The $\LaTeX 2_{\varepsilon}$ Sources

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

1995/06/01

Contents

File a
ltdirchk.dtx

1 LaTeX System Dependant Initialisations

This file implements the semi-automatic determination of various system dependant 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, 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, `/`.

`\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 TeX 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. LaTeX does not set this variable exactly, the automatic tests set it to:
2 for any version, v, $v < 3.0$
3 for any version, v, $3.0 \leq v \leq 3.14$
⟨*undefined*⟩ otherwise.
However these values are accurate enough for LaTeX to take appropriate action for these old TeXs.

If your TeX is older than version 3.141, then you should define `\@TeXversion` (using `\def`) to be the version number. If you do not do this[1] , LaTeX will not work around a bug in old TeX 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.
 ...

l.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.
! .
 ...

l.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 TeX, 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.

[1] Actually if you TeX is really old, version 2, LaTeX can detect this, and set `\@TeXversion` to 2 if it is not set in the `cfg` file

2.1 INITEX

```
⟨*dircheck⟩
⟨*initex⟩
⟨initex⟩\ifnum\catcode`\{=1
⟨initex⟩  \errmessage
⟨initex⟩  {LaTeX must be made using an initex with no format preloaded}
⟨initex⟩\fi
\catcode`\{=1
\catcode`\}=2
\catcode`\#=6
\catcode`\^=7
\chardef\active=13
\catcode`\@=11
\countdef\count@=255
\let\bgroup={ \let\egroup=}
\ifx\@@input\@undefined\let\@@input\input\fi
\ifx\@@end\@undefined\let\@@end\end\fi
\chardef\@inputcheck0
\chardef\sixt@@n=16
\newlinechar`\^^J
\def\typeout{\immediate\write17}
\def\dospecials{\do\ \do\\\do\{\do\}\do\$\do\&%
  \do\#\do\^\do\_\do\%\do\~}
\def\@makeother#1{\catcode`#1=12\relax}
\def\space{ }
\def\@tempswafalse{\let\if@tempswa\iffalse}
\def\@tempswatrue{\let\if@tempswa\iftrue}
\let\if@tempswa\iffalse
\def\loop#1\repeat{\def\iterate{#1\relax\expandafter\iterate\fi}%
  \iterate \let\iterate\relax}
\let\repeat\fi
⟨/initex⟩
```

2.2 Some bits of 2e

```
⟨*2ekernel⟩
\def\two@digits#1{\ifnum#1<10 0\fi\number#1}
\long\def\@firstoftwo#1#2{#1}
\long\def\@secondoftwo#1#2{#2}

\def\ProvidesFile#1{%
  \begingroup
    \catcode`\ 10 %
    \@makeother\/%
    \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}}
\def\@providesfile#1[#2]{%
    \wlog{File: #1 #2}%
    \expandafter\xdef\csname ver@#1\endcsname{#2}%
  \endgroup}
```

```
\def\@empty{}
\catcode`\%=12
\def\@percentchar{%}
\catcode`\%=14
\let\@currdir\@undefined
\let\input@path\@undefined
\let\filename@parse\@undefined
```

`\strip@prefix`

```
\def\strip@prefix#1>{}
⟨/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.

```
⟨*docstrip⟩
\openin15=texsys.cfg
\ifeof15
\typeout{** Writing a default texsys.cfg}
\immediate\openout15=texsys.cfg
\begingroup
\catcode`\^^M\active%
\let^^M\par%
\def\reserved@a#1^^M{%
 \def\reserved@b{#1}%
 \ifx\reserved@b\reserved@c\endgroup\else%
      \immediate\write15{#1}%
      \expandafter\reserved@a\fi}%
\def\reserved@d#1START^^M{\let\do\@makeother\dospecials\reserved@a}%
\catcode`\%=12
\def\reserved@c{%END}
\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 `dircheck.dtx`. You *are* allowed to edit this file to add definitions of these macros appropriate to your system.

David Carlisle 1994/11/17

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, TEXwill 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 willcover 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 `dircheck.dtx`, and so this file may be empty. The definitions below are therefore just for information.

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

```
% \def\@currdir{./}
% \def\input@path{%
%   {/usr/local/lib/tex/inputs/distrib/}%
%   {/usr/local/lib/tex/inputs/contrib/}%
%   {/usr/local/lib/tex/inputs/local/}%
% }
```

3.4 MSDOS (emtex)

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

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

```
% \def\@currdir{./}
% \def\input@path{%
%   {c:/tex/inputs/distrib/}%
%   {c:/tex/inputs/contrib/}%
%   {c:/tex/inputs/local/}%
% }
```

3.6 VMS (???)

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

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

```
% \def\@currdir{[]}
% \def\input@path{%
%   {texinputs:}%
%   {[SOME.TEX.DIRECTORY]}%
% }
```

