoat space
\@fptop : glue at top of box
\@fpbot : glue at bottom of box.
```

If it succeeds, then it does the following:

```
* \@outputbox :=L the composed oat box.
* \@fcolmade  :=G true
* \FLIST      :=G \FLIST - oats put in box
* \@freelist  :=G \@freelist + oats put in box
```

If it fails, then:

```
* \@fcolmade :=G false
```

NOTE: BIT MUST BE A SINGLE TOKEN!

`\@makefcolumn \FLIST` : Same as `\@tryfcolumn` except that it fails to make a oat column only if `\FLIST` is empty. Otherwise, it makes a oat column containing at least the first box in `\FLIST`, disregarding `\@fpmin`.

`\@startcolumn` :

- Calls `\@tryfcolumn\@deferlist`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:
 - * Globally sets `\@toplist` and `\@botlist` to oats 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 oat placement parameters `\@topnum`, `\@toproom`, `\@botnum`, `\@botroom`, `\@colnum` and `\textfraction`.
 - * Globally sets `\@colroom` to `\@colht` minus the height of the added oats.

`\@startdblcolumn` :

- Calls `\@tryfcolumn\@dbldeferlist{8}`. If `\@tryfcolumn` returns with (globally set) `@fcolmade = false`, then:
 - * Globally sets `\@dbltoplist` to oats 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 oats.

`\@combinefloats` : Combines the text from box `\@outputbox` with the oats from `\@toplist` 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 oats in `\@toplist` 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 oat 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 oat environment obtains an insertion number B from the `\@freelist` (see below for a description of list manipulation), puts the oat into box B and sets `\count B` to a FLOAT SPECIFIER. For a normal (not double-column) oat, it then causes a page break in one of the following two ways:

- In outer hmode: `\adjust{\penalty -10002}`
- In vmode : `\penalty -10003`.

For a double-column oat, it puts B onto the `\@dbldeferlist`.

The oat specifier has two components:

- * A PLACEMENT SPECIFICATION, describing where the oat may be placed.
- * A TYPE, which is a power of two—e.g., figures might be

type 1 oats, tables type 2 oats, programs type 4 oats, etc.
The oat specifier is encoded as follows, where bit 0 is the least significant bit.

Bit	Meaning
0	1 iff the oat may go where it appears in the text.
1	1 iff the oat may go on the top of a page.
2	1 iff the oat may go on the bottom of a page.
3	1 iff the oat may go on a oat page.
4	1 unless the PLACEMENT includes a !
5	1 iff a type 1 oat
6	1 iff a type 2 oat
etc.	

A negative oat specifier is used to indicate a marginal note.

MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS

A FLOAT LIST consisting of the oats 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 oat 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 oat 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 `log2 \NUM` of the oat specifiers of all the oats in `\LIST`.
I.e., `@test` is set to true iff there is at least one oat in `\LIST` having bit `log2 \NUM` of its oat specifier equal to 1.

Note: $\log_2 [(\backslash\text{count I})/32]$ is the bit number corresponding to the type of oat I. To see if there is any oat in $\backslash\text{LIST}$ having the same type as oat I, you run $\backslash\text{@bitor}$ with $\backslash\text{NUM} = [(\backslash\text{count I})/32] * 32$.

```
\@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
```

$\backslash\text{@cons\LIST\NUM}$: Globally sets $\backslash\text{LIST} := \backslash\text{LIST} * \backslash\text{@elt \NUM}$

```
\@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 oats.
\@toplist       : List of oats to go at top of current column.
\@midlist       : List of oats in middle of current column.
\@botlist       : List of oats to go at bottom of current column.
\@deferlist     : List of oats to go after current column.
\@dbltoplist    : List of double-col. oats to go at top of current
                  page.
\@dbldeferlist  : List of double-column oats to go on subsequent
                  pages.
```

FLOAT-PLACEMENT ALGORITHMS

$\backslash\text{@addtobot}$: Tries to put insert $\backslash\text{@currbox}$ on $\backslash\text{@botlist}$.

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 oat of the same type is
                put on the bottom.
                If the oat is put in the text, then
                \penalty\interlinepenalty is put
                right after the oat, 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.

\@addtodblcol : Tries to add \@currbox to the next double-column page,
                adding it to \@dbltoplist if it succeeds and
                \@dbldeferlist 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
                + \marginparpush
\@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 @tempcnta > 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.

```

7 \maxdeadcycles = 100

8 \let\@elt\relax

9 \def\@next#1#2#3#4{\ifx#2\@empty #4\else
10   \expandafter\@xnext #2\@@#1#2#3\fi}

11 \def\@xnext \@elt #1#2\@@#3#4{\def#3{#1}\gdef#4{#2}}

   \changes{v1.1v}{1996/07/26}{put \cs{global} into definition}

12 \def\@testfalse{\global\let@if@test\iffalse}
13 \def\@testtrue {\global\let@if@test\iftrue}
14 \@testfalse

```

```

\changes{v1.1v}{1996/07/26}{remove \cs{global} before \cs{@test...}}
15 \def\@bitor#1#2{\@testfalse {\let\@elt\@xbitor
16   \@tempcnta #1\relax #2}}

```

RmS 91/11/22: Added test for $|\text{count}\#1| = 0$.

Suggested by Chris Rowley.

```

\changes{v1.1v}{1996/07/26}{remove \cs{global} before \cs{@test...}}
17 \def\@xbitor #1{\@tempcntb \count#1
18   \ifnum \@tempcnta =\z@
19   \else
20     \divide\@tempcntb\@tempcnta
21     \ifodd\@tempcntb \@testtrue\fi
22   \fi}

```

DEFINITION OF FLOAT BOXES:

```

23 \newinsert\bx@A
24 \newinsert\bx@B
25 \newinsert\bx@C
26 \newinsert\bx@D
27 \newinsert\bx@E
28 \newinsert\bx@F
29 \newinsert\bx@G
30 \newinsert\bx@H
31 \newinsert\bx@I
32 \newinsert\bx@J
33 \newinsert\bx@K
34 \newinsert\bx@L
35 \newinsert\bx@M
36 \newinsert\bx@N
37 \newinsert\bx@O
38 \newinsert\bx@P
39 \newinsert\bx@Q
40 \newinsert\bx@R

41 \gdef\@freelist{\@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
42   \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
43   \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
44   \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}

45 \gdef\@toplist{}
46 \gdef\@botlist{}
47 \gdef\@midlist{}
48 \gdef\@currlist{}
49 \gdef\@deferlist{}
50 \gdef\@dbltoplist{}
51 \gdef\@dbldeferlist{}

```

PAGE LAYOUT PARAMETERS

```

52 \newdimen\topmargin
53 \newdimen\oddsidemargin
54 \newdimen\evensidemargin
55 \let\themargin=\oddsidemargin

```

```

56 \newdimen\headheight
57 \newdimen\headsep
58 \newdimen\footskip
59 \newdimen\textheight
60 \newdimen\textwidth
61 \newdimen\columnwidth
62 \newdimen\columnsep
63 \newdimen\columnseprule
64 \newdimen\marginparwidth
65 \newdimen\marginparsep
66 \newdimen\marginparpush

\AtBeginDvi We use a box register in which to put stuff that must appear before anything else
\@beginvibox in the .dvi file.
    The stuff in the box should not add any typeset material to the page when it
    is unboxed.
67 \newbox\@beginvibox
68 \def \AtBeginDvi #1{%
69     \global \setbox \@beginvibox
70     \vbox{\unvbox \@beginvibox #1}%
71 }

\@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?
72 \newdimen\@maxdepth
73 \@maxdepth = \maxdepth

\paperheight New \paper... registers.
\paperwidth 74 \newdimen\paperheight
75 \newdimen\paperwidth

\if@insert Local switches first:
\if@fcolmade 76 \newif \if@insert
\if@specialpage These should definitely be global:
\if@firstcolumn 77 \newif \if@fcolmade
\if@twocolumn 78 \newif \if@specialpage \@specialpagefalse
\if@twoside
\if@reversemarginpar These should be global but are not always set globally in other files.
\if@mparswitch 79 \newif \if@firstcolumn \@firstcolumntrue
\col@number 80 \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?
81 \newif \if@twoside \@twosidefalse
82 \newif \if@reversemargin \@reversemarginfalse
83 \newif \if@mparswitch \@mparswitchfalse

This counter has been imported from ‘multicol’.
84 \newcount \col@number
85 \col@number \@ne

```

INTERNAL REGISTERS

```

86 \newcount\@topnum
87 \newdimen\@toproom
88 \newcount\@dbltopnum
89 \newdimen\@dbltoproom
90 \newcount\@botnum
91 \newdimen\@botroom
92 \newcount\@colnum
93 \newdimen\@textmin
94 \newdimen\@fpmin
95 \newdimen\@colht
96 \newdimen\@colroom
97 \newdimen\@pageht
98 \newdimen\@pagedp
99 \newdimen\@mparbottom \@mparbottom\z@
100 \newcount\@currtype
101 \newbox\@outputbox
102 \newbox\@leftcolumn
103 \newbox\@holdpg

104 \def\@thehead{\@oddhead} % initialization
105 \def\@thefoot{\@oddfoot}

```

`\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 oat box, never to be seen again. We hope that it does not produce wrong formatting in other cases.

```

106 \def\clearpage{%
107   \ifvmode
108     \ifnum \@dbltopnum =\m@ne
109       \ifdim \pagetotal <\topskip
110         \hbox{}%
111       \fi
112     \fi
113   \fi
114   \newpage
115   \write\m@ne{}%
116   \vbox{}%
117   \penalty -\@Mi
118 }

```

`\cleardoublepage`

```

119 \def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
120   \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
121 </2ekernel | autoload>

```

`\onecolumn`

```

122 <*2ekernel | autoload | trace >
123 \def\onecolumn{%
124   \clearpage
125   \global\columnwidth\textwidth
126   \global\hsize\columnwidth

```

```

127 \global\linewidth\columnwidth
128 \global\@twocolumnfalse
129 \col@number \@one
130 \@floatplacement}

```

\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 ags explicitly, even when it would appear that this would be done by a **\leavevmode**.

```

131 \def \newpage {%
132   \if@noskipsec
133     \ifx \@nodocument\relax
134       \leavevmode
135       \global \@noskipsecfalse
136     \fi
137   \fi
138   \if@inlabel
139     \leavevmode
140     \global \@inlabelfalse
141   \fi
142   \if@nobreak \@nobreakfalse \everypar{}\fi
143   \par
144   \vfil
145   \penalty -\@M}

```

\@emptycol It may be better to use an invisible rule rather than an empty box here.

```

146 \def \@emptycol {\vbox{}}\penalty -\@M}

```

\twocolumn There are several bug fixes to the two-column stuff here.

```

\@topnewpage 147 \def \twocolumn {%
148   \clearpage
149   \global\columnwidth\textwidth
150   \global\advance\columnwidth-\columnsep
151   \global\divide\columnwidth\tw@
152   \global\hsize\columnwidth
153   \global\linewidth\columnwidth
154   \global\@twocolumntrue
155   \global\@firstcolumntrue
156   \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.

```

157   \ifnextchar [\@topnewpage\@floatplacement
158 }

```

Note that here, getting a box from the freelist can assume success since this comes just after a **\clearpage**.

```

159 \long\def \@topnewpage [#1]{%
160   \@nodocument
161   \@next\@currbox\@freelist{}}}%

```

```

162 \global \setbox\@currbox
163 \color@vbox
164 \normalcolor
165 \vbox {%
166 \hsize\textwidth
167 \@parboxrestore
168 \col@number \@ne
169 #1%
170 \vskip -\dbltextfloatsep
171 }%
172 \color@endbox

```

Added size test and warning message; perhaps we should use an error message.

```

173 \ifdim \ht\@currbox>\textheight
174 \ht\@currbox \textheight
175 \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 `\@addtodblcol`, plus some extra checks for error trapping.

```

176 \global \count\@currbox \tw@
177 \@tempdima -\ht\@currbox
178 \advance \@tempdima -\dbltextfloatsep
179 \global \advance \@colht \@tempdima
180 \ifx \@dbltoplist \@empty
181 \else
182 \@latexerr{Float(s) lost}\@ehb
183 \let \@dbltoplist \@empty
184 \fi
185 \@cons \@dbltoplist \@currbox

```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```

186 \global \@dbltopnum \m@ne
187 \if*trace
188 \tr@ce{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
189 \fi

```

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.

```

190 \ifdim \@colht<2.5\baselineskip
191 \latex@warning@no@line {Optional argument of \noexpand\twocolumn
192 too tall on page \thepage}%
193 \@emptycol
194 \if@firstcolumn
195 \else
196 \emptycol
197 \fi
198 \else
199 \global \vsize \@colht
200 \global \@colroom \@colht
201 \floatplacement
202 \fi
203 }

```

`\output` This needs some small adjustments. We cannot guarantee that the oat 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.

```

204 \output {%
205 \let \par \@par
206 \ifnum \outputpenalty<-\@M
207 \specialoutput
208 \else
209 \makecol
210 \opcol

```

Moved to `\@opcol`: `\@floatplacement`.

```

211 \@startcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```

212 \@whilesw \if@fcolmade \fi
213 {%
214 (*trace)
215 \tr@ce{PAGE: float \if@twocolumn column \else page \fi
216 completed}%
217 </trace>
218 \@opcol\@startcolumn}%
219 \fi
220 \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 oat 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 oats, 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`.)

```

221 \ifdim \@colroom<1.5\baselineskip
222 \ifdim \@colroom<\textheight
223 \latex@warning@no@line {Text page \thepage\space
224 contains only floats}%
225 \@emptycol
226 % \if@twocolumn
227 % \if@firstcolumn
228 % \else
229 % \@emptycol
230 % \fi
231 % \fi
232 \else
233 \global \vsize \@colroom
234 \fi
235 \else
236 \global \vsize \@colroom
237 \fi
238 \else
239 \global \vsize \maxdimen
240 \fi
241 }
242 </2ekernel | autoload | trace >

```

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.)

```

243 <*2ekernel | def1 | autoload | trace >
244 \gdef\@specialoutput{%
245 \ifnum \outputpenalty>-\@Mii
246 \doclearpage
247 \else
248 \ifnum \outputpenalty<-\@Miii
249 \ifnum \outputpenalty<-\@MMM \deadcycles \z@ \fi
250 \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
251 \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.

```
252      \global \setbox\@holdpg \vbox{%
253          \unvbox\@holdpg
254          \unvbox\cclv
```

We must now remove the box added by the oat mechanism and the `\topskip` glue therefore added above it by \TeX .

```
255          \setbox\@tempboxa \lastbox
256          \unskip
257      }%
```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```
258      \@pagedp \dp\@holdpg
259      \@pageht \ht\@holdpg
260      \unvbox \@holdpg
261      \@next\@currbox\@currlist{%
262          \ifnum \count\@currbox>\z@
```

Putting the whole size into `\@pageht` (see above).

```
263          \advance \@pageht \@pagedp
264          \ifvoid\footins \else
265              \advance \@pageht \ht\footins
266              \advance \@pageht \skip\footins
267              \advance \@pageht \dp\footins
268          \fi
269 \*2ekernel | defl)
270          \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 oats, but maybe it still could make some adjustment here even so?

```
271          \ifdim \wd\@kludgeins=\z@
272              \advance \@pageht \ht\@kludgeins
273 \*trace)
274          \tr@ce {Extra size added: \the \ht\@kludgeins}%
275 \trace)
276          \fi
277          \fi
278 \2ekernel | defl)
```

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.

```
279          \reinserts
280          \addtocurcol
281      \else
282          \reinserts
283          \@addmarginpar
284      \fi
```

```
285 } \@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 ag is true, a `\nobreak` must be correct.

```
286 \ifnum \outputpenalty<\z@
287 \if@nobreak
288 \nobreak
289 \else
290 \addpenalty \interlinepenalty
291 \fi
292 \fi
293 \fi
294 \fi
295 }
296 \<2kernel | def1 | autoload | trace >
```

`\@docclearpage` This is a very much an emergency action, just dumping everything: footnotes first then oats. 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 oat 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 uncancellable.

```
297 \<*2kernel | autoload)
298 \def \@docclearpage {%
299 \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 oats, but who cares? The whole of this stuff needs completely redoing for many such reasons.

```
300 \ifvbox\@kludgeins
301 {\setbox \@tempboxa \box \@kludgeins}%
302 \<*trace)
303 \tr@ce {kludgeins box made void}%
304 \</trace)
305 \fi
306 \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
307 \setbox\@tempboxa\box\@cclv
308 \xdef\@deferlist{\@toplist\@botlist\@deferlist}%

309 \global \let \@toplist \@empty
310 \global \let \@botlist \@empty
311 \global \@colroom \@colht
312 \ifx \@currlist\@empty
313 \else
314 \latexerr{Float(s) lost}\@ehb

315 \global \let \@currlist \@empty
316 \fi
317 \makefcolumn\@deferlist
318 \@whilesw\if@colmade \fi{\@opcol\@makefcolumn\@deferlist}%
319 \if@twocolumn
```

```

320     \if@firstcolumn
321     \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

322     \global \let \@dbltoplist \@empty
323     \global \@colht \textheight
324     \begingroup
325     \@dblfloatplacement
326     \@makefcolumn\@dbldeferlist
327     \@whiles\if@fcolmade \fi{\@outputpage
328                                     \@makefcolumn\@dbldeferlist}%
329     \endgroup
330     \else
331     \vbox{}\clearpage
332     \fi
333     \fi
334     \else
335     \setbox\@cc1v\vbox{\box\@cc1v\vf1}%
336     \@makecol\@opcol
337     \clearpage
338     \fi
339 }
340 </2ekernel | autoloading>

```

`\@opcol` Several changes in detail here.

```

341 <*2ekernel | autoloading | trace >
342 \def \@opcol {%
343     \if@twocolumn
344     \@outputdblcol
345     \else
346     \@outputpage
347 <*trace>
348     \trace{PAGE: one column (float? see above) page completed}%
349 </trace>

```

Not needed since it comes after `\@outputpage`:

```

350 %     \global\@colht\textheight
351 \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.

```

352 \global \mparbottom \z@ \global \textfloatsheight \z@
353 \floatplacement
354 }
355 </2ekernel | autoloading | trace >

```

`\@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.

```

356 <*2ekernel | def1 | autoloading>
357 \gdef \@makecol {%
358     \ifvoid\footins
359     \setbox\@outputbox \box\@cc1v
360     \else
361     \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 oats.

```

362      \boxmaxdepth \@maxdepth
363 %      \@tempdima\dp\@cclv
364      \unvbox \@cclv
365 %      \vskip-\@tempdima
366      \vskip \skip\footins

367      \color@begingroup
368      \normalcolor
369      \footnoterule
370      \unvbox \footins
371      \color@endgroup
372      }%
373 \fi

```

The h oats have now been finally committed to this page so we can reset their list. The top and bottom oats are then added to the page.

```

374 \let\@elt\relax
375 \xdef\@freelist{\@freelist\@midlist}%

376 \global \let \@midlist \@empty
377 \@combinefloats

```

The variations start here in case `\enlargethispage` has been used.

```

378 (*2ekernel | def1)
379 \ifvbox\@kludgeins
380 \@makespecialcolbox
381 \else
382 /2ekernel | def1)

```

This extra reboxing is only needed to add the `\@texttop` and `\@textbottom` but this could be done earlier, when the oats 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 `\dimen@`.

```

383 \setbox\@outputbox \vbox to\@colht {%
384 % \boxmaxdepth \maxdepth %??
385 \@texttop
386 \dimen@ \dp\@outputbox

```

```

387      \unvbox \@outputbox
388      \vskip -\dimen@
389      \@textbottom
390      }%
391 \<*2ekernel | def1>
392   \fi
393 \</2ekernel | def1>
394   \global \maxdepth \@maxdepth
395 }
396 \</2ekernel | def1 | autoloading>

```

\@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.

```

397 \<*2ekernel | def1 | autoloading>
398 \gdef \@reinserts{%
399   \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
400 \<+2ekernel | def1>   \ifvbox\@kludgeins\insert\@kludgeins
401 \<+2ekernel | def1>   {\unvbox\@kludgeins}\fi
402 }
403 \</2ekernel | def1 | autoloading>

```

\@makespecialcolbox This implements certain variations in page-makeup.

```

404 \<*2ekernel | def1 | trace >
405 \gdef \@makespecialcolbox {%
406 \<*trace>
407   \tr@ce{Kludgeins ht \the\ht\@kludgeins\space
408           dp \the\dp\@kludgeins\space
409           wd \the\wd\@kludgeins}%
410 \</trace>

```

First we find the natural height of the column.

See above for discussion of what is happening here.

This needs further investigation, especially in this ‘special case’.

```

411   \setbox\@outputbox \vbox {%
412     \@texttop
413     \dimen@ \dp\@outputbox
414     \unvbox\@outputbox
415     \vskip-\dimen@
416     }%
417   \@tempdima \@colht
418   \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 T_EX level!).

This needs T_EX3 otherwise \pageshrink is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate

of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for both the ush- 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.

```

419     \advance \@tempdima -\ht\@outputbox
420     \advance \@tempdima \pageshrink
421 (*trace)
422     \tr@ce {Natural ht of col: \the \ht\@outputbox}%
423     \tr@ce {\string \@colht: \the \@colht}%
424     \tr@ce {Pageshrink added: \the \pageshrink}%
425     \tr@ce {Hence, space added: \the \@tempdima}%
426 (/trace)
427     \setbox\@outputbox \vbox to \@colht {%
428 %       \boxmaxdepth \maxdepth
429     \unvbox\@outputbox
430     \vskip \@tempdima
431     \@textbottom
432     }%
```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the ush-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.

```

433     \else
434     \advance \@tempdima -\ht\@kludgeins
435 (*trace)
436     \tr@ce {Natural ht of col: \the \ht\@outputbox}%
437     \tr@ce {\string \@colht: \the \@colht}%
438     \tr@ce {Extra size added: -\the \ht \@kludgeins}%
439     \tr@ce {Hence, height of inner box: \the \@tempdima}%
440     \tr@ce {Max? pageshrink available: \the \pageshrink}%
441 (/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.

```

442     \setbox \@outputbox \vbox to \@colht {%
443     \vbox to \@tempdima {%
444     \unvbox\@outputbox
445     \@textbottom}%
446     \vss}%
447     \fi
```

Finally we need to explicitly make the insert box void.

```

448     {\setbox \@tempboxa \box \@kludgeins}%
449 (*trace)
450     \tr@ce {kludgeins box made void}%
```

```

451 </trace>
452 }
453 </2ekernel | def1 | trace >

\@texttop These do nothing as a default.
\@textbottom 454 <*2ekernel | autoload>
455 \let \@texttop \relax
456 \let \@textbottom \relax

\@resetactivechars RmS 93/09/06: added hook to protect against certain active characters in the
\@activechar@info output routine. Default checks are for active space and end-of-line.
457 \def\@activechar@info #1{%
458     \@latex@info@no@line {Active #1 character found while
459         output routine is active
460         \MessageBreak
461         This may be a bug in a package file
462         you are using}%
463 }

    Do not put any spaces in this next bit!
464 \begingroup
465 \obeylines\obeyspaces%
466 \catcode'\'\active%
467 \gdef\@resetactivechars{%
468 \def~M{\@activechar@info{EOL}\space}%
469 \def {\@activechar@info{space}\space}%
470 \let'\active@math@prime}%
471 \endgroup

\@outputpage The \color@hbox hooks here are used to avoid putting just a colour special into
\@shipoutsetup an otherwise empty box (in a header or footer). These boxes are often set to be
\@writesetup 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 con-
    structed 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.
472 \def\@outputpage{%
473 \begingroup % the \endgroup is put in by \aftergroup

```

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`.

```
474 \let \protect \noexpand
```

RmS 93/08/19: Redefined accents to allow changes in font encoding; but exactly why was this needed?

The `\catcode'\ = 10` was removed as it was considered useless (presumably because nothing gets tokenised during shipout).

This was put in as some error produced active spaces in a mark, I think.

Why was the hyphen reset?

```
475 \@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 ag.

```
476 \global\let\@if@newlist@if@newlist
```

```
477 \global\@newlistfalse
```

This next hook replaces the following:

```
\let\-\@dischyph
\let'\@acci\let'\@accii\let\=\@acciii
\let\\\@normalcr
\let\par\@par %% 15 Sep 87 (this was once inside the box)
```

and it does more than they did; in particular it sets:

```
\parindent\z@
\parskip\z@skip
\everypar{}%
\leftskip\z@skip
\rightskip\z@skip
\parfillskip\@flushglue
\lineskip\normallineskip
\baselineskip\normalbaselineskip
\sloppy
```

```
478 \@parboxrestore
```

... to here was in the command `\@writesetup`.

```
479 \shipout \vbox{%
```

```
480 \set@typeset@protect
```

```
481 \aftergroup \endgroup
```

```
482 \aftergroup \set@typeset@protect
```

```
483                                     % correct? or just restore by ending
```

```
484                                     % the group?
```

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`.

```
485 \if@specialpage
```

```
486 \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
```



```

487 \fi
488 \if@twoside
489 \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
490 \let\@themargin\oddsidemargin
491 \else \let\@thehead\@evenhead
492 \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
493 \fi
494 \fi

The rest was always inside the box.
RmS 91/08/15: aded this line:

495 \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.

496 \normalsize

Reset the space factors.

497 \normalsfcodes

Reset these here (previously reset separately for head and foot)

498 \let\label\@gobble
499 \let\index\@gobble
500 \let\glossary\@gobble
501 \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@

... to here was in the command \@shipoutsetup.

502 \@beginndvi
503 \vskip \topmargin
504 \moveright\@themargin \vbox {%
505 \setbox\@tempboxa \vbox to\headheight{%
506 \vfil
507 \color@hbox
508 \normalcolor
509 \hb@xt@\textwidth{\@thehead}%
510 \color@endbox
511 }% %% 22 Feb 87
512 \dp\@tempboxa \z@
513 \box\@tempboxa
514 \vskip \headsep
515 \box\@outputbox
516 \baselineskip \footskip
517 \color@hbox
518 \normalcolor
519 \hb@xt@\textwidth{\@thefoot}%
520 \color@endbox
521 }%
522 }%

\endgroup now inserted by \aftergroup
Restore \if@newlist

523 \global\let\if@newlist\@if@newlist
524 \global \colht \textheight
525 \stepcounter{page}%

```

It is now clear that this does something useful, thanks to Piet van Oostrum. It is needed because a oat page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```
526 \let\firstmark\botmark
527 }
```

`\@beginndvi` 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.

```
528 \def \@beginndvi{%
529   \unvbox \@beginndvibox
530   \global\let \@beginndvi \@empty
531 }
```

`\@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.

```
532 \def \@combinefloats {%
533 %   \boxmaxdepth \maxdepth
534   \ifx \@toplist\@empty \else \@cflt \fi
535   \ifx \@botlist\@empty \else \@cflb \fi
536 }

537 \def \@cflt{%
538   \let \@elt \@comflelt
539   \setbox\@tempboxa \vbox{%
540     \@toplist
541     \setbox\@outputbox \vbox{%
542       \boxmaxdepth \maxdepth
543       \unvbox\@tempboxa
544       \vskip -\floatsep
545       \topfigrule
546       \vskip \textfloatsep
547       \unvbox\@outputbox
548     }%
549   \let\@elt\relax
550   \xdef\@freelist{\@freelist\@toplist}%
551   \global\let\@toplist\@empty
552 }

553 \def \@cflb {%
554   \let\@elt\@comflelt
555   \setbox\@tempboxa \vbox{%
556     \@botlist
557     \setbox\@outputbox \vbox{%
558       \unvbox\@outputbox
559       \vskip \textfloatsep
560       \botfigrule
561       \unvbox\@tempboxa
562       \vskip -\floatsep
563     }%
564   \let\@elt\relax
565   \xdef\@freelist{\@freelist\@botlist}%
566   \global \let \@botlist\@empty
```

567 }

\@comflelt

```
\@comdblfilelt 568 \def\@comflelt#1{\setbox\@tempboxa
\@combinedblfloats 569 \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
570 \def\@comdblfilelt#1{\setbox\@tempboxa
571 \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
572 \def \@combinedblfloats{%
573 \ifx \@dbltoplist \@empty
574 \else
575 \setbox\@tempboxa \vbox{}%
576 \let \@elt \@comdblfilelt
577 \@dbltoplist
578 \let \@elt \relax
579 \xdef \@freelist {\@freelist\@dbltoplist}%
580 \global\let \@dbltoplist \@empty
581 \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.

```
582 {\boxmaxdepth\maxdepth %% probably not needed, CAR
583 \unvbox\@tempboxa\vskip-\dblfloatsep
```

Here we need different typesetting if the top oat comes from \@topnewpage.

```
584 \ifnum \@dbltopnum>\m@ne
585 \dblfigrule
586 \fi
587 \vskip \dbltextfloatsep
588 \box\@outputbox
589 }%
590 \fi
591 }
592 </2ekernel | autoload>
```

\@startcolumn We could combine (most of) these two into \@startcol <list>. Note that
\@startdblcolumn \@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.

```
593 < *2ekernel | autoload | trace >
594 \def \@startcolumn {%
595 \global \@colroom \@colht
596 \@tryfcolumn \@deferlist
597 \if@fcolmade
598 < *trace>
599 \tr@ce{PAGE: float \if@twocolumn column \else page \fi
600 completed}%
601 </trace>
602 \else
603 \begingroup
604 \let \reserved@b \@deferlist
```

```

605     \global \let \@deferlist \@empty
606     \let \@elt \@scolelt
607     \reserved@b
608   \endgroup
609 \fi
610 }

```

This one does not need to set \@colht.

```

611 \def \@startdblcolumn {%
    Not needed since this always comes after \@outputpage:
612 % \global \@colht \textheight
613   \@tryfcolumn \@dbldeferlist
614   \if@fcolmade
615 (*trace)
616   \tr@ce{PAGE: double float page completed}%
617 </trace>
618   \else
619     \begingroup
620     \let \reserved@b \@dbldeferlist
621     \global \let \@dbldeferlist \@empty
622     \let \@elt \@sdblcolelt
623     \reserved@b
624   \endgroup
625 \fi
626 }

```

\@tryfcolumn Now tests if its list is empty before any further exertion.

```

627 \def \@tryfcolumn #1{%
628   \global \@fcolmadefalse
629   \ifx #1\@empty
630     \else
631 (*trace)
632   \tr@ce{PAGE: try float \if@twocolumn column/page\else page\fi
633         ---\string #1}%
634   \tr@ce{----- \string #1: #1}%
635 </trace>
636   \xdef\@trylist{#1}%
637   \global \let \@failedlist \@empty
638   \begingroup
639     \let \@elt \@xtryfc \@trylist
640   \endgroup
641   \if@fcolmade
642     \@vtryfc #1%
643   \fi
644 \fi
645 }
646 </2ekernel | autoload | trace >
647 (*2ekernel | autoload)

```

\@scolelt

```

648 \def\@scolelt#1{\def\@currbox{#1}\@addtonextcol}

```

```

\@sdblcolelt
649 \def\@sdblcolelt#1{\def\@currbox{#1}\@addtodblcol}

\@vtryfc
650 \def\@vtryfc #1{%
651   \global\setbox\@outputbox\vbox{%
652     \let\@elt\@wtryfc
653     \@flsucceed
654     \global\setbox\@outputbox \vbox to\@colht{%
655       \vskip \@fptop
656       \vskip -\@fpsep
657       \unvbox \@outputbox
658       \vskip \@fpbot}%
659     \let\@elt\relax
660     \xdef #1{\@failedlist\@flfail}%
661     \xdef\@freelist{\@freelist\@flsucceed}}

\@wtryfc
662 \def\@wtryfc #1{%
663   \global\setbox\@outputbox\vbox{%
664     \unvbox\@outputbox
665     \vskip\@fpsep
666     \box #1}}

\@xtryfc
667 \def\@xtryfc #1{%
668   \@next\reserved@a\@trylist{}{}%
669   \@currtype \count #1%
670   \divide\@currtype\@xxxii
671   \multiply\@currtype\@xxxii
672   \@bitor \@currtype \@failedlist
673   \@testfp #1%
674   \ifdim \ht #1>\@colht
675     \@testtrue
676   \fi
677   \if@test
678     \@cons\@failedlist #1%
679   \else
680     \@ytryfc #1%
681   \fi}

\@ytryfc
682 \def\@ytryfc #1{%
683   \begingroup
684   \gdef\@flsucceed{\@elt #1}%
685   \global\let\@flfail\@empty
686   \@tempdima\ht #1%
687   \let\@elt\@ztryfc
688   \@trylist
689   \ifdim \@tempdima >\@fpmin
690     \global\@fcolmadetrue
691   \else
692     \@cons\@failedlist #1%

```

```

693   \fi
694 \endgroup
695 \if@fcolmade
696   \let\@elt\@gobble
697 \fi}

```

\@ztryfc

```

698 \def\@ztryfc #1{%
699   \@tempcnta \count#1%
700   \divide\@tempcnta\@xxxii
701   \multiply\@tempcnta\@xxxii
702   \@bitor \@tempcnta {\@failedlist \@flfail}%
703   \@testfp #1%
704   \@tempdimb\@tempdima
705   \advance\@tempdimb \ht#1%
706   \advance\@tempdimb\@fpsep
707   \ifdim \@tempdimb >\@colht
708     \testtrue
709   \fi
710   \if@test
711     \@cons\@flfail #1%
712   \else
713     \@cons\@flsucceed #1%
714     \@tempdima\@tempdimb
715   \fi}

```

```

716 </2ekernel | autoload>

```

The major changes for oat suppression and the changes to the oat mechanism to make it conform to the documentation are in these next macros.

\@addtobot Lots of changes.

```

717 <*2ekernel | autoload | trace >
718 \def \@addtobot {%
719 <*trace>
720   \tr@ce{***Start addtobot}%
721 </trace>
722   \@getfpsbit 4\relax
723 <*trace>
724   \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi bot:
725                                           \the \@fpstype}%
726 </trace>
727   \ifodd \@tempcnta
728     \@flsetnum \@botnum
729     \ifnum \@botnum>\z@
730       \@tempswafalse
731       \@flcheckspace \@botroom \@botlist
732     \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?

```

733     \global \maxdepth \z@
734     \@flupdates \@botnum \@botroom \@botlist
735 <*trace>

```

```

736      \tr@ce{colroom (after-bot) = \the \@colroom}%
737      \tr@ce{colnum (after-bot) = \the \@colnum}%
738      \tr@ce{botnum (after-bot) = \the \@botnum}%
739      \tr@ce{***Success: bot}%
740 \
```

\@addtotoporbot Lots of changes.

```

754 \def \@addtotoporbot {%
755 \
```

```

788     \fi
789 < *trace >
790     \else
791         \tr@ce{Fail: topnum = \the \@topnum: fpstype
792                                     \the \@fpstype=ORD?}%
793         \ifnum \@fpstype<\sist@n
794             \tr@ce{ERROR: !t float not successful (addtotoporbot)}%
795         \fi
796 < /trace >
797     \fi
798     \fi
799     \if@insert
800     \else
801 < *trace >
802         \tr@ce{sent to addtobot (addtotoporbot)}%
803 < /trace >
804         \@addtobot
805     \fi
806 }
807 < /2ekernel | autoloading | trace >

```

\@addtocurcol Lots of changes.

```

808 < *2ekernel | autoloading | trace | after >
809 \def \@addtocurcol {%
810 < *trace >
811     \tr@ce{***Start addtocurcol}%
812 < /trace >
813     \@insertfalse
814     \@setfloattypecounts
815     \ifnum \@fpstype=8
816 < *trace >
817         \tr@ce{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
818 < /trace >
819     \else
820         \ifnum \@fpstype=24
821 < *trace >
822             \tr@ce{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
823 < /trace >
824         \else
825             \@flsettextmin

```

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 oats, so before comparing it with \@textmin, we add this to \@textmin also.

```

826 < *trace >
827     \tr@ce{textfloatsheight (before) = \the \@textfloatsheight}%
828 < /trace >
829     \advance \@textmin \@textfloatsheight
830     \@reqcolroom \@pageht

```

This line must be removed since \@specialoutput changed.

```

831 %         \advance \@reqcolroom \@pagedp
832 < *trace >

```



```

833      \tr@ce{textmin + textfloatsheight: \the \@textmin}%
834      \tr@ce{page-so-far: \the \@reqcolroom}%
835 \tr@ce{
836      \ifdim \@textmin>\@reqcolroom
837      \@reqcolroom \@textmin
838 }*trace>
839      \tr@ce{ORD? textmin being used}%
840 \tr@ce{
841      \fi
842      \advance \@reqcolroom \ht\@currbox
843 }*trace>
844      \tr@ce{float size = \the \ht \@currbox (addtocurcol)}}%
845      \tr@ce{colroom = \the \@colroom (addtocurcol)}}%
846      \tr@ce{reqcolroom = \the \@reqcolroom (addtocurcol)}}%
847 \tr@ce{
848      \ifdim \@colroom>\@reqcolroom
849      \@flsetnum \@colnum
850      \ifnum \@colnum>\z@
851      \@bitor\@currtype\@deferlist
852 }*trace>
853      \tr@ce{deferlist: \@deferlist: (addtocurcol-before)}}%
854 \tr@ce{
855      \if@test
856 }*trace>
857      \tr@ce{type already on list: defer (addtocurcol)}}%
858 \tr@ce{
859      \else
860      \@bitor\@currtype\@botlist
861 }*trace>
862      \tr@ce{botlist: \@botlist: (addtocurcol-before)}}%
863 \tr@ce{
864      \if@test
865 }*trace>
866      \tr@ce{type already on list: bot---sent to addtobot}}%
867 \tr@ce{
868      \@addtobot
869      \else
870 }*trace>
871      \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi
872      here: \the \@fpstype}}%
873 \tr@ce{
874      \ifodd \count\@currbox
875      \advance \@reqcolroom \intextsep
876      \ifdim \@colroom>\@reqcolroom
877      \global \advance \@colnum \m@ne
878      \global \advance \@textfloatsheight \ht\@currbox
879      \global \advance \@textfloatsheight 2\intextsep
880      \@cons \@midlist \@currbox
881 }*trace>
882      \tr@ce{***Success: here}}%
883      \tr@ce{textfloatsheight (after-here) =
884      \the \@textfloatsheight}}%
885      \tr@ce{colnum (after-here) = \the \@colnum}}%

```

This may sometimes give an overestimate.

```

886 </trace>
      CHANGE TO \@addtocurcol:
      \penalty\z@ changed to \penalty\interlinepenalty so \samepage works
      properly with figure and table environments. (Changed 23 Oct 86)
      There is also an \addpenalty\interlinepenalty above.
      Since in 2e \samepage is no longer supported, these could be removed.
      Although it is best to use \addvspace in case two h oats come together, this
      makes other spacing more difficult to adjust; whereas if a user specifies two h oats
      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 oat is
      added between paragraphs (i.e. when in vertical mode).
      If the nobreak switch is true we need to reset it and clear \everypar sionce
      the oat may not reset the ag and cannot reset the \everypar globally.
      Typesetting starts here (we are in vertical mode).
887      \if@nobreak
888      \nobreak
889      \@nobreakfalse
890      \everypar{}%
891      \else
892      \addpenalty \interlinepenalty
893      \fi
894      \vskip \intextsep
895      \box\@currbox
896      \penalty\interlinepenalty
897      \vskip\intextsep
898      \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
      Typesetting ends here.
899      \outputpenalty \z@
900      \@inserttrue
901 <*trace>
902      \else
903      \tr@ce{Fail---no room at 2nd test of colroom
904      (addtocurcol \string\intextsep)}%
905 </trace>
906      \fi
907      \fi
908      \if@insert
909      \else
910 <*2ekernel | autoload | trace >
911 <*trace>
912      \tr@ce{not here: sent to addtotoporbot}%
913 </trace>
914      \@addtotoporbot
915 </2ekernel | autoload | trace >
916 <!*2ekernel&!autoload&!trace >
917 <*trace>
918      \tr@ce{not here: sent to addtobot}%
919 </trace>
920      \@addtobot
921 <!/2ekernel&!autoload&!trace >
922      \fi
923      \fi

```

```

924         \fi
925 < *trace >
926         \else
927             \tr@ce{Fail: colnum = \the \@colnum:
928                 fpstype \the \@fpstype=ORD?}%
929             \ifnum \@fpstype<\sist@n
930                 \tr@ce{ERROR: BANG float not successful (addtocurcol)}%
931             \fi
932 < /trace >
933         \fi
934 < *trace >
935         \else
936             \tr@ce{Fail---no room: fl box ht: \the \ht \@currbox
937                 (addtocurcol)}%
938 < /trace >
939         \fi
940     \fi
941 \fi
942 \if@insert
943 \else
944     \@resethfps
945 < *trace >
946     \tr@ce{put on deferlist (addtocurcol)}%
947 < /trace >
948     \@cons\@deferlist\@currbox
949 < *trace >
950     \tr@ce{deferlist: \@deferlist: (addtocurcol-after)}%
951 < /trace >
952 \fi
953 }
954 < /2ekernel | autoload | trace | after >

```

\@addtonextcol Lots of changes.

```

955 < *2ekernel | autoload | trace >
956 \def\@addtonextcol{%
957     \begingroup
958 < *trace >
959     \tr@ce{***Start addtonextcol}%
960 < /trace >
961     \@insertfalse
962     \@setfloattypescounts
963     \ifnum \@fpstype=8
964 < *trace >
965     \tr@ce{fpstype not curcol: \the \@fpstype = 8?}%
966 < /trace >
967     \else
968     \ifnum \@fpstype=24
969 < *trace >
970     \tr@ce{fpstype not curcol: \the \@fpstype = 24?}%
971 < /trace >
972     \else
973     \@flsettextmin
974 < *trace >
975     \tr@ce{text-so-far: 0pt (top of col)}%

```

```

976 </trace>
977 \@reqcolroom \ht\@currbox
978 <*trace>
979 \tr@ce{float size: \the \@reqcolroom (addtonextcol)}%
980 </trace>
981 \advance \@reqcolroom \@textmin
982 <*trace>
983 \tr@ce{colroom = \the \@colroom (addtonextcol)}%
984 \tr@ce{reqcolroom = \the \@reqcolroom (addtonextcol)}%
985 </trace>
986 \ifdim \@colroom>\@reqcolroom
987 \@flsetnum \@colnum
988 \ifnum \@colnum>z@
989 \@bitor\@currtype\@deferlist
990 <*trace>
991 \tr@ce{deferlist: \@deferlist: (addtonextcol-before)}%
992 </trace>
993 \if@test
994 <*trace>
995 \tr@ce{type already on list: defer (addtonextcol)}%
996 </trace>
997 \else
998 <*trace>
999 \tr@ce{sent to addtotoporbot (addtonextcol)}%
1000 </trace>
1001 \@addtotoporbot
1002 \fi
1003 \fi
1004 <*trace>
1005 \else
1006 \tr@ce{Fail---no room: fl box ht: \the \ht \@currbox
1007 (addtonextcol)}%
1008 </trace>
1009 \fi
1010 \fi
1011 \fi
1012 \if@insert
1013 \else
1014 <*trace>
1015 \tr@ce{put back on deferlist (addtonextcol)}%
1016 </trace>
1017 \@cons\@deferlist\@currbox
1018 <*trace>
1019 \tr@ce{deferlist: \@deferlist: (addtonextcol-after)}%
1020 </trace>
1021 \fi
1022 <*trace>
1023 \tr@ce{End of addtonextcol -- locally counts:}%
1024 \tr@ce{ col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
1025 </trace>
1026 \endgroup
1027 <*trace>
1028 \tr@ce{End of addtonextcol -- globally counts:}%
1029 \tr@ce{col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%