3.8 MACINTOSH (OzTeX 1.6)

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

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

```
% \def\@currdir{:}
% \def\input@path{%
%   {Hard-Disk:Applications:TeX:TeX-inputs:}%
%   {Hard-Disk:Applications:TeX:My-inputs:}%
% }
```

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:

```
% \def\@dir#1#2 {%
```

```
%    \@d@r{#1}#2..\@nil}
% \def\@d@r#1#2.#3.#4\@nil{%
%    <\ifx\@dir#1\@dir\else#1\ifx\@dir#3\@dir\else.\fi\fi#3>#2 }
%
% \def\@currdir{\@dir{}}
% \def\input@path{%
%    {\@dir{area.one}}%
%    {\@dir{area.two}}%
% }
```

END

```
\immediate\closeout15
```

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

```
\else
\typeout{** Using the existing texsys.cfg}
\closein15
\input texsys.cfg
\fi
⟨/docstrip⟩
```

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

```
⟨dircheck⟩\input texsys.cfg
```

4 Setting `\@currdir`

`\@currdir`
`\IfFileExists`

This is a local definition of `\IfFileExists`. It tries to relocate `texsxys.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.

```
\begingroup
\count@\time
\divide\count@ 60
\count2=-\count@
\multiply\count2 60
\advance\count2 \time
```

`\today`

The current date and time stamp.

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

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

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

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

```
\def\IfFileExists#1#2#3{%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck
     #3\relax
  \else
    \read\@inputcheck to \reserved@a
    \ifx\reserved@a\today
      \typeout{#1 found}#2\relax
    \else
      \typeout{BAD: old file \reserved@a (should be \today)}%
      #3\relax
    \fi
  \fi
  \closein\@inputcheck}
```

```
\endlinechar=-1
```

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

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

```
  \ifx\@currdir\@undefined
    \global\let\@currdir\@empty
    \typeout{^^J***^^J%
      No syntax for the current directory could be found^^J%
      ***}%
  \fi
```

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

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

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

```
\fi
\immediate\closeout15 %
\endgroup
\typeout{^^J***^^J%
         \noexpand\@currdir set to:
           \expandafter\strip@prefix\meaning\@currdir.^^J%
         ***}
```

Stop here if the file is being used unstripped.

```
⟨*docstrip⟩
\relax\endinput
⟨/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 LaTeX 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 LaTeX 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.

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

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

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

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

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

6 Filename Parsing

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

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

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

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

Search for the last /.

```
    \def\filename@path#1/#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1/}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

```
  \else\def\reserved@a{[]}\ifx\@currdir\reserved@a
```

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

```
    \typeout{^^J***^^JDefining VMS style filename parser.^^J***}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@path#1]\\}
```

Search for the last].

```
    \def\filename@path#1]#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1]}%
         \def\reserved@a{\filename@path#2\\}%
      \fi
      \reserved@a}
```

```
  \else\def\reserved@a{:}\ifx\@currdir\reserved@a
```

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

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

Search for the last :.

```
    \def\filename@path#1:#2\\{%
      \ifx\\#2\\%
         \def\reserved@a{\filename@simple#1.\\}%
      \else
         \edef\filename@area{\filename@area#1:}%
         \def\reserved@a{\filename@path#2\\}%
```

```
      \fi
      \reserved@a}
  \else
```

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

```
    \typeout{^^J***^^JDefining generic filename parser.^^J***}
    \def\filename@parse#1{%
      \let\filename@area\@empty
      \expandafter\filename@simple#1.\\}
  \fi\fi\fi
```

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

```
  \def\filename@simple#1.#2\\{%
    \ifx\\#2\\%
       \let\filename@ext\relax
    \else
       \edef\filename@ext{\filename@dot#2\\}%
    \fi
    \edef\filename@base{#1}}
```

Remove a final dot, added earlier.

```
  \def\filename@dot#1.\\{#1}
\else
```

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

```
  \typeout{^^J***^^J%
    \noexpand\filename@parse was defined in texsys.cfg:^^J%
    \expandafter\strip@prefix\meaning\filename@parse.^^J%
    ***}
\fi
```

7 TeX Versions

`\@TeXversion` TeX 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 TeX3.14 would be detected as TeX3, but LaTeX only needs to take account of TeX's older than 3, or between 3 and 3.14.

```
\ifx\@TeXversion\@undefined
  \ifx\undefined\inputlineno
    \def\@TeXversion{2}
  \else
   {\catcode`\^^J=\active
     \def\reserved@a#1#2\@@{\if#1\string^3\fi}
     \edef\reserved@a{\expandafter\reserved@a\string^^J\@@}
     \ifx\reserved@a\@empty\else\gdef\@TeXversion{3}\fi}
```

```
  \fi
\fi
```

⟨/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.l2e`, the format may be checked by running the file `ltxcheck.tex`.