```

```

1030 </trace>
1031 }

```

\@addtodblcol Lots of changes.

```

1032 \def\@addtodblcol{%
1033   \begingroup
1034   \*trace
1035   \tr@ce{***Start addtodblcol}%
1036 </trace>
1037   \@insertfalse
1038   \@setfloattypecounts
1039   \@getfpsbit \tw@
1040 \*trace
1041   \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
1042                                           \the \@fpstype}%
1043 </trace>
1044   \ifodd \@tempcnta
1045     \@flsetnum \@dbltopnum
1046     \ifnum \@dbltopnum>\z@
1047       \@tempswafalse
1048       \ifdim \@dbltoproom>\ht\@currbox
1049         \@tempwattrue
1050 \*trace
1051     \tr@ce{Space OK: \@dbltoproom =
1052           \the \@dbltoproom > \the \ht \@currbox
1053           (dbltoproom)}%
1054 </trace>
1055   \else
1056 \*trace
1057   \tr@ce{fpstype: \the \@fpstype (addtodblcol)}%
1058 </trace>
1059   \ifnum \@fpstype<\sist@@n
1060 \*trace
1061   \tr@ce{BANG float ignoring \@dbltoproom}%
1062   \tr@ce{\@spaces \@dbltoproom = \the \@dbltoproom.
1063         Ht float: \the \ht \@currbox-BANG}%
1064 </trace>

```

Need to check that there is room on the page, using the local value of \@textmin to make the necessary adjustment to \@dbltoproom.

```

1065   \advance \@dbltoproom \@textmin
1066 \*trace
1067   \tr@ce{Local value of texmin: \the \@textmin}%
1068   \tr@ce{\@spaces space on page = \the \@dbltoproom.
1069         Ht float: \the \ht \@currbox-BANG}%
1070 </trace>
1071   \ifdim \@dbltoproom>\ht\@currbox
1072     \@tempwattrue
1073 \*trace
1074   \tr@ce{Space OK BANG: space on page = \the \@dbltoproom >
1075         \the \ht \@currbox}%
1076   \else
1077     \tr@ce{fpstype: \the \@fpstype}%
1078     \tr@ce{Fail---no room dbltoproom-BANG?:}%

```

```

1079          \tr@ce{\@spaces space on page = \the \@dbltoproom.
1080                  Ht float: \the \ht \@currbox}%
1081 \</trace>
1082 \fi
1083 \advance \@dbltoproom -\@textmin
1084 \<trace>
1085 \else
1086 \tr@ce{fpstype: \the \@fpstype}%
1087 \tr@ce{Fail---no room dbltoproom-ORD?:}%
1088 \tr@ce{\@spaces \@dbltoproom = \the \@dbltoproom.
1089         Ht float: \the \ht \@currbox}%
1090 \</trace>
1091 \fi
1092 \fi
1093 \if@tempswa
1094 \bitor \@currtype \@dbldeferlist
1095 \<trace>
1096 \tr@ce{dbldeferlist: \@dbldeferlist: (before)}%
1097 \</trace>
1098 \if@test
1099 \<trace>
1100 \tr@ce{type already on list: dbldefer}%
1101 \</trace>
1102 \else
1103 \@tempdima -\ht\@currbox
1104 \advance\@tempdima
1105 -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
1106 \dblfloatsep \fi
1107 \global \advance \@dbltoproom \@tempdima
1108 \global \advance \@colht \@tempdima
1109 \global \advance \@dbltopnum \m@ne
1110 \@cons \@dbltoplist \@currbox
1111 \<trace>
1112 \tr@ce{dbltopnum (after) = \the \@dbltopnum}%
1113 \tr@ce{***Success: dbltop}%
1114 \</trace>
1115 \@inserttrue
1116 \fi
1117 \fi
1118 \<trace>
1119 \else
1120 \tr@ce{Fail: dbltopnum = \the \@dbltopnum: fpstype
1121 \the \@fpstype=ORD?}%
1122 \ifnum \@fpstype<\sist@n
1123 \tr@ce{ERROR: !t float not successful (addtodblcol)}%
1124 \fi
1125 \</trace>
1126 \fi
1127 \fi
1128 \if@insert
1129 \else
1130 \<trace>
1131 \tr@ce{put on dbldeferlist}%
1132 \</trace>

```

```

1133     \@cons\@dbldeferlist\@currbox
1134 (*trace)
1135     \tr@ce{dbldeferlist: \@dbldeferlist: (after)}%
1136 </trace>
1137 \fi
1138 (*trace)
1139     \tr@ce{End of addtodblcol -- locally count:}%
1140     \tr@ce{dbltop: \the \@dbltopnum.}%
1141 </trace>
1142 \endgroup
1143 (*trace)
1144     \tr@ce{End of addtodblcol -- globally count:}%
1145     \tr@ce{dbltop: \the \@dbltopnum.}%
1146 </trace>
1147 }
1148 </2ekernel | autoload | trace >

```

\@addmarginpar

```

1149 (*2ekernel | autoload)
1150 \def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
1151     \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
1152     \if@twocolumn
1153         \if@firstcolumn \@tempcnta\m@ne \fi
1154     \else
1155         \if@mparswitch
1156             \ifodd\c@page \else\@tempcnta\m@ne \fi
1157         \fi
1158         \if@reversemargin \@tempcnta -\@tempcnta \fi
1159     \fi
1160     \ifnum\@tempcnta <\z@ \global\setbox\@marbox\box\@currbox \fi
1161     \@tempdima\@mparbottom
1162     \advance\@tempdima -\@pageht
1163     \advance\@tempdima\ht\@marbox
1164     \ifdim\@tempdima >\z@
1165         \@latexwarning@no@line {Marginpar on page \thepage\space moved}%
1166     \else
1167         \@tempdima\z@
1168     \fi
1169     \global\@mparbottom\@pageht
1170     \global\advance\@mparbottom\@tempdima
1171     \global\advance\@mparbottom\dp\@marbox
1172     \global\advance\@mparbottom\marginparpush
1173     \advance\@tempdima -\ht\@marbox

```

Putting box movement inside the ‘marbox’:

```

1174     \global\setbox \@marbox
1175         \vbox {\vskip \@tempdima
1176             \box \@marbox}%
1177     \global \ht\@marbox \z@
1178     \global \dp\@marbox \z@

```

Sticking (rather than gluing:-) the ‘marbox’ to the line above, changed vskip to kern:

```

1179     \kern -\@pagedp
1180     \nointerlineskip

```

```

1181 \hb@xt@\columnwidth
1182 {\ifnum \@tempcnta >\z@
1183 \hskip\columnwidth \hskip\marginparsep
1184 \else
1185 \hskip -\marginparsep \hskip -\marginparwidth
1186 \fi
1187 \box\@marbox \hss}%

```

For this reason the following code can vanish:

```

\ nobreak          %% No longer needed. CAR92/12
\ vskip -\@tempdima %% No longer needed. CAR92/12

1188 \nointerlineskip
1189 \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}
1190 </2kernel | autoload>

```

66.1.1 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

: please give keep them clear of such places.

`\@kludgeins` The insert which makes TeX do a lot of the necessary work. All we need to put into it is the amount by which the pagegoal should be changed.

```
1191 <*2ekernel | def1>
1192 \newinsert \@kludgeins
1193 \global\dimen\@kludgeins \maxdimen
1194 \global\count\@kludgeins 1000
1195 </2ekernel | def1>
```

`\enlargethispage` The user command.

```
\enlargethispage* 1196 <*2ekernel | def1>
1197 \gdef \enlargethispage {%
1198   \@ifstar
1199   {%
1200 <*trace>
1201   \tr@ce{Enlarging page height * }%
1202 </trace>
1203   \@enlargepage{\hbox{\kern\p@}}}%
1204   {%
1205 <*trace>
1206   \tr@ce{Enlarging page height exactly---}%
1207 </trace>
1208   \@enlargepage\@empty}%
1209 }
1210 </2ekernel | def1>
1211 <*autoload>
1212 \def\enlargethispage{\@autoload{out1}\enlargethispage}
1213 </autoload>
```

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
1214 <*2ekernel | def1>
1215 \gdef\@enlargepage#1#2{%
1216 <*trace>
1217   \tr@ce{\@spaces\@spaces by #2}%
1218 </trace>
1219   \@tempskipa#2\relax
1220   \ifdim \@tempskipa>.5\maxdimen
1221     \@latexerr{Suggested\space extra\space height\space
1222               (\the\@tempskipa)\space dangerously\space
1223               large}\@eha
1224   \else
1225     \ifdim \vsize<.5\maxdimen
1226 <*trace>
1227     \tr@ce {Kludgeins added--pagegoal before: \the\pagegoal}%
1228 </trace>
1229     \@bsphack
1230     \insert\@kludgeins{#1\vskip-\@tempskipa}%
1231     \@esphack
```

This next bit is for tracing only:

```
1232 <*trace>
1233   \ifvmode \par
1234   \tr@ce {Kludgeins added--pagegoal after: \the \pagegoal}%
```

```

1235         \fi
1236 \end{trace}
1237     \else
1238         \@latexerr{Page\space height\space already\space
1239                 too\space large}\@eha
1240     \fi
1241 \fi
1242 }
1243 \end{kernel} \defl)

```

66.1.2 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the document level, the following command for inclusion in L^AT_EX 2_ε.

```
\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.

```

\tr@ce Set-up tracing for floats independent of other tracing as it produces mega-output.
\notrace Default is no tracing.
\tracefloats 1244 \begin{trace}
\@traceval 1245 \def \@tracemessage #1{\typeout{LaTeX2e: #1}}
\tracefloatvals 1246 \def \tracefloats{\let \tr@ce \@tracemessage}
\@tracemessage 1247 \def \notrace {\let \tr@ce \@gobble}
1248 \notrace
1249 \def \@traceval #1{\tr@ce{\string #1 = \the #1}}
1250 \def \tracefloatvals{%
1251     \@dblfloatplacement
1252     \floatplacement
1253     \@traceval\@colnum

```

```

1254 \traceval\@colroom
1255 \traceval\@topnum
1256 \traceval\@toproom
1257 \traceval\@botnum
1258 \traceval\@botroom
1259 \traceval\@fpmin
1260 \tr@ce{\string\textfraction = \textfraction}%
1261 \traceval\@dbltopnum
1262 \traceval\@dbltoproom
1263 }
1264 \trace
1265 \after
1266 \providecommand\tr@ce[1]{}
1267 \after

```

\suppressfloats Float suppression commands: these set the relevant counter globally to zero. Thus they are overridden for a particular oat by an **!** specifier.

\@flstop

```

1268 \*2kernel | autoload)
1269 \def \suppressfloats {%
1270   \ifnextchar [%
1271     \@flstop
1272     {\global \@colnum \z@}%
1273 }

```

Maybe this should be a loop over **#1**?

```

1274 \def \@flstop [#1]{%
1275   \if t#1%
1276     \global \@topnum \z@
1277   \fi
1278   \if b#1%
1279     \global \@botnum \z@
1280   \fi
1281 }

```

Manipulation of oat 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 oats 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.

```

1282 \newcount \@fpstype
1283 \newdimen \@reqcolroom
1284 \newdimen \@textfloatsheight
1285 \*2kernel | autoload)

```

\@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@\@capytype \endcsname}%
\edef \reserved@a {\the\@temptokena}%
\@onelevel@sanitize \reserved@a
\edef \@fps {\@fps\reserved@a}%
}

1286 \<*2ekernel | autoload | trace >
1287 \def \@fpsadddefault {%
1288 \<*trace>
1289   \tr@ce{fps changed from: \@fps}%
1290 \</trace>
1291   \edef \@fps {\@fps\csname fps@\@capytype \endcsname}%
1292   \@latex@warning {%
1293     No positions in optional float specifier.\MessageBreak
1294     Default added (so using '\@fps')}%
1295 }

```

`\@setfloattypecounts` Sets counters `\@fpstype` and `\@currtype`.
 BANG == bit4 of `\count\@currbox` = 0.

```

1296 \def \@setfloattypecounts {%
1297   \@currtype \count\@currbox
1298   \@fpstype \count\@currbox
1299   \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
1300   \advance \@fpstype -\@currtype
1301 \<*trace>
1302   \tr@ce{(mod 32) fpstype: \the \@fpstype}%
1303   \tr@ce{(mult of 32) currtype: \the \@currtype}%
1304 % Tracing only: but some should be changed into real errors/warnings?
1305   \ifnum \@fpstype<\sist@n
1306     \ifnum \@fpstype=\z@
1307       \tr@ce{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
1308     \fi
1309     \ifnum \@fpstype=\@ne
1310       \tr@ce{WARNING: only h, fpstype = \the \@fpstype = 1?}%
1311     \fi
1312     \tr@ce{BANG float}%
1313   \else
1314     \ifnum \@fpstype=\sist@n
1315       \tr@ce{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
1316     \fi
1317     \ifnum \@fpstype=17
1318       \tr@ce{WARNING: only h, fpstype = \the \@fpstype = 17?}%
1319     \fi
1320     \tr@ce{ORD float}%
1321   \fi
1322 \</trace>
1323 }
1324 \</2ekernel | autoload | trace >

```

Macros for getting, testing and setting bits of the fps.

`\@getfpsbit` Sets `\@tempcnta` to required bit of `\count\@currbox`.

```

1325 \<*2ekernel | autoload>

```

```

1326 \def \@getfpsbit {%
1327   \@boxfpsbit \@currbox
1328 }

```

\@boxfpsbit Used above.

```

1329 \def \@boxfpsbit #1#2{%
1330   \@tempcnta \count#1%
1331   \divide \@tempcnta #2\relax
1332 }

```

\@testfp New definition of the oat page test.

```

1333 \def \@testfp #1{%
1334   \@boxfpsbit #18\relax % Really ‘#1 8’ for human readers!
1335   \ifodd \@tempcnta
1336   \else
1337     \@testtrue
1338   \fi
1339 }

```

\@setfpsbit Sets required bit of \@tempcnta (to 1).

```

1340 \def \@setfpsbit #1{%
1341   \@tempcntb \@tempcnta
1342   \divide \@tempcntb #1\relax
1343   \ifodd \@tempcntb
1344   \else
1345     \advance \@tempcnta #1\relax
1346   \fi
1347 }
1348 \<2kernel | autoload>

```

\@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.

```

1349 \<*2kernel | autoload | trace>
1350 \def \@resethfps {%
1351   \let\reserved@a\@empty
1352   \ifnum \@fpstype=\@one
1353     \def \reserved@a {!}%
1354     \@fpstype 17
1355   \fi
1356   \ifnum \@fpstype=17
1357     \global \advance \count\@currbox \tw@
1358     \@latex@warning@no@line {%
1359       ‘\reserved@a h’ float specifier changed to ‘\reserved@a ht’}%
1360 \<*trace>
1361   \tr@ce{%
1362     ‘t’ added to ‘\reserved@a h’- new Count: \the \count\@currbox}%
1363 \</trace>
1364   \fi
1365 }

```

Special stuff for BANG oats.

`\@flsetnum` Ignores any zero oat 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 oat’) 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 oat which was on the deferlist. Almost identical considerations pertain to `\@addtodblcol`. There may be more efficient ways to handle this, but the group seems to be the simplest.

```
1366 \def \@flsetnum #1{%
1367   \tr@ce{*trace}
1368   \tr@ce{fpstype: \the \@fpstype (flsetnum \string#1)}%
1369   \tr@ce{}
1370   \ifnum \@fpstype<\sist@n
1371     \ifnum #1=\z@
1372     \tr@ce{*trace}
1373     \tr@ce{BANG float resetting \string#1 to 1}%
1374     \tr@ce{}
1375     #1\@ne
1376     \fi
1377   \fi
1378   \tr@ce{*trace}
1379   \tr@ce{#1 (before) = \the #1}%
1380   \tr@ce{}
1381 }
```

`\@flsettextmin` This ignores `\textfraction` space restriction in case BANG.

```
1382 \def \@flsettextmin {%
1383   \tr@ce{*trace}
1384   \tr@ce{fpstype: \the \@fpstype (flsettextmin)}%
1385   \tr@ce{}
1386   \ifnum \@fpstype<\sist@n
1387   \tr@ce{*trace}
1388   \tr@ce{BANG ignoring textmin}%
1389   \tr@ce{}
1390   \@textmin \z@
1391   \else
1392   \@textmin \textfraction\@colht
1393   \tr@ce{*trace}
1394   \tr@ce{ORD textmin = \the \@textmin}%
1395   \tr@ce{}
1396   \fi
1397 }
```

`\@flcheckspace` This ignores space restriction in case BANG; this is still slightly conervative since it does not allow for the fact that, if there is no text in the column then `\textfloatsep` is not needed. Sets `@tempwa` true if there is room for `\@currbox`.

```
1398 \def \@flcheckspace #1#2{%
1399   \advance \@reqcolroom
1400   \ifx #2\@empty \textfloatsep \else \floatsep \fi
```

```

1401 <*trace>
1402   \tr@ce{colroom = \the \@colroom (flcheckspace \string#1 \string#2)}%
1403   \tr@ce{reqcolroom = \the \@reqcolroom
1404           (flcheckspace \string#1 \string#2)}%
1405 </trace>
1406   \ifdim \@colroom>\@reqcolroom
1407     \ifdim #1>\ht\@currbox
1408       \@tempwattrue
1409 <*trace>
1410       \tr@ce{Space OK: #1 = \the #1 > \the \ht \@currbox
1411               (flcheckspace \string#1 \string#2)}%
1412 </trace>
1413     \else
1414 <*trace>
1415       \tr@ce{fpstype: \the \@fpstype
1416               (flcheckspace \string#1 \string#2)}%
1417 </trace>
1418       \ifnum \@fpstype<\sist@
1419 <*trace>
1420       \tr@ce{BANG float ignoring #1
1421               (flcheckspace \string#1 \string#2):}%
1422       \tr@ce{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
1423               BANG}%
1424 </trace>
1425       \@tempwattrue
1426 <*trace>
1427     \else
1428       \tr@ce{Fail---no room (flcheckspace \string#1 \string#2)
1429               (fpstype \the \@fpstype=ORD?):}%
1430       \tr@ce{\@spaces #1 = \the #1. Ht float: \the \ht \@currbox
1431               ORD?}%
1432 </trace>
1433     \fi
1434   \fi
1435 <*trace>
1436   \else
1437     \tr@ce{Fail---no room at 2nd test of colroom
1438             (flcheckspace \string#1 \string#2)}%
1439 </trace>
1440   \fi
1441 }
1442 </2ekernel | autoloading | trace >

```

\@flupdates This updates everything when a oat is placed.

```

1443 <*2ekernel | autoloading>
1444 \def \@flupdates #1#2#3{%
1445   \global \advance #1\m@ne
1446   \global \advance \@colnum \m@ne
1447   \@tempdima -\ht\@currbox
1448   \advance \@tempdima
1449   -\ifx #3\@empty \textfloatsep \else \floatsep \fi
1450   \global \advance #2\@tempdima
1451   \global \advance \@colroom \@tempdima
1452   \@cons #3\@currbox

```

1453 }
1454 </2ekernel | autoload>

Interesting facts about oat mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol oats: this seems sensible, but should be documented.
2. `\twocolumn` oatplacement 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 ‘oat’ from `\@topnewpage`.
5. If the specifier was just h and the oat 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 `\dbltextfloatsep`’. But it was never used! Now gone completely, like `\@maxsep`.
7. After an h oat 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 oats are separated by twice `\intertextsep`: 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 oat 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 `\@docclearpage`?
12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on oat pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what oats can go on the page. Not changed.
13. `\clearpage` can still lose non-boxes, at least when oats 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 oat. (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` ag 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 \@mparbottom \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.

```
1455 <*2ekernel | autoload | trace >
1456 \def\@makefcolumn #1{%
1457   \begingroup
1458     \@fpmin \z@
1459     \let \@testfp \@gobble
1460     \@tryfcolumn #1%
1461   \endgroup
1462 <*trace>
1463   \if@fcolmade
1464     \tr@ce{PAGE: in \string\clearpage \if@twocolumn ---twocolumn\fi---}%
1465     \tr@ce{----- float column/page completed from \string#1}%
1466   \fi
1467 </trace>
1468 }
```

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.

```
1469 \def\@outputdblcol{%
1470   \if@firstcolumn
1471     \global \@firstcolumnfalse
1472     \global \setbox\@leftcolumn \box\@outputbox
1473 <*trace>
1474     \tr@ce{PAGE: first column boxed}%
1475 </trace>
1476   \else
1477     \global \@firstcolumntrue
1478     \setbox\@outputbox \vbox {%
1479       \hb@xt@\textwidth {%
1480         \hb@xt@\columnwidth {%
1481           \box\@leftcolumn \hss}%
1482         \hfil
1483         {\normalcolor\vrule \@width\columnseprule}%
1484         \hfil
1485         \hb@xt@\columnwidth {%
```

```

1486                                \box\@outputbox \hss}%
1487                                }%
1488                                }%
1489 < *trace >
1490     \tr@ce{PAGE: second column also boxed}%
1491 < /trace >
1492     \@combinedblfloats
1493     \@outputpage
1494 < *trace >
1495     \tr@ce{PAGE: two column page completed}%
1496 < /trace >
1497     \begingroup
1498     \@dblfloatplacement
1499     \@startdblcolumn

```

This loop could be replaced by an `\expandafter` tail recursion in `\@startdblcolumn`.

```

1500     \@whiles\if@fcolmade \fi
1501     {\@outputpage
1502 < *trace >
1503     \tr@ce{PAGE: double float page completed}%
1504 < /trace >
1505     \@startdblcolumn}%
1506     \endgroup
1507 \fi
1508 }
1509 < /2kernel | autoload | trace >

```

66.1.3 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.

```

1510 < *2kernel | autoload >
1511 \newcount\c@topnumber
1512 \setcounter{topnumber}{2}

```

\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.

```

1513 \newcommand\topfraction{.7}

```

\c@bottomnumber This counter holds the maximum number of floats that can appear at the bottom of a text page or column.

```

1514 \newcount\c@bottomnumber
1515 \setcounter{bottomnumber}{1}

```

`\bottomfraction` This macro holds the maximum proportion (as a decimal number) of a text page or column that can be occupied by oats at the bottom.

```
1516 \newcommand\bottomfraction{.3}
```

`\c@totalnumber` This counter holds the maximum number of oats that can appear on any text page or column.

```
1517 \newcount\c@totalnumber
1518 \setcounter{totalnumber}{3}
```

`\textfraction` This macro holds the minimum proportion (as a decimal number) of a text page or column that must be occupied by text.

```
1519 \newcommand\textfraction{.2}
```

`\floatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by oating objects before a ‘oat page’ is produced.

```
1520 \newcommand\floatpagefraction{.5}
```

`\c@dbltopnumber` This counter holds the maximum number of double-column oats that can appear on the top of a two-column text page.

```
1521 \newcount\c@dbltopnumber
1522 \setcounter{dbltopnumber}{2}
```

`\dbltopfraction` This macro holds the maximum proportion (as a decimal number) of a two-column text page that can be occupied by double-column oats at the top.

```
1523 \newcommand\dbltopfraction{.7}
```

`\dblfloatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column oating objects before a ‘double-column oat page’ is produced.

```
1524 \newcommand\dblfloatpagefraction{.5}
```

Floats on a text page

`\floatsep` When a oating object is placed on a page with text, these parameters control the separation between the oat and the other objects on the page. These parameters are used for both one-column mode and single-column oats in two-column mode.

`\textfloatsep` They are all rubber lengths.

`\intextsep`

`\floatsep` is the space between adjacent oats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and oats at the top or bottom of the page or column.

`\intextsep` is the space between in-text oats and the text.

```
1525 \newskip\floatsep
1526 \newskip\textfloatsep
1527 \newskip\intextsep
1528 \setlength\floatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1529 \setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
1530 \setlength\intextsep {12\p@ \@plus 2\p@ \@minus 2\p@}
```

`\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.

```
1531 \newskip\dblfloatsep
1532 \newskip\dbltextfloatsep
1533 \setlength\dblfloatsep {12\p@ \@plus 2\p@ \@minus 2\p@}
1534 \setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
```

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.

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.

```
1535 \newskip\@fptop
1536 \newskip\@fpsep
1537 \newskip\@fpbot
1538 \setlength\@fptop{0\p@ \@plus 1fil}
1539 \setlength\@fpsep{8\p@ \@plus 2fil}
1540 \setlength\@fpbot{0\p@ \@plus 1fil}
```

`\@dblftop` Double-column ‘float pages’ in two-column mode use similar parameters.

```
\@dblfpsep 1541 \newskip\@dblftop
\@dblfpbot 1542 \newskip\@dblfpsep
1543 \newskip\@dblfpbot
1544 \setlength\@dblftop{0\p@ \@plus 1fil}
1545 \setlength\@dblfpsep{8\p@ \@plus 2fil}
1546 \setlength\@dblfpbot{0\p@ \@plus 1fil}
```

`\topfigrule` The macros can be used to put in rules between floats and text; whatever they insert should be vertical mode material which takes up zero space.

```
\dblfigrule 1547 \let\topfigrule=\relax
1548 \let\botfigrule=\relax
1549 \let\dblfigrule=\relax
1550 </2kernel | autoload>
```

File L

ltclass.dtx

67 Introduction

This file implements the following declarations, which replace `\documentstyle` in \LaTeX 2_ε 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 \LaTeX 2_ε and \LaTeX 2.09 is that \LaTeX 2_ε 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.

68 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 \LaTeX 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 \LaTeX 2.09 compatibility mode. As far as most packages are concerned, this only affects the warnings and errors \LaTeX 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 $\langle package-option-list \rangle$, each package processes the $\langle main-option-list \rangle$. 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`

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 is allowed only before `\documentclass` to ensure that all packages or options necessary for this particular run are present when needed. The begin and end tags should each be on a line by itself. There is also a star-form; this does not write extra comments into the file.

68.1 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.

69 Class and Package interface

69.1 Class name and version

`\ProvidesClass`

A class can identify itself with the `\ProvidesClass{\langle name \rangle}[\langle version \rangle]` command. The $\langle version \rangle$ should begin with a date in the format YYYY/MM/DD.

69.2 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.

69.3 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}
\RequirePackage[baz]{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` `\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.

<code>\@ifpackageloaded</code> <code>\@ifclassloaded</code> <code>\@ifpackagelater</code> <code>\@ifclasslater</code> <code>\@ifpackagewith</code> <code>\@ifclasswith</code>	<p>To find out if a package has already been loaded, use <code>\@ifpackageloaded{<package>}{<true>}{<false>}</code>.</p> <p>To find out if a package has already been loaded with a version more recent than <code><version></code>, use <code>\@ifpackagelater{<package>}{<version>}{<true>}{<false>}</code>.</p> <p>To find out if a package has already been loaded with at least the options <code><options></code>, use <code>\@ifpackagewith{<package>}{<options>}{<true>}{<false>}</code>.</p> <p>There exists one package that can't be tested with the above commands: the <code>fontenc</code> package pretends that it was never loaded to allow for repeated reloading with different options (see <code>ltoutenc.dtx</code> for details).</p>
--	--

69.4 Declaring new options

Options for classes and packages are built using the same macros.

<code>\DeclareOption</code> <code>\DeclareOption*</code>	<p>To define a builtin option, use <code>\DeclareOption{<name>}{<code>}</code>.</p> <p>To define the default action to perform for local options which have not been declared, use <code>\DeclareOption*{<code>}</code>.</p>
---	--

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*{\@unknownoptionerror}`

Complain about unknown local options. (The initial setting for package files.)

`\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{<pkg-name>}`

Handle the 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.

69.5 Safe Input Macros

<code>\InputIfFileExists</code> <code>\IfFileExists</code> <code>\@missingfileerror</code>	<p><code>\InputIfFileExists{<file>}{<then>}{<else>}</code> Inputs <code><file></code> if it exists. Immediately before the input, <code><then></code> is executed. Otherwise <code><else></code> is executed.</p> <p>As above, but does not input the file.</p> <p>One thing you might like to put in the <code><else></code> clause is</p> <p>This starts an interactive request for a filename, supplying default extensions.</p>
--	--

Just hitting return causes the whole input to be skipped and entering `x` quits the current run,

`\input` This has been redefined from the L^AT_EX2.09 definition, in terms of the new commands `\InputIfFileExists` and `\@missingfileerror`.

`\listfiles` 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 `\ProvidesPackage` are listed alongside the file name. So files in standard (and other non-standard) distributions can put informative strings in this argument.

70 Implementation

```

1 (*2ekernel)

@if@compatibility The ag for compatibilty mode.
2 \newif@if@compatibility

\@documentclasshook The hook called after the first \documentclass command. By default this checks
to see if \@normalsize is undefined, and if so, sets it to \normalsize.
3 \def\@documentclasshook{%
4   \ifx\@normalsize\undefined
5     \let\@normalsize\normalsize
6   \fi
7 }

\@declaredoptions This list is automatically built by \DeclareOption. It is the list of options (separated by commas) declared in the class or package file and it defines the order in which the the corresponding \ds@<option> commands are executed. All local <option>s which are not declared will be processed in the order defined by the optional argument of \documentclass or \usepackage.
8 \let\@declaredoptions\empty

\@classoptionslist List of options of the main class.
9 \let\@classoptionslist\relax
10 \@onlypreamble\@classoptionslist

\@unusedoptionlist List of options of the main class that haven't been declared or loaded as class option files.
11 \let\@unusedoptionlist\empty
12 \@onlypreamble\@unusedoptionlist

\CurrentOption Name of current package or option.
13 \let\CurrentOption\empty

\@currname Name of current package or option.
14 \let\@currname\empty

\@currentx The current file extension.
15 \global\let\@currentx=\empty

```

```

\@clsextension The two possible values of \@currentt.
\@pkgextension 16 \def\@clsextension{cls}
                17 \def\@pkgextension{sty}
                18 \@onlypreamble\@clsextension
                19 \@onlypreamble\@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.
                20 \def\@pushfilename{%
                21   \xdef\@currnamestack{%
                22     {\@currname}%
                23     {\@currentt}%
                24     {\the\catcode'\@}%
                25     \@currnamestack}}
                26 \@onlypreamble\@pushfilename

                27 \def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
                28 \@onlypreamble\@popfilename

                29 \def\@p@pfilename#1#2#3#4\@nil{%
                30   \gdef\@currname{#1}%
                31   \gdef\@currentt{#2}%
                32   \catcode'\@#3\relax
                33   \gdef\@currnamestack{#4}}
                34 \@onlypreamble\@p@pfilename

                35 \gdef\@currnamestack{}
                36 \@onlypreamble\@currnamestack

\@optionlist Returns the option list of the file.
                37 \def\@optionlist#1{%
                38   \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}
                39 \@onlypreamble\@optionlist

\@ifpackageloaded \@ifpackageloaded{<name>} Checks to see whether a file has been loaded.
\@ifclassloaded  40 \def\@ifpackageloaded{\@ifl@aded\@pkgextension}
                41 \def\@ifclassloaded{\@ifl@aded\@clsextension}
                42 \@onlypreamble\@ifpackageloaded
                43 \@onlypreamble\@ifclassloaded

                44 \def\@ifl@aded#1#2{%
                45   \expandafter\ifx\csname ver@#2.#1\endcsname\relax
                46     \expandafter\@secondoftwo
                47   \else
                48     \expandafter\@firstoftwo
                49   \fi}
                50 \@onlypreamble\@ifl@aded

\@ifpackagelater \@ifpackagelater{<name>}{YYYY/MM/DD} Checks that the package loaded is
\@ifclasslater   more recent than the given date.
                51 \def\@ifpackagelater{\@ifl@ter\@pkgextension}
                52 \def\@ifclasslater{\@ifl@ter\@clsextension}

```

```

53 \onlypreamble\ifpackagelater
54 \onlypreamble\ifclasslater

55 \def\@ifl@ter#1#2{%
56   \expandafter\@ifl@t@r
57   \csname ver@#2.#1\endcsname}
58 \onlypreamble\@ifl@ter

    This internal macro is also used in \NeedsTeXFormat.

59 \def\@ifl@t@r#1#2{%
60   \ifnum\expandafter\@parse@version#1//00\@nil<%
61     \expandafter\@parse@version#2//00\@nil
62     \expandafter\@secondoftwo
63   \else
64     \expandafter\@firstoftwo
65   \fi}
66 \onlypreamble\@ifl@t@r

67 \def\@parse@version#1/#2/#3#4#5\@nil{#1#2#3#4 }
68 \onlypreamble\@parse@version

\@ifpackagewith \@ifpackagewith{<name>}{<option-list>} Checks that <option-list> is a subset of
\@ifclasswith the options with which <name> was loaded.

69 \def\@ifpackagewith{\@ifoptions\@pkgextension}
70 \def\@ifclasswith{\@ifoptions\@clsextension}
71 \onlypreamble\@ifpackagewith
72 \onlypreamble\@ifclasswith

73 \def\@ifoptions#1#2{%
74   \@expandtwoargs\@ifptions{\@optionlist{#2.#1}}
75 \onlypreamble\@ifptions

    Probably shouldnt use \CurrentOption here...(changed to \reserved@b.)

76 \def\@ifptions#1#2{%
77   \let\reserved@a\@firstoftwo
78   \@for\reserved@b:=#2\do{%
79     \expandafter\in@\expandafter{\expandafter,\reserved@b,}{, #1,}%
80     \ifin@\else\let\reserved@a\@secondoftwo\fi}%
81   \reserved@a}
82 \onlypreamble\@ifptions

\ProvidesPackage Checks that the current filename is correct, and defines \ver@filename.

83 \def\ProvidesPackage#1{%
84   \xdef\@gtempa{#1}%
85   \ifx\@gtempa\@currname\else
86     \@latex@warning@no@line{You have requested
87       \@cls@pkg\space'\@currname',\MessageBreak
88       but the \@cls@pkg\space provides '#1'}%
89   \fi
90   \@ifnextchar[\@pr@videpackage{\@pr@videpackage[]}]%
91 \onlypreamble\ProvidesPackage

92 \def\@pr@videpackage[#1]{%
93   \expandafter\xdef\csname ver@\@currname.\@currentx\endcsname{#1}%
94   \ifx\@currentx\@clsextension
95     \typeout{Document Class: \@gtempa\space#1}%

```

```

96 \else
97 \wlog{Package: \@gtempa\space#1}%
98 \fi}
99 \@onlypreamble\@pr@videpackage

\ProvidesClass Like \ProvidesPackage, but for classes.
100 \let\ProvidesClass\ProvidesPackage
101 \@onlypreamble\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
102 \def\ProvidesFile#1{%
103 \begingroup
104 \catcode'\ 10 %
105 \ifnum \endlinechar<256 %
106 \ifnum \endlinechar>\m@ne
107 \catcode\endlinechar 10 %
108 \fi
109 \fi
110 \@makeother\/%
111 \@makeother\&%

112 \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{macrocode}

\PassOptionsToPackage If the package has been loaded, we check that it was first loaded with the options.
\PassOptionsToClass Otherwise we add the option list to that of the package.
113 \def\@pass@options#1#2#3{%
114 \expandafter\xdef\csname opt@#3.#1\endcsname{%
115 \@ifundefined{opt@#3.#1}\@empty
116 {\csname opt@#3.#1\endcsname},}%
117 \zap@space#2 \@empty}}
118 \@onlypreamble\@pass@options

119 \def\PassOptionsToPackage{\@pass@options\@pkgextension}
120 \def\PassOptionsToClass{\@pass@options\@clsextension}
121 \@onlypreamble\PassOptionsToPackage
122 \@onlypreamble\PassOptionsToClass

\DeclareOption Adds an option as a \ds@ command, or the default \default@ds command.
\DeclareOption* 123 \def\DeclareOption{%
124 \let\@fileswith@ptions\@badrequireerror
125 \ifstar\@defdefault@ds\@declareoption}
126 \long\def\@declareoption#1#2{%
127 \xdef\@declaredoptions{\@declaredoptions,#1}%

```

```

128 \toks@{#2}%
129 \expandafter\edef\csname ds@#1\endcsname{\the\toks@}
130 \long\def\@defdefault@ds#1{%
131 \toks@{#1}%
132 \edef\default@ds{\the\toks@}
133 \@onlypreamble\DeclareOption
134 \@onlypreamble\@declareoption
135 \@onlypreamble\@defdefault@ds

```

`\OptionNotUsed` If we are in a class file, add `\CurrentOption` to the list of unused options. Otherwise, in a package file do nothing.

```

136 \def\OptionNotUsed{%
137 \ifx\@current\@clsextension
138 \xdef\@unusedoptionlist{%
139 \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
140 \CurrentOption}%
141 \fi}
142 \@onlypreamble\OptionNotUsed

```

`\default@ds` The default default option code. Set by `\@onefilewithoptions` to either `\OptionNotUsed` for classes, or `\@unknownoptionerror` for packages. This may be reset in either case with `\DeclareOption*`.

```

143 % \let\default@ds\OptionNotUsed

```

`\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.

```

144 \def\ProcessOptions{%
145 \let\ds@\@empty
146 \edef\@curroptions{\@optionlist{\@currname.\@current}}%
147 \@ifstar\@xprocessoptions\@processoptions}
148 \@onlypreamble\ProcessOptions

149 \def\@processoptions{%
150 \@for\CurrentOption:=\@declaredoptions\do{%
151 \ifx\CurrentOption\@empty\else
152 \expandtwoargs\in@{,\CurrentOption,}%
153 ,\ifx\@current\@clsextension\else\@classoptionslist,\fi
154 \@curroptions,}%
155 \ifin@
156 \use@option
157 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
158 \fi
159 \fi}%
160 \@processoptions}
161 \@onlypreamble\@processoptions

162 \def\@xprocessoptions{%

```

```

163 \ifx\@current\@clsextension\else
164 \@for\CurrentOption:=\@classoptionslist\do{%
165 \ifx\CurrentOption\@empty\else
166 \expandtwoargs\in@{,\CurrentOption,}{,\@declaredoptions,}%
167 \ifin@
168 \@use@option
169 \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
170 \fi
171 \fi}%
172 \fi
173 \@process@pti@ns}
174 \@onlypreamble\@xprocess@ptions

```

The common part of `\ProcessOptions` and `\ProcessOptions*`.

```

175 \def\@process@pti@ns{%
176 \@for\CurrentOption:=\@curroptions\do{%
177 \@ifundefined{ds@\CurrentOption}%
178 {\@use@option
179 \default@ds}%

```

There should not be any non-empty definition of `\CurrentOption` at this point, as all the declared options were executed earlier. This is for compatibility with 2.09 styles which use `\def\ds@...` directly, and so have options which do not appear in `\@declaredoptions`.

```

180 \@use@option}%

```

Clear all the definitions for option code. First set all the declared options to `\relax`, then reset the ‘default’ and ‘empty’ options. and the list of declared options.

```

181 \@for\CurrentOption:=\@declaredoptions\do{%
182 \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
183 \let\CurrentOption\@empty
184 \let\@fileswith@pti@ns\@fileswith@pti@ns
185 \AtEndOfPackage{\let\unprocessedoptions\relax}}
186 \@onlypreamble\@process@pti@ns

```

`\@options` `\@options` is a synonym for `\ProcessOptions*` for upward compatibility with L^AT_EX 2.09 style files.

```

187 \def\@options{\ProcessOptions*}
188 \@onlypreamble\@options

```

`\@use@option` Execute the code for the current option.

```

189 \def\@use@option{%
190 \expandtwoargs\@removeelement\CurrentOption
191 \@unusedoptionlist\@unusedoptionlist
192 \csname ds@\CurrentOption\endcsname}
193 \@onlypreamble\@use@option

```

`\ExecuteOptions` `\ExecuteOptions{<option-list>}` executes the code declared for each option.

```

194 \def\ExecuteOptions#1{%
195 \def\reserved@a##1\@nil{%
196 \@for\CurrentOption:=#1\do{\csname ds@\CurrentOption\endcsname}%
197 \edef\CurrentOption{##1}}%

```

```

198 \expandafter\reserved@a\CurrentOption\@nil}
199 \@onlypreamble\ExecuteOptions

```

The top-level commands, which just set some parameters then call the internal command, \@fileswithoptions.

`\documentclass` The main new-style class declaration.

```

200 \def\documentclass{%
201   \let\documentclass\@twoclasseserror
202   \if@compatibility\else\let\usepackage\RequirePackage\fi
203   \@fileswithoptions\@clsextension}
204 \@onlypreamble\documentclass

```

`\documentstyle` 2.09 style class ‘style’ declaration.

```

205 \def\documentstyle{%
206   \makeatletter\input{latex209.def}\makeatother
207   \documentclass}
208 \@onlypreamble\documentstyle

```

`\RequirePackage` Load package if not already loaded.

```

209 \def\RequirePackage{%
210   \@fileswithoptions\@pkgextension}
211 \@onlypreamble\RequirePackage

```

`\LoadClass` Load class.

```

212 \def\LoadClass{%
213   \ifx\@currentx\@pkgextension
214     \@latex@error
215     {\noexpand\LoadClass in package file}%
216     {You may only use \noexpand\LoadClass in a class file.}%
217   \fi
218   \@fileswithoptions\@clsextension}
219 \@onlypreamble\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.

```

220 \def\@loadwithoptions#1#2#3{%
221   \expandafter\let\csname opt@#3.#1\expandafter\endcsname
222     \csname opt@\@currname.\@currentx\endcsname
223   #2{#3}}
224 \@onlypreamble\@loadwithoptions

```

`\LoadClassWithOptions` Load class ‘#1’ with the current option list.

```

225 \def\LoadClassWithOptions{%
226   \@loadwithoptions\@clsextension\LoadClass}
227 \@onlypreamble\LoadClassWithOptions

```

`\RequirePackageWithOptions` Load package ‘#1’ with the current option list.

```

228 \def\RequirePackageWithOptions{%
229   \AtEndOfPackage{\let\@unprocessedoptions\relax}%
230   \@loadwithoptions\@pkgextension\RequirePackage}
231 \@onlypreamble\RequirePackageWithOptions

```


`\usepackage` To begin with, `\usepackage` produces an error. This is reset by `\documentclass`.

```

232 \def\usepackage#1{%
233   \@latex@error
234     {\noexpand \usepackage before \string\documentclass}%
235     {\noexpand \usepackage may only appear in the document
236      preamble, i.e.,\MessageBreak
237      between \noexpand\documentclass and
238      \string\begin{document}.}%
239   \@gobble}
240 \@onlypreamble\usepackage

```

`\NeedsTeXFormat` Check that the document is running on the correct system.

```

241 \def\NeedsTeXFormat#1{%
242   \def\reserved@a{#1}%
243   \ifx\reserved@a\fmtname
244     \expandafter\@needsformat
245   \else
246     \@latex@error{This file needs format ‘\reserved@a’%
247       \MessageBreak but this is ‘\fmtname’}{%
248       The current input file will not be processed
249       further,\MessageBreak
250       because it was written for some other flavor of
251       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.

```

252   \endinput \fi}
253 \@onlypreamble\NeedsTeXFormat

254 \def\@needsformat{%
255   \@ifnextchar[%]
256     \@needsformat
257   {}}
258 \@onlypreamble\@needsformat

259 \def\@needsformat[#1]{%
260   \@ifl@t@r\fmtversion{#1}{}%
261   {\@latex@warning@no@line
262     {You have requested release ‘#1’ of LaTeX,\MessageBreak
263     but only release ‘\fmtversion’ is available}}%
264 \@onlypreamble\@needsformat

```

`\zap@space` `\zap@space foo<space>\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```

265 \def\zap@space#1 #2{%
266   #1%
267   \ifx#2\@empty\else\expandafter\zap@space\fi
268   #2}

```

`\@fileswithoptions` The common part of `\documentclass` and `\usepackage`.

```

269 \def\@fileswithoptions#1{%
270   \@ifnextchar[%]
271     {\@fileswithoptions#1}%

```

```

272     {\@fileswith@ptions#1[]}}
273 \@onlypreamble\@fileswith@ptions

274 \def\@fileswith@ptions#1[#2]#3{%
275     \ifnextchar[%]
276     {\@fileswith@ptions#1[#2]#3}%
277     {\@fileswith@ptions#1[#2]#3[]}}
278 \@onlypreamble\@fileswith@ptions

```

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.

```

279 \def\@fileswith@ptions#1[#2]#3[#4]{%
280     \ifx#1\@clsextension
281         \ifx\@classoptionslist\relax
282             \xdef\@classoptionslist{\zap@space#2 \@empty}%
283             \def\reserved@a{%
284                 \@onefilewithoptions#3[#2] [#4]#1%
285                 \@documentclasshook}%
286         \else
287             \def\reserved@a{%
288                 \@onefilewithoptions#3[#2] [#4]#1}%
289         \fi
290     \else

```

build up a list of calls to \@onefilewithoptions (one for each package) without thrashing the parameter stack.

```

291     \def\reserved@b##1,{%
292         \ifx\@nil##1\relax\else
293             \ifx\relax##1\relax\else
294                 \noexpand\@onefilewithoptions##1[#2] [#4]\noexpand\@pkgextension
295             \fi
296             \expandafter\reserved@b
297         \fi}%
298     \edef\reserved@a{\zap@space#3 \@empty}%
299     \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
300     \fi
301     \reserved@a}
302 \@onlypreamble\@fileswith@ptions

```

Have the main argument as #1, so we only need one \expandafter above.

```

303 \def\@onefilewithoptions#1[#2] [#3]#4{%
304     \@pushfilename
305     \xdef\@currname{#1}%
306     \global\let\@current#4%
307     \expandafter\let\csname\@currname.\@current-h@k\endcsname\@empty
308     \let\CurrentOption\@empty
309     \@reset@ptions

```

```

310 \makeatletter
Grab everything in a macro, so the parameter stack is popped before any process-
ing begins.
311 \def\reserved@a{%
312 \ifl@aded\@currentx{#1}%
313 {\if@options\@currentx{#1}{#2}{}}%
314 {\@latex@error
315 {Option clash for \cls@pkg\space #1}%
316 {The package #1 has already been loaded
317 with options:\MessageBreak
318 \space\space[\@optionlist{#1.\@currentx}]\MessageBreak
319 There has now been an attempt to load it
320 with options:\MessageBreak
321 \space\space[#2]\MessageBreak
322 Adding the global options:\MessageBreak
323 \space\space
324 \@optionlist{#1.\@currentx},#2\MessageBreak
325 to your \noexpand\documentclass declaration may fix this.%
326 \MessageBreak
327 Try typing \space <return> \space to proceed.}}}%
328 {\@pass@options\@currentx{#2}{#1}%

329 \global\expandafter
330 \let\csname ver@\@currname.\@currentx\endcsname\@empty
331 \InputIfFileExists
332 {\@currname.\@currentx}%
333 {}%
334 {\@missingfileerror\@currname\@currentx}%

\@unprocessedoptions will generate an error for each specified option in a pack-
age unless a \ProcessOptions has appeared in the package file.
335 \let\@unprocessedoptions\@@unprocessedoptions
336 \csname\@currname.\@currentx-h@k\endcsname
337 \expandafter\let\csname\@currname.\@currentx-h@k\endcsname
338 \@undefined
339 \@unprocessedoptions}
340 \ifl@ter\@currentx{#1}{#3}{}%
341 {\@latex@warning@no@line
342 {You have requested,\on@line,
343 version\MessageBreak
344 ‘#3’ of \cls@pkg\space #1,\MessageBreak
345 but only version\MessageBreak
346 ‘\csname ver@#1.\@currentx\endcsname’\MessageBreak
347 is available}}}%

348 \ifx\@currentx\clsextension\let\LoadClass\@twoloadclasserror\fi
349 \popfilename
350 \reset@options}%
351 \reserved@a}
352 \@onlypreamble\@onefilewithoptions

\@fileswith@ptions Save the definition (for error checking).
353 \let\@fileswith@ptions\@fileswith@ptions
354 \@onlypreamble\@fileswith@ptions

```

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

```
355 \def\@reset@ptions{%
356   \global\ifx\@currentx\@clsextension
357     \let\default@ds\OptionNotUsed
358   \else
359     \let\default@ds\@unknownoptionerror
360   \fi
361   \global\let\ds@\@empty
362   \global\let\@declaredoptions\@empty}
363 \@onlypreamble\@reset@ptions
```

70.1 Hooks

Allow code do 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 begining or end of the document.

```
\@enddocumenthook 364 \ifx\@begindocumenthook\@undefined
365   \let\@begindocumenthook\@empty
366 \fi
367 \let\@enddocumenthook\@empty
```

`\g@addto@macro` Globally add to the end of a macro.

```
368 \long\def\g@addto@macro#1#2{%
369   \begingroup
370   \toks@\expandafter{#1#2}%
371   \xdef#1{\the\toks@}%
372   \endgroup}
```

`\AtEndOfPackage` The access functions.

```
\AtEndOfClass 373 \def\AtEndOfPackage{%
\AtBeginDocument 374   \expandafter\g@addto@macro\csname\@currname.\@currentx-h@k\endcsname}
\AtEndDocument 375 \let\AtEndOfClass\AtEndOfPackage
376 \@onlypreamble\AtEndOfPackage
377 \@onlypreamble\AtEndOfClass

378 \def\AtBeginDocument{\g@addto@macro\@begindocumenthook}
379 \def\AtEndDocument{\g@addto@macro\@enddocumenthook}
380 \@onlypreamble\AtBeginDocument
```

`\@cls@pkg` The current file type.

```
381 \def\@cls@pkg{%
382   \ifx\@currentx\@clsextension
383     document class%
384   \else
385     package%
386   \fi}
387 \@onlypreamble\@cls@pkg
```

`\@unknownoptionerror` Bad option.

```
388 \def\@unknownoptionerror{%
```

```

389 \latex@error
390 {Unknown option '\CurrentOption' for \cls@pkg\space'\currname'}%
391 {The option '\CurrentOption' was not declared in
392 \cls@pkg\space'\currname', perhaps you\MessageBreak
393 misspelled its name.
394 Try typing \space <return>
395 \space to proceed.}}
396 \onlypreamble\unknownoptionerror

\@unprocessedoptions Declare an error for each option, unless a \ProcessOptions occurred.
397 \def\@unprocessedoptions{%
398 \ifx\@current\pkgextension
399 \edef\@curroptions{\@optionlist{\currname.\current}}%
400 \for\CurrentOption:=\@curroptions\do{%
401 \ifx\CurrentOption\empty\else\unknownoptionerror\fi}%
402 \fi}
403 \onlypreamble\@unprocessedoptions
404 \onlypreamble\@unprocessedoptions

\@badrequireerror \RequirePackage or \LoadClass occurs in the options section.
405 \def\@badrequireerror#1[#2]#3[#4]{%
406 \latex@error
407 {\noexpand\RequirePackage or \noexpand\LoadClass
408 in Options Section}%
409 {The \cls@pkg\space '\currname' is defective.\MessageBreak
410 It attempts to load '#3' in the options section, i.e.,\MessageBreak
411 between \noexpand\DeclareOption and \string\ProcessOptions.}}
412 \onlypreamble\@badrequireerror

\@twoloadclasserror Two \LoadClass in a class.
413 \def\@twoloadclasserror{%
414 \latex@error
415 {Two \noexpand\LoadClass commands}%
416 {You may only use one \noexpand\LoadClass in a class file}}
417 \onlypreamble\@twoloadclasserror

\@twoclasseserror Two \documentclass or \documentstyle.
418 \def\@twoclasseserror#1{%
419 \latex@error
420 {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
421 {The document may only declare one class.}\@gobble}
422 \onlypreamble\@twoclasseserror

```

70.2 Providing shipment

```

\two@digits Prefix a number less than 10 with '0'.
423 \def\two@digits#1{\ifnum#1<10 0\fi\number#1}

\filecontents This environment implements inline files. The star-form does not write extra
\endfilecontents comments into the file.
424 \begingroup%
425 \catcode'\*=11 %

```

```

426 \catcode'\^M\active%
427 \catcode'\^L\active\let^L\relax%
428 \catcode'\^I\active%

429 \gdef\filecontents{\@tempswatrue\filecontents}%
430 \gdef\filecontents*{\@tempswafalse\filecontents}%

431 \gdef\filecontents#1{%
432   \openin\@inputcheck#1 %
433   \ifeof\@inputcheck%
434     \@latex@warning@no@line%
435     {Writing file '@currdir#1'}%

436   \chardef\reserved@c15 %
437   \ch@ck7\reserved@c\write%
438   \immediate\openout\reserved@c#1\relax%
439   \else%

440     \closein\@inputcheck%
441     \@latex@warning@no@line%
442     {File '#1' already exists on the system.\MessageBreak%
443     Not generating it from this source}%
444     \let\write\@gobbletwo%
445     \let\closeout\@gobble%
446   \fi%
447   \if@tempswa%

448     \immediate\write\reserved@c{%
449       \@percentchar\@percentchar\space%
450       \expandafter\@gobble\string\LaTeX2e file '#1'^J%
451       \@percentchar\@percentchar\space generated by the %
452       '@currenvir' \expandafter\@gobblefour\string\newenvironment^J%
453       \@percentchar\@percentchar\space from source '\jobname' on %
454       \number\year/\two@digits\month/\two@digits\day.^J%
455       \@percentchar\@percentchar}%
456   \fi%
457   \let\do\@makeother\dospecials%

458   \edef\E{\@backslashchar end\string{\@currenvir\string}}%
459   \edef\reserved@b{%
460     \def\noexpand\reserved@b%
461       ###1\E###2\E###3\relax}%
462   \reserved@b{%
463     \ifx\relax##3\relax%

There was no \end{filecontents}

464     \immediate\write\reserved@c{##1}%
465     \else%

There was a \end{filecontents}, so stop this time.

466     \edef^M{\noexpand\end{\@currenvir}}%
467     \ifx\relax##1\relax%
468     \else%

Text before the \end, write it with a warning.

469     \@latex@warning{Writing text '##1' before %
470     \string\end{\@currenvir}\MessageBreak as last line of #1}%

```

```

471      \immediate\write\reserved@c{##1}%
472      \fi%
473      \ifx\relax##2\relax%
474      \else%
    Text after the \end, ignore it with a warning.
475      \@latex@warning{%
476      Ignoring text ‘##2’ after \string\end{\@currenvir}}%
477      \fi%
478      \fi%
479      ^^M}%

480      \catcode‘\^^L\active%
481      \let\L\@undefined%
482      \def^^L{\@ifundefined L^^J^^J^^J}%
483      \catcode‘\^^I\active%
484      \let\I\@undefined%
485      \def^^I{\@ifundefined I\space\space}%
486      \catcode‘\^^M\active%
487      \edef^^M##1^^M{%
488      \noexpand\reserved@b##1\E\E\relax}}%
489 \endgroup%

490 \begingroup
491 \catcode‘|= \catcode‘\%
492 \catcode‘\%=12
493 \catcode‘*=11
494 \gdef\@percentchar{%}
495 \gdef\endfilecontents{|
496      \immediate\closeout\reserved@c
497      \def\T##1##2##3{|
498      \ifx##1\@undefined\else
499      \@latex@warning@no@line{##2 has been converted to Blank ##3e}|
500      \fi}|
501      \T\L{Form Feed}{Lin}|
502      \T\I{Tab}{Spac}|
503      \immediate\write\@unused{}}
504 \global\let\endfilecontents*\endfilecontents
505 \@onlypreamble\filecontents
506 \@onlypreamble\endfilecontents
507 \@onlypreamble\filecontents*
508 \@onlypreamble\endfilecontents*
509 \endgroup
510 \@onlypreamble\filec@ntents

511 </2ekernel>

```

71 After Preamble

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

```

512 <*afterpreamble>
513 \NeedsTeXFormat{LaTeX2e}
514 \ProvidesPackage{pkgindoc}

```

```
515          [1994/10/20 v1.1 Package Interface in Document (DPC)]
516 \def\reserved@a#1\do\@classoptionslist#2\do\filecontents#3\relax{%
517   \gdef\@preamblecmds{#1#3}}
518 \expandafter\reserved@a\@preamblecmds\relax
519 \</afterpreamble>
```


File M

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 ghyph31
\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 N

ltfinal.dtx

72 Final settings

This section contains the final settings for L^AT_EX. It initialises some debugging and typesetting parameters, sets the default `\catcodes` and `uc/lc` codes, and inputs the hyphenation file.

72.1 Debugging

By default, L^AT_EX shows statistics:

```
1 \*2ekernel
2 \tracingstats1
```

72.2 Typesetting parameters

```
\@lowpenalty These are penalties used internally.
\@medpenalty 3 \newcount\@lowpenalty
\@highpenalty 4 \newcount\@medpenalty
5 \newcount\@highpenalty
```

The default values of the `picture` and `\fbox` parameters:

```
6 \unitlength = 1pt
7 \fboxsep = 3pt
8 \fboxrule = .4pt
```

The saved value of T_EX's `\maxdepth`:

```
9 \@maxdepth = \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}`

```
10 \vsize = 1000pt
11 \@colroom = \vsize
12 \@colht = \vsize
```

Initialise `\textheight` `\textwidth` and page style, to avoid internal errors if they are not set by the class.

```
13 \textheight=.5\maxdimen
14 \textwidth=\textheight
15 \ps@empty
```

72.3 Lccodes for hyphenation

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.

```
16 \def\reserved@a#1#2{%
```

```

17 \@tempcnta#1\relax
18 \@tempcntb#2\relax
19 \reserved@b
20 }
21 \def\reserved@b{%
22 \ifnum\@tempcnta>\@tempcntb\else
23 \reserved@c\@tempcnta
24 \advance\@tempcnta\@ne
25 \expandafter\reserved@b
26 \fi
27 }

```

Depending on the T_EX version, we might not be allowed to do this for non-ASCII characters.

```

28 \def\reserved@c#1{%
29 \count@=#1\advance\count@ by -"20
30 \uccode#1=\count@
31 \lccode#1=#1
32 }
33 \reserved@a{'\a}{'\z}
34 \ifnum\inputlineno=\m@ne\else
35 \reserved@a{"A0}{'BC}
36 \reserved@a{"E0}{'FF}
37 \fi

```

The upper case characters need their \uccode and \lccode values set, and their \sfcode set to 999.

```

38 \def\reserved@c#1{%
39 \count@=#1\advance\count@ by "20
40 \uccode#1=#1
41 \lccode#1=\count@
42 \sfcode#1=999
43 }
44 \reserved@a{'\A}{'\Z}
45 \ifnum\inputlineno=\m@ne\else
46 \reserved@a{"80}{'9C}
47 \reserved@a{"C0}{'DF}
48 \fi

```

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.

```

49 \uccode'\^Y='I % dotless i
50 \lccode'\^Y='^Y % dotless i
51 \uccode'\^Z='J % dotless j, ae in OT1
52 \lccode'\^Z='^Z % dotless j, ae in OT1
53 \ifnum\inputlineno=\m@ne\else
54 \lccode'\^9d='i % dotted I
55 \uccode'\^9d='^9d % dotted I
56 \lccode'\^9e='^9e % d-bar
57 \uccode'\^9e='^d0 % d-bar
58 \fi

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

59 \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).

```
60 \lccode'\- ='\- % default hyphen char
61 \lccode 127=127 % alternate hyphen char
62 \lccode 23 =23 % textcompwordmark in T1
```

72.4 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.

```
63 \InputIfFileExists{hyphen.cfg}
64     {\typeout{=====^^J%
65             Local configuration file hyphen.cfg used^^J%
66             =====}%
67     \def\@addtofilelist##1{\xdef\@filelist{\@filelist,##1}}%
68     }
69     {\input{hyphen.ltx}}
70 \let\@addtofilelist\@gobble
```

72.5 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.

```
71 % \changes{v1.1c}{2000/08/23}{Fix typo in warning}
72 \ifdim \font@submax >\z@
73     \font@warning{Size substitutions with differences\MessageBreak
74                 up to \font@submax\space have occurred.\MessageBreak
75                 \MessageBreak
76                 Please check the transcript file
77                 carefully\MessageBreak
78                 and redo the format generation if necessary!
79                 \@gobbletwo}%
80     \errhelp{Only stopped, to give you time to
81             read the above message.}
82     \errmessage{}
```

We reset the macro. Otherwise every user will get a warning on every job.

```
83 \def\font@submax{0pt}
84 \fi
```

72.6 Input encoding

We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```
85 \def\reserved@a#1#2{%
86     \@tempcnta#1\relax
87     \@tempcntb#2\relax
88     \reserved@b
```

```

89 }
90 \def\reserved@b{%
91   \ifnum\@tempcnta>\@tempcntb\else
92     \reserved@c\@tempcnta
93     \advance\@tempcnta\@ne
94     \expandafter\reserved@b
95   \fi
96 }

```

Set the special catcodes (although some of these are useless, since an error will have occurred if the catcodes have changed). Note that `^^J` has catcode ‘other’ for use in warning messages.

```

97 \catcode'\ =10
98 \catcode'\#=6
99 \catcode'\$=3
100 \catcode'\%=14
101 \catcode'\&=4
102 \catcode'\'=0
103 \catcode'\^=7
104 \catcode'\_ =8
105 \catcode'\{=1
106 \catcode'\}=2
107 \catcode'\~=13
108 \catcode'\@=11
109 \catcode'\^^I=10
110 \catcode'\^^J=12
111 \catcode'\^^L=13
112 \catcode'\^^M=5

```

Set the ‘other’ catcodes.

```

113 \def\reserved@c#1{\catcode#1=12\relax}
114 \reserved@c{'!}
115 \reserved@c{'\"}
116 \reserved@a{'\' }{'\?}
117 \reserved@c{'\[]
118 \reserved@c{'\]}
119 \reserved@c{'\' }
120 \reserved@c{'\|}

```

Set the ‘letter’ catcodes.

```

121 \def\reserved@c#1{\catcode#1=11\relax}
122 \reserved@a{'\A}{'\Z}
123 \reserved@a{'\a}{'\z}

```

All the characters in the range 0–31 and 127–255 are illegal, *except* `tab` (`^^I`), `nl` (`^^J`), `ff` (`^^L`) and `cr` (`^^M`).

Now allow 8-bit characters, although their use in this way is strongly discouraged. See `inputenc.dtx` for a supported mechanism for 8-bit input.

```

124 \def\reserved@c#1{\catcode#1=15\relax}
125 \reserved@a{0}{'\^H}
126 \reserved@c{'\^K}
127 \reserved@a{'\^N}{31}
128 %\ifnum\inputlineno=\m@ne
129   \catcode"7F=15
130 %\else

```

```

131 % \reserved@a{"7F}{FF}
132 %\fi

```

72.7 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 T_EX version, we might not be allowed to do this for non-ASCII characters.

```

133 \def\reserved@c#1{%
134   \count@=#1\advance\count@ by -"20
135   \uccode#1=\count@
136   \lccode#1=#1
137 }
138 \reserved@a{'\a}{'\z}
139 \ifnum\inputlineno=\m@ne\else
140   \reserved@a{"A0}{BC}
141   \reserved@a{"E0}{FF}
142 \fi

```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```

143 \def\reserved@c#1{%
144   \count@=#1\advance\count@ by "20
145   \uccode#1=#1
146   \lccode#1=\count@
147   \sfcode#1=999
148 }
149 \reserved@a{'\A}{'\Z}
150 \ifnum\inputlineno=\m@ne\else
151   \reserved@a{"80}{9C}
152   \reserved@a{"C0}{DF}
153 \fi

```

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.

```

154 \uccode'\^^Y=' \I      % dotless i
155 \lccode'\^^Y=' \^^Y    % dotless i
156 \uccode'\^^Z=' \J      % dotless j, ae in OT1
157 \lccode'\^^Z=' \^^Z    % dotless j, ae in OT1
158 \ifnum\inputlineno=\m@ne\else
159   \lccode'\^^9d=' \i     % dotted I
160   \uccode'\^^9d=' \^^9d % dotted I
161   \lccode'\^^9e=' \^^9e % d-bar
162   \uccode'\^^9e=' \^^d0 % d-bar
163 \fi

```

Finally here is one that helps hyphenation in the OT1 encoding.

```

164 \lccode'\^^[=' \^^[    % oe in OT1

```

`\MakeUppercase` And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.

`\MakeUppercase` *Note* that this implementation is subject to change! At the moment we're not providing any way to extend the list of uc/lc commands, since finding a good interface is difficult. These commands have some nasty features, such as uppercasing

`\@uclclist`

mathematics, environment names, labels, etc. A much better long-term solution is to use all-caps fonts, but these aren't generally available.

```

165 \DeclareRobustCommand{\MakeUppercase}[1]{%
166     \def\i{I}\def\j{J}%
167     \def\reserved@a##1##2{\let##1##2\reserved@a}%
168     \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b@gobble}%
169     \protected@edef\reserved@a{\uppercase{#1}}%
170     \reserved@a
171 }
172 \DeclareRobustCommand{\MakeLowercase}[1]{%
173     \def\reserved@a##1##2{\let##1##2\reserved@a}%
174     \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b@gobble}%
175     \protected@edef\reserved@a{\lowercase{#1}}%
176     \reserved@a
177 }
178 \def\@uclclist{\oe\OE\o\O\ae\AE
179     \dh\DH\dj\DJ\l\L\ng\NG\ss\SS\th\TH}

```

The above code works, but has the nasty side-effect that if you say something like:

```

\markboth{\MakeUppercase\contentsname}
         {\MakeUppercase\contentsname}

```

then the uppcasing is only done to the first letter of the contents name, since the mark expands out to:

```

\mark{\protect\MakeUppercase Table of Contents}
      {\protect\MakeUppercase Table of Contents}

```

In order to get round this, we redefine `\MakeUppercase` and `\MakeLowercase` to grab their argument and brace it. This is a very low-level hack, and is *not* recommended practice! This is an instance of a general problem that makes it unsafe to grab arguments unbraced, and probably needs a more general solution. For the moment though, this hack will do:

```

180 \protected@edef\MakeUppercase#1{\MakeUppercase{#1}}
181 \protected@edef\MakeLowercase#1{\MakeLowercase{#1}}

```

72.8 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

```

182 \IfFileExists{ltpatch.ltx}
183   {\typeout{=====^^J%
184             Applying patch file ltpatch.ltx^^J%
185             =====}}
186   \def\fmtversion@topatch{unknown}
187   \input{ltpatch.ltx}
188   \ifx\fmtversion\fmtversion@topatch
189     \ifx\patch@level\@undefined
190       \typeout{^^J^^J^^J%
191               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
192               !! Patch file 'ltpatch.ltx' not suitable for this^^J%
193               !! version of LaTeX.^^J^^J%
194               !! Please check if initex found an old patch file:^^J%

```



```

195      !! -- if so, rename it or delete it, and redo the^^J%
196      !! initex run.^^J%
197      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
198      \batchmode \@@end
199      \else

```

```

200 \def\fmtversion@topatch{0}%
201 \ifx\fmtversion@topatch\patch@level\else
202 \def\reserved@a\typeout{##1##2\reserved@a{%
203 \typeout{##1 patch level \patch@level}##2}
204 \everyjob\expandafter\expandafter\expandafter{%
205 \expandafter\reserved@a\the\everyjob\reserved@a}
206 \let\reserved@a\relax
207 \the\everyjob
208 \fi
209 \fi
210 \else
211 \typeout{^^J^^J^^J%
212 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
213 !! Patch file 'ltpatch.ltx' (for version <\fmtversion@topatch>)^^J%
214 !! is not suitable for version <\fmtversion> of LaTeX.^^J^^J%
215 !! Please check if initex found an old patch file:^^J%
216 !! --- if so, rename it or delete it, and redo the^^J%
217 !! initex run.^^J%
218 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
219 \batchmode \@@end
220 \fi
221 \let\fmtversion@topatch\relax
222 }{}

```

`\reserved@a` And just to make sure nobody relies on those definitions of `\reserved@b` and `\reserved@b` friends. These macros are reserved for use in the kernel. *Do not use them as general scratch macros.*

\toks

`\errhelp` Empty the error help message, which may have some rubbish:

```
234 \errhelp{}
```

72.10 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`.

```
235 \def\@providesfile#1[#2]{%
236     \wlog{File: #1 #2}%
237     \expandafter\xdef\csname ver@#1\endcsname{#2}%
238     \endgroup}
```

`\@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.

```
239 \let\@filelist\@gobble
240 \def\@addtofilelist#1{\xdef\@filelist{\@filelist,#1}}%
```

72.11 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.

```
241 \makeatother
242 \errorstopmode
243 \dump
244 \</2ekernel>
```

File O

ltpatch

Things we did wrong...

```
1 %%%
2 %%% Patch file for the LaTeX2e kernel dated 2005/12/01
3 %%% (2005/12/01)
4
5 \def\fmtversion@topatch{2005/12/01} % This patch will not work with
6                                     % any other release.
7
8 \def\patch@level{0}
9
10
11
12
13 %%%%%%%%%%%%%%%
14 \iffalse
15
16 \typeout{%
17 ^^J%
18 *****^^J%
19 ltpatch.ltx has fixed certain problems with the 'kernel' of LaTeX.^^J%
20 Certain other files in the LaTeX distribution have also been updated^^J%
21 since the last release (list correct as of 2005/12/01):^^J%
22 base/xxxxxxx.dtx.....(patch 1)^^J%
23 unpacked/yyyyyyy.cls.....(patch 1)^^J%
24 ^^J%
25 See the file patches.txt for more details.^^J%
26 *****}
27
28 \fi
29
30 \endinput
31
32
33
34
35
36
37
```

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

<code>\!</code>	b152, b154, <u>z144</u> , N114	<code>\@@defaultsubs</code>	<u>o414</u>
<code>\"</code>	l181, l280, l318, l356, l364, l436, l468, l495, l503, l509, l513, l519, l523, l529, l535, l542, l543, l549, l553, l593, l636, o367, N115	<code>\@@enc@update</code>	l138, o207, <u>o211</u>
<code>\#</code>	a9, a22, b6, b14, b219, d278, o306, o354, N98	<code>\@@end</code> ..	a16, a169, d8, k186, k187, s121, y39, y49, M18, N198, N219
<code>\\$</code>	a21, b4, b13, d277, l261, l343, l350, l425, l648, l655, N99	<code>\@@endpbox</code>	C175, C206, <u>C354</u>
<code>\%</code> a22, a52, a54, a74, b14, b217, d278, l386, l388, o356, L491, L492, N100		<code>\@@eqnocr</code>	z228, z246, z249, z329
<code>\&</code> a21, b5, b13, b218, d277, L111, N101		<code>\@@fileswith@pti@ns</code>	L184, <u>L353</u>
<code>\'</code> ..	b239, l182, l281, l320, l354, l361, l438, l448, l454, l456, l459, l461, l469, l475, l481, l483, l486, l488, l496, l500, l507, l511, l516, l521, l524, l526, l533, l538, l539, l546, l551, l554, l594, l638, l657, l659, l660, l661, l664, l666, l667, l668, l670, l671, o366, s154, t172, y145, z151, B181, C63, K466, N116	<code>\@@hyph</code>	d9
<code>\(</code>	z168, z193	<code>\@@hyphenation</code>	<u>l160</u>
<code>\)</code>	b239, <u>z168</u> , z194	<code>\@@if@newlist</code>	K476, K523
<code>*</code>	o359, <u>z148</u> , L425, L493	<code>\@@if@definable</code>	<u>d101</u> , l17
<code>\+</code>	C63	<code>\@@input</code>	a15, d7, d293, k162, k163, k172, o307
<code>\,</code> ..	b153, b155, <u>i187</u> , t414, y145, z7, z8, z40, z108, z110, z113, z127, z144	<code>\@@italiccorr</code>	<u>d12</u> , v95, v99
<code>\-</code>	b121, <u>d9</u> , d11, i178, l315, l316, l431, l632, l633, o361, y145, B180, C63, N60	<code>\@@line</code>	B289
<code>\.</code>	b152, b154, k39, l183, l282, l351, l352, l370, l444, l445, l471, l472, l498, l595, l662, l669, o360	<code>\@@math@bgroup</code>	v113, v120
<code>\/</code>	a44, d12, o274, o362, L110	<code>\@@math@egroup</code>	<u>v110</u> , <u>v110</u>
<code>\:</code>	b153, b155, d270, d271, <u>z149</u>	<code>\@@par</code>	<u>d6</u> , h4, y49, y104, y108, y111, 57, A82, A85, B162, B179, C181, F50, F101, K205
<code>\;</code>	b153, b155, t408, z128, <u>z144</u>	<code>\@@patterns</code>	<u>l160</u>
<code>\<</code> ...	l432, l586, o357, y145, C62, C100	<code>\@@protect</code>	d228, d234, d243
<code>\=</code> ..	l184, l283, l369, l596, s154, B181, C62	<code>\@@startpbox</code>	C175, C206, <u>C354</u>
<code>\></code> ..	l429, l587, o358, y145, <u>z144</u> , z149, C62	<code>\@@underline</code>	B259, B262, B263
<code>\?</code>	b152, b154, N116	<code>\@@unprocessedoptions</code> ...	L335, <u>L397</u>
<code>\@</code>	a12, d272, d273, g22, <u>i190</u> , j2, L24, L32, N108	<code>\@@warning</code>	<u>g188</u>
<code>\@@</code>	a254, a255, f15, f19, f20, f21, f22, f24, f27, f28, f30, f31, k208, k224, p494, p498, p499, C208, C209, C210, C220, K10, K11	<code>\@Alph</code>	m38, <u>m54</u>

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx, t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx, y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx, D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx, I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx, N=ltfinal.dtx, O=ltpatch.ltx

- \@acol C150,
C160, C230, C231, C243, C244,
C247, C264, C277, C285, C295
- \@acolampacol C228, C245,
C247, C254, C262, C294, C297
- \@activechar@info K457
- \@addamp C221, C230,
C231, C246, C260, C295, C296
- \@addfield C45,
C55, C77, C84, C116, C127, C129
- \@addmarginpar K283, K1149
- \@addtobot .. K717, K804, K868, K920
- \@addtocurcol K280, K808
- \@addtodblcol K649, K1032
- \@addtofilelist a48, a50, k54, k162,
k205, s111, s129, s133, s140,
s143, s150, s153, N67, N70, N239
- \@addtonextcol K648, K955
- \@addtopreamble C279, C292,
C298, C299, C300, C302, C314
- \@addtoreset m16, m28, m33
- \@addtotoporbot .. K754, K914, K1001
- \@afterheading F75, F108
- \@afterindentfalse F28
- \@afterindenttrue .. F26, F107, F153
- \@alph m37, m50, G254
- \@ampacol C228, C245, C256, C297
- \@arabic m32, m34, m40, G252
- \@argarraycr C185, C186
- \@argdef d47
- \@argrsbox B269
- \@argtabularcr C192, C193
- \@array C163, C164
- \@arrayacol C150, C228
- \@arrayclassiv C151, C299
- \@arrayclassz C150, C245
- \@arraycr C152, C183, C185
- \@arrayparboxrestore B176, B190, C352
- \@arrayrule C277,
C279, C283, C285, C287, C314
- \@arstrut C174, C207, C311
- \@arstrutbox . C167, C200, C311, C353
- \@author F5
- \@autoerr b75, b247, b255,
g19, g88, g119, g150, g213,
g220, g223, g226, g230, g236,
g239, g242, g246, g249, g258,
g264, g267, g270, g273, g278,
g281, g286, k174, l56, o483, r78
- \@autoload d286,
g213, p417, C146, D15, K1212
- \@auxout I7, I8, I19, I29, I37,
I43, k81, k87, k103, k118, x33, F145
- File Key:** a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- \@backslashchar
.... d191, g216, g218, t185, L458
- \@badcrerr g274
- \@badend g233, y65
- \@badlinearg g260, D79,
D88, D89, D93, D137, D142, D153
- \@badmath g237, z168,
z169, z172, z184, z189, z277, z291
- \@badpoptabs g243, C76, C138
- \@badrequireerror L124, L405
- \@badtab g247,
C24, C78, C99, C105, C112, C135
- \@begin@tempboxa
B11, B26, B114, B162, B270, B278
- \@begindocumenthook
.. I33, k48, k51, o301, L364, L378
- \@beginndvi K502, K528
- \@beginndvibox K67, K529
- \@beginparpenalty
.... i30, z280, z294, A23, A160
- \@begintheorem E30, E35
- \@bezier D342, D343
- \@bibitem I3, I8
- \@biblabel I4, I54
- \@bitor K15, K672, K702,
K769, K851, K860, K989, K1094
- \@botlist K46, K308, K310,
K535, K556, K565, K566, K731,
K734, K769, K771, K860, K862
- \@botnum G172, K90,
K728, K729, K734, K738, K745,
K1024, K1029, K1257, K1279
- \@botroom G173, K91, K731, K734, K1258
- \@boxfpsbit K1327, K1329, K1334
- \@break@tfor f31, k157, v80
- \@bsphack I39,
i9, i63, i138, i154, x32, G48,
G189, H6, H18, H23, H35, K1229
- \@caption G12, G14
- \@capttype G5, G9, G12, G36, G84, K1291
- \@car 29, d30, j14, l82
- \@carcube d32, d104
- \@cclv b16, K250, K254, K306,
K307, K335, K359, K363, K364
- \@ccclvi b21, b50, b53, b56
- \@cdr 29, d30, d251, d252
- \@centercr y68, y76, y83, y89
- \@centering z208,
z209, z216, z219, z222, z322, z326
- \@cflb K532
- \@cflt K532
- \@changed@cmd . I3, l68, l178, o106, o215
- \@changed@x I3, l166, l174

- \@changed@x@mouth [I166](#), [I174](#)
- \@charlb [k121](#), [k129](#)
- \@charrb [k123](#), [k129](#)
- \@chclass [C241](#), [C242](#), [C303](#), [C316](#), [C321](#)
- \@check@c [d162](#), [d164](#)
- \@check@eq [d168](#), [d169](#), [d173](#)
- \@checkend [y11](#), [y61](#), [y64](#)
- \@chnum [C249](#),
[C268](#), [C303](#), [C318](#), [C319](#), [C320](#)
- \@circ [D308](#), [D309](#), [D310](#)
- \@circle [D296](#), [D297](#)
- \@circlefnt [D47](#), [D49](#),
[D257](#), [D275](#), [D301](#), [D316](#), [D333](#)
- \@cite [I16](#), [I52](#)
- \@cite@ofmt [I24](#), [I53](#)
- \@citea [I15](#), [I17](#)
- \@citeb [I16](#), [I18](#), [I19](#),
[I20](#), [I23](#), [I24](#), [I41](#), [I42](#), [I43](#), [I44](#), [I45](#)
- \@citex [I13](#), [I14](#)
- \@classi [C241](#), [C275](#)
- \@classii [C241](#), [C289](#)
- \@classiii [C241](#), [C294](#)
- \@classiv [C151](#), [C162](#), [C242](#)
- \@classoptionslist
[L9](#), [L153](#), [L164](#), [L281](#), [L282](#), [L516](#)
- \@classv [C242](#), [C300](#)
- \@classz [C150](#), [C161](#), [C241](#)
- \@cline [C335](#)
- \@clnht ... [D95](#), [D96](#), [D104](#), [D106](#),
[D108](#), [D118](#), [D125](#), [D151](#), [D327](#)
- \@clnwd [D97](#),
[D103](#), [D107](#), [D109](#), [D110](#), [D327](#)
- \@cls@pkg [L87](#), [L88](#),
[L315](#), [L344](#), [L381](#), [L390](#), [L392](#), [L409](#)
- \@clsextension
... [L16](#), [L41](#), [L52](#), [L70](#), [L94](#),
[L120](#), [L137](#), [L153](#), [L163](#), [L203](#),
[L218](#), [L226](#), [L280](#), [L348](#), [L356](#), [L382](#)
- \@clubpenalty [k9](#), [k19](#), [A186](#), [F89](#), [F118](#)
- \@colht ... [k16](#), [G171](#), [G173](#), [G176](#),
[G179](#), [G180](#), [K95](#), [K179](#), [K190](#),
[K199](#), [K200](#), [K311](#), [K323](#), [K350](#),
[K383](#), [K417](#), [K423](#), [K427](#), [K437](#),
[K442](#), [K524](#), [K595](#), [K612](#), [K654](#),
[K674](#), [K707](#), [K1108](#), [K1392](#), [N12](#)
- \@colnum [G174](#), [K92](#), [K737](#),
[K782](#), [K849](#), [K850](#), [K877](#), [K885](#),
[K927](#), [K987](#), [K988](#), [K1024](#),
[K1029](#), [K1253](#), [K1272](#), [K1446](#)
- \@colroom [k17](#),
[K96](#), [K200](#), [K221](#), [K222](#), [K233](#),
[K236](#), [K311](#), [K595](#), [K736](#), [K781](#),
[K845](#), [K848](#), [K876](#), [K983](#), [K986](#),
[K1254](#), [K1402](#), [K1406](#), [K1451](#), [N11](#)
- \@combinedblfloats [K568](#), [K1492](#)
- \@combinefloats [K377](#), [K532](#)
- \@comdblfilelt [K568](#)
- \@comflelt [K538](#), [K554](#), [K568](#)
- \@cons [29](#),
[d29](#), [m33](#), [G118](#), [G135](#), [G234](#),
[K185](#), [K678](#), [K692](#), [K711](#), [K713](#),
[K880](#), [K948](#), [K1017](#), [K1110](#),
[K1133](#), [K1150](#), [K1151](#), [K1452](#)
- \@contfield [C52](#), [C128](#), [C140](#)
- \@ctrerr [g228](#), [m53](#), [m57](#), [m60](#)
- \@curfield [C16](#), [C43](#),
[C49](#), [C53](#), [C54](#), [C56](#), [C121](#), [C122](#)
- \@curline
[C16](#), [C29](#), [C41](#), [C46](#), [C55](#), [C56](#),
[C57](#), [C81](#), [C82](#), [C94](#), [C119](#), [C120](#)
- \@curr@enc [I119](#), [I121](#)
- \@currbox [G56](#),
[G87](#), [G91](#), [G118](#), [G135](#), [G155](#),
[G157](#), [G159](#), [G198](#), [G201](#),
[G206](#), [G210](#), [K161](#), [K162](#), [K173](#),
[K174](#), [K176](#), [K177](#), [K185](#), [K261](#),
[K262](#), [K648](#), [K649](#), [K842](#), [K844](#),
[K874](#), [K878](#), [K880](#), [K895](#), [K936](#),
[K948](#), [K977](#), [K1006](#), [K1017](#),
[K1048](#), [K1052](#), [K1063](#), [K1069](#),
[K1071](#), [K1075](#), [K1080](#), [K1089](#),
[K1103](#), [K1110](#), [K1133](#), [K1151](#),
[K1160](#), [K1297](#), [K1298](#), [K1327](#),
[K1357](#), [K1362](#), [K1407](#), [K1410](#),
[K1422](#), [K1430](#), [K1447](#), [K1452](#)
- \@currdir [1](#), [4](#), [a55](#), [a77](#), [a79](#),
[a85](#), [a87](#), [a93](#), [a95](#), [a100](#), [a102](#),
[a112](#), [a125](#), [a190](#), [a203](#), [a216](#), [L435](#)
- \@current@cmd [l25](#), [o219](#)
- \@currentlabel [x34](#),
[x37](#), [x40](#), [z212](#), [z307](#), [B243](#), [G277](#)
- \@currenvir [g234](#), [y3](#), [y55](#), [y65](#), [A112](#),
[B74](#), [L452](#), [L458](#), [L466](#), [L470](#), [L476](#)
- \@currenvline ... [g234](#), [y56](#), [y66](#), [B75](#)
- \@currentext [L15](#), [L23](#), [L31](#), [L93](#), [L94](#),
[L137](#), [L146](#), [L153](#), [L163](#), [L213](#),
[L222](#), [L306](#), [L307](#), [L312](#), [L313](#),
[L318](#), [L324](#), [L328](#), [L330](#), [L332](#),
[L334](#), [L336](#), [L337](#), [L340](#), [L346](#),
[L348](#), [L356](#), [L374](#), [L382](#), [L398](#), [L399](#)
- \@currlist [G118](#), [G234](#),
[K48](#), [K261](#), [K312](#), [K315](#), [K1150](#)
- \@currname
[k216](#), [k217](#), [L14](#), [L22](#), [L30](#), [L85](#),
[L87](#), [L93](#), [L146](#), [L222](#), [L305](#),
[L307](#), [L330](#), [L332](#), [L334](#), [L336](#),
[L337](#), [L374](#), [L390](#), [L392](#), [L399](#), [L409](#)
- \@currnamestack [L20](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

- \@curroptions L146, L154, L176, L399, L400
 - \@currsiz s58
 - \@currtype K100, K669, K670, K671, K672, K769, K851, K860, K989, K1094, K1297, K1299, K1300, K1303
 - \@curtab C11, C28, C77, C78, C79, C85, C86, C89, C93, C94, C98, C133, C134
 - \@curtabmar C11, C27, C28, C40, C46, C80, C93, C97, C98
 - \@d@r a108, a109
 - \@dashbox D196, D197, D198, D199, D200, D203, D206, D208, D217, D219, D220, D221, D222, D225, D228, D231, D329
 - \@dashcnt D190, D191, D192, D193, D194, D195, D205, D207, D210, D211, D212, D213, D215, D216, D227, D230, D329
 - \@dashdim D189, D190, D191, D192, D194, D197, D199, D200, D201, D205, D207, D209, D210, D211, D212, D215, D219, D221, D222, D223, D229, D232, D329
 - \@date F7
 - \@dbflt G32, G162
 - \@dblarg 28, d275, F37, F125, G12
 - \@dbldeferlist . . . G135, K51, K321, K326, K328, K613, K620, K621, K1094, K1096, K1133, K1135
 - \@dblfloat G31
 - \@dblfloatplacement k25, G177, K325, K1251, K1498
 - \@dblflset G26
 - \@dblpbot G185, K1541
 - \@dblpsep G184, K1541
 - \@dblpftop G183, K1541
 - \@dbltolist K50, K180, K183, K185, K321, K322, K573, K577, K579, K580, K1105, K1110
 - \@dbltopnum G178, K88, K108, K186, K188, K584, K1045, K1046, K1109, K1112, K1120, K1140, K1145, K1261
 - \@dbltoproom G179, G181, K89, K1048, K1051, K1052, K1061, K1062, K1065, K1068, K1071, K1074, K1079, K1083, K1088, K1107, K1262
 - \@dec@text@cmd l3
 - \@declaredoptions L8, L127, L150, L166, L181, L362
 - \@declareoption ... L125, L126, L134
 - \@defaultsubs o404, o414, y26
 - \@defaultunits o185, o245, p133, p135
 - \@defdefault@ds ... L125, L130, L135
 - \@deferlist K49, K308, K317, K318, K596, K604, K605, K851, K853, K948, K950, K989, K991, K1017, K1019
 - \@definecounter m12, m25, z197, A217, A218, A219, A220, E8, E16, G251, G253
 - \@depth d13, p145, t464, t465, t467, t468, B258, B288, C169, C201, D127, D178, D181, D196, D203, D376, K1189
 - \@dir a107, a110, a112, a114, a115
 - \@dischyph d11, B180
 - \@doclearpage K246, K297
 - \@documentclasshook L3, L285
 - \@doendpe y62, A123
 - \@dofilelist k214, k230, y21
 - \@donoparitem A134, A148
 - \@dot D296, D309
 - \@dotsep F160
 - \@dottedtocline F149
 - \@downline D175, D179, D184
 - \@downvector D146, D184
 - \@eha I47, g195, g219, g222, g225, g235, g238, g280, k88, l54, l939, l949, o33, o77, o119, o162, o201, o256, p106, r16, r57, r118, r233, r265, r306, r351, r356, r411, r519, r523, r527, r561, r571, r655, r660, r663, r695, r698, r752, r755, r758, r825, r831, v128, y54, K1223, K1239
 - \@ehb g195, g229, g263, g266, g269, K182, K314
 - \@ehc d97, d124, g195, g274, g277, g285, g288, y130, y141, z253, A210, F4
 - \@ehd . . . g195, g232, g241, g245, g248, g256, r76, C91, C100, G6, L251
 - \@elt d29, k122, m20, m24, K8, K11, K15, K41, K42, K43, K44, K374, K538, K549, K554, K564, K576, K578, K606, K622, K639, K652, K659, K684, K687, K696
 - \@empty f14
 - \@emptycol K146, K193, K196, K225, K229
 - \@end@tempboxa B20, B29, B119, B175, B276, B286
 - \@enddocumenthook . . . y10, L364, L379
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=ltterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx, t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx, y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx, D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx, I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx, N=ltfinal.dtx, O=ltpatch.ltx

`\@endfloatbox` [G115](#), [G132](#), [G142](#)
`\@endparenv` [A120](#), [A123](#)
`\@endparpenalty`
 [i31](#), [z281](#), [z295](#), [A23](#), [A124](#)
`\@endpbox` [C175](#),
 [C206](#), [C236](#), [C301](#), [C352](#), [C355](#)
`\@endpfalse` [y59](#), [A126](#), [A127](#), [A128](#), [B77](#)
`\@endpeltrue` [A128](#)
`\@endpetrue` [A124](#), [A125](#)
`\@endtheorem` [E13](#), [E19](#), [E25](#), [E35](#)
`\@enlargepage` .. [K1203](#), [K1208](#), [K1214](#)
`\@ensuredmath` [z264](#), [z266](#)
`\@enumctr` [A224](#), [A227](#), [A228](#)
`\@enumdepth` .. [A216](#), [A222](#), [A223](#), [A224](#)
`\@eqcnt` [z205](#),
 [z250](#), [z255](#), [z309](#), [z324](#), [z325](#), [z327](#)
`\@eqncr` ... [z217](#), [z235](#), [z256](#), [z257](#), [z311](#)
`\@eqnnum` .. [z199](#), [z200](#), [z254](#), [z268](#), [z302](#)
`\@eqnset` [z205](#), [z323](#)
`\@eqnswfalse` [z234](#)
`\@eqnswtrue` [z207](#), [z213](#), [z255](#), [z308](#)
`\@eqpen` [z205](#), [z238](#), [z240](#), [z247](#)
`\@err@` [g40](#),
 [g44](#), [g47](#), [g55](#), [g67](#), [g71](#), [g74](#), [g82](#)
`\@esphack` . [I50](#), [i11](#), [i69](#), [i143](#), [i160](#),
 [x35](#), [G240](#), [H17](#), [H19](#), [H34](#), [K1231](#)
`\@evenfoot` [J12](#), [J15](#), [K492](#)
`\@evenhead` [J12](#), [J15](#), [K491](#)
`\@expandtwoargs`
 [d189](#), [L74](#), [L152](#), [L166](#), [L190](#)
`\@expast` [C209](#), [C237](#)
`\@failedlist` [K637](#),
 [K660](#), [K672](#), [K678](#), [K692](#), [K702](#)
`\@fcolmadefalse` [K628](#)
`\@fcolmadetrue` [K690](#)
`\@file@und` ... [k144](#), [k154](#), [k162](#), [k172](#)
`\@filelist`
 .. [k53](#), [k204](#), [k205](#), [k216](#), [s111](#),
 [s129](#), [s140](#), [s150](#), [N67](#), [N223](#), [N239](#)
`\@filesfalse` [k64](#)
`\@fileswith@pti@ns` [L124](#),
 [L184](#), [L276](#), [L277](#), [L279](#), [L302](#), [L353](#)
`\@fileswith@ptions`
 [L271](#), [L272](#), [L274](#), [L278](#)
`\@fileswithoptions`
 [L203](#), [L210](#), [L218](#), [L269](#)
`\@filestrue` [k7](#)
`\@finalstrut` . [B247](#), [B287](#), [C353](#), [G282](#)
`\@firstampfalse` ... [C224](#), [C247](#), [C264](#)
`\@firstamptrue` [C232](#)
`\@firstcolumnfalse` [K1471](#)
`\@firstcolumntrue`
 [k22](#), [K79](#), [K155](#), [K1477](#)
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

`\@firstofone` [I18](#),
 [I42](#), [d184](#), [k47](#), [I73](#), [I118](#), [p300](#),
 [r40](#), [r99](#), [r606](#), [y9](#), [z262](#), [C340](#), [G10](#)
`\@firstoftwo` [a34](#), [d184](#),
 [d247](#), [d274](#), [k155](#), [I102](#), [I911](#),
 [I927](#), [r610](#), [x19](#), [J16](#), [L48](#), [L64](#), [L77](#)
`\@firsttab` [C2](#), [C65](#), [C66](#), [C67](#), [C97](#), [C109](#)
`\@flcheckspace` ... [K731](#), [K767](#), [K1398](#)
`\@flfail` [K660](#), [K685](#), [K702](#), [K711](#)
`\@float` [G26](#), [G32](#)
`\@floatboxreset` [G97](#), [G99](#)
`\@floatpenalty` [G3](#),
 [G49](#), [G51](#), [G54](#), [G116](#), [G119](#),
 [G124](#), [G126](#), [G133](#), [G137](#),
 [G190](#), [G192](#), [G196](#), [G200](#), [G234](#)
`\@floatplacement` [k25](#), [G169](#),
 [K130](#), [K157](#), [K201](#), [K353](#), [K1252](#)
`\@flsetnum` [K728](#),
 [K764](#), [K849](#), [K987](#), [K1045](#), [K1366](#)
`\@flsettextmin` ... [K825](#), [K973](#), [K1382](#)
`\@flstop` [K1268](#)
`\@flsucceed` .. [K653](#), [K661](#), [K684](#), [K713](#)
`\@fltovf` [g268](#), [G89](#), [G201](#)
`\@flupdates` [K734](#), [K779](#), [K1443](#)
`\@flushglue`
 [e17](#), [y77](#), [y83](#), [y90](#), [y103](#), [A76](#), [B187](#)
`\@fnsymbol` [m39](#), [m58](#)
`\@font@info` [o108](#), [o146](#),
 [o152](#), [o285](#), [o334](#), [o452](#), [p30](#),
 [p38](#), [p46](#), [p74](#), [p87](#), [p126](#), [p154](#),
 [p168](#), [p179](#), [p193](#), [p209](#), [p215](#),
 [p228](#), [p235](#), [p242](#), [p247](#), [p257](#),
 [p269](#), [p281](#), [p465](#), [p481](#), [p490](#),
 [p503](#), [p532](#), [p544](#), [r126](#), [r141](#),
 [r175](#), [r216](#), [r291](#), [r335](#), [r348](#),
 [r431](#), [r510](#), [r552](#), [r645](#), [r793](#), [r822](#)
`\@font@warning` ... [o4](#), [o400](#), [o405](#),
 [p19](#), [p33](#), [p41](#), [p49](#), [p61](#), [p77](#),
 [p448](#), [p464](#), [p480](#), [p489](#), [p502](#),
 [p531](#), [p543](#), [q20](#), [r285](#), [y23](#), [N73](#)
`\@fontswitch` [v108](#), [v110](#)
`\@footnotemark`
 .. [G264](#), [G270](#), [G288](#), [G294](#), [G295](#)
`\@footnotetext` [B217](#),
 [G264](#), [G270](#), [G271](#), [G304](#), [G310](#)
`\@for` . [I16](#), [I41](#), [f16](#), [k99](#), [k216](#), [L78](#),
 [L150](#), [L164](#), [L176](#), [L181](#), [L196](#), [L400](#)
`\@forloop` [f19](#), [f20](#)
`\@fornoop` [f15](#), [f23](#), [f29](#)
`\@fortmp` [f17](#), [f18](#), [f26](#)
`\@fpbot` [G185](#), [K658](#), [K1535](#)
`\@fpmin` [G176](#),
 [G182](#), [K94](#), [K689](#), [K1259](#), [K1458](#)

`\@fps` G37, G38, G40,
 G43, G60, K1289, K1291, K1294
`\@fpsadddefault` ... G41, G44, [K1286](#)
`\@fpsep` G184, K656, K665, K706, [K1535](#)
`\@fpstype` K725,
 K746, K747, K761, K792, K793,
 K815, K817, K820, K822, K872,
 K928, K929, K963, K965,
 K968, K970, K1042, K1057,
 K1059, K1077, K1086, K1121,
 K1122, [K1282](#), K1298, K1300,
 K1302, K1305, K1306, K1307,
 K1309, K1310, K1314, K1315,
 K1317, K1318, K1352, K1354,
 K1356, K1368, K1370, K1384,
 K1386, K1415, K1418, K1429
`\@fptop` G183, K655, [K1535](#)
`\@frameb@x` B104, B118, [B120](#)
`\@framebox` B107, [B108](#)
`\@framepicbox` B107, [B141](#)
`\@freelist` G56, G198,
 G199, K41, K161, K375, K550,
 K565, K579, K661, K1150, K1151
`\@getcirc` [D247](#), [D271](#), [D299](#)
`\@getfpsbit` [K722](#), [K758](#), [K1039](#), [K1325](#)
`\@getlarrow` [D144](#), [D152](#), [D154](#)
`\@getlinechar` D90, [D129](#)
`\@getpen` i7, i10, i21, [i55](#)
`\@getrarrow` [D145](#), [D152](#), [D161](#)
`\@glossaryfile` H21, H22, H31
`\@gnewline` i46, i48, [i49](#)
`\@gobble` .. I11, I25, I26, d79, d102,
 [d181](#), d191, d210, d214, d292,
 f6, f9, g109, g140, g171, g180,
 i42, i207, k54, k204, l29, l889,
 o401, p299, q16, r19, r21, r179,
 r190, r249, r296, r297, r326,
 r332, r340, r345, r363, r377,
 r387, r396, r409, r426, r435,
 r513, r555, r648, r711, r784,
 r815, s133, s143, s153, F126,
 F127, F128, F129, F130, F146,
 G7, K498, K499, K500, K696,
 K1247, K1459, L239, L421,
 L445, L450, N70, N168, N174, N239
`\@gobblecr` i205, i206
`\@gobblefour` [d181](#),
 r15, r176, r287, r289, r293,
 r295, r305, r309, r433, r485, L452
`\@gobbletwo` d148,
 d149, [d181](#), f12, k26, o406, r93,
 y16, y24, J11, J13, L444, N79
`\@gtempa` d95, d96, d154, d156, k183,
 k184, k186, k187, k188, C3, C5,
 C6, C7, C8, L84, L85, L95, L97
`\@halfwidth` [D2](#), [D48](#),
 D50, D57, D127, D177, D180,
 D196, D203, D217, D227, D230,
 D335, D361, D374, D375, D376
`\@halignto` .. C152, C156, C159, C173
`\@hangfrom` F49, F100, [F121](#)
`\@height` b190,
 d13, i148, i156, l247, l249, p144,
 t246, t464, t465, t467, t468,
 B88, B93, B127, B137, B258,
 B288, C168, C201, C327, C344,
 D127, D178, D181, D196, D203,
 D219, D226, D293, D375, K1189
`\@highpenalty` i56, [N3](#)
`\@hightab` ... [C11](#), C23, C25, C65,
 C77, C86, C87, C102, C133, C134
`\@hline` D81, [D126](#), [D143](#)
`\@holdpg` K103, K250,
 K252, K253, K258, K259, K260
`\@hspace` i191, [i192](#)
`\@hspacer` i191, [i193](#)
`\@hvector` D139, [D143](#)
`\@icentercr` y71, [y72](#)
`\@iden` [d187](#)
`\@if` d143, d144, [d146](#)
`\@if@pti@ns` L74, L76, L82
`\@if@ptions` .. L69, L70, L73, L75, L313
`\@ifatmargin` [C57](#), [C97](#)
`\@ifclasslater` [424](#), [L51](#)
`\@ifclassloaded` [424](#), [L40](#)
`\@ifclasswith` [424](#), [L69](#)
`\@ifdefinable` 28, d51,
 d53, d99, [d101](#), d212, l14, l17,
 m11, n3, s54, B54, E7, E15, E22
`\@iffileonpath` k140, [k148](#)
`\@ifl@aded` .. L40, L41, L44, L50, L312
`\@ifl@t@r` L56, L59, L66, L260
`\@ifl@ter` l875,
 l876, L51, L52, L55, L58, L340
`\@ifl@ter@@` l875, l876
`\@ifnch` d257, [d259](#), [d271](#)
`\@ifnextchar`
 .. I3, I13, a45, 28, [d253](#), d258,
 d274, i44, i206, k163, m13,
 p371, y70, z203, A133, B5, B7,
 B10, B31, B56, B57, B61, B65,
 B106, B107, B109, B142, B146,
 B150, B154, B195, B199, B203,
 B250, B266, B268, C59, C163,
 C185, C192, D12, D63, D74,
 D263, E3, E5, E28, G27, G162,
 G203, G262, G285, G302, K157,
 K1270, L90, L255, L270, L275
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=lt pictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

- \@iforloop [f21](#), [f22](#)
- \@ifpackagelater [424](#), [L51](#)
- \@ifpackageloaded [424](#), [L40](#)
- \@ifpackagewith [424](#), [L69](#)
- \@ifframebox [B110](#), [B111](#), [B112](#)
- \@ifframepicbox [B142](#), [B143](#)
- \@ifstar [28](#), [d40](#), [d274](#), [i38](#), [i132](#), [i191](#),
[o181](#), [q111](#), [y69](#), [y136](#), [z237](#),
[C58](#), [C184](#), [C191](#), [D73](#), [D296](#),
[F35](#), [F125](#), [K1198](#), [L125](#), [L147](#)
- \@ifundefined [I20](#), [I44](#), [28](#), [d96](#), [d103](#),
[d123](#), [d131](#), [d156](#), [d167](#), [d245](#),
[l891](#), [m3](#), [m7](#), [m16](#), [o75](#), [o161](#),
[p384](#), [r211](#), [x23](#), [y44](#), [y53](#), [E21](#),
[J3](#), [J7](#), [L38](#), [L115](#), [L177](#), [L482](#), [L485](#)
- \@ignorefalse [y4](#), [y58](#), [y63](#), [G239](#)
- \@ignoretrue
[i82](#), [y4](#), [y7](#), [z196](#), [z199](#), [z231](#), [z332](#)
- \@iiiminipage
... [B197](#), [B201](#), [B204](#), [B205](#), [B206](#)
- \@iiiparbox [B148](#),
[B152](#), [B155](#), [B156](#), [B157](#), [B234](#)
- \@iiminipage [B200](#), [B202](#)
- \@iinput [k163](#), [k164](#)
- \@iiparbox [B151](#), [B153](#)
- \@iirsbox [B268](#), [B277](#)
- \@imakebox [B10](#), [B25](#), [B63](#)
- \@imakepicbox ... [B31](#), [B32](#), [B68](#), [B144](#)
- \@iminipage [B196](#), [B198](#)
- \@include [k89](#), [k90](#)
- \@index [H18](#), [H19](#), [H35](#)
- \@indexfile [H4](#), [H5](#), [H14](#)
- \@inlabelfalse [A28](#), [A104](#), [A174](#), [K140](#)
- \@inlabeltrue [A28](#), [A168](#)
- \@inmatherr .. [g282](#), [A112](#), [A132](#), [D296](#)
- \@inmathwarn [l3](#)
- \@input [k28](#), [k93](#), [k171](#), [F135](#)
- \@input@ [I31](#), [k108](#), [k173](#), [o344](#)
- \@inputcheck [a17](#), [a138](#), [a139](#), [a142](#),
[a150](#), [d24](#), [k3](#), [k135](#), [k136](#), [k143](#),
[k152](#), [k153](#), [k156](#), [L432](#), [L433](#), [L440](#)
- \@insertfalse [K813](#), [K961](#), [K1037](#)
- \@inserttrue [K741](#), [K786](#), [K900](#), [K1115](#)
- \@invalidchar [g288](#)
- \@iparbox [B147](#), [B149](#)
- \@irsbox [B266](#), [B268](#), [B269](#)
- \@isavebox [B61](#), [B62](#)
- \@isavepicbox [B66](#), [B67](#)
- \@ishortstack [D64](#), [D72](#)
- \@istackcr [D74](#), [D75](#)
- \@itabcr [C59](#), [C60](#)
- \@item [A133](#), [A146](#)
- \@itemdepth .. [A231](#), [A233](#), [A234](#), [A235](#)
- \@itemfudge [C40](#), [C46](#), [C73](#)
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- \@itemitem [A235](#), [A238](#)
- \@itemlabel [A44](#), [A96](#), [A133](#)
- \@itempenalty [i32](#), [A23](#), [A165](#)
- \@iwhiledim [f7](#)
- \@iwhilenum [f3](#)
- \@iwhilesw [f10](#)
- \@ixpt [o490](#)
- \@ixstackcr [D73](#)
- \@killglue [D30](#), [D38](#), [D44](#)
- \@kludgeins ... [K270](#), [K271](#), [K272](#),
[K274](#), [K300](#), [K301](#), [K379](#), [K400](#),
[K401](#), [K407](#), [K408](#), [K409](#), [K418](#),
[K434](#), [K438](#), [K448](#), [K1191](#), [K1230](#)
- \@labels [A27](#),
[A136](#), [A137](#), [A179](#), [A196](#), [A197](#)
- \@largefloatcheck . [G117](#), [G134](#), [G154](#)
- \@lastchclass [C232](#),
[C242](#), [C243](#), [C245](#), [C253](#), [C276](#),
[C290](#), [C294](#), [C303](#), [C316](#), [C317](#)
- \@latex@e@error [d292](#)
- \@latex@error [I47](#),
[d97](#), [d124](#), [g150](#), [g191](#), [g215](#),
[g222](#), [g225](#), [g229](#), [g232](#), [g234](#),
[g238](#), [g241](#), [g244](#), [g248](#), [g254](#),
[g261](#), [g266](#), [g269](#), [g272](#), [g274](#),
[g276](#), [g280](#), [g285](#), [g288](#), [k88](#), [l52](#),
[o8](#), [o33](#), [o77](#), [o119](#), [o162](#), [o201](#),
[o256](#), [p105](#), [q90](#), [q101](#), [r14](#),
[r55](#), [r75](#), [r116](#), [r137](#), [r153](#), [r233](#),
[r265](#), [r305](#), [r309](#), [r351](#), [r356](#),
[r411](#), [r479](#), [r485](#), [r519](#), [r523](#),
[r527](#), [r561](#), [r571](#), [r655](#), [r660](#),
[r663](#), [r695](#), [r698](#), [r752](#), [r755](#),
[r758](#), [r825](#), [r831](#), [s36](#), [s86](#), [v125](#),
[y54](#), [y129](#), [y141](#), [z253](#), [A209](#),
[C91](#), [C100](#), [F4](#), [G6](#), [L214](#), [L233](#),
[L246](#), [L314](#), [L389](#), [L406](#), [L414](#), [L419](#)
- \@latex@info [d197](#), [g150](#)
- \@latex@info@no@line [g150](#), [K458](#)
- \@latex@warning
[I22](#), [I45](#), [g150](#), [g188](#), [l60](#), [x14](#),
[D259](#), [G158](#), [K1292](#), [L469](#), [L475](#)
- \@latex@warning@no@line
..... [d175](#), [g150](#), [g189](#),
[k13](#), [k202](#), [x8](#), [x26](#), [x27](#), [y31](#),
[F6](#), [K191](#), [K223](#), [K1165](#), [K1358](#),
[L86](#), [L261](#), [L341](#), [L434](#), [L441](#), [L499](#)
- \@latexbug [g271](#), [K285](#), [K1151](#)
- \@latexerr
[g188](#), [K182](#), [K314](#), [K1221](#), [K1238](#)
- \@lbibitem [I3](#), [I4](#)
- \@ldots [t412](#), [t414](#)
- \@leftcolumn [K102](#), [K1472](#), [K1481](#)
- \@leftmark [J16](#), [J36](#)

`\@let@token` d257,
 d260, d263, d271, i172, i173,
 i180, v65, v78, z153, z155, z158
`\@lign` z138, z140
`\@linechar` .. D90, D91, D92, D96,
 D97, D99, D104, D106, D107,
 D108, D109, D111, D115, D116,
 D119, D120, D125, D150, D325
`\@linefmt` D47, D49, D90,
 D143, D151, D182, D185, D332
`\@linelen` D78,
 D79, D103, D110, D119, D121,
 D126, D127, D128, D136, D137,
 D178, D181, D183, D184, D326
`\@listctr` I9, A192, A215
`\@listdepth`
 . A23, A35, A38, A43, A99, B218
`\@listfiles` k52, k208, k223
`\@loadwithoptions` . L220, L226, L230
`\@lowpenalty` i55, N3
`\@ltab` C62, C97
`\@m` I17, b21, b150, b152, b153, b186,
 b187, i190, k39, A80, D113, D117
`\@mainaux`
 . k5, k31, k32, k81, k93, k118, y15
`\@makebox` B7, B9
`\@makecaption` G24
`\@makecol` K209, K336, K356
`\@makefcolumn`
 . K317, K318, K326, K328, K1456
`\@makefnmark` G255, G298
`\@makefntext` B246, G281
`\@makeother`
 a23, a44, a73, d277, d278, o357,
 o358, o359, o360, o361, o362,
 o363, o364, o365, o366, o367,
 y113, y123, y134, L110, L111, L457
`\@makepicbox` B6, B30, D232
`\@makespecialcolbox` K380, K404
`\@marbox` . G199, G201, G205, G209,
 G210, G234, K1150, K1160,
 K1163, K1171, K1173, K1174,
 K1176, K1177, K1178, K1187
`\@marginparreset` G218, G225
`\@markright` J29, J34
`\@maxdepth` . k50, K72, K362, K394, N9
`\@maxtab` C2, C85
`\@medpenalty` i56, N3
`\@midlist` K47,
 K375, K376, K769, K771, K880
`\@minipagefalse` A171, B191,
 B193, B231, G112, G144, G220
`\@minipagerestore` B219, B221
`\@minipagetrue` B192, G111
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

`\@minus` d13, K1528,
 K1529, K1530, K1533, K1534
`\@missingfileerror`
 k167, k174, 424, L334
`\@miv` e3
`\@mkboth` J11, J13
`\@mklab` A45, A130
`\@mkpream` C171, C204, C232
`\@mparbottom` G242,
 G243, K99, K352, K1161,
 K1169, K1170, K1171, K1172
`\@mpargs` B210, B234
`\@mparswitchfalse` K83
`\@mpfn` . B216, G262, G267, G307, G311
`\@mpfootins` B225,
 B226, B229, B235, B238, B239
`\@mpfootnotetext` B217, B237
`\@mplistdepth` B218, B235
`\@multicnt`
 C338, C340, C341, C342, C349,
 C350, C351, D38, D39, D41,
 D322, D359, D361, D362, D363,
 D364, D368, D372, D383, D387
`\@multiplelabels` k27, x25, x31, y29, y35
`\@multiput` D36, D37
`\@multispan` C339, C343, C347
`\@namedef` 28, d27, k128,
 l894, o110, o111, o135, p378,
 x28, y121, z257, z258, C157,
 E12, E13, E18, E19, E23, E24, E25
`\@nameuse`
 28, d28, k116, k127, E23, J5, K486
`\@nbitem` A158, A211
`\@ne` b16
`\@needsf@rmat` L256, L259, L264
`\@needsformat` L244, L254, L258
`\@negargfalse` D86
`\@negargtrue` D85
`\@newcommand` d46, d47
`\@newctr` m13, m15, E8
`\@newenv` d119, d120, d130
`\@newenva` d117, d118
`\@newenvb` d119, d120
`\@newl@bel` I10, x22, y17
`\@newline` i45, i47
`\@newlistfalse`
 ... A29, A33, A108, A172, K477
`\@newlisttrue` A29, A33, A87
`\@next` G56, G198, G199,
 K9, K161, K261, K668, K1150
`\@nextchar`
 .. C239, C240, C298, C299, C300
`\@nil` a108,
 a109, c5, c11, d30, d31, d32,

d104, d251, d252, f13, f19, f27,
 j14, l82, o275, o288, o373, o433,
 o436, o437, o445, p306, p307,
 p309, p322, p328, p332, p333,
 p373, p394, p401, p509, p523,
 q16, q34, q43, q47, r31, r275,
 r283, r316, r836, r838, v41, v45,
 C335, C336, L27, L29, L60,
 L61, L67, L195, L198, L292, L299
 \@nmbrlistfalse A33, A46, A91
 \@nmbrlisttrue A215
 \@nnil f13, f20, f21, f22, f28,
 o185, p133, p135, p299, p301,
 p315, p317, p322, p336, p338,
 p345, p360, p361, p363, p394, p401
 \@no@font@optfalse q119
 \@no@lnbk i13, i14, i15
 \@no@pgbk i3, i4, i5
 \@nobreakfalse i58, i60, A183, F77,
 F112, F140, G107, K142, K889
 \@nobreaktrue i59, F109, G106
 \@nocnterr g224
 \@nocounterr g224, m4, m8, m16, E21
 \@nodocument
 . g231, k58, y50, G35, K133, K160
 \@noitemargfalse A32, A190
 \@noitemargtrue A32, A133
 \@noitemerr
 . g275, i111, i130, A69, A81, A107
 \@noligs y114, y135, y151
 \@nolnerr g221, i17, i51, y68
 \@nomath o2, o254, s31, s49, s51, s56
 \@noparitemfalse A30, A135
 \@noparitemtrue A30, A66
 \@noparlistfalse A31, A70
 \@noparlisttrue A31, A67
 \@normalcr i35, i43, B190
 \@normalsize L4, L5
 \@noskipsecfalse k45, F81, K135
 \@noskipsectrue F21, F78
 \@notdefinable d105, d106, d110, g214
 \@notprerr g279, k56
 \@nthm E3, E4
 \@nxttabmar C11, C23, C25,
 C27, C66, C102, C103, C109, C110
 \@obsoletefile k201
 \@oddfoot J11, J14, J15, K105, K489
 \@oddhead J11, J14, K104, K489
 \@onefilewithoptions
 L284, L288, L294, L303, L352
 \@onelevel@sanitize d279, G38
 \@onlypreamble I40, d33, d161,
 d163, d172, d180, k61, k70, k85,
 File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx
 k203, k229, l23, l24, l66, l67,
 l71, l94, l114, l144, l145, l159,
 l895, o26, o90, o92, o98, o114,
 o142, o157, o178, o183, o198,
 o384, p379, q18, q26, q32, q69,
 q73, q78, q83, q88, q98, q116,
 q117, q118, q124, q128, q133,
 r8, r10, r35, r37, r63, r72, r97,
 r167, r168, r171, r203, r236,
 r238, r240, r253, r268, r315,
 r317, r359, r398, r414, r491,
 r530, r533, r574, r577, r580,
 r600, r613, r666, r701, r705,
 r708, r761, r781, r785, r849,
 v122, v123, x30, H12, H29,
 L10, L12, L18, L19, L26, L28,
 L34, L36, L39, L42, L43, L50,
 L53, L54, L58, L66, L68, L71,
 L72, L75, L82, L91, L99, L101,
 L118, L121, L122, L133, L134,
 L135, L142, L148, L161, L174,
 L186, L188, L193, L199, L204,
 L208, L211, L219, L224, L227,
 L231, L240, L253, L258, L264,
 L273, L278, L302, L352, L354,
 L363, L376, L377, L380, L387,
 L396, L403, L404, L412, L417,
 L422, L505, L506, L507, L508, L510
 \@opargbegintheorem E32, E35
 \@opcol K210, K218, K318, K336, K341
 \@options L187
 \@othm E3, E20
 \@outerparskip
 A8, A88, A117, A142, A212
 \@outputbox K101,
 K359, K361, K383, K386, K387,
 K411, K413, K414, K419, K422,
 K427, K429, K436, K442, K444,
 K515, K541, K547, K557, K558,
 K581, K588, K651, K654, K657,
 K663, K664, K1472, K1478, K1486
 \@outputdblcol K344, K1469
 \@outputpage
 K327, K346, K472, K1493, K1501
 \@oval D263, D264
 \@ovbtrue D265
 \@ovdx D239, D273, D279, D281,
 D292, D294, D348, D349, D350,
 D351, D365, D366, D368, D382
 \@ovdy D239, D274, D280, D281,
 D286, D290, D355, D356, D357,
 D358, D369, D370, D372, D386
 \@ovhorz D278, D279, D291
 \@ovltrtrue D265

\@ovri .. B17, [D239](#), D272, D286, D295
 \@ovro ... [D239](#), D272, D279, D280,
 D285, D290, D291, D300, D307
 \@ovrtrue D265
 \@ovttrue D265
 \@ovvert [D276](#), [D277](#), [D283](#)
 \@ovxx [D239](#),
 D267, D269, D273, D277, D278,
 D291, D345, D346, D347, D351,
 D360, D361, D367, D368, D381
 \@ovyy ... [D239](#), D268, D269, D274,
 D279, D283, D352, D353, D354,
 D358, D360, D371, D372, D385
 \@p@pfilename L27, L29, L34
 \@pagedp K98,
 K258, K263, K831, K1179, K1189
 \@pageht
 K97, K259, K263, K265, K266,
 K267, K272, K830, K1162, K1169
 \@par [h3](#), [h6](#)
 \@parboxrestore
 B162, B190, B215, B242,
 G19, G96, G217, G276, K167, K478
 \@parboxto [B157](#)
 \@parmoderr [g265](#), [G54](#), [G195](#)
 \@parse@version .. L60, L61, L67, L68
 \@partaux [k5](#), [k87](#), [k103](#),
 [k105](#), [k106](#), [k112](#), [k121](#), [k123](#), [k126](#)
 \@partlist [k84](#), [k99](#)
 \@partswfalse [k8](#)
 \@partswtrue [k83](#)
 \@pass@ptions
 L113, L118, L119, L120, L328
 \@pboxswfalse B160, B208
 \@pboxswtrue B170
 \@penup [z129](#), [z130](#)
 \@percentchar
 [a53](#), [L449](#), [L451](#), [L453](#), [L455](#), [L494](#)
 \@picbox [D6](#), D21, D27, D28
 \@picht [D6](#), D20, D27
 \@picture D12, [D18](#)
 \@picture@warn [D123](#), [D251](#), [D255](#), [D259](#)
 \@pkgextension [L16](#), [L40](#), [L51](#), [L69](#),
 L119, L210, L213, L230, L294, L398
 \@plus [d13](#), [i197](#), [F16](#), [F151](#),
 J40, K1528, K1529, K1530,
 K1533, K1534, K1538, K1539,
 K1540, K1544, K1545, K1546
 \@pnumwidth F163
 \@popfilename L20, L349
 \@pr@videpackage L90, L92, L99
 \@preamble C172, C174,
 C182, C207, C226, C228, C229,
 C233, C248, C266, C267, C302
 \@preamblecmds .. [d33](#), [k57](#), L517, L518
 \@preamerr ... [g250](#), C181, C244, C323
 \@process@pti@ns
 L160, L173, L175, L186
 \@process@ptions .. L147, L149, L161
 \@protected@testopt [d57](#), [d69](#)
 \@providesfile .. [a45](#), [a46](#), [L102](#), [N235](#)
 \@optionlist
 [L37](#), [L74](#), [L146](#), [L318](#), [L324](#), [L399](#)
 \@pushfilename [L20](#), [L304](#)
 \@put [D262](#), D281, D307
 \@qend [d105](#), [d251](#), [g218](#)
 \@qrelax [d106](#), [d251](#)
 \@rc@ifdefinable [d99](#), [d101](#), [d212](#), [i14](#)
 \@reargdef [d91](#)
 \@refundefined [k46](#), [x3](#), [y27](#)
 \@reinserts K279, K282, [K397](#)
 \@removeelement [f32](#), L190
 \@reqcolroom K830,
 K831, K834, K836, K837, K842,
 K846, K848, K875, K876, K977,
 K979, K981, K984, K986,
 [K1282](#), K1399, K1403, K1406
 \@reset@ptions L309, L350, [L355](#)
 \@resetactivechars [K457](#), [K475](#)
 \@resethfps [K944](#), [K1349](#)
 \@restorepar .. [h6](#), [i139](#), [i155](#), [57](#), [A126](#)
 \@reversemarginfalse G243, K82
 \@reversemargintrue G242
 \@rightmark [J16](#), [J37](#)
 \@rightskip [y79](#), [y83](#), [A75](#), [B186](#)
 \@rjfieldfalse C36, C68
 \@rjfieldtrue C116
 \@roman [m35](#), [m41](#)
 \@rsbox [B266](#), [B267](#)
 \@rtab C62, [C77](#)
 \@rule B250, [B251](#)
 \@sanitize .. [d277](#), [H7](#), [H18](#), [H24](#), [H35](#)
 \@savebox B57, B60
 \@savemarbox . G205, G206, G209, [G212](#)
 \@savepicbox B57, [B64](#)
 \@savsf [i61](#), [i67](#), [i72](#), [i80](#)
 \@savsk [i61](#), [i66](#), [i73](#), [i81](#)
 \@scolelt K606, [K648](#)
 \@sdblcolelt K622, [K649](#)
 \@secntformat F43, [F94](#)
 \@secondoftwo
 [a35](#), [d184](#), [d249](#), [k149](#), [i100](#),
 1913, 1929, [x21](#), [J17](#), [L46](#), [L62](#), [L80](#)
 \@secpenalty [i33](#), [F19](#), [F33](#)
 \@sect [F37](#), [F38](#)
 \@seqncr [z256](#)
 \@setckpt [k121](#), [k128](#), [y16](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfssscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

- \@setfloattypecounts [K814](#), [K962](#), [K1038](#), [K1296](#)
- \@setfontsize [s56](#)
- \@setfps [G34](#)
- \@setfpsbit ... [G70](#), [G73](#), [G76](#), [K1340](#)
- \@setminipage
.. [B220](#), [G21](#), [G102](#), [G110](#), [G231](#)
- \@setnobreak [G104](#), [G230](#)
- \@setpar [h3](#), [57](#), [A78](#)
- \@setref [x10](#)
- \@setsize [s56](#)
- \@settab [C62](#), [C84](#)
- \@settodim [n6](#)
- \@settopoint [n11](#)
- \@sharp ... [C178](#), [C205](#), [C235](#), [C250](#),
[C251](#), [C269](#), [C271](#), [C273](#), [C301](#)
- \@shipoutsetup [K472](#)
- \@shortstack [D63](#), [D64](#)
- \@sline [D81](#), [D84](#), [D147](#)
- \@slowromancap [m42](#), [m43](#)
- \@spaces [g192](#)
- \@specialoutput [K204](#)
- \@specialpagefalse [K78](#), [K486](#)
- \@specialpagetrue [J9](#)
- \@specialstyle [J9](#), [K486](#)
- \@sptoken [d260](#), [d270](#)
- \@sqrt [z203](#)
- \@ssect [F36](#), [F95](#)
- \@stackcr [D70](#), [D73](#)
- \@star@or@long [d39](#), [d44](#),
[d93](#), [d115](#), [d121](#), [d151](#), [d160](#), [d194](#)
- \@startcolumn [K211](#), [K218](#), [K593](#)
- \@startdblcolumn [K593](#), [K1499](#), [K1505](#)
- \@startfield
.. [C30](#), [C48](#), [C83](#), [C95](#), [C116](#), [C124](#)
- \@startline .. [C22](#), [C59](#), [C60](#), [C61](#), [C74](#)
- \@startpbox [C175](#),
[C206](#), [C236](#), [C300](#), [C352](#), [C354](#)
- \@startsection [F22](#)
- \@starttoc [F132](#)
- \@stopfield [C34](#), [C50](#), [C61](#),
[C77](#), [C84](#), [C116](#), [C118](#), [C127](#), [C129](#)
- \@stopline [C32](#), [C58](#), [C76](#)
- \@stpelt [m20](#), [m23](#)
- \@strip@args [l79](#)
- \@svector [D139](#), [D147](#)
- \@sverb [y136](#), [y137](#), [y144](#)
- \@svsec [F40](#), [F43](#), [F49](#), [F61](#)
- \@svsechd [F59](#), [F84](#), [F104](#)
- \@sxverbatim [y95](#), [y121](#)
- \@tabacckludge .. [l178](#), [l180](#), [l353](#), [l354](#)
- \@tabacol [C160](#), [C228](#)
- \@tabarray [C152](#), [C162](#), [C163](#)
- \@tabclassiv [C162](#), [C298](#)
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- \@tabclassz [C161](#), [C252](#)
- \@tabcr [C58](#), [C64](#)
- \@tabfbox [C16](#), [C71](#), [C73](#)
- \@tablab [C63](#), [C117](#)
- \@tabminus [C63](#), [C108](#)
- \@tabplus [C63](#), [C101](#)
- \@tabpush
[C11](#), [C68](#), [C76](#), [C127](#), [C130](#), [C132](#)
- \@tabrj [C63](#), [C115](#)
- \@tabular [C156](#), [C159](#), [C160](#)
- \@tabularcr [C162](#), [C190](#)
- \@tempboxa [e13](#), [l74](#), [n6](#), [n7](#),
[A195](#), [A201](#), [A202](#), [A204](#), [B13](#),
[B14](#), [B15](#), [B16](#), [B21](#), [B22](#), [B23](#),
[B24](#), [B100](#), [B116](#), [B123](#), [B133](#),
[B211](#), [B234](#), [B273](#), [B274](#), [B275](#),
[B282](#), [B283](#), [B284](#), [B285](#), [D182](#),
[D183](#), [D257](#), [D258](#), [D272](#), [D275](#),
[D280](#), [D281](#), [D300](#), [D301](#), [D306](#),
[D307](#), [D373](#), [D391](#), [F121](#), [F122](#),
[G201](#), [G235](#), [K255](#), [K301](#), [K306](#),
[K307](#), [K448](#), [K505](#), [K512](#), [K513](#),
[K539](#), [K543](#), [K555](#), [K561](#), [K568](#),
[K569](#), [K570](#), [K571](#), [K575](#), [K583](#)
- \@tempcnta [e7](#), [r582](#), [r583](#),
[r584](#), [r585](#), [r589](#), [C212](#), [C213](#),
[C214](#), [C215](#), [D87](#), [D88](#), [D114](#),
[D115](#), [D116](#), [D129](#), [D130](#), [D131](#),
[D132](#), [D134](#), [D135](#), [D148](#), [D149](#),
[D154](#), [D156](#), [D157](#), [D158](#), [D159](#),
[D160](#), [D163](#), [D165](#), [D166](#), [D167](#),
[D168](#), [D169](#), [D170](#), [D171](#), [D172](#),
[D173](#), [D174](#), [D204](#), [D205](#), [D206](#),
[D207](#), [D208](#), [D226](#), [D227](#), [D228](#),
[D229](#), [D230](#), [D231](#), [D248](#), [D249](#),
[D250](#), [D252](#), [D254](#), [D256](#), [D258](#),
[D284](#), [D288](#), [D302](#), [D303](#), [D304](#),
[D305](#), [D311](#), [D312](#), [D313](#), [D314](#),
[D315](#), [D316](#), [D364](#), [D380](#), [G58](#),
[G64](#), [G66](#), [G79](#), [G80](#), [G86](#), [G87](#),
[K16](#), [K18](#), [K20](#), [K699](#), [K700](#),
[K701](#), [K702](#), [K724](#), [K727](#), [K760](#),
[K763](#), [K871](#), [K1041](#), [K1044](#),
[K1151](#), [K1153](#), [K1156](#), [K1158](#),
[K1160](#), [K1182](#), [K1330](#), [K1331](#),
[K1335](#), [K1341](#), [K1345](#), [N17](#),
[N22](#), [N23](#), [N24](#), [N86](#), [N91](#), [N92](#), [N93](#)
- \@tempcntb [e7](#),
[r583](#), [r587](#), [r589](#), [D157](#), [D158](#),
[D159](#), [D161](#), [D162](#), [D163](#), [D284](#),
[D285](#), [D288](#), [D289](#), [G84](#), [G85](#),
[G86](#), [K17](#), [K20](#), [K21](#), [K1341](#),
[K1342](#), [K1343](#), [N18](#), [N22](#), [N87](#), [N91](#)
- \@tempdima [e10](#), [z116](#)

- z119, z125, B27, B28, B115,
B116, B121, B122, B123, B125,
B161, B162, B209, B213, B254,
B257, B258, B271, B273, B279,
B282, C37, C38, C39, C79, C80,
C81, C82, C200, C201, D110,
D111, D113, D114, D115, D116,
D117, D118, D247, D248, D249,
D258, D273, D274, D276, D277,
D303, D305, D310, D311, D312,
F156, F157, F166, G121, G123,
G156, G157, G158, K177, K178,
K179, K363, K365, K417, K419,
K420, K425, K430, K434, K439,
K443, K686, K689, K704, K714,
K1103, K1104, K1107, K1108,
K1161, K1162, K1163, K1164,
K1167, K1170, K1173, K1175,
K1447, K1448, K1450, K1451
- \@tempdimb [e10](#), o455,
o459, p133, p134, p399, p432,
p433, p442, p443, p447, p469,
p472, p475, p477, B164, B165,
B255, B258, B272, B274, B280,
B283, D111, D112, D269, D270,
D271, D298, D299, D308, D309,
K704, K705, K706, K707, K714
- \@tempdimc . [e10](#), p426, p427, p429,
p430, p432, p433, B256, B257, B258
- \@tempskipa
. [e14](#), i19, i22, i23, p135, p136,
A116, A117, A118, A140, A142,
A143, A144, A212, A213, A214,
F25, F27, F28, F33, F45, F46,
F71, F72, F74, F86, F87, F96,
F97, K1219, K1220, K1222, K1230
- \@tempskipb [e14](#),
i87, i89, i91, i94, i96, i106, i122, i125
- \@tempswafalse
I13, a25, k97, o69, r205, r255,
r319, r400, r824, r830, v79, y18,
y105, K730, K766, K1047, L430
- \@tempswatruer I13,
a26, k95, k100, o72, r208, r258,
r322, r403, r787, v69, y42, y110,
K1049, K1072, K1408, K1425, L429
- \@temptokena [e16](#), y45,
y46, J22, J23, J30, J31, J34, J35
- \@testdef y17, [y40](#)
- \@testfalse K12, K14, K15
- \@testfp .. K673, K703, [K1333](#), K1459
- \@testopt d20, d46,
[d67](#), d71, d117, i3, i4, i13, i14, z243
- \@testpatch [C240](#), [C316](#)
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=lt pictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- \@testpatch [C316](#)
- \@testtrue K13, K21, K675, K708, [K1337](#)
- \@text@composite [179](#)
- \@text@composite@x [179](#)
- \@textbottom
J40, J42, K389, K431, K445, [K454](#)
- \@textfloatsheight .. K352, K827,
K829, K878, K879, K884, [K1282](#)
- \@textmin G180,
G181, K93, K829, K833, K836,
K837, K981, K1065, K1067,
K1083, K1390, K1392, K1394
- \@textsuperscript . G256, G258, [G259](#)
- \@texttop . J40, J42, K385, K412, [K454](#)
- \@tf@r f25, f26
- \@tfor [f25](#), k150,
k210, v70, B36, C238, D266, G59
- \@tforloop f27, f28, f30
- \@thanks F10, [F13](#)
- \@thefnmark B244,
G255, G256, G263, G268,
G278, G287, G292, G303, G308
- \@thefoot ... K105, K489, K492, K519
- \@thehead ... K104, K489, K491, K509
- \@themargin ... K55, K490, K492, K504
- \@themark . J21, J22, J29, J30, J35, [J38](#)
- \@thirdofthree [d188](#), [i152](#)
- \@thm E12, E18, E24, [E26](#)
- \@thmcounter E11, E17, [E33](#)
- \@thmcountersep E10, [E33](#)
- \@title [F3](#)
- \@tocrmarg F152
- \@toodeep [g240](#), A36, A222, A233
- \@toplist K45, K308, K309, K534,
K540, K550, K551, K767, K779
- \@topnewpage [K147](#)
- \@topnum G169, K86,
K764, K765, K779, K783, K791,
K1024, K1029, K1255, K1276
- \@toproom G171, K87, K767, K779, K1256
- \@topsep A1, A71, A73, A161
- \@topsepadd . A1, A59, A61, A71, A124
- \@totalleftmargin y102,
[A9](#), A53, A54, B185, C37, C67, C72
- \@tracemessage [K1244](#)
- \@traceval [K1244](#)
- \@trivlist A48, [A57](#), A92
- \@tryfcolumn K596, K613, [K627](#), K1460
- \@trylist ... K636, K639, K668, K688
- \@twoclasseserror L201, [L418](#)
- \@twocolumnfalse K80, K128
- \@twocolumntrue K154
- \@twoloadclasserror [L348](#), [L413](#)
- \@twosidefalse K81

`\@typein` d19, d20, d26
`\@typeset@protect` d70, d217,
 d224, d226, l26, l32, l165, l173, s57
`\@uclclist` l815, l816, l863, N165
`\@undefined` . I33, a15, a16, a55, a56,
 a57, a78, a86, a94, a101, a152,
 a156, a182, a189, a249, a250,
 b86, b89, d196, g31, k51, k52,
 k137, l150, l152, o310, o401,
 o469, v104, D338, G5, L4, L338,
 L364, L481, L484, L498, N189,
 N224, N225, N226, N227, N228
`\@unexpandable@noexpand` d192
`\@unexpandable@protect`
 d192, d229, d235, d240, k75, C234
`\@unknownoptionerror` L359, L388, L401
`\@unprocessedoptions`
 ... L185, L229, L335, L339, L403
`\@unused` .. d4, g15, g35, g62, k3, L503
`\@unusedoptionlist`
 . k12, k14, L11, L138, L139, L191
`\@upline` D175, D176, D182
`\@upordown` D95, D96, D104, D125, D151
`\@upvector` D146, D182
`\@use@ption`
 ... L156, L168, L178, L180, L189
`\@use@text@encoding` l115, l1106
`\@vbsphack` i86
`\@verb` y136, y144
`\@verbatim` y100, y118, y121
`\@vereq` t365, t366
`\@viipt` o489
`\@vipt` o488
`\@vipt` o487
`\@vline` D80, D175
`\@vobeyspaces` y93, y118, y144
`\@vpt` o486
`\@vspacer` i132
`\@vspace` i132
`\@vtryfc` K642, K650
`\@vvector` D138, D146
`\@warning` g188
`\@wckptelt` k122, k125
`\@whiledim` f7, D44, D103
`\@whilenoop` f3
`\@whilenum` f3, C214, D39,
 D205, D207, D227, D230, D380
`\@whilesw` f10, K212, K318, K327, K1500
`\@whileswnoop` f10
`\@wholewidth`
 ... B85, B87, B88, B90, B92,
 B93, B94, B95, D2, D48, D50,
 D57, D177, D180, D218, D225,
 D287, D293, D334, D335, D377
`\@width` b193, d13, i193, l245,
 l248, p146, t522, B90, B92,
 B129, B136, B258, B288, C170,
 C201, C315, C334, D127, D177,
 D180, D197, D204, D218, D225,
 D287, D377, G250, K1189, K1483
`\@wrglossary` H25, H30
`\@wrindex` H8, H13
`\@writeckpt` k110, k119
`\@writefile` k26, y43, F147
`\@writesetup` K472
`\@wrong@font@char` .. l126, o402, o413
`\@wtryfc` K652, K662
`\@x@protect` d73, d216
`\@x@sf` G297, G299
`\@xDeclareMathDelimiter` .. r612, r667
`\@xaddvskip` i86, i107
`\@xarg` D77, D80, D85,
 D89, D90, D126, D128, D133,
 D134, D138, D144, D152, D319
`\@xargarraycr` C187, C196, C200
`\@xargdef` d47
`\@xarraycr` C184, C185
`\@xbitor` K15, K17
`\@xcentercr` y69, y70
`\@xdblarg` d275
`\@xdblfloat` G162
`\@xdim` D34, D40, D42, D323,
 D381, D382, D383, D384, D390
`\@xeqncr` z235
`\@exexnoop` C208, C218
`\@exexpast` C209, C210
`\@xfloat` G28, G29, G34, G164
`\@xfootnote` G262, G265
`\@xfootnotemark` G285, G289
`\@xfootnotenext` G302, G305
`\@xhline` C328, C329
`\@xifnch` d261, d271
`\@xipt` o493, t83, t85, t86
`\@xipt` o492, t82
`\@xivpt` o494, t84, t86
`\@xmpar` G203, G204
`\@xnewline` i39, i40, i44
`\@xnext` K10, K11
`\@xnthm` E5, E6
`\@xobeysp` i182, y94, y95
`\@xprocess@ptions` . L147, L162, L174
`\@xpt` o491, t81, t84, t85
`\@xsect` F69, F70, F106
`\@xtabcr` C58, C59
`\@xtabularcr` C191, C192
`\@xthm` E28, E29

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

- `\@xtryfc` [K639](#), [K667](#)
`\@xtypein` [d20](#), [d21](#)
`\@xverbatim` [y95](#), [y118](#)
`\@xvipt` [o495](#), [t85](#), [t87](#)
`\@xxDeclareMathDelimiter` .. [r597](#), [r601](#)
`\@xxpt` [o496](#), [t86](#), [t87](#)
`\@xxvpt` [o497](#), [t87](#)
`\@xxxii` [e2](#), [l324](#), [l326](#), [G85](#),
[K670](#), [K671](#), [K700](#), [K701](#), [K1299](#)
`\@xympar` [G207](#), [G211](#), [G233](#)
`\@yarg` [D77](#),
[D81](#), [D85](#), [D86](#), [D95](#), [D133](#),
[D139](#), [D146](#), [D148](#), [D175](#), [D319](#)
`\@yargarraycr` [C188](#), [C198](#), [C202](#)
`\@yargd@f` [d75](#)
`\@yargdef` [d51](#), [d62](#), [d75](#), [d92](#)
`\@ydim` [D35](#), [D40](#), [D42](#), [D324](#),
[D385](#), [D386](#), [D387](#), [D388](#), [D389](#)
`\@yeqncr` [z235](#)
`\@ympar` [G203](#), [G208](#)
`\@ynthm` [E5](#), [E14](#)
`\@ythm` [E28](#), [E29](#)
`\@ytryfc` [K680](#), [K682](#)
`\@yyarg` [D85](#), [D86](#), [D87](#), [D90](#), [D152](#), [D319](#)
`\@ztryfc` [K687](#), [K698](#)
`\[` [o363](#), [z170](#), [z195](#), [z276](#), [N117](#)
`\@` [a21](#), [a194](#), [a195](#), [a196](#), [a197](#),
[a200](#), [a207](#), [a208](#), [a209](#), [a210](#),
[a213](#), [a220](#), [a221](#), [a222](#), [a223](#),
[a226](#), [a233](#), [a235](#), [a236](#), [a239](#),
[a242](#), [b13](#), [d191](#), [d277](#), [g274](#),
[i35](#), [i204](#), [k209](#), [k224](#), [l420](#), [o351](#),
[t170](#), [y76](#), [y83](#), [y89](#), [y97](#), [z217](#),
[z311](#), [59](#), [B190](#), [B272](#), [B274](#),
[C64](#), [C152](#), [C162](#), [C176](#), [D70](#), [N102](#)
`\{` [a3](#), [a7](#), [a21](#), [b2](#), [b13](#), [g25](#), [l262](#), [l422](#),
[o352](#), [t168](#), [y96](#), [z59](#), [z108](#), [N105](#)
`\}` [a8](#), [a21](#), [b3](#), [b13](#), [g24](#), [l263](#),
[l423](#), [o353](#), [t169](#), [y96](#), [z59](#), [N106](#)
`\]` .. [b239](#), [o364](#), [z170](#), [z196](#), [z286](#), [N118](#)
`\^` [a10](#), [a19](#), [a22](#), [a66](#), [a253](#),
[b7](#), [b9](#), [b11](#), [b14](#), [b158](#), [b159](#),
[b173](#), [b174](#), [d5](#), [d278](#), [i204](#), [i206](#),
[i208](#), [l186](#), [l241](#), [l284](#), [l355](#), [l362](#),
[l418](#), [l501](#), [l508](#), [l512](#), [l517](#), [l522](#),
[l527](#), [l534](#), [l540](#), [l541](#), [l547](#), [l552](#),
[l597](#), [o349](#), [o350](#), [o355](#), [L426](#),
[L427](#), [L428](#), [L480](#), [L483](#), [L486](#),
[N49](#), [N50](#), [N51](#), [N52](#), [N54](#), [N55](#),
[N56](#), [N57](#), [N59](#), [N103](#), [N109](#),
[N110](#), [N111](#), [N112](#), [N125](#), [N126](#),
[N127](#), [N154](#), [N155](#), [N156](#), [N157](#),
[N159](#), [N160](#), [N161](#), [N162](#), [N164](#)
File Key: `a=ltldirchk.dtx`, `b=ltplain.dtx`, `c=ltvers.dtx`, `d=ltdefns.dtx`,
`e=ltalloc.dtx`, `f=ltcntrl.dtx`, `g=lterror.dtx`, `h=ltpar.dtx`, `i=ltspace.dtx`,
`j=ltlogos.dtx`, `k=ltfiles.dtx`, `l=ltoutenc.dtx`, `m=ltcounts.dtx`, `n=ltlength.dtx`,
`o=ltfssbas.dtx`, `p=ltfssstrc.dtx`, `q=ltfsscmp.dtx`, `r=ltfssdcl.dtx`, `s=ltfssini.dtx`,
`t=fontdef.dtx`, `u=preload.dtx`, `v=ltfntcmd.dtx`, `w=ltpageno.dtx`, `x=ltxref.dtx`,
`y=ltmiscen.dtx`, `z=ltmath.dtx`, `A=ltlists.dtx`, `B=ltboxes.dtx`, `C=lttab.dtx`,
`D=ltpictur.dtx`, `E=ltthm.dtx`, `F=ltsect.dtx`, `G=ltfloat.dtx`, `H=ltidxglo.dtx`,
`I=ltbibl.dtx`, `J=ltpage.dtx`, `K=ltoutput.dtx`, `L=ltclass.dtx`, `M=lthyphen.dtx`,
`N=ltfinal.dtx`, `O=ltpatch.ltx`
- `_` [a22](#), [b8](#), [b14](#),
[d278](#), [l268](#), [t173](#), [z166](#), [z167](#), [N104](#)
`\‘` [l187](#),
[l285](#), [l319](#), [l353](#), [l360](#), [l437](#), [l499](#),
[l506](#), [l510](#), [l515](#), [l520](#), [l525](#), [l532](#),
[l536](#), [l537](#), [l545](#), [l550](#), [l598](#), [l637](#),
[o365](#), [s154](#), [y145](#), [B181](#), [C63](#), [N119](#)
`\|` [l421](#), [m59](#), [t479](#), [N120](#)
`\~` [a22](#), [b10](#), [b14](#),
[d278](#), [g23](#), [i184](#), [l194](#), [l242](#), [l286](#),
[l363](#), [l419](#), [l502](#), [l514](#), [l518](#), [l528](#),
[l544](#), [l548](#), [l599](#), [y139](#), [y149](#), [N107](#)
`_` [l17](#), [a21](#), [a38](#), [b13](#), [b158](#),
[b176](#), [d277](#), [d291](#), [g22](#), [g23](#), [g24](#),
[g25](#), [g28](#), [i183](#), [o348](#), [o480](#), [q11](#),
[t171](#), [y93](#), [y94](#), [E36](#), [E38](#), [L104](#), [N97](#)
A
`\A` [N44](#), [N122](#), [N149](#)
`\a` [l178](#), [C1](#), [N33](#), [N123](#), [N138](#)
`\AA` [b164](#), [l195](#), [l327](#), [l389](#)
`\aa` [b164](#), [l200](#), [l321](#), [l399](#)
`\abovedisplayskip` .. [b139](#), [z319](#)
`\abovedisplayskip` [b138](#),
[z312](#), [z314](#), [z316](#), [z317](#), [z318](#), [z319](#)
`\accent` [l76](#), [l295](#), [l322](#), [l375](#), [l610](#)
`\accent@spacefactor` [l75](#), [l76](#), [l77](#)
`\active` [a11](#), [a66](#), [a253](#), [b10](#),
[b11](#), [b173](#), [b174](#), [b176](#), [y93](#), [y94](#),
[y138](#), [y147](#), [z151](#), [z166](#), [K466](#),
[L426](#), [L427](#), [L428](#), [L480](#), [L483](#), [L486](#)
`\active@math@prime` .. [z150](#), [z151](#), [K470](#)
`\acute` [t424](#)
`\add@accent` [l70](#), [l72](#)
`\addcontentsline` [F53](#), [F63](#), [F142](#), [G16](#)
`\addpenalty` [i113](#), [A124](#),
[A160](#), [A165](#), [F33](#), [K290](#), [K892](#)
`\addto@hook`
. [o127](#), [o129](#), [o485](#), [r187](#), [r278](#),
[r282](#), [r299](#), [r423](#), [r429](#), [r437](#),
[r453](#), [r456](#), [r459](#), [r796](#), [r803](#), [r806](#)
`\addtocontents` [F143](#), [F144](#)
`\addtocounter` [m6](#), [m18](#), [l21](#)
`\addtolength` [n5](#), [z314](#), [z316](#), [l25](#)
`\addtoversion` [q130](#)
`\addvspace` [i100](#), [y70](#), [A124](#),
[A161](#), [A162](#), [A166](#), [A214](#), [F33](#)
`\adjdemerits` [b118](#)
`\AE` [l196](#), [l300](#), [l390](#), [l615](#), [N178](#)
`\ae` [l201](#), [l303](#), [l400](#), [l619](#), [N178](#)
`\afterassignment` [b189](#), [b192](#),
[d230](#), [d236](#), [l167](#), [l175](#), [o245](#), [z129](#)

`\aftergroup` o66,
 o259, p156, p222, r70, r82, r90,
 v47, y142, B73, K473, K481, K482
`\aleph` t227
`\alloc@` b47, b48, b49, b50,
 b51, b53, b54, b55, b56, b57, o23
`\allocationnumber` b37, b59,
 b60, b61, b67, b68, b69, C4, C9
`\allowbreak` b196, z40
`\Alpha` m38, 121
`\alph` m37, 121
`\alpha` t187
`\alpha@elt`
 . r36, r191, r373, r475, r795, r796
`\alpha@list` r32, r34, r200, r361, r373,
 r418, r473, r474, r791, r797, r798
`\amalg` t293
`\and` 330, F14
`\angle` t243
`\approx` t333
`\arabic` m34, 121, E33
`\arccos` z13
`\arcsin` z10
`\arctan` z16
`\arg` z26
`\array` C150
`\arraycolsep`
 z220, z221, z324, z325, C228, C306
`\arrayrulewidth`
 C292, C306, C314, C315,
 C327, C331, C334, C344, C346
`\arraystretch` C168, C169, C310
`\Arrowvert` t475
`\arrowvert` t473
`\ast` t151, t309
`\asympt` t357
`\AtBeginDocument` . I34, I48, k47, L373
`\AtBeginDvi` K67
`\AtEndDocument` y9, L373
`\AtEndOfClass` z275, L373
`\AtEndOfPackage` . . . L185, L229, L373
`\atopwithdelims` z57, z58, z59
`\aut@global` d54, d86, d127,
 d137, d141, d147, d202, d285, d289
`\author` 330, F5

B

`\b` I188, I291, I371, I606
`\backslash` t170, t494
`\badness` b97
`\bar` t428
`\baselineskip` b157, b187,
 b223, p140, p141, p142, p144,
 p145, t418, z112, z113, z121,

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfssscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=ltthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

 z127, z131, B188, C180, D67,
 D187, K190, K221, K501, K516
`\baselinestretch`
 o236, p118, p119, p138, p199
`\batchmode`
 k186, k187, q96, s121, N198, N219
`\begin` g232, g234,
 l563, p7, t4, u4, y51, y52, z279,
 F14, F17, G150, G152, L238, M3
`\belowdisplayshortskip` .. b141, z318
`\belowdisplayskip` b140, z317
`\best@size` ... p400, p434, p440, p446
`\beta` t188
`\bezier` 304, D341, D342
`\bfdefault` s14, t32
`\bfseries`
 . I20, s12, s13, v19, x13, E36, E38
`\bgroup` b171
`\bibcite` I7, I9, I10
`\bibdata` I25, I29
`\bibitem` I3
`\bibliography` I27, 359
`\bibliographystyle` I32, 359
`\bibstyle` I25, I37
`\Big` t525, z44, z45, z46
`\big` t524, z41
`\bigbreak` b203
`\bigcap` t263
`\bigcirc` t306
`\bigcup` t264
`\Bigg` t527, z50, z51, z52
`\bigg` t526, z47, z48, z49
`\Biggl` z50
`\biggl` z47
`\Biggm` z51
`\biggm` z48
`\Biggr` z52
`\biggr` z49
`\Bigl` z44
`\bigl` z41
`\Bigm` z45
`\bigm` z42
`\bigodot` t271
`\bigoplus` t270
`\bigotimes` t269
`\Bigr` z46
`\bigr` z43
`\bigskip` b208, i162
`\bigskipamount` . b207, i164, i165, G246
`\bigsqcup` t274
`\bigtriangledown` t279, t280
`\bigtriangleup` t278, t281
`\biguplus` t262
`\bigvee` t260

<code>\bigwedge</code>	t261	<code>\c@bottomnumber</code>	G167 , G172 , K1514
<code>\binoppenalty</code>	b109	<code>\c@dbltopnumber</code>	G166 , G178 , K1521
<code>\bm@b</code>	B21	<code>\c@enumi</code>	A217
<code>\bm@c</code>	B21	<code>\c@enumii</code>	A217 , A217
<code>\bm@l</code>	B21	<code>\c@enumiv</code>	A217
<code>\bm@r</code>	B21	<code>\c@equation</code>	z197 , z230 , z331
<code>\bm@s</code>	B21	<code>\c@errorcontextlines</code>	g181
<code>\bm@t</code>	B21	<code>\c@footnote</code>	F11 , G252 , G291
<code>\bmod</code>	z35	<code>\c@mpfootnote</code>	B216 , G254
<code>\boldmath</code>	j14 , s49	<code>\c@ncel</code>	t369 , t370
<code>\bordermatrix</code>	z115	<code>\c@page</code>	w3 , w6 , w7 , K119 , K1156
<code>\bot</code>	t242	<code>\c@secnumdepth</code>	F39 , F54 , F64 , F123
<code>\botfigrule</code>	K560 , K1547	<code>\c@tocdepth</code>	F123 , F150
<code>\botmark</code>	J36 , K526	<code>\c@topnumber</code>	G165 , G169 , K1510
<code>\bottomfraction</code>	G173 , K1516	<code>\c@totalnumber</code>	G168 , G174 , K1517
<code>\bowtie</code>	t390	<code>\cal</code>	s155
<code>\Box</code>	s92	<code>\calculate@math@sizes</code>	o450 , p173
<code>\boxmaxdepth</code>	b132 , D264 , D298 , K362 , K384 , K428 , K533 , K542 , K582	<code>\cap</code>	t286
<code>\brace</code>	z59	<code>\capitalacute</code>	l683 , l967
<code>\braced</code>	t459 , t463 , t464 , t466 , t468	<code>\capitalbreve</code>	l690 , l974
<code>\bracelu</code>	t461 , t465 , t467	<code>\capitalcaron</code>	l689 , l973
<code>\bracerd</code>	t460 , t465 , t467	<code>\capitalcedilla</code>	l676 , l964
<code>\braceru</code>	t462 , t464 , t468	<code>\capitalcircumflex</code>	l684 , l968
<code>\bracevert</code>	t512	<code>\capitaldieresis</code>	l686 , l970
<code>\brack</code>	z58	<code>\capitaldotaccent</code>	l692 , l976
<code>\break</code>	b196 , b201 , i53	<code>\capitalgrave</code>	l682 , l966
<code>\breve</code>	t429	<code>\capitalhungarumlaut</code>	l687 , l971
<code>\brokenpenalty</code>	b114	<code>\capitalmacron</code>	l691 , l975
<code>\buildrel</code>	t377 , z107	<code>\capitalnewtie</code>	l696 , l1037
<code>\bullet</code>	t295	<code>\capitalogonek</code>	l679 , l965
<code>\bx@A</code>	K23 , K41	<code>\capitalring</code>	l688 , l972
<code>\bx@B</code>	K24 , K41	<code>\capitaltie</code>	l694 , l1035
<code>\bx@C</code>	K25 , K41	<code>\capitaltilde</code>	l685 , l969
<code>\bx@D</code>	K26 , K41	<code>\caption</code>	G4
<code>\bx@E</code>	K27 , K41	<code>\cases</code>	z108
<code>\bx@F</code>	K28 , K42	<code>\cdot</code>	t308
<code>\bx@G</code>	K29 , K42	<code>\cdottp</code>	t410 , t416
<code>\bx@H</code>	K30 , K42	<code>\cdotts</code>	t416
<code>\bx@I</code>	K31 , K42	<code>\cdp@elt</code>	o71 , o91 , o102 , o103 , o124 , o127 , o129 , r125 , r207 , r257 , r321 , r402
<code>\bx@J</code>	K32 , K42	<code>\cdp@list</code>	o73 , o89 , o103 , o131 , o132 , r143 , r209 , r259 , r323 , r404
<code>\bx@K</code>	K33 , K43	<code>\center</code>	y73
<code>\bx@L</code>	K34 , K43	<code>center (environment)</code>	y73
<code>\bx@M</code>	K35 , K43	<code>\centering</code>	y73 , y75
<code>\bx@N</code>	K36 , K43	<code>\centerline</code>	B289
<code>\bx@O</code>	K37 , K44	<code>\cf@encoding</code>	l34 , l41 , l44 , l53 , l119 , o204 , o214 , o224 , o243
<code>\bx@P</code>	K38 , K44	<code>\ch@ck</code>	b58 , b63 , b64 , b65 , b66 , b70 , L437
<code>\bx@Q</code>	K39 , K44	<code>\changes</code>	l560 , G151 , N71
<code>\bx@R</code>	K40 , K44	<code>\char</code>	l293 , l296 , l329 , l332 , l343 , l350 , l373 , l377 , l382 , l384 , l386 , l388 , l580 , l608 , l611 ,

C

`\c` [l189](#), [l294](#), [l374](#), [l463](#),
[l465](#), [l490](#), [l492](#), [l505](#), [l531](#), [l609](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=ltthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

- l641, l648, l655, l678, l681, l729,
s55, y150, z148, D132, D160,
D174, D182, D185, D258, D285,
D289, D302, D303, D305, D316
- \chardef a11, a17, a18, b10,
b16, b17, b18, b19, b20, b51,
b54, b55, b56, b68, b217, b218,
b219, e2, l18, o23, C4, C9, L436
- \chardef@text@cmd l3
- \check t430
- \check@command d160, d162
- \check@ic1
.. v9, v27, v32, v38, v46, v53, v55
- \check@icr
.. v9, v27, v33, v39, v47, v56, v61
- \check@mathfonts
..... j5, l256, o265, o267, p204
- \check@nocorr@ v29
- \check@orange o316, p335, p336
- \check@single o317, p334, p359
- \CheckCommand d160
- \CheckEncodingSubset l906,
l962, l963, l1031, l1091, l1093, l1106
- \chi t207
- \choose z57
- \circ t305
- \circle D260, D296
- \citation l11, l19, l43
- \cite l12, 359
- \cl@ckpt k122, m24
- \cl@page w4
- \ClassError g88
- \ClassInfo g88
- \ClassWarning g88
- \ClassWarningNoLine g88
- \cleaders b237, t454, t457
- \cleardoublepage K119
- \clearpage k91,
k109, y12, y49, K106, K119,
K124, K148, K331, K337, K1464
- \cline C335
- \clubpenalty b111, k10, k19,
A184, A186, F83, F89, F113, F118
- \clubsuit t255
- \col@number ... K76, K129, K156, K168
- \colon t411
- \color@begingroup o476,
z87, z103, B13, B47, B101,
B212, B245, C49, C53, G280, K367
- \color@endbox B47,
G147, G223, K172, K510, K520
- \color@endgroup .. o481, z87, z103,
B13, B47, B59, B80, B103,
B232, B248, C51, G283, K371
- \color@hbox B47, K507, K517
- \color@setgroup B47, B59, B78
- \color@vbox B47, G92, G214, G236, K163
- \columnsep k21, K62, K150
- \columnseprule K63, K1483
- \columnwidth k18,
k21, k22, k24, B214, B241, G95,
G276, K61, K125, K126, K127,
K149, K150, K151, K152, K153,
K1181, K1183, K1480, K1485
- \cong t365
- \contentsline F143, F148
- \coprod t259
- \copyright l240, l270, s75
- \cos z12
- \cosh z14
- \cot z18
- \coth z19
- \count@ a13, a126, a127, a128, a133,
b41, b193, b194, c6, c7, c8,
c9, c10, c12, d140, d145, p22,
p256, p258, p280, p281, r184,
r186, r190, r500, r501, r502,
r542, r543, r544, r585, r586,
r587, r625, r626, r627, r633,
r634, r635, r678, r679, r680,
r686, r687, r688, r728, r729,
r730, r736, r737, r738, v97,
v100, D379, D380, D381, D384,
D385, D388, D392, N29, N30,
N39, N41, N134, N135, N144, N146
- \countdef
.. a13, b37, b38, b39, b41, b47, w3
- \cr b167, l955, l958, z118, z122, z225,
z255, z328, C174, C185, C192,
C201, C202, C345, D72, D74, D75
- \cr cr b224, l292, l296, l299, l372,
l376, l380, l382, l384, l579, l607,
l611, l614, l678, l681, l728, l959,
s77, t243, t244, t246, t367, t370,
t374, t438, t439, t440, t441,
t442, t443, t444, t445, t446,
t447, t448, t450, z109, z111,
z112, z113, z118, z120, z121,
z122, z140, z141, C153, C154, D72
- \cs l560
- \csc z21
- \cup t287
- \curr@fontshape
.. l135, o63, o280, o290, o294,
o296, o387, o393, o396, o402,
- File Key:** a=ltltdirchk.dtx, b=ltlplain.dtx, c=ltlvers.dtx, d=ltldefs.dtx,
e=ltlalloc.dtx, f=ltlcntrl.dtx, g=ltlerror.dtx, h=ltlpar.dtx, i=ltltspace.dtx,
j=ltllogos.dtx, k=ltlfiles.dtx, l=ltloutenc.dtx, m=ltlcounts.dtx, n=ltllength.dtx,
o=ltlssbas.dtx, p=ltlssstrc.dtx, q=ltlssscmp.dtx, r=ltlssdcl.dtx, s=ltlssini.dtx,
t=fontdef.dtx, u=ltlpreload.dtx, v=ltlfontcmd.dtx, w=ltlpageno.dtx, x=ltltxref.dtx,
y=ltlmiscen.dtx, z=ltlmath.dtx, A=ltllists.dtx, B=ltlboxes.dtx, C=ltltab.dtx,
D=ltlpictur.dtx, E=ltlthm.dtx, F=ltlsect.dtx, G=ltlfloat.dtx, H=ltlidxglo.dtx,
I=ltlbibl.dtx, J=ltlpage.dtx, K=ltloutput.dtx, L=ltlclass.dtx, M=ltlthyphen.dtx,
N=ltlfinal.dtx, O=ltlpatch.ltx

o409, o411, p92, p100, p121, p449, p473, p519, p536, r147, r152	\CYRF	1822
\curr@math@size	\cyrf	1822
.... o269, p210, p216, p221, p238	\CYRFITA	1823
\CurrentOption	\cyrfita	1822
1859, L13, L140, L150, L151, L152, L157, L164, L165, L166, L169, L176, L177, L181, L182, L183, L190, L192, L196, L197, L198, L308, L390, L391, L400, L401	\CYRG	1823
\CYRA	\cyrg	1823
\cyra	\CYRGDSC	1823
\CYRABHCH	\cyrgdsc	1823
\cyrabhch	\CYRGDSCHCRS	1823
\CYRABHCHDSC	\cyrgdschcrs	1823
\cyrabhchdsc	\CYRGHCRS	1824
\CYRABHDZE	\cyrgchcrs	1824
\cyrabhdze	\CYRGHK	1824
\CYRABHHA	\cyrghk	1824
\cyrabhha	\CYRGUP	1824
\CYRAE	\cyrgup	1824
\cyrae	\CYRH	1824
\CYRB	\cyrh	1824
\cyrb	\CYRHDSC	1825
\CYRBYUS	\cyrhdsc	1825
\cyrbyus	\CYRHHCRS	1825
\CYRC	\cyrhhcrs	1825
\cyrc	\CYRHHK	1825
\CYRCH	\cyrhkh	1825
\cyrch	\CYRHRDSN	1826
\CYRCHLDSC	\cyrhrdsn	1825
\cyrchldsc	\CYRI	1826
\CYRCHRDSC	\cyri	1826
\cyrchrdsc	\CYRIE	1826
\CYRCHVCRS	\cyrie	1826
\cyrchvcrs	\CYRII	1826
\CYRD	\cyrii	1826
\cyrd	\CYRISHRT	1826
\CYRDELTA	\cyrishrt	1826
\cyrdelta	\CYRISHRTDSC	1827
\CYRDJE	\cyrishrtdsc	1827
\cyrdje	\CYRIZH	1827
\CYRDZE	\cyrizh	1827
\cyrdze	\CYRJE	1827
\CYRDZHE	\cyrje	1827
\cyrdzhe	\CYRK	1827
\CYRE	\cyrk	1827
\cyre	\CYRKBEAK	1828
\CYREPS	\cyrkbeak	1828
\cyreps	\CYRKDSC	1828
\CYREREV	\cyrkdsc	1828
\cyrerev	\CYRKHCRS	1828
\CYRERY	\cyrkchcrs	1828
\cyrery	\CYRKHK	1829
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx, t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx, y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx, D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx, I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=ltthyphen.dtx, N=ltfinal.dtx, O=ltpatch.ltx	\CYRL	1829
	\cyr1	1829

\CYRLDSC	1829	\CYRSH	1836
\cyrldsc	1829	\cyrsh	1836
\CYRLHK	1830	\CYRSHCH	1836
\cyrllhk	1829	\cyrshch	1836
\CYRLJE	1830	\CYRSHHA	1836
\cyrlje	1830	\cyrshha	1836
\CYRM	1830	\CYRT	1837
\cyrm	1830	\cyrt	1837
\CYRMDSC	1830	\CYRTDSC	1837
\cyrmdsc	1830	\cyrtldsc	1837
\CYRMHK	1830	\CYRTETSE	1837
\cyrmhk	1830	\cyrtetse	1837
\CYRN	1831	\CYRTSHE	1837
\cyrn	1831	\cyrtshe	1837
\CYRNDSC	1831	\CYRU	1838
\cyrndsc	1831	\cyr	1838
\CYRNG	1831	\CYRUSHRT	1838
\cyrng	1831	\cyrushrt	1838
\CYRNHK	1831	\CYRV	1838
\cyrnhk	1831	\cyrv	1838
\CYRNJE	1832	\CYRW	1838
\cyrnje	1831	\cyrw	1838
\CYRNLHK	1832	\CYRY	1838
\cyrnlhk	1832	\cyry	1838
\CYRO	1832	\CYRYA	1839
\cyro	1832	\cyrya	1839
\CYROTLD	1832	\CYRYAT	1839
\cyrotld	1832	\cyryat	1839
\CYRP	1832	\CYRYHCRS	1839
\cyrp	1832	\cyrhcrs	1839
\CYRPHK	1833	\CYRYI	1839
\cyrphk	1833	\cyryi	1839
\CYRQ	1833	\CYRYO	1840
\cyrq	1833	\cyryo	1839
\CYRR	1833	\CYRYU	1840
\cyrr	1833	\cyryu	1840
\CYRRDSC	1833	\CYRZ	1840
\cyrldsc	1833	\cyrz	1840
\CYRRHK	1834	\CYRZDSC	1840
\cyrhkh	1833	\cyrzldsc	1840
\CYRRTICK	1834	\CYRZH	1840
\cyrtrick	1834	\cyrzh	1840
\CYRS	1834	\CYRZHDSC	1841
\cyrs	1834	\cyrzldsc	1841
\CYRSACRS	1834		
\cyrscrs	1834		
\CYRSCHWA	1835		
\cyrschwa	1835		
\CYRSDSC	1835		
\cyrldsc	1835		
\CYRSEMISFTSN	1835		
\cyrsemisftsn	1835		
\CYRSFTSN	1836		
\cyrldsc	1836		

D

\d	1190, 1297, 1378, 1612
\dag	1266
\dagger	1266, m58, m59, t289
\dashbox	D187
\dashv	t317
\date	330, F7
\day	a132, L454
\dblfigrule	K585, K1547

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=ltthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

`\dblfloatpagefraction` .. [G182](#), [K1524](#)
`\dblfloatsep` [K571](#), [K583](#), [K1106](#), [K1531](#)
`\dbltextfloatsep`
[K170](#), [K178](#), [K587](#), [K1105](#), [K1531](#)
`\dbltopfraction` [G179](#), [K1523](#)
`\ddag` [l267](#)
`\ddagger` [l267](#), [m58](#), [m60](#), [t288](#)
`\ddot` [t426](#)
`\ddots` [t421](#)
`\deadcycles` [k115](#), [y39](#), [y49](#), [K249](#)
`\declare@robustcommand` [d194](#)
`\DeclareEncodingSubset` [l890](#),
[l897](#), [l898](#), [l899](#), [l900](#), [l1116](#),
[l1117](#), [l1118](#), [l1119](#), [l1120](#), [l1121](#),
[l1122](#), [l1123](#), [l1124](#), [l1125](#), [l1126](#),
[l1127](#), [l1128](#), [l1129](#), [l1130](#), [l1131](#),
[l1132](#), [l1133](#), [l1134](#), [l1135](#), [l1136](#),
[l1137](#), [l1138](#), [l1139](#), [l1140](#), [l1141](#),
[l1142](#), [l1143](#), [l1144](#), [l1145](#), [l1146](#),
[l1147](#), [l1148](#), [l1149](#), [l1150](#), [l1151](#),
[l1152](#), [l1153](#), [l1154](#), [l1155](#), [l1156](#),
[l1157](#), [l1158](#), [l1159](#), [l1160](#), [l1161](#)
`\DeclareErrorFont` [o369](#), [r156](#), [s101](#), [t28](#)
`\DeclareFixedFont` [o48](#)
`\DeclareFontEncoding`
..... [l279](#), [l359](#), [l559](#), [l585](#),
[l591](#), [l674](#), [o93](#), [t45](#), [t46](#), [t47](#), [t48](#)
`\DeclareFontEncoding@` . [o97](#), [o99](#), [o114](#)
`\DeclareFontEncodingDefaults` ...
..... [o143](#), [q80](#), [q81](#), [t15](#)
`\DeclareFontFamily` [o68](#), [q75](#), [q76](#)
`\DeclareFontShape`
..... [o27](#), [o29](#), [q15](#), [q17](#), [q71](#), [q72](#)
`\DeclareFontShape@` [o30](#), [o31](#)
`\DeclareFontSubstitution`
..... [l592](#), [l675](#),
[o116](#), [t16](#), [t17](#), [t49](#), [t50](#), [t51](#), [t52](#)
`\DeclareMathAccent`
..... [r492](#), [r530](#), [t424](#), [t425](#),
[t426](#), [t427](#), [t428](#), [t429](#), [t430](#),
[t431](#), [t432](#), [t433](#), [t434](#), [t435](#), [t436](#)
`\DeclareMathAlphabet`
..... [q109](#), [q113](#), [q115](#), [q122](#),
[r318](#), [r481](#), [r492](#), [t70](#), [t71](#), [t72](#), [t73](#)
`\DeclareMathAlphabetCharacter` . [r591](#)
`\DeclareMathDelimiter`
..... [r593](#), [t174](#), [t175](#),
[t176](#), [t177](#), [t178](#), [t179](#), [t182](#),
[t184](#), [t185](#), [t469](#), [t471](#), [t473](#),
[t475](#), [t477](#), [t480](#), [t482](#), [t484](#),
[t486](#), [t488](#), [t490](#), [t492](#), [t494](#),
[t496](#), [t498](#), [t500](#), [t502](#), [t504](#),
[t506](#), [t508](#), [t510](#), [t512](#), [t514](#), [t516](#)
`\DeclareMathRadical` [r709](#), [t437](#)
File Key: a=lt`dirchk`.dtx, b=lt`plain`.dtx, c=lt`vers`.dtx, d=lt`defs`.dtx,
e=lt`alloc`.dtx, f=lt`cntrl`.dtx, g=lt`error`.dtx, h=lt`par`.dtx, i=lt`space`.dtx,
j=lt`logos`.dtx, k=lt`files`.dtx, l=lt`outenc`.dtx, m=lt`counts`.dtx, n=lt`length`.dtx,
o=lt`fssbas`.dtx, p=lt`fssstrc`.dtx, q=lt`fsscmp`.dtx, r=lt`fssdcl`.dtx, s=lt`fssini`.dtx,
t=lt`fontdef`.dtx, u=lt`preload`.dtx, v=lt`fntcmd`.dtx, w=lt`pageno`.dtx, x=lt`xref`.dtx,
y=lt`miscen`.dtx, z=lt`math`.dtx, A=lt`lists`.dtx, B=lt`boxes`.dtx, C=lt`ttab`.dtx,
D=lt`pictur`.dtx, E=lt`thm`.dtx, F=lt`ssect`.dtx, G=lt`float`.dtx, H=lt`idxglo`.dtx,
I=lt`bibl`.dtx, J=lt`page`.dtx, K=lt`output`.dtx, L=lt`class`.dtx, M=lt`hyphen`.dtx,
N=lt`final`.dtx, O=lt`patch`.ltx

- t387, t409, t410, t411, t459,
t460, t461, t462, t518, t519, t520
- `\DeclareMathVersion` [r169](#), [s1](#), [s2](#)
- `\DeclareOldFontCommand` [v107](#), [v123](#)
- `\DeclareOption` [l844](#), [l897](#), [l898](#),
[l899](#), [l900](#), [l901](#), [l903](#), [p29](#), [p37](#),
[p45](#), [p53](#), [p56](#), [p60](#), [424](#), [L123](#), [L411](#)
- `\DeclareOption*` [424](#), [L123](#)
- `\DeclarePreloadSizes`
. [o160](#), [q85](#), [q86](#), [u19](#), [u21](#),
[u22](#), [u23](#), [u25](#), [u26](#), [u27](#), [u28](#),
[u29](#), [u30](#), [u34](#), [u38](#), [u43](#), [u45](#),
[u49](#), [u50](#), [u53](#), [u54](#), [u57](#), [u58](#), [u64](#)
- `\DeclareRobustCommand` [l12](#),
[d194](#), [g4](#), [g11](#), [g33](#), [g60](#), [i35](#),
[i43](#), [i132](#), [i168](#), [i182](#), [i187](#), [i191](#),
[j3](#), [j13](#), [l261](#), [l262](#), [l263](#), [l264](#),
[l265](#), [l266](#), [l267](#), [l268](#), [l270](#), [l272](#),
[l274](#), [l1101](#), [o199](#), [o227](#), [o228](#),
[o229](#), [o233](#), [o235](#), [o253](#), [p113](#),
[s3](#), [s6](#), [s9](#), [s12](#), [s15](#), [s18](#), [s21](#),
[s24](#), [s27](#), [s30](#), [s75](#), [s79](#), [t364](#),
[t368](#), [t371](#), [t376](#), [t378](#), [t380](#),
[t383](#), [t389](#), [t391](#), [t393](#), [t395](#),
[t397](#), [t399](#), [t401](#), [t403](#), [t405](#),
[t407](#), [t413](#), [t415](#), [t417](#), [t420](#), [v3](#),
[v108](#), [z203](#), [z260](#), [G257](#), [N165](#), [N172](#)
- `\DeclareSizeFunction` [p377](#), [p464](#),
[p465](#), [p480](#), [p481](#), [p489](#), [p490](#),
[p502](#), [p503](#), [p531](#), [p532](#), [p543](#), [p544](#)
- `\DeclareSymbolFont`
. [q126](#), [r204](#), [t60](#), [t61](#), [t62](#), [t63](#)
- `\DeclareSymbolFontAlphabet`
. [r782](#), [t67](#), [t68](#), [t69](#)
- `\DeclareSymbolFontAlphabet@` [r783](#), [r786](#)
- `\DeclareTextAccent` [l69](#), [l280](#),
[l281](#), [l282](#), [l283](#), [l284](#), [l285](#), [l286](#),
[l287](#), [l288](#), [l289](#), [l290](#), [l360](#), [l361](#),
[l362](#), [l363](#), [l364](#), [l365](#), [l366](#), [l367](#),
[l368](#), [l369](#), [l370](#), [l588](#), [l593](#), [l594](#),
[l595](#), [l596](#), [l597](#), [l598](#), [l599](#), [l600](#),
[l601](#), [l602](#), [l603](#), [l682](#), [l683](#), [l684](#),
[l685](#), [l686](#), [l687](#), [l688](#), [l689](#), [l690](#),
[l691](#), [l692](#), [l693](#), [l694](#), [l695](#), [l696](#)
- `\DeclareTextAccentDefault`
[l140](#), [l181](#), [l182](#), [l183](#), [l184](#), [l185](#),
[l186](#), [l187](#), [l188](#), [l189](#), [l190](#), [l191](#),
[l192](#), [l193](#), [l194](#), [l234](#), [l237](#), [l964](#),
[l965](#), [l966](#), [l967](#), [l968](#), [l969](#), [l970](#),
[l971](#), [l972](#), [l973](#), [l974](#), [l975](#), [l976](#)
- `\DeclareTextCommand` [l3](#), [l63](#), [l70](#),
[l291](#), [l294](#), [l297](#), [l313](#), [l314](#), [l321](#),
[l323](#), [l325](#), [l327](#), [l333](#), [l335](#), [l337](#),
[l344](#), [l371](#), [l374](#), [l378](#), [l381](#), [l383](#),
[l385](#), [l387](#), [l577](#), [l604](#), [l606](#), [l609](#),
[l612](#), [l642](#), [l649](#), [l676](#), [l679](#), [l726](#)
- `\DeclareTextCommandDefault`
[l62](#), [l141](#), [l143](#), [l238](#), [l241](#), [l242](#),
[l243](#), [l244](#), [l246](#), [l250](#), [l254](#), [l255](#),
[l257](#), [l258](#), [l259](#), [l260](#), [l1030](#),
[l1032](#), [l1033](#), [l1034](#), [l1035](#), [l1036](#),
[l1037](#), [l1038](#), [l1039](#), [l1040](#), [l1041](#),
[l1042](#), [l1043](#), [l1044](#), [l1045](#), [l1046](#),
[l1047](#), [l1048](#), [l1049](#), [l1050](#), [l1051](#),
[l1052](#), [l1053](#), [l1054](#), [l1055](#), [l1056](#),
[l1057](#), [l1058](#), [l1059](#), [l1060](#), [l1061](#),
[l1062](#), [l1063](#), [l1064](#), [l1065](#), [l1066](#),
[l1067](#), [l1068](#), [l1069](#), [l1070](#), [l1071](#),
[l1072](#), [l1073](#), [l1074](#), [l1075](#), [l1076](#),
[l1077](#), [l1078](#), [l1079](#), [l1080](#), [l1081](#),
[l1082](#), [l1083](#), [l1084](#), [l1085](#), [l1086](#),
[l1087](#), [l1088](#), [l1089](#), [l1090](#), [l1092](#)
- `\DeclareTextComposite`
. [l79](#), [l351](#), [l352](#), [l444](#),
[l445](#), [l446](#), [l447](#), [l448](#), [l449](#), [l450](#),
[l451](#), [l452](#), [l453](#), [l454](#), [l455](#), [l456](#),
[l457](#), [l458](#), [l459](#), [l460](#), [l461](#), [l462](#),
[l463](#), [l464](#), [l465](#), [l466](#), [l467](#), [l468](#),
[l469](#), [l470](#), [l471](#), [l472](#), [l473](#), [l474](#),
[l475](#), [l476](#), [l477](#), [l478](#), [l479](#), [l480](#),
[l481](#), [l482](#), [l483](#), [l484](#), [l485](#), [l486](#),
[l487](#), [l488](#), [l489](#), [l490](#), [l491](#), [l492](#),
[l493](#), [l494](#), [l495](#), [l496](#), [l497](#), [l498](#),
[l499](#), [l500](#), [l501](#), [l502](#), [l503](#), [l504](#),
[l505](#), [l506](#), [l507](#), [l508](#), [l509](#), [l510](#),
[l511](#), [l512](#), [l513](#), [l514](#), [l515](#), [l516](#),
[l517](#), [l518](#), [l519](#), [l520](#), [l521](#), [l522](#),
[l523](#), [l524](#), [l525](#), [l526](#), [l527](#), [l528](#),
[l529](#), [l530](#), [l531](#), [l532](#), [l533](#), [l534](#),
[l535](#), [l536](#), [l537](#), [l538](#), [l539](#), [l540](#),
[l541](#), [l542](#), [l543](#), [l544](#), [l545](#), [l546](#),
[l547](#), [l548](#), [l549](#), [l550](#), [l551](#), [l552](#),
[l553](#), [l554](#), [l556](#), [l557](#), [l558](#), [l559](#),
[l560](#), [l561](#), [l562](#), [l563](#), [l564](#), [l565](#),
[l566](#), [l567](#), [l568](#), [l569](#), [l570](#), [l571](#)
- `\DeclareTextCompositeCommand`
. [l79](#), [l330](#), [l353](#),
[l354](#), [l355](#), [l356](#), [l555](#), [l556](#), [l639](#)
- `\DeclareTextFontCommand`
. [v1](#), [v15](#), [v16](#), [v17](#), [v18](#), [v19](#),
[v20](#), [v21](#), [v22](#), [v23](#), [v24](#), [v25](#), [v122](#)
- `\DeclareTextSymbol` [l3](#),
[l300](#), [l301](#), [l302](#), [l303](#), [l304](#), [l305](#),
[l306](#), [l307](#), [l308](#), [l309](#), [l310](#), [l311](#),
[l312](#), [l315](#), [l316](#), [l317](#), [l318](#), [l319](#),
[l320](#), [l389](#), [l390](#), [l391](#), [l392](#), [l393](#),
[l394](#), [l395](#), [l396](#), [l397](#), [l398](#), [l399](#),
[l400](#), [l401](#), [l402](#), [l403](#), [l404](#), [l405](#),
- File Key:** a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

- 1406, 1407, 1408, 1409, 1410, 1411,
1412, 1413, 1414, 1415, 1416, 1417,
1418, 1419, 1420, 1421, 1422, 1423,
1424, 1425, 1426, 1427, 1428, 1429,
1430, 1431, 1432, 1433, 1434, 1435,
1436, 1437, 1438, 1439, 1440, 1441,
1442, 1443, 1564, 1565, 1566, 1567,
1568, 1569, 1570, 1571, 1572, 1573,
1574, 1575, 1576, 1586, 1587, 1615,
1616, 1617, 1618, 1619, 1620, 1621,
1622, 1623, 1624, 1625, 1626, 1627,
1628, 1629, 1630, 1631, 1632, 1633,
1634, 1635, 1636, 1637, 1638, 1697,
1698, 1699, 1700, 1701, 1702, 1703,
1704, 1705, 1706, 1707, 1708, 1709,
1710, 1711, 1712, 1713, 1714, 1715,
1716, 1717, 1718, 1719, 1720, 1721,
1722, 1723, 1724, 1725, 1732, 1733,
1734, 1735, 1736, 1737, 1738, 1739,
1740, 1741, 1742, 1743, 1744, 1745,
1746, 1747, 1748, 1749, 1750, 1751,
1752, 1753, 1754, 1755, 1756, 1757,
1758, 1759, 1760, 1761, 1762, 1763,
1764, 1765, 1766, 1767, 1768, 1769,
1770, 1771, 1772, 1773, 1774, 1775,
1776, 1777, 1778, 1779, 1780, 1781,
1782, 1783, 1784, 1785, 1786, 1787,
1788, 1789, 1790, 1791, 1792, 1793,
1794, 1795, 1796, 1797, 1798, 1799,
1800, 1801, 1802, 1803, 1804, 1805,
1806, 1807, 1808, 1809, 1810, 1811
- `\DeclareTextSymbolDefault`
 . . . [1140](#), 1195, 1196, 1197, 1198,
 1199, 1200, 1201, 1202, 1203, 1204,
 1205, 1206, 1207, 1208, 1209, 1210,
 1211, 1212, 1213, 1214, 1215, 1216,
 1217, 1218, 1219, 1220, 1221, 1222,
 1223, 1224, 1225, 1226, 1227, 1228,
 1229, 1230, 1231, 1232, 1233, 1235,
 1236, 1977, 1978, 1979, 1980, 1981,
 1982, 1983, 1984, 1985, 1986, 1987,
 1988, 1989, 1990, 1991, 1992, 1993,
 1994, 1995, 1996, 1997, 1998, 1999,
 11000, 11001, 11002, 11003, 11004,
 11005, 11006, 11007, 11008, 11009,
 11010, 11011, 11012, 11013, 11014,
 11015, 11016, 11017, 11018, 11019,
 11020, 11021, 11022, 11023, 11024,
 11025, 11026, 11027, 11028, 11029
- `\default@ds`
 . . . [L132](#), [L143](#), [L179](#), [L357](#), [L359](#)
- `\default@family`
 [o104](#), [o136](#), [o375](#), [o378](#), [o397](#)
- `\default@M` [o111](#), [o151](#), [o154](#), [o158](#)
- File Key: a=lt`dirchk.dtx`, b=lt`plain.dtx`, c=lt`vers.dtx`, d=lt`defs.dtx`,
e=lt`alloc.dtx`, f=lt`cntrl.dtx`, g=lt`error.dtx`, h=lt`par.dtx`, i=lt`space.dtx`,
j=lt`logos.dtx`, k=lt`files.dtx`, l=lt`outenc.dtx`, m=lt`counts.dtx`, n=lt`length.dtx`,
o=lt`ssbas.dtx`, p=lt`ssstrc.dtx`, q=lt`sscmp.dtx`, r=lt`ssdcl.dtx`, s=lt`ssini.dtx`,
t=lt`fontdef.dtx`, u=lt`preload.dtx`, v=lt`fontcmd.dtx`, w=lt`pageno.dtx`, x=lt`xref.dtx`,
y=lt`miscen.dtx`, z=lt`math.dtx`, A=lt`lists.dtx`, B=lt`boxes.dtx`, C=lt`tab.dtx`,
D=lt`pictur.dtx`, E=lt`thm.dtx`, F=lt`tsect.dtx`, G=lt`float.dtx`, H=lt`idxglo.dtx`,
I=lt`bibl.dtx`, J=lt`page.dtx`, K=lt`output.dtx`, L=lt`class.dtx`, M=lt`hyphen.dtx`,
N=lt`final.dtx`, O=lt`patch.ltx`
- `\default@mextra` [q1](#), [q79](#)
- `\default@series`
 [o104](#), [o137](#), [o376](#), [o379](#), [o394](#)
- `\default@shape`
 [o105](#), [o138](#), [o377](#), [o380](#), [o392](#)
- `\default@T` [o145](#), [o148](#), [o158](#), [o220](#)
- `\defaultthyphenchar` [b121](#)
- `\defaultscriptratio` [o455](#), [o466](#)
- `\defaultscriptscriptratio` [o456](#), [o466](#)
- `\defaultskewchar` [b122](#)
- `\define@mathalphabet` [q1](#), [q121](#)
- `\define@mathgroup` [q1](#), [q125](#)
- `\define@newfont` [o272](#), [o282](#)
- `\deg` [z34](#)
- `\delcode` [r707](#)
- `\delimiter` [r638](#), [r703](#)
- `\delimiterfactor` [b123](#)
- `\delimitershortfall` [b133](#)
- `\Delta` [t217](#)
- `\delta` [t190](#)
- `\depth` [B16](#), [B19](#)
- `\det` [z30](#)
- `\DH` [l391](#), [N179](#)
- `\dh` [l401](#), [N179](#)
- `\Diamond` [s93](#)
- `\diamond` [t294](#)
- `\diamondsuit` [t256](#)
- `\dim` [z28](#)
- `\dimen@` [b41](#), [b190](#),
 b191, [b227](#), [b228](#), [b230](#), [b232](#),
 [g31](#), [g32](#), [i147](#), [i152](#), [l328](#), [l329](#),
 [l331](#), [l332](#), [l640](#), [l641](#), [l956](#), [l958](#),
 [o185](#), [o188](#), [o192](#), [o454](#), [o455](#),
 [o456](#), [o460](#), [p423](#), [p424](#), [p425](#),
 [p426](#), [p430](#), [z72](#), [z73](#), [z129](#), [z130](#),
 [z131](#), [z132](#), [B281](#), [B284](#), [C158](#),
 [C159](#), [K386](#), [K388](#), [K413](#), [K415](#)
- `\dimen@i` [b41](#)
- `\dimen@ii` [b41](#)
- `\dimendef` [b42](#), [b43](#), [b44](#), [b48](#)
- `\discretionary` [d10](#), [z148](#)
- `\displ@y` [z134](#), [z138](#), [z139](#)
- `\displaylines` [z133](#)
- `\displaymath` [z195](#)
- `displaymath` (environment) [z193](#)
- `\displaystyle` [t440](#), [t443](#),
 [t446](#), [t448](#), [z62](#), [z140](#), [z219](#),
 [z222](#), [z259](#), [z283](#), [z299](#), [z323](#), [z326](#)
- `\displaywidowpenalty` [b113](#)
- `\displaywidth` [z140](#), [z218](#), [z271](#), [z302](#)
- `\div` [t297](#)
- `\DJ` [l392](#), [N179](#)
- `\dj` [l402](#), [N179](#)

- `\do` I16, I41,
a21, a22, a73, b13, b14, d36, f3,
f7, f16, f26, k56, k59, k99, k151,
k210, k216, o310, o311, o312,
o313, o314, o315, o316, o317,
o318, o319, o320, o321, o322,
o323, o324, o325, o326, v72,
y113, y134, y145, y151, B36,
C214, C239, D39, D44, D103,
D206, D208, D228, D231, D266,
D380, G61, L78, L150, L164,
L176, L181, L196, L400, L457, L516
`\do@noligs` y146, y151
`\do@subst@correction` . o58, p454, p527
`\DocInput` p8, t5, u5, M4
`\document` I40, k11, 72
`\document@select@group` r98, r160
`\documentclass` .. p2, t2, u2, L200,
L207, L234, L237, L325, L420, M2
`\documentstyle` L205, L420
`\dorestore@version` r70, r80
`\dospecials`
.. a21, a73, b13, y113, y134, L457
`\dot` t433
`\doteq` t377
`\dotfill` b234
`\dots` l274, l276
`\doublehyphendemerits` b116
`\doublerulesep` C279, C306, C330
`\Downarrow` t490
`\downarrow` t484
`\downbracefill` t445, t463
`\ds@` L145, L361
`\dt@pfalse` z135
`\dt@ptrue` z134
`\dump` N243
- E**
- `\E` L458, L461, L488
`\egroup` b171
`\eject` b201
`\ell` t231
`\em` s30, s31, v25
`\emergencystretch` b96, J45, J51
`\emph` v25
`\empty` b169
`\empty@sfcnt` o321,
p464, p465, p468, p486, p495, p539
`\emptyset` t238
`\enc@update` o205, o207, o223, o226, p129
`\encodingdefault`
..... l845, l871, r161, s80, t38
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- `\end` a16, d8, d251, g235, p9, t6,
u6, y60, y97, y98, z290, A112,
F15, F17, L466, L470, L476, M5
`\end@dblfloat` G130
`\end@float` G114, G139, G238
`\endarray` C153
`\endcenter` y74
`\enddisplaymath` z196
`\enddocument` y8
`\endenumerate` A230
`\endeqnarray` z227, z258
`\endequation` z199
`\endfilecontents` L424
`\endflushleft` y81
`\endflushright` y87
`\endgraf` b166
`\enditemize` A241
`\endline` b166, z118
`\endlinechar` .. a39, a40, a41, a151,
d23, d25, k182, L105, L106, L107
`\endlist` A98, A230, A241
`\endlrbox` B80
`\endmath` z194
`\endminipage` B222
`\endpicture` D25
`\endsloppypar` J49
`\endtabbing` C75
`\endtabular` C153
`\endtabular*` C153
`\endtrivlist` y74, y81, y87,
y119, z304, A100, A101, C76, E39
`\endverbatim` y118, y122
`\enlargethispage` K1196
`\enlargethispage*` K1196
`\enskip` i201
`\enspace` i198
`\ensuremath` m58, z260, G260
`\enumerate` A221
enumerate (environment) A221
environments:
center y73
displaymath z193
enumerate A221
eqnarray z205, z305
eqnarray* z256
equation z197, z293
filecontents 422
flushleft y80
flushright y86
itemize A232
lrbox 271
math z193
minipage 272

- sloppypar [J48](#)
thebibliography [359](#)
verbatim* [y121](#)
\epsilon [t191](#)
\eqnarray [z210](#), [z257](#)
eqnarray (environment) [z205](#), [z305](#)
eqnarray* (environment) [z256](#)
\eqno [z199](#)
\equation [z198](#)
equation (environment) [z197](#), [z293](#)
\equiv [t356](#)
\err@rel@i [q89](#), [q122](#), [q126](#)
\errhelp [a164](#),
 [c23](#), [g42](#), [g69](#), [M12](#), [N80](#), [N234](#)
\errmessage
 [a4](#), [a169](#), [b74](#), [c24](#), [g50](#), [g75](#),
 [o389](#), [p385](#), [p515](#), [q55](#), [M16](#), [N82](#)
\error@fontshape
 [o370](#), [o390](#), [p107](#), [p517](#), [r146](#)
\errorcontextlines
 [b94](#), [b126](#), [b252](#), [g181](#)
\errorstopmode [b244](#), [N242](#)
\escapechar . [d95](#), [d140](#), [d145](#), [d154](#),
 [o286](#), [o420](#), [p183](#), [r45](#), [r104](#), [r145](#)
\eta [t193](#)
\evensidemargin [K54](#), [K492](#)
\every@math@size [o51](#), [p189](#), [p201](#)
\everycr .. [b222](#), [z135](#), [z138](#), [z218](#), [z320](#)
\everydisplay [o262](#), [o263](#), [o268](#)
\everyjob . [c28](#), [r165](#), [N204](#), [N205](#), [N207](#)
\everymath [o261](#), [o263](#), [o266](#)
\everypar ... [k37](#), [o477](#), [y50](#), [y116](#),
 [57](#), [A126](#), [A127](#), [A170](#), [A187](#),
 [B183](#), [C72](#), [F31](#), [F79](#), [F90](#),
 [F110](#), [F119](#), [G112](#), [K142](#), [K890](#)
\execute@size@function
 [p318](#), [p346](#), [p364](#), [p381](#)
\ExecuteOptions . [l904](#), [p57](#), [p70](#), [L194](#)
\exhyphenpenalty [b108](#), [b195](#)
\exists [t249](#)
\exp [z31](#)
\external@font [p84](#),
 [p87](#), [p98](#), [p102](#), [p104](#), [p347](#),
 [p365](#), [p439](#), [p477](#), [p549](#), [p551](#), [p553](#)
\extra@def [q1](#), [q74](#)
\extracolsep [C149](#)
\extract@alph@from@version
 [o426](#), [o432](#), [r108](#)
\extract@font [o297](#), [p81](#)
\extract@fontinfo [p314](#), [p321](#)
\extract@range@fontinfo
 [o314](#), [p331](#), [p338](#), [p361](#), [p394](#)
\extract@size@fn [o311](#), [p305](#), [p328](#)
File Key: a=lt`dirchk`.dtx, b=lt`plain`.dtx, c=lt`vers`.dtx, d=lt`defs`.dtx,
e=lt`alloc`.dtx, f=lt`cntrl`.dtx, g=lt`error`.dtx, h=lt`par`.dtx, i=lt`space`.dtx,
j=lt`logos`.dtx, k=lt`files`.dtx, l=lt`outenc`.dtx, m=lt`counts`.dtx, n=lt`length`.dtx,
o=lt`fssbas`.dtx, p=lt`fssstrc`.dtx, q=lt`fsscmp`.dtx, r=lt`fssdcl`.dtx, s=lt`fssini`.dtx,
t=lt`fontdef`.dtx, u=lt`preload`.dtx, v=lt`fontcmd`.dtx, w=lt`pageno`.dtx, x=lt`txref`.dtx,
y=lt`miscen`.dtx, z=lt`math`.dtx, A=lt`lists`.dtx, B=lt`boxes`.dtx, C=lt`ttab`.dtx,
D=lt`pictur`.dtx, E=lt`thm`.dtx, F=lt`tsect`.dtx, G=lt`float`.dtx, H=lt`tidxglo`.dtx,
I=lt`bibl`.dtx, J=lt`page`.dtx, K=lt`output`.dtx, L=lt`class`.dtx, M=lt`hyphen`.dtx,
N=lt`final`.dtx, O=lt`patch`.ltx
- F**
- \f@baselineskip
 [o234](#), [o241](#), [o382](#), [p119](#),
 [p136](#), [p140](#), [p155](#), [p169](#), [p180](#), [p194](#)
\f@encoding . [l133](#), [o199](#), [o218](#), [o221](#),
 [o222](#), [o224](#), [o243](#), [o275](#), [o280](#),
 [o333](#), [o335](#), [o337](#), [o342](#), [o344](#),
 [o374](#), [o386](#), [p91](#), [p261](#), [p507](#), [r131](#)
\f@family [l921](#), [l924](#), [l938](#),
 [l948](#), [l954](#), [l1109](#), [o227](#), [o237](#),
 [o276](#), [o280](#), [o333](#), [o335](#), [o337](#),
 [o342](#), [o344](#), [o378](#), [o397](#), [p91](#), [r131](#)
\f@linespread ... [o237](#), [p118](#), [p137](#),
 [p138](#), [p141](#), [p149](#), [p152](#), [p163](#), [p166](#)
\f@series [j14](#), [o227](#),
 [o238](#), [o277](#), [o280](#), [o379](#), [o394](#), [s67](#)
\f@shape
 [o227](#), [o239](#), [o278](#), [o280](#), [o380](#), [o392](#)
\f@size [l135](#), [l956](#), [o63](#),
 [o234](#), [o240](#), [o279](#), [o381](#), [o411](#),
 [o453](#), [o454](#), [o457](#), [o458](#), [p119](#),
 [p121](#), [p134](#), [p154](#), [p169](#), [p172](#),
 [p175](#), [p180](#), [p187](#), [p194](#), [p206](#),
 [p209](#), [p215](#), [p221](#), [p238](#), [p239](#),
 [p242](#), [p247](#), [p315](#), [p322](#), [p341](#),
 [p343](#), [p362](#), [p425](#), [p427](#), [p429](#),
 [p445](#), [p446](#), [p451](#), [p469](#), [p485](#),
 [p494](#), [p512](#), [p520](#), [p525](#), [p537](#), [p555](#)
\f@user@size .. [p445](#), [p450](#), [p512](#), [p525](#)
\fam [o24](#)
?familydefault [r162](#), [s81](#), [t38](#)
\fbox [271](#), [B98](#), [B107](#)
\fboxrule [B96](#), [B118](#),
 [B121](#), [B127](#), [B129](#), [B136](#), [B137](#), [N8](#)
\fboxsep [B96](#),
 [B102](#), [B117](#), [B122](#), [B132](#), [B134](#), [N7](#)
\filbreak [b199](#)
\filecontents
 ... [L429](#), [L430](#), [L431](#), [L510](#), [L516](#)
\filecontents [L424](#)
filecontents (environment) [422](#)
\filename@area [a193](#), [a199](#),
 [a206](#), [a212](#), [a219](#), [a225](#), [a232](#),
 [k168](#), [k193](#), [k196](#), [k212](#), [k224](#), [k226](#)
\filename@base
 [a241](#), [k168](#), [k193](#), [k196](#), [k219](#), [k224](#)
\filename@dot [a239](#), [a242](#)
\filename@ext [a237](#), [a239](#),
 [k169](#), [k189](#), [k190](#), [k193](#), [k196](#), [k220](#)
\filename@parse
 . [1](#), [5](#), [a57](#), [a189](#), [k166](#), [k188](#), [k217](#)
\filename@path .. [a194](#), [a195](#), [a200](#),
 [a207](#), [a208](#), [a213](#), [a220](#), [a221](#), [a226](#)

- \filename@simple [a197](#), [a210](#), [a223](#), [a233](#), [a235](#)
 - \fill [i195](#)
 - \finalhyphemerits [b117](#)
 - \finph@nt [z87](#), [z89](#), [z90](#)
 - \finsm@sh [z103](#), [z105](#), [z106](#)
 - \firstmark [J37](#), [K526](#)
 - \fix@penalty [v83](#)
 - \fixed@sfcnt .. [o326](#), [p543](#), [p544](#), [p547](#)
 - \flat [t252](#)
 - \floatingpenalty [G275](#)
 - \floatpagefraction [G176](#), [K1520](#)
 - \floatsep [K544](#),
[K562](#), [K569](#), [K1400](#), [K1449](#), [K1525](#)
 - \flushbottom [J41](#)
 - \flushleft [y80](#)
 - flushleft (environment) [y80](#)
 - \flushright [y86](#)
 - flushright (environment) [y86](#)
 - \fmtname [c1](#), [c28](#), [c31](#), [L243](#), [L247](#)
 - \fmtversion
 . [c1](#), [c11](#), [c30](#), [c33](#), [g1](#), [o1](#), [C1](#),
 [D1](#), [K4](#), [L260](#), [L263](#), [N188](#), [N214](#)
 - \fmtversion@topatch [N186](#),
 [N188](#), [N200](#), [N201](#), [N213](#), [N221](#), [O5](#)
 - \fnsymbol [m39](#), [i21](#)
 - \font [b227](#), [b232](#),
 [l251](#), [l252](#), [l253](#), [l338](#), [l345](#), [l643](#),
 [l650](#), [o54](#), [o62](#), [o64](#), [p84](#), [s31](#),
 [s54](#), [s66](#), [u8](#), [u9](#), [u10](#), [v67](#), [y115](#)
 - \font@info [p99](#), [p321](#), [p394](#), [p401](#)
 - \font@name [l134](#), [l137](#), [o61](#), [o169](#), [o171](#),
 [o271](#), [o288](#), [o410](#), [p84](#), [p88](#),
 [p90](#), [p105](#), [p120](#), [p123](#), [p126](#),
 [p284](#), [p285](#), [p286](#), [p287](#), [p288](#), [p293](#)
 - \font@submax [p407](#), [p442](#),
 [p443](#), [y22](#), [y24](#), [N72](#), [N74](#), [N83](#)
 - \fontdimen [b227](#), [b232](#),
 [l251](#), [l252](#), [l253](#), [l338](#), [l345](#), [l643](#),
 [l650](#), [s31](#), [s66](#), [v67](#), [D48](#), [D50](#), [D334](#)
 - \fontencoding [l871](#), [o199](#), [o230](#), [r161](#), [t14](#)
 - \fontfamily [l940](#), [o227](#), [r162](#), [s5](#), [s8](#), [s11](#)
 - \fontname [o64](#)
 - \fontseries [o227](#), [r163](#), [s14](#), [s17](#)
 - \fontshape [l348](#),
 [l653](#), [o227](#), [r164](#), [s20](#), [s23](#), [s26](#), [s29](#)
 - \fontsize [j6](#),
 [l256](#), [l958](#), [o52](#), [o235](#), [s60](#), [G260](#)
 - \fontsubfuzz [p407](#), [p447](#), [y22](#)
 - \footins [G245](#),
 [G271](#), [K264](#), [K265](#), [K266](#), [K267](#),
 [K299](#), [K358](#), [K366](#), [K370](#), [K399](#)
 - \footnote [G262](#)
 - \footnotemark [F9](#), [G284](#)
 - File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
 - \footnoterule [B228](#), [G249](#), [K369](#)
 - \footnotesep . [B247](#), [G261](#), [G274](#), [G282](#)
 - \footnotesize [B240](#), [G272](#)
 - \footnotetext [F11](#), [G301](#)
 - \footskip [K58](#), [K516](#)
 - \forall [t248](#)
 - \fps@dbl [G34](#)
 - \frac [z202](#)
 - \frame [B82](#), [B144](#)
 - \framebox [271](#), [B105](#)
 - \frenchspacing . [b152](#), [k40](#), [y118](#), [y144](#)
 - \frown [t359](#)
 - \frozen@everydisplay [o261](#), [o267](#)
 - \frozen@everymath [o261](#), [o265](#)
 - \fussy [J50](#)
 - \futurelet [d257](#),
 [d271](#), [i172](#), [i180](#), [v65](#), [z153](#), [C327](#)
- ## G
- \g@addto@macro [L368](#), [L374](#), [L378](#), [L379](#)
 - \G@refundefinedfalse [x5](#)
 - \G@refundefinedtrue . [l21](#), [l44](#), [x3](#), [x12](#)
 - \Gamma [t216](#)
 - \gamma [t189](#)
 - \gcd [z33](#)
 - \ge [t330](#)
 - \gen@sfcnt ... [o322](#), [p480](#), [p481](#), [p484](#)
 - \genb@sfcnt ... [o323](#), [p489](#), [p490](#), [p493](#)
 - \genb@x [p494](#), [p497](#)
 - \genb@y [p497](#)
 - \GenericError [g19](#), [g91](#), [g122](#), [g153](#), [p62](#)
 - \GenericInfo [g4](#),
 [g112](#), [g143](#), [g173](#), [p31](#), [p34](#), [p39](#), [p75](#)
 - \GenericWarning [g11](#),
 [g102](#), [g133](#), [g164](#), [p42](#), [p47](#), [p50](#), [p78](#)
 - \geq [t329](#), [t330](#)
 - \get@cdp [r275](#), [r283](#), [r316](#)
 - \get@external@font ... [p83](#), [p96](#), [p526](#)
 - \getanddefine@fonts [o421](#),
 [o439](#), [p274](#), [r46](#), [r93](#), [r105](#),
 [r187](#), [r246](#), [r280](#), [r282](#), [r299](#),
 [r422](#), [r423](#), [r455](#), [r456](#), [r802](#), [r803](#)
 - \GetFileInfo [t3](#)
 - \getlinechar [D129](#)
 - \gets [t348](#)
 - \gg [t343](#)
 - \glb@currsz [k35](#),
 [o258](#), [p171](#), [p206](#), [p210](#), [p216](#), [p239](#)
 - \glb@settings . [o259](#), [p171](#), [p218](#), [p249](#)
 - \globaldefs [o422](#), [p185](#), [r47](#), [r106](#)
 - \glossary [357](#),
 [F146](#), [H23](#), [H35](#), [J20](#), [J28](#), [K500](#)
 - \glossaryentry [H32](#)

- \goodbreak [b199](#)
- \grave [t425](#)
- \group@elt [r26](#),
[r185](#), [r217](#), [r218](#), [r239](#), [r243](#), [r834](#)
- \group@list
.. [r189](#), [r224](#), [r237](#), [r242](#), [r243](#),
[r272](#), [r494](#), [r536](#), [r616](#), [r619](#),
[r669](#), [r672](#), [r719](#), [r722](#), [r789](#), [r840](#)
- \guillemotleft [l403](#), [l620](#)
- \guillemotright [l404](#), [l621](#)
- \guilsinglleft [l405](#)
- \guilsinglright [l406](#)
- H**
- \H [g27](#), [l185](#), [l287](#),
[l365](#), [l458](#), [l466](#), [l485](#), [l493](#), [l600](#)
- \h@false [z77](#)
- \h@true [z78](#), [z79](#)
- \halign ... [b222](#), [z96](#), [z140](#), [z218](#), [z320](#)
- \hangindent [F122](#)
- \hat [t431](#)
- \hb@xt@ [b237](#), [d16](#),
[l324](#), [z140](#), [z223](#), [z269](#), [z283](#),
[z298](#), [z328](#), [B28](#), [B43](#), [B116](#),
[B289](#), [B293](#), [B294](#), [C39](#), [D21](#),
[D31](#), [D40](#), [D143](#), [D177](#), [D180](#),
[D183](#), [D185](#), [D187](#), [D262](#), [D291](#),
[D390](#), [F163](#), [F166](#), [K509](#), [K519](#),
[K1181](#), [K1479](#), [K1480](#), [K1485](#)
- \hbadness [b104](#), [o480](#)
- \hbar [t228](#)
- \headheight [K56](#), [K505](#)
- \headsep [K57](#), [K514](#)
- \heartsuit [t257](#)
- \height [B15](#), [B18](#)
- \hexnumber@ [r509](#),
[r517](#), [r532](#), [r551](#), [r559](#), [r567](#),
[r576](#), [r579](#), [r588](#), [r589](#), [r628](#),
[r636](#), [r681](#), [r689](#), [r703](#), [r704](#),
[r707](#), [r732](#), [r740](#), [r745](#), [r747](#), [s71](#)
- \hfuzz [b127](#), [J46](#), [J47](#), [J53](#), [J54](#)
- \hgl@ [b192](#), [b193](#)
- \hglue [b189](#)
- \hideskip [b79](#), [b213](#)
- \hidewidth
.. [b213](#), [l292](#), [l293](#), [l296](#), [l299](#),
[l372](#), [l373](#), [l377](#), [l380](#), [l382](#), [l384](#),
[l607](#), [l608](#), [l611](#), [l614](#), [l678](#), [l681](#)
- \hline [C326](#), [C329](#)
- \hmode@bgroup [l72](#),
[l78](#), [l292](#), [l298](#), [l326](#), [l337](#), [l344](#),
[l372](#), [l379](#), [l382](#), [l384](#), [l577](#), [l607](#),
[l613](#), [l642](#), [l649](#), [l677](#), [l680](#), [l726](#), [v7](#)
- File Key:** a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- \hmode@start@before@group
..... [l73](#), [l116](#), [l118](#), [l124](#), [l139](#)
- \holdinginserts [b95](#)
- \hom [z29](#)
- \hookleftarrow [t388](#)
- \hookrightarrow [t386](#)
- \hphantom [z75](#)
- \hrule [b190](#),
[b234](#), [i148](#), [i156](#), [l245](#), [l248](#),
[t246](#), [t522](#), [B88](#), [B93](#), [B127](#),
[B137](#), [C327](#), [C344](#), [D293](#), [G250](#)
- \hrulefill [b234](#)
- \hspace [l191](#)
- \hyphenation [l160](#)
- \hyphenchar [y115](#)
- \hyphenpenalty [b107](#)
- I**
- \I [b158](#), [L484](#), [L502](#), [N49](#), [N154](#)
- \i [l202](#),
[l304](#), [l351](#), [l352](#), [l353](#), [l354](#), [l355](#),
[l356](#), [l407](#), [l444](#), [l445](#), [l537](#), [l539](#),
[l541](#), [l543](#), [l622](#), [N54](#), [N159](#), [N166](#)
- \ialign [b222](#), [b224](#),
[t243](#), [t367](#), [t438](#), [t441](#), [t444](#),
[t447](#), [z109](#), [z111](#), [z119](#), [C173](#), [D72](#)
- \if@afterindent [F107](#), [F114](#)
- \if@compatibility [L2](#), [L202](#)
- \if@endpe [y62](#), [A128](#)
- \if@eqnsw [z205](#), [z254](#)
- \if@fcolmade
[K76](#), [K212](#), [K318](#), [K327](#), [K597](#),
[K614](#), [K641](#), [K695](#), [K1463](#), [K1500](#)
- \if@filesw [I4](#), [I8](#),
[I19](#), [I28](#), [I36](#), [I43](#), [k7](#), [k30](#), [k92](#),
[k104](#), [k111](#), [k120](#), [y14](#), [y28](#), [F136](#)
- \if@firstamp [C221](#)
- \if@firstcolumn [K76](#),
[K194](#), [K227](#), [K320](#), [K1153](#), [K1470](#)
- \if@ignore [y4](#), [y63](#)
- \if@inlabel
[A28](#), [A65](#), [A102](#), [A150](#), [A173](#), [K138](#)
- \if@insert [K76](#),
[K799](#), [K908](#), [K942](#), [K1012](#), [K1128](#)
- \if@minipage [i102](#),
[i115](#), [y101](#), [A139](#), [B191](#), [C70](#), [G20](#)
- \if@mparswitch [K76](#), [K1155](#)
- \if@multiplelabels [x31](#)
- \if@negarg [D76](#), [D98](#), [D112](#), [D151](#)
- \if@newlist [y119](#), [A29](#), [A33](#),
[A69](#), [A78](#), [A106](#), [A156](#), [K476](#), [K523](#)
- \if@nmbrlist [A33](#), [A191](#)
- \if@no@font@opt [q100](#), [q119](#)

`\if@nobreak` [i58](#), [i117](#), [k67](#), [k79](#), [A157](#),
[A182](#), [B177](#), [F30](#), [F111](#), [G105](#),
[G228](#), [J25](#), [J33](#), [K142](#), [K287](#), [K887](#)
`\if@noitemarg` [A32](#), [A189](#)
`\if@noparitem` [A30](#), [A147](#)
`\if@noparlist` [A31](#), [A114](#)
`\if@noskipsec` [A58](#),
[B178](#), [F21](#), [F23](#), [F80](#), [G229](#), [K132](#)
`\if@ovb` [D233](#), [D279](#), [D284](#)
`\if@ovl` [D233](#), [D277](#), [D294](#)
`\if@ovr` [D233](#), [D276](#), [D292](#)
`\if@ovt` [D233](#), [D278](#), [D288](#)
`\if@partsw` [k7](#), [k96](#)
`\if@pboxsw` [B174](#), [B249](#)
`\if@reversemargin` [K82](#), [K1158](#)
`\if@reversemarginpar` [K76](#)
`\if@rjfield` [C21](#), [C35](#)
`\if@specialpage` [K76](#), [K485](#)
`\if@tempswa` [I52](#), [a25](#),
[a26](#), [a27](#), [e9](#), [k102](#), [o74](#), [r210](#),
[r260](#), [r324](#), [r405](#), [r833](#), [v73](#), [y30](#),
[y107](#), [K732](#), [K768](#), [K1093](#), [L447](#)
`\if@test` .. [K12](#), [K13](#), [K677](#), [K710](#),
[K774](#), [K855](#), [K864](#), [K993](#), [K1098](#)
`\if@twocolumn` ... [k20](#), [G32](#), [G131](#),
[K76](#), [K120](#), [K215](#), [K226](#), [K319](#),
[K343](#), [K599](#), [K632](#), [K1152](#), [K1464](#)
`\if@twoside` [K76](#), [K119](#), [K488](#)
`\ifdt@p` [z133](#), [z135](#)
`\iff` [t408](#)
`\IfFileExists` [a125](#),
[k134](#), [k161](#), [k172](#), [72](#), [424](#), [N182](#)
`\ifG@refundefined` [x3](#), [x4](#), [x5](#)
`\ifh@` [z76](#), [z93](#)
`\ifin@` [l861](#), [l864](#), [q40](#), [q42](#), [r1](#),
[r13](#), [r174](#), [r271](#), [r273](#), [r334](#), [r347](#),
[r417](#), [r419](#), [r447](#), [r495](#), [r506](#),
[r537](#), [r548](#), [r617](#), [r620](#), [r640](#),
[r670](#), [r673](#), [r717](#), [r720](#), [r723](#),
[r790](#), [r792](#), [r821](#), [L80](#), [L155](#), [L167](#)
`\ifinner` [z169](#), [z183](#), [G53](#), [G194](#)
`\ifmath@fonts` [o179](#), [p176](#)
`\ifnot@nil` [p297](#), [p316](#), [p337](#)
`\ifodd` [r764](#), [D192](#), [D212](#), [G64](#), [K21](#),
[K119](#), [K489](#), [K724](#), [K727](#), [K760](#),
[K763](#), [K871](#), [K874](#), [K1041](#),
[K1044](#), [K1156](#), [K1335](#), [K1343](#)
`\iftc@forced` [l896](#), [l906](#), [l1115](#)
`\ifv@` [z75](#), [z92](#)
`\ifvbox` [K270](#), [K300](#), [K379](#), [K400](#)
`\ignorespaces` .. [I7](#), [I9](#), [i24](#), [i74](#), [i83](#),
[i207](#), [k60](#), [o232](#), [y63](#), [y71](#), [y72](#),
[z191](#), [A55](#), [A207](#), [B79](#), [B247](#),
[C59](#), [C60](#), [C61](#), [C74](#), [C83](#), [C96](#),
[C100](#), [C107](#), [C114](#), [C116](#), [C125](#),
[C207](#), [C269](#), [C271](#), [C273](#), [C300](#),
[D24](#), [D32](#), [D43](#), [D74](#), [D75](#),
[E30](#), [E32](#), [F93](#), [G17](#), [G24](#), [G282](#)
`\ignorespacesafterend` [y7](#)
`\IJ` [l205](#), [l335](#), [l410](#)
`\ij` [l204](#), [l333](#), [l409](#)
`\Im` [t234](#)
`\imath` [t229](#)
`\in` [t340](#), [t369](#)
`\in@` [l859](#), [l862](#), [q39](#), [q41](#), [r1](#),
[r12](#), [r173](#), [r270](#), [r272](#), [r330](#), [r343](#),
[r416](#), [r418](#), [r445](#), [r493](#), [r504](#),
[r535](#), [r546](#), [r615](#), [r618](#), [r638](#),
[r668](#), [r671](#), [r715](#), [r718](#), [r721](#),
[r788](#), [r791](#), [r819](#), [L79](#), [L152](#), [L166](#)
`\in@@` [r3](#), [r5](#)
`\in@false` [r4](#)
`\in@true` [r4](#)
`\include` [k86](#), [72](#)
`\includeonly` [k82](#), [72](#)
`\indent` [A151](#), [C72](#)
`\index` [357](#), [F146](#), [H6](#), [H18](#), [J20](#), [J28](#), [K499](#)
`\indexentry` [H15](#)
`\inf` [z25](#)
`\infty` [t236](#)
`\init@restore@glb@settings`
..... [p219](#), [p222](#), [p224](#)
`\init@restore@version` [r49](#), [r64](#), [r84](#), [r85](#)
`\input` [a15](#), [a121](#), [a124](#),
[a181](#), [d7](#), [k163](#), [l1094](#), [p16](#), [q8](#),
[q96](#), [s131](#), [s142](#), [s152](#), [t10](#), [t11](#),
[t12](#), [t13](#), [t20](#), [t21](#), [t25](#), [t26](#),
[t55](#), [t56](#), [t57](#), [t58](#), [t540](#), [t541](#),
[t542](#), [y19](#), [72](#), [425](#), [L206](#), [N69](#), [N187](#)
`\input@path` [1](#), [5](#), [a56](#),
[a78](#), [a80](#), [a86](#), [a88](#), [a94](#), [a96](#),
[a101](#), [a103](#), [a113](#), [a180](#), [k137](#), [k151](#)
`\InputIfFileExists`
..... [k160](#), [k165](#), [k173](#), [k192](#),
[l849](#), [l1162](#), [o342](#), [s105](#), [s123](#),
[s134](#), [s144](#), [72](#), [424](#), [L331](#), [M8](#), [N63](#)
`\inputlineno` [a250](#), [b86](#),
[b87](#), [b88](#), [g183](#), [g186](#), [s104](#), [N34](#),
[N45](#), [N53](#), [N128](#), [N139](#), [N150](#), [N158](#)
`\insec@unt` [b37](#), [b47](#), [b48](#), [b49](#),
[b51](#), [b62](#), [b63](#), [b64](#), [b65](#), [b66](#), [b67](#)
`\insert` [b69](#), [G271](#), [K399](#), [K400](#), [K1230](#)
`\install@mathalphabet`
. [o416](#), [o433](#), [o440](#), [r193](#), [r196](#),
[r277](#), [r278](#), [r375](#), [r427](#), [r430](#),
[r437](#), [r452](#), [r453](#), [r460](#), [r804](#), [r806](#)
`\int` [t266](#)
File Key: `a=ltldirchk.dtx`, `b=ltplain.dtx`, `c=ltvers.dtx`, `d=ltdefns.dtx`,
`e=ltalloc.dtx`, `f=ltcntrl.dtx`, `g=lterror.dtx`, `h=ltpar.dtx`, `i=ltspace.dtx`,
`j=ltlogos.dtx`, `k=ltfiles.dtx`, `l=ltoutenc.dtx`, `m=ltcounts.dtx`, `n=ltlength.dtx`,
`o=ltfssbas.dtx`, `p=ltfssstrc.dtx`, `q=ltfsscmp.dtx`, `r=ltfssdcl.dtx`, `s=ltfssini.dtx`,
`t=fontdef.dtx`, `u=preload.dtx`, `v=ltfntcmd.dtx`, `w=ltpageno.dtx`, `x=ltxref.dtx`,
`y=ltmiscen.dtx`, `z=ltmath.dtx`, `A=ltlists.dtx`, `B=ltboxes.dtx`, `C=lttab.dtx`,
`D=ltpictur.dtx`, `E=ltthm.dtx`, `F=ltsect.dtx`, `G=ltfloat.dtx`, `H=ltidxglo.dtx`,
`I=ltbibl.dtx`, `J=ltpage.dtx`, `K=ltoutput.dtx`, `L=ltclass.dtx`, `M=lthyphen.dtx`,
`N=ltfinal.dtx`, `O=ltpatch.ltx`

- `\interdisplaylinepenalty` [i29](#), [z55](#), [z137](#), [z240](#)
`\interfootlinepenalty` [b148](#)
`\interfootnotelinepenalty` [b148](#), [i34](#), [G273](#)
`\interlinepenalty` [i27](#), [y108](#), [y111](#), [F50](#), [F101](#), [F154](#), [G273](#), [K290](#), [K892](#), [K896](#)
`\intextsep` [K875](#), [K879](#), [K894](#), [K897](#), [K904](#), [K1525](#)
`\intop` [t265](#), [t266](#)
`\iota` [t195](#)
`\is@range` [o315](#), [p332](#), [p333](#)
`\ishortstack` [D62](#)
`\itdefault` [s29](#), [t34](#)
`\item` [I4](#), [I8](#), [g277](#), [y73](#), [y80](#), [y86](#), [y100](#), [z282](#), [z297](#), [A131](#), [A209](#), [C69](#), [E36](#), [E38](#)
`\itemindent` . [A9](#), [A42](#), [A95](#), [A177](#), [A198](#)
`\itemize` [A232](#)
`itemize (environment)` [A232](#)
`\itemsep` [A1](#), [A166](#)
`\iterate` [a28](#), [a29](#), [b178](#)
`\itshape` [l346](#), [l651](#), [s27](#), [s28](#), [s32](#), [v21](#), [E36](#), [E38](#), [G254](#)
- ### J
- `\J` [N51](#), [N156](#)
`\j` [l203](#), [l305](#), [l408](#), [l623](#), [N166](#)
`\jmath` [t230](#)
`\Join` [s91](#)
`\joinrel` [t379](#), [t386](#), [t388](#), [t390](#), [t392](#), [t394](#), [t396](#), [t398](#), [t400](#), [t404](#), [t406](#)
`\jot` [z53](#), [z134](#), [z247](#)
- ### K
- `\k` [l381](#), [l447](#), [l452](#), [l474](#), [l479](#), [l555](#), [l556](#), [l604](#), [l605](#), [l656](#), [l658](#), [l663](#), [l665](#)
`\kappa` [t196](#)
`\ker` [z27](#)
`\kernel@ifnextchar` [d48](#), [d68](#), [d119](#), [d258](#), [d275](#), [L112](#)
`\kill` [C61](#)
- ### L
- `\L` [l197](#), [l323](#), [l393](#), [l618](#), [L481](#), [L501](#), [N179](#)
`\l` [l206](#), [l325](#), [l411](#), [l624](#), [N179](#)
`\l@ngrel@x` .. [d41](#), [d42](#), [d43](#), [d89](#), [d138](#)
`\label` [x32](#), [F146](#), [J20](#), [J28](#), [K498](#)
`\labelsep` .. [A9](#), [A200](#), [A206](#), [E36](#), [E38](#)
`\labelwidth` [A9](#), [A93](#), [A199](#), [A201](#), [A204](#)
`\Lambda` [t219](#)
`\lambda` [t197](#)
File Key: a=lt`dirchk`.dtx, b=lt`plain`.dtx, c=lt`vers`.dtx, d=lt`defs`.dtx, e=lt`alloc`.dtx, f=lt`cntrl`.dtx, g=lt`error`.dtx, h=lt`par`.dtx, i=lt`space`.dtx, j=lt`logos`.dtx, k=lt`files`.dtx, l=lt`outenc`.dtx, m=lt`counts`.dtx, n=lt`length`.dtx, o=lt`tfssbas`.dtx, p=lt`tfssstrc`.dtx, q=lt`tfsscmp`.dtx, r=lt`tfssdcl`.dtx, s=lt`tfssini`.dtx, t=lt`fontdef`.dtx, u=lt`preload`.dtx, v=lt`fontcmd`.dtx, w=lt`pageno`.dtx, x=lt`xref`.dtx, y=lt`miscen`.dtx, z=lt`math`.dtx, A=lt`lists`.dtx, B=lt`boxes`.dtx, C=lt`tab`.dtx, D=lt`pictur`.dtx, E=lt`thm`.dtx, F=lt`ssect`.dtx, G=lt`float`.dtx, H=lt`idxglo`.dtx, I=lt`bibl`.dtx, J=lt`page`.dtx, K=lt`output`.dtx, L=lt`class`.dtx, M=lt`hyphen`.dtx, N=lt`final`.dtx, O=lt`patch`.ltx
- `\land` [t283](#)
`\langle` [t498](#)
`\language` ... [b35](#), [b56](#), [b89](#), [b90](#), [M10](#)
`\last@fontshape` [o388](#), [o403](#)
`\lastbox` [z123](#), [z124](#), [A127](#), [A175](#), [F82](#), [F115](#), [K255](#)
`\LastDeclaredEncoding` ... [o112](#), [o115](#)
`\lastpenalty` [v94](#), [v97](#)
`\lastskip` [b202](#), [b203](#), [b205](#), [b207](#), [i19](#), [i66](#), [i87](#), [i88](#), [i92](#), [i94](#), [i95](#), [i103](#), [i119](#), [i122](#), [i123](#), [v84](#), [v87](#), [A115](#), [A116](#), [A140](#), [A141](#), [D44](#)
`\LaTeX` [j3](#), [j15](#), [L450](#)
`\LaTeXe` [j13](#)
`\lbrace` [l262](#), [t502](#)
`\lbrack` [b162](#)
`\lccode` [g22](#), [g23](#), [g24](#), [g25](#), [g26](#), [g27](#), [l109](#), [y139](#), [y149](#), [N31](#), [N41](#), [N50](#), [N52](#), [N54](#), [N56](#), [N59](#), [N60](#), [N61](#), [N62](#), [N136](#), [N146](#), [N155](#), [N157](#), [N159](#), [N161](#), [N164](#)
`\lceil` [t506](#)
`\ldotp` [t409](#), [t412](#), [t523](#)
`\ldots` [l276](#), [t413](#)
`\le` [t328](#)
`\leaders` ... [b234](#), [t246](#), [t464](#), [t465](#), [t467](#), [t468](#), [C344](#), [D287](#), [D293](#), [F159](#)
`\leadsto` [s94](#)
`\leavevmode` . [l14](#), [b193](#), [b220](#), [b223](#), [b234](#), [b236](#), [i169](#), [i183](#), [l78](#), [l139](#), [l243](#), [l245](#), [l295](#), [l324](#), [l328](#), [l331](#), [l375](#), [l610](#), [l640](#), [l952](#), [v105](#), [y108](#), [y119](#), [y132](#), [y150](#), [z282](#), [z297](#), [A58](#), [A103](#), [B4](#), [B8](#), [B81](#), [B83](#), [B99](#), [B113](#), [B159](#), [B207](#), [B252](#), [B265](#), [C160](#), [D65](#), [D187](#), [F23](#), [F155](#), [G296](#), [K134](#), [K139](#)
`\left` [t524](#), [t525](#), [t526](#), [t527](#), [z108](#), [z114](#), [z125](#)
`\Leftarrow` [t324](#), [t400](#), [t406](#)
`\leftarrow` [t347](#), [t348](#), [t388](#), [t398](#), [t404](#), [t456](#)
`\leftarrowfill` [t442](#), [t456](#)
`\lefteqn` [z259](#)
`\leftharpoondown` [t361](#), [t375](#)
`\leftharpoonup` [t360](#)
`\lefthyphenmin` [b92](#), [M11](#)
`\leftline` [B289](#)
`\leftmargin` [A9](#), [A52](#), [A53](#), [A94](#), [A136](#), [A138](#)
`\leftmargini` [z275](#), [A17](#)
`\leftmarginii` [A17](#)
`\leftmarginiii` [A17](#)
`\leftmarginiv` [A17](#)

- `\leftmarginv` [A17](#)
 - `\leftmarginvi` [A17](#)
 - `\leftmark` [J34](#)
 - `\Leftrightarrow` [t323](#)
 - `\leftrightharrow` [t346](#)
 - `\leftskip` [b215](#), [y77](#), [y84](#),
[y90](#), [y102](#), [A74](#), [B186](#), [F152](#), [F157](#)
 - `\leq` [t327](#), [t328](#)
 - `\lfloor` [t510](#)
 - `\lg` [z4](#)
 - `\lggroup` [t512](#)
 - `\lhd` [s97](#)
 - `\lhook` [t385](#), [t386](#)
 - `\lim` [z6](#)
 - `\liminf` [z8](#)
 - `\limits` [t446](#), [t450](#), [z107](#), [z201](#)
 - `\limsup` [z7](#)
 - `\line` [g262](#), [D77](#), [D260](#)
 - `\linebreak` [i13](#), [59](#)
 - `\linepenalty` [b106](#)
 - `\lineskip`
 [b156](#), [b188](#), [b223](#), [t366](#), [z130](#),
 [B187](#), [C62](#), [C180](#), [D67](#), [D188](#), [K501](#)
 - `\lineskiplimit` .. [b157](#), [b188](#), [b225](#),
 [b226](#), [t366](#), [t418](#), [z132](#), [z136](#), [K501](#)
 - `\linespread` [o233](#)
 - `\linethickness` [D56](#)
 - `\linewidth` .. [k24](#), [z176](#), [z283](#), [z298](#),
 [z302](#), [z320](#), [A15](#), [A51](#), [A52](#),
 [A54](#), [B184](#), [C38](#), [G164](#), [K127](#), [K153](#)
 - `\list` [A34](#), [A226](#), [A237](#)
 - `\listfiles` [k206](#), [425](#)
 - `\listparindent` [A9](#), [A41](#), [A50](#)
 - `\ll` [t344](#)
 - `\llap` [A228](#), [A239](#), [B293](#), [B294](#)
 - `\lmoustache` [t469](#)
 - `\ln` [z5](#)
 - `\lnot` [t251](#)
 - `\LoadClass` [423](#),
 [L212](#), [L226](#), [L348](#), [L407](#), [L415](#), [L416](#)
 - `\LoadClassWithOptions` [423](#), [L225](#)
 - `\log` [z3](#)
 - `\loggingall` [b248](#)
 - `\loggingoutput` [b241](#), [b252](#)
 - `\Longleftarrow` [t400](#)
 - `\longleftarrow` [t397](#)
 - `\Longleftrightarrow` [t406](#), [t408](#)
 - `\longleftarrow` [t404](#)
 - `\longmapsto` [t402](#)
 - `\Longrightarrow` [t394](#)
 - `\longrightarrow` [t395](#), [t402](#)
 - `\loop` [a28](#), [b178](#), [C350](#)
 - `\lor` [t285](#)
 - File Key:** `a=ltltdirchk.dtx`, `b=ltlplain.dtx`, `c=ltlvers.dtx`, `d=ltldefns.dtx`,
`e=ltlalloc.dtx`, `f=ltlcntrl.dtx`, `g=ltlerror.dtx`, `h=ltlpar.dtx`, `i=ltltspace.dtx`,
`j=ltllogos.dtx`, `k=ltlfiles.dtx`, `l=ltloutenc.dtx`, `m=ltlcounts.dtx`, `n=ltllength.dtx`,
`o=ltlfsbas.dtx`, `p=ltlfsstrc.dtx`, `q=ltlfsncmp.dtx`, `r=ltlfsdcl.dtx`, `s=ltlfsini.dtx`,
`t=fontdef.dtx`, `u=ltlpreload.dtx`, `v=ltlfontcmd.dtx`, `w=ltlpageno.dtx`, `x=ltltxref.dtx`,
`y=ltlmiscen.dtx`, `z=ltlmath.dtx`, `A=ltllists.dtx`, `B=ltlboxes.dtx`, `C=ltltab.dtx`,
`D=ltlpictur.dtx`, `E=ltlthm.dtx`, `F=ltlsect.dtx`, `G=ltlfloat.dtx`, `H=ltlidxglo.dtx`,
`I=ltlbibl.dtx`, `J=ltlpage.dtx`, `K=ltloutput.dtx`, `L=ltlclass.dtx`, `M=ltlthyphen.dtx`,
`N=ltlfinal.dtx`, `O=ltlpatch.ltx`
 - `\lower` [j2](#), [t366](#), [B125](#),
 [D23](#), [D96](#), [D183](#), [D184](#), [D221](#), [D222](#)
 - `\lower@bound` [p342](#), [p343](#), [p355](#)
 - `\lowercase` [g29](#), [l110](#),
 [l847](#), [o249](#), [o341](#), [y143](#), [y150](#), [N175](#)
 - `\lq` [b160](#)
 - `\lrbox` [B69](#)
 - `lrbox` (environment) [271](#)
 - `\ltx@sh@ft` [b229](#),
 [l292](#), [l299](#), [l372](#), [l380](#), [l607](#), [l614](#)
- ## M
- `\M` [b158](#)
 - `\m@ne` [b39](#)
 - `\m@th` .. [b209](#), [b221](#), [j13](#), [t243](#), [t367](#),
 [t369](#), [t370](#), [t373](#), [t414](#), [t438](#),
 [t441](#), [t444](#), [t447](#), [t453](#), [t456](#),
 [t463](#), [t466](#), [t528](#), [z68](#), [z71](#), [z89](#),
 [z105](#), [z108](#), [z110](#), [z115](#), [z134](#),
 [z214](#), [z283](#), [z298](#), [z308](#), [B174](#),
 [B263](#), [C163](#), [F159](#), [G255](#), [G260](#)
 - `\magstep` [b149](#)
 - `\magstephalf` [b149](#)
 - `\makeatletter` [d272](#), [d288](#), [k26](#), [o305](#),
 [o347](#), [y19](#), [F134](#), [K2](#), [L206](#), [L310](#)
 - `\makeatother` [d272](#), [L206](#), [N241](#)
 - `\makebox` [z176](#), [271](#), [B3](#)
 - `\makeglossary` [k69](#), [357](#), [H20](#)
 - `\makeindex` [k68](#), [357](#), [H3](#)
 - `\makelabel`
 [A45](#), [A97](#), [A195](#), [A208](#), [A228](#), [A239](#)
 - `\MakeLowercase` [N172](#), [N181](#)
 - `\makeph@nt` [z84](#), [z86](#)
 - `\makesm@sh` [z100](#), [z102](#)
 - `\maketitle` [330](#)
 - `\MakeUppercase` [N165](#), [N165](#)
 - `\mandatory@arg` [p374](#), [p477](#),
 [p485](#), [p494](#), [p507](#), [p509](#), [p514](#),
 [p516](#), [p521](#), [p523](#), [p538](#), [p549](#), [p551](#)
 - `\mapsto` [t352](#)
 - `\mapstochar` [t351](#), [t352](#), [t402](#)
 - `\marginpar` [G187](#)
 - `\marginparpush` [K66](#), [K1172](#)
 - `\marginparsep` [K65](#), [K1183](#), [K1185](#)
 - `\marginparwidth` ... [G216](#), [K64](#), [K1185](#)
 - `\mark` [J23](#), [J31](#), [J39](#)
 - `\markboth` [J18](#)
 - `\markright` [J18](#)
 - `\math` [z193](#)
 - `math` (environment) [z193](#)
 - `\math@bgroup` [o447](#), [p260](#),
 [p266](#), [r40](#), [r99](#), [v112](#), [v113](#), [v120](#)
 - `\math@egroup`
 [o447](#), [p264](#), [p265](#), [v113](#), [v114](#), [v121](#)

$\backslash\math@fontsfalse$
 $\backslash\math@fontstrue$ [o179](#), [o461](#)
 $\backslash\math@version$ [o10](#), [o253](#),
[o421](#), [o425](#), [o427](#), [o428](#), [o430](#),
[p184](#), [r43](#), [r46](#), [r51](#), [r52](#), [r56](#),
[r67](#), [r68](#), [r69](#), [r87](#), [r88](#), [r89](#), [r102](#),
[r105](#), [r109](#), [r111](#), [r113](#), [r117](#), [s53](#)
 $\backslash\mathaccent$ [r504](#), [r532](#)
 $\backslash\mathalpha$
[r603](#), [r762](#), [t88](#), [t89](#), [t90](#), [t91](#),
[t92](#), [t93](#), [t94](#), [t95](#), [t96](#), [t97](#), [t98](#),
[t99](#), [t100](#), [t101](#), [t102](#), [t103](#), [t104](#),
[t105](#), [t106](#), [t107](#), [t108](#), [t109](#),
[t110](#), [t111](#), [t112](#), [t113](#), [t114](#),
[t115](#), [t116](#), [t117](#), [t118](#), [t119](#),
[t120](#), [t121](#), [t122](#), [t123](#), [t124](#),
[t125](#), [t126](#), [t127](#), [t128](#), [t129](#),
[t130](#), [t131](#), [t132](#), [t133](#), [t134](#),
[t135](#), [t136](#), [t137](#), [t138](#), [t139](#),
[t140](#), [t141](#), [t142](#), [t143](#), [t144](#),
[t145](#), [t146](#), [t147](#), [t148](#), [t149](#),
[t216](#), [t217](#), [t218](#), [t219](#), [t220](#),
[t221](#), [t222](#), [t223](#), [t224](#), [t225](#),
[t226](#), [t424](#), [t425](#), [t426](#), [t427](#),
[t428](#), [t429](#), [t430](#), [t431](#), [t433](#), [t436](#)
 $\backslash\mathbf$ [s13](#), [t70](#)
 $\backslash\mathbin$ [r767](#),
[t151](#), [t152](#), [t154](#), [t276](#), [t277](#),
[t278](#), [t279](#), [t282](#), [t284](#), [t286](#),
[t287](#), [t288](#), [t289](#), [t290](#), [t291](#),
[t292](#), [t293](#), [t294](#), [t295](#), [t296](#),
[t297](#), [t298](#), [t299](#), [t300](#), [t301](#),
[t302](#), [t303](#), [t304](#), [t305](#), [t306](#),
[t307](#), [t308](#), [t309](#), [t310](#), [t311](#), [z37](#)
 $\backslash\mathcal$ [t69](#)
 $\backslash\mathchar$
[b221](#), [r546](#), [r588](#), [t228](#), [t240](#), [t521](#)
 $\backslash\mathchar@type$ [r532](#),
[r576](#), [r579](#), [r588](#), [r604](#), [r703](#), [r763](#)
 $\backslash\mathchardef$ [b21](#), [b22](#),
[b23](#), [b24](#), [e3](#), [e4](#), [e5](#), [e6](#), [l75](#), [r579](#)
 $\backslash\mathchoice$ [z61](#)
 $\backslash\mathclose$ [r770](#), [t150](#),
[t159](#), [t161](#), [t164](#), [t169](#), [t175](#),
[t177](#), [t179](#), [t472](#), [t497](#), [t501](#),
[t505](#), [t509](#), [t515](#), [z43](#), [z46](#), [z49](#), [z52](#)
 $\backslash\mathcode$ [r576](#), [t171](#), [t172](#), [t173](#)
 $\backslash\mathdollar$ [l261](#), [t518](#)
 $\backslash\mathellipsis$ [l275](#), [t523](#)
 $\backslash\mathgroup$ [l1104](#), [o22](#), [p257](#),
[p263](#), [p269](#), [p270](#), [p281](#), [s68](#), [t529](#)
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
 $\backslash\mathhexbox$ [b221](#), [s78](#)
 $\backslash\mathindent$ [z273](#), [z284](#), [z300](#), [z310](#)
 $\backslash\mathinner$ [t412](#), [t416](#), [t421](#), [t523](#)
 $\backslash\mathit$ [s28](#), [t72](#), [t75](#), [t521](#)
 $\backslash\mathnormal$ [t68](#)
 $\backslash\mathop$ [r766](#),
[t259](#), [t260](#), [t261](#), [t262](#), [t263](#),
[t264](#), [t265](#), [t267](#), [t268](#), [t269](#),
[t270](#), [t271](#), [t272](#), [t274](#), [t275](#),
[t444](#), [t447](#), [z3](#), [z4](#), [z5](#), [z6](#), [z7](#), [z8](#),
[z9](#), [z10](#), [z11](#), [z12](#), [z13](#), [z14](#), [z15](#),
[z16](#), [z17](#), [z18](#), [z19](#), [z20](#), [z21](#), [z22](#),
[z23](#), [z24](#), [z25](#), [z26](#), [z27](#), [z28](#), [z29](#),
[z30](#), [z31](#), [z32](#), [z33](#), [z34](#), [z107](#), [z201](#)
 $\backslash\mathopen$ [r769](#), [t160](#), [t163](#), [t168](#), [t174](#),
[t176](#), [t178](#), [t470](#), [t499](#), [t503](#),
[t507](#), [t511](#), [t513](#), [z41](#), [z44](#), [z47](#), [z50](#)
 $\backslash\mathord$ [r603](#),
[r765](#), [t155](#), [t162](#), [t165](#), [t170](#),
[t182](#), [t183](#), [t184](#), [t186](#), [t187](#),
[t188](#), [t189](#), [t190](#), [t191](#), [t192](#),
[t193](#), [t194](#), [t195](#), [t196](#), [t197](#),
[t198](#), [t199](#), [t200](#), [t201](#), [t202](#),
[t203](#), [t204](#), [t205](#), [t206](#), [t207](#),
[t208](#), [t209](#), [t210](#), [t211](#), [t212](#),
[t213](#), [t214](#), [t215](#), [t227](#), [t229](#),
[t230](#), [t231](#), [t232](#), [t233](#), [t234](#),
[t235](#), [t236](#), [t237](#), [t238](#), [t239](#),
[t241](#), [t242](#), [t247](#), [t248](#), [t249](#),
[t250](#), [t252](#), [t253](#), [t254](#), [t255](#),
[t256](#), [t257](#), [t258](#), [t432](#), [t434](#),
[t435](#), [t455](#), [t456](#), [t459](#), [t460](#),
[t461](#), [t462](#), [t474](#), [t476](#), [t478](#),
[t481](#), [t495](#), [t517](#), [t518](#), [t519](#), [t520](#)
 $\backslash\mathpalette$
[t365](#), [t369](#), [t372](#), [z60](#), [z69](#), [z82](#), [z98](#)
 $\backslash\mathparagraph$ [l264](#), [m59](#), [t518](#)
 $\backslash\mathph\nt$ [z82](#), [z88](#)
 $\backslash\mathpunct$
[r771](#), [t153](#), [t157](#), [t409](#), [t410](#), [t411](#)
 $\backslash\mathrel$ [r768](#), [t156](#), [t158](#),
[t166](#), [t167](#), [t180](#), [t181](#), [t244](#),
[t312](#), [t313](#), [t314](#), [t315](#), [t316](#),
[t317](#), [t318](#), [t319](#), [t320](#), [t321](#),
[t322](#), [t323](#), [t324](#), [t325](#), [t327](#),
[t329](#), [t331](#), [t332](#), [t333](#), [t334](#),
[t335](#), [t336](#), [t337](#), [t338](#), [t339](#),
[t340](#), [t341](#), [t343](#), [t344](#), [t345](#),
[t346](#), [t347](#), [t349](#), [t351](#), [t353](#),
[t354](#), [t355](#), [t356](#), [t357](#), [t358](#),
[t359](#), [t360](#), [t361](#), [t362](#), [t363](#),
[t365](#), [t369](#), [t372](#), [t379](#), [t381](#),
[t384](#), [t385](#), [t387](#), [t390](#), [t392](#),
[t483](#), [t485](#), [t487](#), [t489](#), [t491](#),

- t493, z42, z45, z48, z51, z107, z201
- `\mathring` t436
- `\mathrm` s4, t67
- `\mathsection` l265, m59, t518
- `\mathsf` s7, t71, t74
- `\mathsm@sh` z98, z104
- `\mathsterling` l273, t518
- `\mathstrut` z94, z112, z113
- `\mathsurround` b209
- `\mathsymbol` r581
- `\mathtt` s10, t73
- `\mathunderscore` t518
- `\mathversion` o253, s50, s52
- `\matrix` z110, z114
- `\max` z22
- `\maxdeadcycles` K7
- `\maxdepth`
 . b130, k50, K73, K384, K394,
 K428, K533, K542, K582, K733, N9
- `\maxdimen` b79, b131, b132,
 b188, b226, b244, b252, o478,
 p340, p399, t366, D264, D298,
 K239, K1193, K1220, K1225, N13
- `\maybe@ic` v46, v47, v65
- `\maybe@ic@` v65
- `\mb@b` B34, B44
- `\mb@l` B34, B38, B43, D68, D72
- `\mb@r` B34, B38, B43, D68, D72
- `\mb@t` B35, B42
- `\mbox` b221, j13, l247,
 s74, t414, 271, B7, B8, D28, G260
- `\mddefault` s17, t32, t40
- `\mdseries` s15, s16, s77, v20
- `\meaning` a166, a175,
 a246, d201, d281, r331, r344,
 r445, r504, r546, r638, r715, r819
- `\medbreak` b203
- `\medmuskip` t531, z36, z38, z145
- `\medskip` b206, i162
- `\medskipamount` b205, i163, i165
- `\MessageBreak` d177, g3, g6, g13, g36,
 g49, g63, g76, g197, g199, g205,
 g217, l126, l852, l855, l879, l880,
 l881, l882, l883, l884, l885, l886,
 l887, l888, l937, l939, l947, l954,
 l1109, o401, p20, p21, p67, p88,
 p281, p450, p474, p520, p537,
 p554, q21, q23, r286, r295,
 r433, v126, y23, K460, K1293,
 L87, L236, L247, L249, L251,
 L262, L317, L318, L320, L321,
 L322, L324, L326, L343, L344,
 L345, L346, L392, L409, L410,
 L442, L470, N73, N74, N75, N77
- File Key:** a=ltltdirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx
- `\mho` s90
- `\mid` t316
- `\min` z23
- `\minipage` B194
- `minipage (environment)` 272
- `\mit` s155
- `\mkern` . t228, t244, t246, t370, t379,
 t421, t422, t423, t451, t452,
 t453, t454, t455, t456, t457,
 t458, z36, z37, z40, z73, z74, F160
- `\models` t392
- `\month` a132, c9, L454
- `\moveright` K504
- `\mp` t303
- `\mscount` C347
- `\mskip` i188,
 z36, z38, z144, z145, z146, z147
- `\mu` t198
- `\multicolumn` C203
- `\multiput` D33, D37
- `\multispan` C203, C347
- `\muskip` b29, b50, t451, t452
- `\muskipdef` b50
- ## N
- `\n@space` .. t524, t525, t526, t527, t528
- `\nabla` t239
- `\narrower` b214
- `\natural` t253
- `\ne` t326
- `\nearrow` t319
- `\NeedsTeXFormat` p12, L241, L513
- `\neg` t250, t251
- `\negthinspace` i198
- `\neq` t326
- `\new@command`
 . d44, d45, d100, d135, d157, d213
- `\new@environment` ... d115, d116, d129
- `\new@fontshape` q1, q14
- `\new@mathalphabet` ... r328, r349, r360
- `\new@mathgroup` o22, r212
- `\new@mathversion` r11, r170, r172
- `\new@symbolfont` r213, r241
- `\newbox` b47, b84,
 b211, e13, z66, A27, B54, C16,
 C17, C18, C311, D6, D325,
 D330, K67, K101, K102, K103
- `\newcommand`
 29, d44, l4, t29, t30, t31, t32,
 t33, t34, t35, t36, t37, t38, t39,
 t40, t41, D341, K1513, K1516,
 K1519, K1520, K1523, K1524
- `\newcount` b47, b87, b90, b92,
 b93, b94, b95, b97, b99, b148,

- e7, e8, i62, k9, m25, p25, r18, r178, z55, z205, z206, A23, A24, A25, A26, A56, A216, A231, B235, C11, C12, C13, C14, C15, C303, C304, C305, D319, D320, D321, D322, D331, F19, F123, F124, G3, G165, G166, G167, G168, K84, K86, K88, K90, K92, K100, K1282, K1511, K1514, K1517, K1521, N3, N4, N5
- \newcounter m10, 121
- \newdimen b47, b79, b81, b82, b96, b147, e10, e11, e12, i61, p356, p357, z53, z274, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, B96, B97, C3, C5, C6, C7, C8, C143, C306, C307, C308, C309, D3, D4, D5, D7, D239, D240, D241, D242, D243, D244, D323, D324, D326, D327, D328, D329, G261, K52, K53, K54, K56, K57, K58, K59, K60, K61, K62, K63, K64, K65, K66, K72, K74, K75, K87, K89, K91, K93, K94, K95, K96, K97, K98, K99, K1283, K1284
- \newenvironment 29, d115, L452
- \newfam o25
- \newfont s54
- \newgroup r38
- \newhelp b47
- \newif d139, e9, k7, k8, l896, o179, r6, x3, z75, z76, z133, z207, A28, A29, A30, A31, A32, A33, A128, B249, C21, C221, D76, D233, D234, D235, D236, F21, F107, K76, K77, K78, K79, K80, K81, K82, K83, L2
- \newinsert b62, B236, G245, K23, K24, K25, K26, K27, K28, K29, K30, K31, K32, K33, K34, K35, K36, K37, K38, K39, K40, K1192
- \newlabel x22, x34
- \newlanguage b56
- \newlength n3, 125
- \newline i43
- \newlinechar a19, d5
- \newmathalphabet q99
- \newmathalphabet@@ q99
- \newmathalphabet@@@ q99
- \newmuskip b47
- \newpage K114, K120, K131
- \newread b54, k3
- \newsavebox 271, B54
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx, e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx, j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx, o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx, t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx, y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx, D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx, I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx, N=ltfinal.dtx, O=ltpatch.ltx
- \newskip b47, b80, b83, b145, b146, e14, e15, e17, i165, i166, i167, i195, n3, y79, z208, A2, A3, A4, A5, A6, A7, A8, K1525, K1526, K1527, K1531, K1532, K1535, K1536, K1537, K1541, K1542, K1543
- \newtheorem E1
- \newtie l695, l1036
- \newtoks b47, e16, o263, o264, p201
- \newwrite b54, k4, k5, k6, F137, H4, H21
- \nfss@catcodes d290, o28, o95, o305, o338, o339, o346, q10, t19, t24, t54, K3
- \nfss@text l269, l271, s74, v5, v104, x13
- \NG l394, N179
- \ng l412, N179
- \ni t341, t342
- \no@alphabet@error o6, r192, r194, r365, r366, r380, r389, r475, r476
- \noaccents@ o469, t48
- \noalign t245, t439, t442, t444, t445, t449, t450, z112, z113, z118, z121, z135, z247, C202, C327, C346, D75
- \noboundary b98
- \nobreak b191, b194, b196, i38, i53, i149, i157, i176, i183, i193, k67, k79, l334, l336, y69, B288, F73, F157, F158, F162, G297, J25, J33, K288, K888
- \nobreakdashes i168
- \nobreakspace i182
- \nocite I39, 359
- \nocorr v26, v41, v45, v48
- \nocorrlist v71, v103
- \nofiles k63, 72
- \noindent F122
- \nointerlineskip b186, t245, t439, t442, t445, t449, z175, D285, D287, K1180, K1188
- \nolimits t266, t273, z3, z4, z5, z9, z10, z11, z12, z13, z14, z15, z16, z17, z18, z19, z20, z21, z26, z27, z28, z29, z31, z34
- \nolinebreak i13, 59
- \non@alpherr o441, o443, r59, r73, r120, r841
- \nonfrenchspacing b152, b257, k42
- \nonscript z36, z38
- \nonumber z234, z257, z258
- \nopagebreak i3, 59
- \normalbaselines b156, z108, z110

`\normalbaselineskip` [b145](#), [b157](#), [p142](#), [B188](#)
`\normalcolor` [z200](#),
[z270](#), [B47](#), [B227](#), [F163](#), [G93](#),
[K164](#), [K368](#), [K508](#), [K518](#), [K1483](#)
`\normalfont` [o479](#), [s79](#),
[v18](#), [y120](#), [z200](#), [z270](#), [F163](#), [G256](#)
`\normallineskip` ... [b145](#), [b156](#), [B187](#)
`\normallineskiplimit` [b145](#), [b157](#), [z136](#)
`\normalmarginpar` [G242](#)
`\normalsfcodes` [k38](#), [k40](#), [k42](#), [k62](#), [K497](#)
`\normalsize` [k36](#),
[v124](#), [G23](#), [G101](#), [G227](#), [K496](#), [L5](#)
`\not` [t244](#), [t326](#), [t345](#)
`\not@base` [s86](#), [s90](#), [s91](#), [s92](#),
[s93](#), [s94](#), [s95](#), [s96](#), [s97](#), [s98](#), [s99](#), [s100](#)
`\not@math@alphabet` [s4](#), [s7](#),
[s10](#), [s13](#), [s16](#), [s19](#), [s22](#), [s25](#), [s28](#), [s33](#)
`\notin` [t369](#)
`\notrace` [K1244](#)
`\nu` [t199](#)
`\null` [b170](#), [l382](#), [l384](#), [l678](#), [l681](#), [x17](#),
[y108](#), [y132](#), [z91](#), [z110](#), [z128](#), [F157](#)
`\nulldelimiterspace` [b134](#), [t528](#)
`\nullfont` [y51](#)
`\number` [a33](#),
[d2](#), [d82](#), [m40](#), [o425](#), [o428](#), [p401](#),
[r51](#), [r69](#), [r89](#), [r110](#), [s71](#), [L423](#), [L454](#)
`\numberline` [F55](#), [F65](#), [F166](#), [G17](#)
`\narrow` [t321](#)

O

`\O` [l199](#), [l302](#), [l396](#), [l617](#), [N178](#)
`\o` [l208](#), [l307](#), [l414](#), [l625](#), [N178](#)
`\o@lign` [b223](#),
[l292](#), [l299](#), [l372](#), [l380](#), [l607](#), [l614](#)
`\oalign` [b223](#)
`\obeycr` [i204](#)
`\obeylines` [b173](#), [y114](#), [y127](#), [y128](#), [K465](#)
`\obeyspaces` [b173](#), [K465](#)
`\oddsidemargin` [K53](#), [K55](#), [K490](#)
`\odot` [t298](#)
`\OE` [l198](#), [l301](#), [l395](#), [l616](#), [N178](#)
`\oe` [l207](#), [l306](#), [l413](#), [l626](#), [N178](#)
`\of` [z67](#), [z204](#)
`\offinterlineskip` [b186](#)
`\oint` [t273](#)
`\ointop` [t272](#), [t273](#)
`\oldstylenums` [l1101](#), [s64](#)
`\Omega` [t226](#)
`\omega` [t209](#)
`\ominus` [t301](#)
`\omit` [z121](#), [z122](#), [C337](#), [C340](#), [C347](#), [C351](#)
`\on@line` [g8](#), [g15](#), [g183](#), [y56](#), [B75](#), [L342](#)

File Key: a=lt`dirchk.dtx`, b=lt`plain.dtx`, c=lt`vers.dtx`, d=lt`defs.dtx`,
e=lt`alloc.dtx`, f=lt`cntrl.dtx`, g=lt`error.dtx`, h=lt`par.dtx`, i=lt`space.dtx`,
j=lt`logos.dtx`, k=lt`files.dtx`, l=lt`outenc.dtx`, m=lt`counts.dtx`, n=lt`length.dtx`,
o=lt`ssbas.dtx`, p=lt`ssstrc.dtx`, q=lt`ssscmp.dtx`, r=lt`ssdcl.dtx`, s=lt`ssini.dtx`,
t=lt`fontdef.dtx`, u=lt`preload.dtx`, v=lt`fontcmd.dtx`, w=lt`pageno.dtx`, x=lt`xref.dtx`,
y=lt`miscen.dtx`, z=lt`math.dtx`, A=lt`lists.dtx`, B=lt`boxes.dtx`, C=lt`tab.dtx`,
D=lt`pictur.dtx`, E=lt`thm.dtx`, F=lt`tsect.dtx`, G=lt`float.dtx`, H=lt`idxglo.dtx`,
I=lt`bibl.dtx`, J=lt`page.dtx`, K=lt`output.dtx`, L=lt`class.dtx`, M=lt`hyphen.dtx`,
N=lt`final.dtx`, O=lt`patch.ltx`

`\onecolumn` [K122](#)
`\OnlyDescription` [p5](#), [u3](#)
`\oalign` [b223](#),
[l296](#), [l376](#), [l382](#), [l384](#), [l578](#), [l611](#),
[l678](#), [l681](#), [l727](#), [s76](#), [t370](#), [t373](#)
`\openup` [z129](#), [z134](#)
`\operator@font`
..... [t529](#), [z3](#), [z4](#), [z5](#), [z6](#), [z7](#),
[z8](#), [z9](#), [z10](#), [z11](#), [z12](#), [z13](#), [z14](#),
[z15](#), [z16](#), [z17](#), [z18](#), [z19](#), [z20](#), [z21](#),
[z22](#), [z23](#), [z24](#), [z25](#), [z26](#), [z27](#), [z28](#),
[z29](#), [z30](#), [z31](#), [z32](#), [z33](#), [z34](#), [z37](#), [z40](#)
`\oplus` [t302](#)
`\optional@arg`
..... [p375](#), [p470](#), [p472](#), [p548](#), [p551](#)
`\OptionNotUsed` [L136](#), [L143](#), [L357](#)
`\Orb` [l561](#)
`\oslash` [t299](#)
`\otimes` [t300](#)
`\outer` [b11](#)
`\outer@nobreak`
..... [G106](#), [G145](#), [G150](#), [G153](#), [G221](#)
`\outerparskip` [A1](#)
`\output` [K204](#)
`\outputpenalty` [K206](#), [K220](#), [K245](#),
[K248](#), [K249](#), [K286](#), [K898](#), [K899](#)
`\oval` [D260](#), [D263](#)
`\over` [t377](#), [z107](#), [z202](#)
`\overbrace` [t444](#)
`\overfullrule` [b129](#), [J55](#)
`\overleftarrow` [t441](#)
`\overrightarrow` [t438](#)
`\owns` [t342](#)

P

`\P` [l264](#)
`\p@` [b81](#)
`\p@equation` [z212](#), [z307](#)
`\p@reset@font` [s79](#)
`\p@selectfont` [p117](#)
`\PackageError` [g88](#), [l850](#), [l902](#), [l946](#)
`\PackageInfo`
..... [g88](#), [l879](#), [l892](#), [l893](#), [l953](#), [l1163](#)
`\PackageWarning` [g88](#), [l903](#), [l1107](#)
`\PackageWarningNoLine` [g88](#)
`\pagebreak` [i3](#), [59](#)
`\pagegoal` [K1227](#), [K1234](#)
`\pagenumbering` [w5](#), [232](#)
`\pageref` [x10](#)
`\pageshrink` [K420](#), [K424](#), [K440](#)
`\pagestyle` [J2](#)
`\pagetotal` [K109](#)
`\paperheight` [K74](#)
`\paperwidth` [K74](#)

- \par a67, b11, b166, b174, b175, b190,
b199, b200, b201, b203, b205,
b207, d6, h3, h4, h6, y49, y69,
y106, A63, A110, A126, A151,
A154, B179, B223, C177, C353,
F24, F73, F164, G15, G24,
G143, J48, J49, K143, K205, K1233
 - \par@deathcycles A56, A77, A79, A80
 - \paragraphmark F126
 - \parallel t315
 - \parbox 271, B145
 - \parboxrestore B190
 - \parfillskip b144, o478,
y78, y91, y103, A76, B187, F152
 - \parindent b136, b215, b216, y78,
y85, y91, y103, A50, B182, F153
 - \parsep A1, A49, A90
 - \parshape A54
 - \parskip b137,
y70, y101, y103, z316, A49,
A73, A88, A90, A117, A143,
A162, A213, B182, C70, K898
 - \partial t235
 - \partopsep z314, A1, A61
 - \PassOptionsToClass 423, L113
 - \PassOptionsToPackage 423, L113
 - \patch@level N189, N201, N203, O8
 - \patterns l160
 - \penalty l17, b195, b196, b197,
b198, b199, b200, b204, b206,
b208, i7, i10, i21, i120, i124,
v100, y108, y111, z37, z137,
z247, A180, C58, G120, G124,
G126, K117, K145, K146, K896
 - \perp t355
 - \ph@nt z77, z78, z79, z80
 - \phantom z75
 - \Phi t224
 - \phi t206
 - \Pi t221
 - \pi t201
 - \pickup@font l136, o170,
o270, o412, p122, p285, p287, p289
 - \pictur@ D9, D53, D54, D60
 - \picture D9
 - \pm t304
 - \pmatrix z114
 - \pmod z39
 - \poptabs g244, C129
 - \poptracing p130, p294
 - \postdisplaypenalty i28, z281, z295
 - \pounds l272
 - \Pr z32
 - \pr@@s z156, z164
 - \pr@@@t z159, z165
 - \pr@mes z153, z154
 - \prec t332
 - \preceq t335
 - \predisplaypenalty b115, z280, z294
 - \preload@sizes q84
 - \pretolerance b102, o480
 - \prevdepth b186, b190,
b191, i147, i152, z135, G121, G123
 - \prim@s z150, z152, z164
 - \prime t172, t237, z153
 - \prime@s z151
 - \process@table k34, r124
 - \ProcessOptions l870, l905, p71, L144, L187, L411
 - \ProcessOptions* L144
 - \prod t267
 - \propto t312
 - \protect l5,
d70, d192, d208, d217, d222,
d225, d226, d228, d229, d234,
d235, d240, d243, d244, g213,
g232, g234, g235, g244, g252,
g262, g274, g277, g285, k75, l26,
l32, l53, l60, l164, l172, r394,
r846, s57, v125, x12, C234,
F11, F55, F65, F143, G17, K474
 - \protected@edef d227, x37, B243, F43,
G277, N169, N175, N180, N181
 - \protected@write k66, k71, x33, F145, H14, H31
 - \protected@xdef d227, F10, G263, G287, G303
 - \provide@command d151, d152
 - \providexcommand d151, l6, K1266
 - \ProvidesClass 422, L100
 - \ProvidesFile a36, t551, t553, t554, t555, L102
 - \ProvidesPackage p13, 423, L83, L100, L514
 - \ProvideTextCommand l3, l65
 - \ProvideTextCommandDefault l62
 - \ps@empty J10, N15
 - \ps@plain J13
 - \Psi t225
 - \psi t208
 - \pushtabs g244, C126
 - \pushtracing p115, p275
 - \put D29, D197, D198, D199,
D200, D205, D207, D219, D220,
D221, D222, D227, D230, D378
- File Key: a=ltltdirchk.dtx, b=ltlplain.dtx, c=ltlvers.dtx, d=ltldefs.dtx,
e=ltlalloc.dtx, f=ltlcntrl.dtx, g=ltlerror.dtx, h=ltlpar.dtx, i=ltltspace.dtx,
j=ltllogos.dtx, k=ltlfiles.dtx, l=ltloutenc.dtx, m=ltlcounts.dtx, n=ltllength.dtx,
o=ltlssbas.dtx, p=ltlssstrc.dtx, q=ltlssscmp.dtx, r=ltlssdcl.dtx, s=ltlssini.dtx,
t=fontdef.dtx, u=ltlpreload.dtx, v=ltlfontcmd.dtx, w=ltlpageno.dtx, x=ltltxref.dtx,
y=ltlmiscen.dtx, z=ltlmath.dtx, A=ltllists.dtx, B=ltlboxes.dtx, C=ltltab.dtx,
D=ltlpictur.dtx, E=ltlthm.dtx, F=ltlsect.dtx, G=ltlfloat.dtx, H=ltltdxglo.dtx,
I=ltlbibl.dtx, J=ltlpage.dtx, K=ltloutput.dtx, L=ltlclass.dtx, M=ltlthyphen.dtx,
N=ltlfinal.dtx, O=ltlpatch.ltx

Q

\qbezier [304](#), [D341](#)
 \qbeziermax [D337](#), [D362](#)
 \quad [i201](#)
 \quad [i201](#), [z109](#), [z111](#), [z120](#), [F94](#)
 \quotedblbase [l415](#), [l627](#)
 \quotesinglbase [l416](#)

R

\r [b164](#), [b165](#), [l191](#), [l290](#), [l330](#), [l366](#),
 [l467](#), [l494](#), [l504](#), [l530](#), [l603](#), [l639](#)
 \r@t [z66](#)
 \radical [r713](#), [r715](#), [r745](#)
 \raggedbottom [J39](#)
 \raggedleft [y86](#), [y88](#)
 \raggedright [y80](#), [y82](#)
 \raise [l329](#), [l332](#), [l579](#), [l641](#),
 [l728](#), [s77](#), [t373](#), [t421](#), [t423](#), [z73](#),
 [B273](#), [B282](#), [D30](#), [D40](#), [D95](#),
 [D183](#), [D262](#), [D279](#), [D305](#), [D389](#)
 \raisebox [272](#), [B264](#)
 \rangle [t496](#)
 \rbrace [l263](#), [t500](#)
 \rbrack [b162](#)
 \rceil [t504](#)
 \Re [t233](#)
 \ref [x10](#)
 \refstepcounter [x32](#), [z198](#),
 [z296](#), [i21](#), [A192](#), [E27](#), [F42](#), [G9](#)
 \Relbar [t384](#), [t392](#), [t394](#), [t400](#)
 \relbar [t381](#), [t396](#), [t398](#)
 \relpenalty [b110](#)
 \rem@pt [o246](#)
 \remove@angles [p301](#), [p326](#)
 \remove@nil [r27](#)
 \remove@star [p301](#), [p309](#)
 \remove@to@nnil [o245](#), [p301](#), [p329](#), [p458](#)
 \removelastskip [b202](#), [b204](#), [b206](#), [b208](#)
 \renew@command .. [d93](#), [d94](#), [d158](#), [d166](#)
 \renew@environment [d121](#), [d122](#)
 \renewcommand [29](#), [d93](#), [z269](#), [z276](#), [z286](#)
 \renewenvironment [29](#), [d121](#), [z293](#), [z305](#)
 \repeat [a28](#), [a30](#), [b178](#), [C350](#)
 \RequirePackage
 [423](#), [L202](#), [L209](#), [L230](#), [L407](#)
 \RequirePackageWithOptions [423](#), [L228](#)

\reserved@a [a68](#), [a72](#), [a73](#),
 [a142](#), [a143](#), [a146](#), [a164](#), [a168](#),
 [a190](#), [a197](#), [a200](#), [a202](#), [a203](#),
 [a210](#), [a213](#), [a215](#), [a216](#), [a223](#),
 [a226](#), [a228](#), [a254](#), [a255](#), [a256](#),
 [c5](#), [c11](#), [c26](#), [d85](#), [d89](#), [d102](#),
 [d103](#), [d104](#), [d106](#), [d157](#), [d158](#),

[d159](#), [d165](#), [d166](#), [d167](#), [d168](#),
[d171](#), [d190](#), [d199](#), [d204](#), [d255](#),
[d264](#), [f33](#), [f37](#), [g216](#), [i171](#), [i174](#),
[k76](#), [k77](#), [k99](#), [k100](#), [k138](#), [k140](#),
[k145](#), [k147](#), [k149](#), [k155](#), [k159](#),
[k167](#), [k170](#), [k186](#), [k187](#), [k191](#),
[k197](#), [k218](#), [k222](#), [k226](#), [l80](#),
[l82](#), [l90](#), [l107](#), [l112](#), [o38](#), [o41](#),
[o44](#), [o80](#), [o83](#), [o85](#), [o122](#), [o126](#),
[o340](#), [o343](#), [o387](#), [o388](#), [o400](#),
[o403](#), [o408](#), [o433](#), [o436](#), [o437](#),
[o445](#), [p150](#), [p152](#), [p154](#), [p164](#),
[p166](#), [p169](#), [p298](#), [p299](#), [p314](#),
[p315](#), [q43](#), [q47](#), [r275](#), [r284](#),
[r286](#), [r330](#), [r333](#), [r343](#), [r346](#),
[r444](#), [r446](#), [r504](#), [r505](#), [r546](#),
[r547](#), [r638](#), [r639](#), [r715](#), [r716](#),
[r818](#), [r820](#), [r836](#), [r838](#), [r839](#),
[r844](#), [v30](#), [v31](#), [v36](#), [v37](#), [v48](#),
[v51](#), [v70](#), [v77](#), [y41](#), [y42](#), [y54](#),
[y55](#), [y59](#), [y64](#), [y65](#), [z249](#), [z250](#),
[z251](#), [z252](#), [z254](#), [B36](#), [B37](#),
[B40](#), [B70](#), [B76](#), [C211](#), [C215](#),
[C220](#), [C239](#), [C328](#), [C329](#), [D99](#),
[D101](#), [D105](#), [D266](#), [G29](#), [G30](#),
[G32](#), [G33](#), [G59](#), [G63](#), [G69](#), [G72](#),
[G75](#), [G78](#), [K668](#), [K1351](#), [K1353](#),
[K1359](#), [K1362](#), [L77](#), [L80](#), [L81](#),
[L195](#), [L198](#), [L242](#), [L243](#), [L246](#),
[L283](#), [L287](#), [L298](#), [L299](#), [L301](#),
[L311](#), [L351](#), [L516](#), [L518](#), [N16](#),
[N33](#), [N35](#), [N36](#), [N44](#), [N46](#), [N47](#),
[N85](#), [N116](#), [N122](#), [N123](#), [N125](#),
[N127](#), [N131](#), [N138](#), [N140](#), [N141](#),
[N149](#), [N151](#), [N152](#), [N167](#), [N168](#),
[N169](#), [N170](#), [N173](#), [N174](#), [N175](#),
[N176](#), [N202](#), [N205](#), [N206](#), [N223](#)

\reserved@b [a69](#), [a70](#), [d77](#), [d79](#), [d87](#),
[d104](#), [d105](#), [d200](#), [d201](#), [d204](#),
[d256](#), [d266](#), [f33](#), [f34](#), [f37](#), [i172](#),
[i173](#), [i180](#), [k98](#), [k100](#), [k150](#),
[k152](#), [k154](#), [k221](#), [k227](#), [l83](#),
[l90](#), [o70](#), [o72](#), [o125](#), [o126](#), [o434](#),
[o445](#), [q37](#), [q44](#), [q61](#), [q63](#), [r206](#),
[r208](#), [r256](#), [r258](#), [r283](#), [r284](#),
[r285](#), [r320](#), [r322](#), [r401](#), [r403](#),
[r448](#), [r449](#), [r450](#), [r457](#), [v35](#),
[v36](#), [v49](#), [v51](#), [v77](#), [v78](#), [C216](#),
[C218](#), [C220](#), [G39](#), [G40](#), [K604](#),
[K607](#), [K620](#), [K623](#), [L78](#), [L79](#),
[L291](#), [L296](#), [L299](#), [L459](#), [L460](#),
[L462](#), [L488](#), [N19](#), [N21](#), [N25](#),
[N88](#), [N90](#), [N94](#), [N168](#), [N174](#), [N223](#)

\reserved@c [a70](#), [a75](#),

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

- d261, d264, d266, d269, k210,
k211, o71, o72, o435, o438, q38,
q45, q51, q58, r24, r28, r207,
r208, r257, r258, r321, r322,
r402, r403, r425, r434, r449,
r463, r628, r644, r653, r681,
r692, r731, r744, r746, v50, v52,
v59, L436, L437, L438, L448,
L464, L471, L496, N23, N28,
N38, N92, N113, N114, N115,
N117, N118, N119, N120, N121,
N124, N126, N133, N143, N225
- \reserved@d a73, a76,
d254, d263, k209, k211, q51,
q58, q60, q64, r636, r644, r653,
r689, r692, r739, r744, r748, N226
- \reserved@e .. i36, i38, i47, i53, q29,
q35, q60, q63, q64, r25, r30, N227
- \reserved@f
. i37, i38, i53, l846, l847, l848,
l849, l851, l858, o165, o167,
o173, o174, p338, p349, p353,
p361, p367, p370, p421, p458,
p461, q17, q28, q35, q61, q63, N228
- \reset@font ... I20, s79, x13, B240,
G100, G226, G272, J14, K495
- \restglb@settings p222, p232
- \restore@mathversion r63, r66, r86, r94
- \restore@protect d227
- \restorecr i204
- \reversemarginpar G242
- \rfloor t508
- \rgroup t512
- \rhd s99
- \rho t202
- \rhook t387, t388
- \right t524,
t525, t526, t527, z109, z114, z127
- \Rightarrow t325, t394, t406
- \rightarrow t349,
t350, t352, t386, t396, t404, t455
- \rightarrowfill t439, t453
- \rightharpoondown t363
- \rightharpoonup t362, t374
- \righthypenmin b93, M11
- \rightleftharpoons t372
- \rightline B289
- \rightmargin A9, A40, A51
- \rightmark J34
- \rightskip b216, y77,
y83, y90, y102, A75, B186, F152
- \rlap l329,
l332, l641, z259, z270, B293, C72
- \rlh@ t372, t373
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx
- \rmdefault s5, s67, t29, t39
- \rmfamily s3, s4, v15
- \rmoustache t471
- \Roman m36, 121
- \roman m35, 121
- \romannumeral
.... m41, m42, A43, A224, A235
- \root z66, z204
- \rootbox z66
- \rq b160
- \rule 272, B247, B250, G282
- ## S
- \S l265
- \s@fct@ p386, p463
- \s@fct@fixed p542
- \s@fct@gen p479
- \s@fct@genb p488
- \s@fct@sgen p479
- \s@fct@sgenb p488
- \s@fct@sub p501
- \s@fct@subf p530
- \samepage i27, 59
- \savebox 271, B55
- \sb z142
- \sbox b210,
j4, 271, A195, B57, B58, B63, B68
- \scan@fontshape q30, q33
- \scan@fontshape q2,
q3, q4, q5, q6, q7, q8, q16, q27
- \scdefault s26, t34
- \scriptfont p292
- \scriptfont@name p287, p292
- \scriptscriptfont p293
- \scriptscriptstyle z65, z68
- \scriptspace b135
- \scriptstyle t243, z64
- \scshape l254, s24, s25, v23
- \searrow t320
- \sec z20
- \secdef F125
- \sectionmark F126
- \select@group o418, o437, r39, r160,
r197, r330, r383, r392, r430, r462
- \selectfont j7,
l256, l348, l653, l871, l940, l958,
o231, p112, s5, s8, s11, s14,
s17, s20, s23, s26, s29, s60, G258
- \seriesdefault r163, s82, t38
- \set@mathdelimiter r690, r706
- \set@color B46
- \set@display@protect
.... d3, d225, g7, g14, g37, g64

- `\set@fontsize` . o234, o236, p119, [p132](#)
 - `\set@mathaccent` r507, r515, [r531](#)
 - `\set@mathchar` r565, [r575](#)
 - `\set@mathdelimit` r641, r650, [r702](#)
 - `\set@mathradical` r168, r741
 - `\set@mathsymbol` r549, r557, [r578](#)
 - `\set@simple@size@args`
o313, p302, p317, [p324](#), p345, p363
 - `\set@size@funct@args`
. o318, p307, p309, [p371](#)
 - `\set@size@funct@args@` o319, [p371](#)
 - `\set@typeset@protect` [d225](#),
d244, C179, C205, K480, K482
 - `\setcounter`
. k127, [m2](#), m26, [l21](#), A215,
K1512, K1515, K1518, K1522
 - `\setlanguage` b99
 - `\setlength` [n4](#), z312, z317, z318, z319,
[l25](#), B27, B115, B161, B164,
B209, B254, B255, B256, B271,
B272, B279, B280, B281, C158,
C352, K1528, K1529, K1530,
K1533, K1534, K1538, K1539,
K1540, K1544, K1545, K1546
 - `\SetMathAlphabet`
. o14, q131, q132, [r399](#), t74, t75
 - `\SetMathAlphabet@` r337, r406, [r415](#)
 - `\setminus` t307
 - `\SetSymbolFont` [r254](#), t64, t65, t66
 - `\SetSymbolFont@` r227, r261, [r269](#)
 - `\settodepth` [n6](#), [l25](#)
 - `\settoheight` [n6](#), [l25](#)
 - `\settowidth` [n6](#), [l25](#)
 - `\sf@size` j6,
l256, o193, o459, p282, p286, G260
 - `\sfcode` b152, b153, b154,
b155, b239, i178, k39, N42, N147
 - `\sfdefault` s8, [t29](#)
 - `\sffamily` s6, s7, v16
 - `\sh@ft` [b227](#)
 - `\shapedefault` r164, s83, [t38](#)
 - `\sharp` t254
 - `\shipout` K479
 - `\shortstack` D63
 - `\showboxbreadth` b124, b244
 - `\showboxdepth` b125, b244, o480
 - `\showhyphens` [o472](#)
 - `\showoutput` [b241](#)
 - `\showoverfull` [b240](#), b245, b253
 - `\Sigma` t222
 - `\sigma` t203
 - `\sim` t353, t365
 - `\simeq` t354
 - `\sin` z9
 - `\sinh` z11
 - `\sixt@n` a18,
b16, b54, b55, o23, r43, r102,
r499, r501, r541, r543, r584,
r586, r624, r626, r632, r634,
r677, r679, r685, r687, r727,
r729, r735, r737, D156, D171,
D173, G58, G80, K747, K793,
K929, K1059, K1122, K1305,
K1314, K1370, K1386, K1418
 - `\size@update` p128, p139, p158, [p160](#)
 - `\sizefn@info`
. p308, p310, p318, p346, p364
 - `\skew` t451
 - `\skip` b28,
b49, b65, B226, G246, K266, K366
 - `\skip@` [b41](#),
b189, b191, b192, b194, v87, v90
 - `\skipdef` b45, b49
 - `\slash` [b195](#)
 - `\sldefault` s23, [t34](#)
 - `\sloppy` B189, [J43](#), J48
 - `\sloppypar` J48
 - `\sloppypar (environment)` J48
 - `\slshape` l339, l644, s21, s22, v22
 - `\smallbreak` [b203](#)
 - `\smallint` t275
 - `\smallskip` b204, [i162](#)
 - `\smallskipamount` b203, [i162](#), [i165](#)
 - `\smash` t381, t453, t454, t457, t458, [z95](#)
 - `\smile` t358
 - `\sp` [z142](#)
 - `\sp@n` [C347](#)
 - `\space` [b168](#)
 - `\spacefactor` b193, b194, i67, i72,
i80, i178, i190, i75, i76, G297, G299
 - `\spaceskip` s66
 - `\spadesuit` t258
 - `\span` C351
 - `\split@name` [o274](#), o288, o371, p509, p523
 - `\splitmaxdepth` b131, G275
 - `\splittopskip` b143, G274
 - `\sqcap` t290
 - `\sqcup` t291
 - `\sqrt` [z203](#)
 - `\sqrtsign` t437, z71, z203
 - `\sqsubset` s95
 - `\sqsubseteq` t313
 - `\sqsupset` s96
 - `\sqsupseteq` t314
 - `\SS` l258, l397, N179
 - `\ss` l209, l308, l417, l628, N179
 - `\ssf@size` o194, o460, p282, p288
 - `\stackrel` [z201](#)
- File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
D=lt pictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=ltthyphen.dtx,
N=ltfinal.dtx, O=ltpatch.ltx

`\star` [t311](#)
`\stepcounter`
 ... [m17](#), [o430](#), [r39](#), [x36](#), [z211](#),
 [z254](#), [z306](#), [121](#), [G262](#), [G286](#), [K525](#)
`\stop` [y49](#)
`\stretch` [i197](#)
`\strip@prefix`
 [a58](#), [a175](#), [a246](#), [d201](#), [d280](#), [o415](#)
`\strip@pt` [b231](#),
 [o188](#), [o192](#), [o246](#), [o459](#), [o460](#), [p134](#)
`\strut` [b211](#), [z121](#), [z122](#), [C31](#)
`\strutbox` [b211](#), [p143](#),
 [B247](#), [C168](#), [C169](#), [G275](#), [G282](#)
`\sub@sfcnt` ... [o324](#), [p502](#), [p503](#), [p506](#)
`\subf@sfcnt` ... [o325](#), [p531](#), [p532](#), [p535](#)
`\subparagraphmark` [F126](#)
`\subsectionmark` [F126](#)
`\subset` [t337](#)
`\subseteq` [t339](#)
`\subst@correction` [o60](#), [o66](#)
`\subst@fontshape` [q1](#), [q70](#)
`\subst@size` [p437](#)
`\subsubsectionmark` [F126](#)
`\succ` [t331](#)
`\succeq` [t334](#)
`\sum` [t268](#)
`\sup` [z24](#)
`\suppressfloats` [K1268](#)
`\supset` [t336](#)
`\supseteq` [t338](#)
`\surd` [t240](#)
`\sw@slant` [v73](#), [v83](#)
`\swarrow` [t322](#)
`\symbol` [i127](#), [s54](#)
`\symletters` [i1104](#), [s68](#)
`\symoperators` [t529](#)

T

`\T` [g26](#), [L497](#), [L501](#), [L502](#)
`\t` .. [i237](#), [i588](#), [i693](#), [i884](#), [i1092](#), [i1093](#)
`\t@st@ic` [v72](#), [v76](#)
`\tabbing` [C62](#), [C145](#)
`\tabbingsep` [C121](#), [C123](#), [C142](#)
`\tabcolsep` [C229](#), [C306](#)
`\tabskip` [b222](#), [z138](#),
 [z139](#), [z216](#), [z219](#), [z222](#), [z224](#),
 [z310](#), [z323](#), [z326](#), [z328](#), [C149](#), [C174](#)
`\tabular` [C156](#)
`\tabular*` [C157](#)
`\tabularnewline` [C176](#), [C189](#)
`\tan` [z15](#)
`\tanh` [z17](#)
`\tau` [t204](#)

File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefs.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

`\tc@check@accent`
 ... [1962](#), [i1035](#), [i1036](#), [i1037](#)
`\tc@check@symbol` [1962](#),
 [i1032](#), [i1033](#), [i1034](#), [i1038](#), [i1039](#),
 [i1040](#), [i1041](#), [i1042](#), [i1043](#), [i1044](#),
 [i1045](#), [i1046](#), [i1047](#), [i1048](#), [i1049](#),
 [i1050](#), [i1051](#), [i1052](#), [i1053](#), [i1054](#),
 [i1055](#), [i1056](#), [i1057](#), [i1058](#), [i1059](#),
 [i1060](#), [i1061](#), [i1062](#), [i1063](#), [i1064](#),
 [i1065](#), [i1066](#), [i1067](#), [i1068](#), [i1069](#),
 [i1070](#), [i1071](#), [i1072](#), [i1073](#), [i1074](#),
 [i1075](#), [i1076](#), [i1077](#), [i1078](#), [i1079](#),
 [i1080](#), [i1081](#), [i1082](#), [i1083](#), [i1084](#),
 [i1085](#), [i1086](#), [i1087](#), [i1088](#), [i1089](#)
`\tc@error` [1943](#), [1963](#)
`\tc@errorwarn` [1902](#), [1903](#), [1936](#)
`\tc@fake@euro` [1951](#), [i1031](#)
`\tc@forcedfalse` [1896](#)
`\tc@forcedtrue` [1901](#)
`\tc@subst` [1935](#), [1935](#), [1962](#)
`\tencirc` [u10](#), [D47](#), [D333](#)
`\tencircw` [u10](#), [D49](#)
`\tenln` [u9](#), [D47](#), [D48](#), [D332](#), [D334](#)
`\tenlnw` [u9](#), [D49](#), [D50](#)
`\TeX` [j1](#), [j12](#)
`\text@command` [v8](#), [v29](#)
`\textacutedbl` [1748](#), [1995](#)
`\textascendercompwordmark` . [i698](#), [1978](#)
`\textasciicute` [1798](#), [i1019](#)
`\textasciibreve` [1746](#), [1992](#)
`\textasciicaron` [1747](#), [1993](#)
`\textasciicircum` [1241](#), [1418](#)
`\textasciidieresis` [1786](#), [i1009](#)
`\textasciigrave` [1737](#), [1990](#)
`\textasciimacron` [1793](#), [i1014](#)
`\textasciitilde` [1242](#), [1419](#)
`\textasteriskcentered`
 ... [1222](#), [i564](#), [i708](#), [1985](#)
`\textbackslash` [1223](#), [1420](#), [i565](#)
`\textbaht` [1772](#), [i1079](#)
`\textbar` [1224](#), [1421](#), [i566](#)
`\textbardbl` [1225](#), [i567](#), [i752](#), [1998](#)
`\textbf` [v19](#)
`\textbigcircle` [i576](#), [i725](#), [i1055](#)
`\textblank` [i705](#), [i1040](#)
`\textborn` [i738](#), [i1058](#)
`\textbraceleft` .. [i226](#), [i262](#), [i422](#), [i568](#)
`\textbraceright` . [i227](#), [i263](#), [i423](#), [i569](#)
`\textbrokenbar` [i784](#), [i1007](#)
`\textbullet` [i228](#), [i570](#), [i754](#), [i1000](#)
`\textcapitalcompwordmark` .. [i697](#), [1977](#)
`\textcelsius` [i755](#), [i1001](#)
`\textcent` [i780](#), [i1004](#)
`\textcentoldstyle` [i757](#), [i1066](#)

`\textcircled` 1234, 1238, 1254,
 1255, 1577, 1726, 1884, 11090, 11091
`\textcircledP` 1791, 11087
`\textcolonmonetary` 1759, 11067
`\textcompsubstdefault` 1940, 1942
`\textcompwordmark` 1243, 1424
`\textcopyleft` 1789, 11086
`\textcopyright` . 1238, 1271, 1787, 11010
`\textcurrency` .. 1782, 1880, 1883, 11034
`\textdagger` . 1230, 1266, 1572, 1750, 1996
`\textdaggerdbl` 1229, 1267, 1571, 1751, 1997
`\textdblhyphen` 1709, 11041
`\textdblhyphenchar` 1745, 11064
`\textdegree` 1794, 11015
`\textdied` 1740, 11060
`\textdiscount` 1774, 11081
`\textdiv` 1811, 11029
`\textdivorced` 1739, 11059
`\textdollar` 1210, 1261, 1337,
 1425, 1642, 1706, 1983, 11096, 11098
`\textdollaroldstyle` 1756, 11065
`\textdong` 1768, 11076
`\textdownarrow` 1736, 11057
`\texteightoldstyle` 1719, 11050
`\textellipsis` 1250, 1275
`\textemdash` 1211, 1309, 1426, 1629
`\textendash` 1212, 1310, 1427, 1630
`\textestimated` 1775, 1883, 11033
`\texteuro` 1809, 1881, 11030, 11031
`\textexclamdown`
 1213, 1311, 1313, 1428, 1631
`\textfiveoldstyle` 1716, 11047
`\textfloatsep`
 K546, K559, K1400, K1449, K1525
`\textflorin` 1758, 11002
`\textfont` p291, z148
`\textfont@name` p285, p291
`\textfouroldstyle` 1715, 11046
`\textfraction` .. K1260, K1392, K1519
`\textfractionsolidus` 1710, 1986
`\textgravedbl` 1749, 1994
`\textgreater` 1236, 1429, 1587
`\textguarani` 1762, 11070
`\textheight` k16,
 k17, G155, G156, G159, G182,
 K59, K173, K174, K222, K323,
 K350, K524, K581, K612, N13, N14
`\texthyphen` 1215, 1316, 1431, 1633
`\texthyphenchar` . 1214, 1315, 1430, 1632
`\textinterrobang` 1766, 11074
`\textinterrobangdown` 1767, 11075
`\textit` v21
`\textlangle` 1721, 11052
`\textlbrackdbl` 1733, 1988
`\textleaf` 1741, 11061
`\textleftarrow` 1703, 11038
`\textless` 1235, 1432, 1586
`\textlira` 1764, 11072
`\textlnot` 1790, 11012
`\textlquill` 1778, 11084
`\textmarried` 1742, 11062
`\textmd` v19
`\textmho` 1724, 11054
`\textminus` 1722, 1987
`\textmu` 1799, 11020
`\textmusicalnote` 1743, 11063
`\textnaira` 1761, 11069
`\textnineoldstyle` 1720, 11051
`\textnormal` v15
`\textnumero` 1773, 11080
`\textogonekcentered` .. 1383, 1555, 1556
`\textohm` 1732, 1882, 11032
`\textonehalf` 1807, 11026
`\textoneoldstyle` 1712, 11043
`\textonequarter` 1806, 11025
`\textonesuperior` 1803, 11023
`\textopenbullet` 1776, 11082
`\textordfeminine` ... 1259, 1788, 11011
`\textordmasculine` .. 1260, 1804, 11024
`\textparagraph`
 1231, 1264, 1573, 1800, 11021
`\textperiodcentered`
 1232, 1574, 1801, 11022
`\textpertenthousand`
 1387, 1770, 11077, 11100
`\textperthousand` 1385, 1753, 1999, 11099
`\textpeso` 1763, 11071
`\textpilcrow` 1771, 11078
`\textpm` 1795, 11016
`\textquestiondown`
 1216, 1312, 1314, 1433, 1634
`\textquotedbl` 1436
`\textquotedblleft` 1217, 1317, 1434, 1635
`\textquotedblright` 1218, 1318, 1435, 1636
`\textquoteleft` .. 1219, 1319, 1437, 1637
`\textquoteright` . 1220, 1320, 1438, 1638
`\textquotesingle` 1707, 1984
`\textquotestraightbase` ... 1699, 1979
`\textquotestraightdblbase` . 1700, 1980
`\textrangle` 1723, 11053
`\textrbrackdbl` 1734, 1989
`\textrecipe` 1765, 11073
`\textreferencemark` 1802, 11088
`\textregistered` 1254, 1255, 1792, 11013
`\textrightarrow` 1704, 11039
`\textrm` v15
`\textrquill` 1779, 11085
`\textsc` v21
File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfontcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=ltthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx

`\textsection` [1233](#), [1265](#), [1439](#), [1575](#), [1785](#), [11008](#)
`\textservicemark` [1777](#), [11083](#)
`\textsevenoldstyle` [1718](#), [11049](#)
`\textsf` [v15](#)
`\textsixoldstyle` [1717](#), [11048](#)
`\textsl` [v21](#)
`\textsterling` [1221](#), [1273](#), [1344](#),
[1440](#), [1649](#), [1781](#), [11005](#), [11095](#), [11097](#)
`\textstyle` [j15](#), [t377](#), [z63](#)
`\textsuperscript` [1257](#), [1259](#), [1260](#), [G257](#)
`\textsurd` [1805](#), [11089](#)
`\TextSymbolUnavailable` [13](#), [1605](#)
`\textthreeoldstyle` [1714](#), [11045](#)
`\textthreequarters` [1808](#), [11027](#)
`\textthreequartersemdash` .. [1702](#), [1982](#)
`\textthreesuperior` [1797](#), [11018](#)
`\texttildebelow` [1744](#), [1991](#)
`\texttimes` [1810](#), [11028](#)
`\texttrademark` [1257](#), [1769](#), [11003](#)
`\texttt` [v15](#)
`\texttwelveudash` [1701](#), [1981](#)
`\texttwooldstyle` [1713](#), [11044](#)
`\texttwosuperior` [1796](#), [11017](#)
`\textunderscore` [1244](#), [1269](#), [1441](#)
`\textup` [v21](#)
`\textuparrow` [1735](#), [11056](#)
`\textvisiblespace` [1246](#), [1442](#)
`\textwidth` [k18](#),
[B214](#), [G164](#), [K60](#), [K125](#), [K149](#),
[K166](#), [K509](#), [K519](#), [K1479](#), [N14](#)
`\textwon` [1760](#), [11068](#)
`\textyen` [1783](#), [11006](#)
`\textzerooldstyle` [1711](#), [11042](#)
`\tf@size` [o193](#), [o458](#), [p282](#), [p284](#)
`\TH` [1398](#), [N179](#)
`\th` [1443](#), [N179](#)
`\thanks` [330](#), [F9](#)
`thebibliography` (environment) .. [359](#)
`\theequation` ... [z200](#), [z212](#), [z271](#), [z307](#)
`\thefootnote` . [G251](#), [G287](#), [G292](#), [G312](#)
`\thempfn` [B216](#),
[G263](#), [G268](#), [G303](#), [G308](#), [G311](#)
`\thempfootnote` [B216](#), [G253](#)
`\thepage` [I23](#), [k73](#), [w6](#), [x14](#), [x34](#), [F143](#),
[H15](#), [H32](#), [J14](#), [K192](#), [K223](#), [K1165](#)
`\Theta` [t218](#)
`\theta` [t194](#)
`\thicklines` [D46](#)
`\thickmuskip` [t532](#), [z146](#)
`\thinlines` [D46](#)
`\thinmuskip` [i188](#), [t530](#), [z144](#), [z147](#)
`\thinspace` [i188](#), [i198](#), [z119](#), [z148](#)
`\thispagestyle` [J6](#)
File Key: `a=ltldirchk.dtx`, `b=ltplain.dtx`, `c=ltvers.dtx`, `d=ltdefns.dtx`,
`e=ltalloc.dtx`, `f=ltcntrl.dtx`, `g=lterror.dtx`, `h=ltpar.dtx`, `i=ltspace.dtx`,
`j=ltlogos.dtx`, `k=ltfiles.dtx`, `l=ltoutenc.dtx`, `m=ltcounts.dtx`, `n=ltlength.dtx`,
`o=ltfssbas.dtx`, `p=ltfssstrc.dtx`, `q=ltfsscmp.dtx`, `r=ltfssdcl.dtx`, `s=ltfssini.dtx`,
`t=fontdef.dtx`, `u=preload.dtx`, `v=ltfntcmd.dtx`, `w=ltpageno.dtx`, `x=ltxref.dtx`,
`y=ltmiscen.dtx`, `z=ltmath.dtx`, `A=ltlists.dtx`, `B=ltboxes.dtx`, `C=lttab.dtx`,
`D=ltpictur.dtx`, `E=ltthm.dtx`, `F=ltsect.dtx`, `G=ltfloat.dtx`, `H=ltidxglo.dtx`,
`I=ltbibl.dtx`, `J=ltpage.dtx`, `K=ltoutput.dtx`, `L=ltclass.dtx`, `M=lthyphen.dtx`,
`N=ltfinal.dtx`, `O=ltpatch.ltx`

`\thr@@` [b16](#), [p58](#), [p208](#),
[p214](#), [p227](#), [p234](#), [p241](#), [p246](#),
[z223](#), [z327](#), [A222](#), [A233](#), [D165](#),
[D166](#), [D168](#), [D169](#), [D201](#), [D223](#)
`\tilde` [t427](#)
`\time` [a126](#), [a130](#)
`\times` [t310](#)
`\title` [330](#), [F3](#)
`\to` [t350](#)
`\today` [a131](#), [a135](#), [a143](#), [a146](#), [F8](#)
`\toks` [b31](#),
[b53](#), [r372](#), [r373](#), [r383](#), [r392](#), [N229](#)
`\toks@` .. [b41](#), [i170](#), [i171](#), [i176](#), [o123](#),
[o127](#), [o129](#), [o132](#), [r183](#), [r187](#),
[r193](#), [r196](#), [r201](#), [r242](#), [r243](#),
[r245](#), [r246](#), [r276](#), [r278](#), [r282](#),
[r299](#), [r302](#), [r361](#), [r373](#), [r374](#),
[r375](#), [r421](#), [r423](#), [r429](#), [r437](#),
[r441](#), [r453](#), [r456](#), [r459](#), [r467](#),
[r469](#), [r794](#), [r796](#), [r798](#), [r801](#),
[r803](#), [r806](#), [r809](#), [r841](#), [r842](#),
[L128](#), [L129](#), [L131](#), [L132](#), [L370](#), [L371](#)
`\toksdef` [b46](#), [b53](#)
`\tolerance` [b103](#), [o480](#), [J44](#), [J52](#)
`\top` [t241](#)
`\topfigrule` [K545](#), [K1547](#)
`\topfraction` [G171](#), [K1513](#)
`\topmargin` [K52](#), [K503](#)
`\topsep` [z312](#), [A2](#), [A59](#)
`\topskip` [b142](#), [k49](#), [A1](#), [K109](#)
`\totalheight` [B17](#), [B18](#), [B19](#)
`\tr@ce` ... [K188](#), [K215](#), [K274](#), [K303](#),
[K348](#), [K407](#), [K422](#), [K423](#), [K424](#),
[K425](#), [K436](#), [K437](#), [K438](#), [K439](#),
[K440](#), [K450](#), [K599](#), [K616](#), [K632](#),
[K634](#), [K720](#), [K724](#), [K736](#), [K737](#),
[K738](#), [K739](#), [K745](#), [K748](#), [K756](#),
[K760](#), [K771](#), [K776](#), [K781](#), [K782](#),
[K783](#), [K784](#), [K791](#), [K794](#), [K802](#),
[K811](#), [K817](#), [K822](#), [K827](#), [K833](#),
[K834](#), [K839](#), [K844](#), [K845](#), [K846](#),
[K853](#), [K857](#), [K862](#), [K866](#), [K871](#),
[K882](#), [K883](#), [K885](#), [K903](#), [K912](#),
[K918](#), [K927](#), [K930](#), [K936](#), [K946](#),
[K950](#), [K959](#), [K965](#), [K970](#), [K975](#),
[K979](#), [K983](#), [K984](#), [K991](#), [K995](#),
[K999](#), [K1006](#), [K1015](#), [K1019](#),
[K1023](#), [K1024](#), [K1028](#), [K1029](#),
[K1035](#), [K1041](#), [K1051](#), [K1057](#),
[K1061](#), [K1062](#), [K1067](#), [K1068](#),
[K1074](#), [K1077](#), [K1078](#), [K1079](#),
[K1086](#), [K1087](#), [K1088](#), [K1096](#),
[K1100](#), [K1112](#), [K1113](#), [K1120](#),
[K1123](#), [K1131](#), [K1135](#), [K1139](#),

- K1140, K1144, K1145, K1201,
 K1206, K1217, K1227, K1234,
 K1244, K1289, K1302, K1303,
 K1307, K1310, K1312, K1315,
 K1318, K1320, K1361, K1368,
 K1373, K1379, K1384, K1388,
 K1394, K1402, K1403, K1410,
 K1415, K1420, K1422, K1428,
 K1430, K1437, K1464, K1465,
 K1474, K1490, K1495, K1503
 \tracefloats K1244
 \tracefloatvals K1244
 \tracingall b248
 \tracingcommands b249
 \tracingfonts p17, p54, p58,
 p86, p116, p125, p148, p178,
 p192, p208, p214, p227, p234,
 p241, p246, p255, p268, p276, p279
 \tracinglostchars b119, b250
 \tracingmacros b251
 \tracingoff p116, p276
 \tracingon p117, p277
 \tracingonline b240
 \tracingoutput b243
 \tracingpages b250
 \tracingparagraphs b251
 \tracingrestores b251
 \tracingstats b249, N2
 \triangle t247
 \triangleleft t276, t390
 \triangleright t277, t390
 \trivlist y73, y80, y86,
 y100, z297, A89, C69, E35, E37
 \try@load@fontshape
 o291, o331, p510, r132, r149
 \try@simple@size ... o312, p312, p453
 \try@simples .. p401, p405, p412, p421
 \try@size@range o320, p101, p312, p392
 \try@size@substitution .. p103, p397
 \try@sizes o20, o464
 \tryif@simple p414, p420
 \tryis@simple p419
 \ttdefault s11, t29
 \ttfamily s9, s10, v17, y120
 \tw@ b16
 \two@digits a33,
 a132, a133, d2, p498, L423, L454
 \twocolumn K147
 \type@restoreinfo p156, p161
 \typein 29, 29, d18
 \typeout a20, a63,
 a119, a144, a146, a158, a173,
 a180, a191, a204, a217, a230,
 File Key: a=ltldirchk.dtx, b=ltplain.dtx, c=ltvers.dtx, d=ltdefns.dtx,
 e=ltalloc.dtx, f=ltcntrl.dtx, g=lterror.dtx, h=ltpar.dtx, i=ltspace.dtx,
 j=ltlogos.dtx, k=ltfiles.dtx, l=ltoutenc.dtx, m=ltcounts.dtx, n=ltlength.dtx,
 o=ltfssbas.dtx, p=ltfssstrc.dtx, q=ltfsscmp.dtx, r=ltfssdcl.dtx, s=ltfssini.dtx,
 t=fontdef.dtx, u=preload.dtx, v=ltfntcmd.dtx, w=ltpageno.dtx, x=ltxref.dtx,
 y=ltmiscen.dtx, z=ltmath.dtx, A=ltlists.dtx, B=ltboxes.dtx, C=lttab.dtx,
 D=ltpictur.dtx, E=ltthm.dtx, F=ltsect.dtx, G=ltfloat.dtx, H=ltidxglo.dtx,
 I=ltbibl.dtx, J=ltpage.dtx, K=ltoutput.dtx, L=ltclass.dtx, M=lthyphen.dtx,
 N=ltfinal.dtx, O=ltpatch.ltx
- a244, c13, c28, 29, d3, d22, g77,
 k65, k172, k173, k178, k215,
 k225, k228, o285, s106, s113,
 s124, s135, s145, t9, t44, H8,
 H25, K1245, L95, N64, N183,
 N190, N202, N203, N211, O16
- ## U
- \u l192, l288,
 l368, l446, l453, l473, l480, l601
 \uccode N30, N40,
 N49, N51, N55, N57, N135,
 N145, N154, N156, N160, N162
 \uchyph b120
 \unboldmath s51
 \UndeclareTextCommand l146, l1095,
 l1096, l1097, l1098, l1099, l1100
 \undefinedpagestyle J4, J8
 \underbar b210
 \underbrace t447
 \underline b210, 272, B259, B260
 \unhcopy b212, C313, D391
 \unitlength
 B33, B43, D5, D20, D21, D22,
 D23, D30, D31, D34, D35, D42,
 D78, D136, D189, D191, D204,
 D209, D211, D226, D228, D231,
 D268, D298, D309, D345, D346,
 D348, D349, D352, D353, D355,
 D356, D365, D367, D369, D371, N6
 \unlhd s98
 \unpenalty v98, y116
 \unrestored@protected@xdef
 d227, G268, G292, G308, J21, J35
 \unrhd s100
 \unvcopy z123
 \Uparrow t488
 \uparrow t482
 \upbracefill t450, t466
 \update@ucllc@with@cyrillic
 l814, l842, l866, l872
 \updefault s20, t34, t41
 \Updownarrow t492
 \updownarrow t486
 \uplus t292
 \upper@bound .. p339, p340, p341, p355
 \uppercase N169
 \upshape l341, l579,
 l646, l728, s18, s19, s32, s77, v24
 \Upsilon t223
 \upsilon t205
 \use@mathgroup
 o424, o442, o444, p253,
 r50, r343, r445, r448, r819, r843

<code>\usebox</code>	B81	<code>\vfil</code>	b199 ,
<code>\usecounter</code>	A215 , A228		1956 , 1958 , D287 , K144 , K335 , K506
<code>\usefont</code>	o53 , o230 , s67 , s80	<code>\vfilneg</code>	b199
<code>\usepackage</code>	L202 , L232	<code>\vfuzz</code>	b128 , J47 , J54
<code>\UseTextAccent</code>		<code>\vgl@</code>	b189 , b190
	1115 , 1143 , 1963 , 11091 , 11093	<code>\vglue</code>	b189
<code>\UseTextSymbol</code> .	1115 , 1141 , 1962 , 11031	<code>\vline</code>	C334
V			
<code>\v</code>	1193 , 1289 , 1367 , 1449 ,	<code>\voidb@x</code>	b81 , b220 , n7
	1450 , 1451 , 1455 , 1457 , 1460 , 1462 ,	<code>\vphantom</code>	z75 , z94
	1464 , 1470 , 1476 , 1477 , 1478 , 1482 ,	<code>\vrule</code> .	b193 , i193 , 1247 , 1249 , p144 ,
	1484 , 1487 , 1489 , 1491 , 1497 , 1602		t464 , t465 , t467 , t468 , B90 ,
<code>\v@false</code>	z78		B92 , B129 , B136 , B258 , B288 ,
<code>\v@true</code>	z77 , z79		C168 , C201 , C315 , C334 , D127 ,
<code>\vadjust</code> .	i10 , i38 , i47 , i139 , i155 , G126		D177 , D180 , D196 , D203 , D218 ,
<code>\valign</code>	1955		D225 , D287 , D375 , K1189 , K1483
<code>\value</code>	19 , m14 , 121	<code>\vspace</code>	1132 , i162 , i163 , i164
<code>\varbigtriangledown</code>	t280	<code>\vsplit</code>	K306
<code>\varbigtriangleup</code>	t281	W	
<code>\varepsilon</code>	j15 , t210	<code>\warn@rel@i</code>	q15 ,
<code>\varphi</code>	t215		q19 , q71 , q75 , q80 , q85 , q109 , q131
<code>\varpi</code>	t212	<code>\wedge</code>	t282 , t283
<code>\varrho</code>	t213	<code>\widehat</code>	t435
<code>\varsigma</code>	t214	<code>\widetilde</code>	t434
<code>\vartheta</code>	t211	<code>\widowpenalty</code>	b112
<code>\vbadness</code>	b105	<code>\width</code>	B14
<code>\vdash</code>	t318	<code>\wlog</code> ...	a47 , b40 , b61 , b69 , L97 , N236
<code>\vdots</code>	t418	<code>\wp</code>	t232
<code>\vec</code>	t432	<code>\wr</code>	t296
<code>\vector</code>	g262 , D133	<code>\wrong@fontshape</code>	o295 , o385
<code>\vee</code>	t284 , t285	X	
<code>\verb</code>	y130 , y132 , y141	<code>\x</code>	o250 , o251
<code>\verb@balance@group</code>		<code>\x@protect</code>	d205 , d216
	y124 , y125 , y140 , y142	<code>\Xi</code>	t220
<code>\verb@egroup</code> ..	y125 , y129 , y140 , y143	<code>\xi</code>	t200
<code>\verb@eol@error</code>	y126 , y134	Y	
<code>\verbatim</code>	y118	<code>\year</code>	a132 , c6 , L454
<code>verbatim*</code> (environment)	y121	<code>\yxdim</code>	D323
<code>\verbatim@font</code>	y114 , y120 , y135	Z	
<code>\verbatim@nolig@list</code>	y145 , y151	<code>\Z</code>	N44 , N122 , N149
<code>\version@elt</code> r9 , r22 , r23 , r180 , r181 ,		<code>\z</code>	N33 , N123 , N138
r225 , r245 , r336 , r374 , r466 , r799		<code>\z@</code>	b81
<code>\version@list</code>	r7 ,	<code>\z@skip</code>	b81
	r12 , r23 , r173 , r181 , r230 , r251 ,	<code>\zap@space</code>	k84 , L117 , L265 , L282 , L298
	r270 , r341 , r386 , r416 , r471 , r812	<code>\zeta</code>	t192
<code>\Vert</code>	t477 , t479		
<code>\vert</code>	t480		

File Key: a=lt`dirchk`.dtx, b=lt`plain`.dtx, c=lt`vers`.dtx, d=lt`defs`.dtx, e=lt`alloc`.dtx, f=lt`cntrl`.dtx, g=lt`error`.dtx, h=lt`par`.dtx, i=lt`space`.dtx, j=lt`logos`.dtx, k=lt`files`.dtx, l=lt`outenc`.dtx, m=lt`counts`.dtx, n=lt`length`.dtx, o=lt`fssbas`.dtx, p=lt`fssstrc`.dtx, q=lt`fsscmp`.dtx, r=lt`fssdcl`.dtx, s=lt`fssini`.dtx, t=lt`fontdef`.dtx, u=lt`preload`.dtx, v=lt`fntcmd`.dtx, w=lt`pageno`.dtx, x=lt`txref`.dtx, y=lt`miscen`.dtx, z=lt`math`.dtx, A=lt`lists`.dtx, B=lt`boxes`.dtx, C=lt`tab`.dtx, D=lt`pictur`.dtx, E=lt`thm`.dtx, F=lt`sect`.dtx, G=lt`float`.dtx, H=lt`idxglo`.dtx, I=lt`bibl`.dtx, J=lt`page`.dtx, K=lt`output`.dtx, L=lt`class`.dtx, M=lt`hyphen`.dtx, N=lt`final`.dtx, O=lt`patch`.ltx