⟨*ltxcheck⟩

```
\NeedsTeXFormat{LaTeX2e}
\makeatletter
\typeout{^^J%
LaTeX2e installation check file^^J%
==============================}

\typeout{^^J%
 Before running this file through LaTeX2e you should have installed^^J%
 the Standard LaTeX files in their final `system' directories.^^J%
 This file should *not* be run in a directory that contains article.cls}
\def\pause{%
  \typeout{}%
  \message{** Hit return to continue: }%
  \read -1  to \xxx
  \typeout{}}
\typeout{^^J%
  After certain tests, LaTeX will pause so that you can read the^^J%
  output without it scrolling off the screen.^^J%
  When you are ready just hit <return> and LaTeX will continue.^^J%
  When LaTeX pauses, you will see a prompt like the one below.^^J^^J%
  If a test fails, a message will be displayed followed by^^J%
  an error message starting `! BAD'.^^J%
  LaTeX will quit if you try to scroll past some error messages.}
\pause
\typeout{^^J%
  Checking the current directory syntax^^J%
  ====================================}
\newif\iftest\testfalse
\ifx\@currdir\@undefined
  \typeout{^^J%
  \noexpand\@currdir is undefined !!^^J%
  Something is seriously wrong with the LaTeX2e initialisation.^^J%
  Either you have corrupted files or this is a LaTeX bug.}
  \errmessage{BAD LaTeX2e system!!}
```

```
  \expandafter\@@end
\fi
\ifx\@currdir\@empty
  \typeout{^^J%
  \noexpand\@currdir is defined to be empty.^^J%
  This means that LaTeX can not distinguish between a file^^J%
  aaaaa.tex^^J%
  that exists in the current directory, and  a file aaaaa.tex^^J%
  in another directory.^^J%
  It may be that this Operating System has no concept of `directory'^^J%
  in which case the setting is correct. If however it is possible to^^J%
  uniquely refer to a file then a suitable definition of
    \noexpand\@currdir^^J%
  should be added to texsys.cfg, and the format remade.}
  \pause
\else
  \typeout{^^J%
\noexpand\@currdir is defined as
    \expandafter\strip@prefix\meaning\@currdir^^J%
  (Testing...)}
\begingroup
\endlinechar=-1
\count@\time
\divide\count@ 60
\count2=-\count@
\multiply\count2 60
\advance\count2 \time
\edef\today{%
  \the\year/\two@digits{\the\month}/\two@digits{\the\day}:%
    \two@digits{\the\count@}:\two@digits{\the\count2}}
  \immediate\openout15=ltxcheck.aux
  \immediate\write15{\today^^J}
  \immediate\closeout15 %
  \openin\@inputcheck\@currdir ltxcheck.aux %
  \ifeof\@inputcheck
    \typeout{\@currdir ltxcheck.aux  not found}%
  \else
    \read\@inputcheck to \reserved@a
    \ifx\reserved@a\today
      \typeout{\@currdir ltxcheck.aux found}
      \testtrue
    \else
      \typeout{BAD: old file \reserved@a(should be \today)}%
      \testfalse
    \fi
  \fi
  \closein\@inputcheck
```

```
  \iftest
    \endgroup
    \typeout{\noexpand \@currdir OK!}
  \else
  \endgroup
  \typeout{^^J%
    The LaTeX2e installation has defined \noexpand\@currdir^^J%
    to be \expandafter\strip@prefix\meaning\@currdir.^^J%
    This appears to be incorrect.^^J%
    You should add a correct definition to texsys.cfg^^J%
    and rebuild the format.}
  \errmessage{BAD LaTeX2e system!!}
  \expandafter\expandafter\expandafter\@@end
  \fi
  \pause
\fi
\typeout{^^J%
  Checking the input path^^J%
  ======================^^J}
\begingroup
\let\input@path\@undefined
\ifx\@currdir\@empty\else
  \IfFileExists{\@currdir article.cls}
   {\typeout{%
      article.cls appears to be in current directory!^^J^^J%
      If this is the case, install article.cls into a^^J%
      `standard input directory'^^J%
      and copy ltxcheck.tex to another directory before^^J%
      processing with LaTeX.^^J%
      ^^J%
      If article.cls is not in the current directory,^^J%
      then you need to edit texsys.cfg.^^J%
      Read the comments in that file. If nothing else works, add:^^J%
      \string\let\string\@currdir\string\@empty^^J}%
    \errhelp{Move files, or edit texsys.cfg}
    \def\ArticleClassFoundInCurrentDirectory{%
      This file should not be run in a `standard input directory'}
    \errmessage{BAD: \ArticleClassFoundInCurrentDirectory}}
    {}
\fi
\endgroup
\IfFileExists{article.cls}
  {\typeout{input path OK!}}
  {\typeout{^^J%
     LaTeX claims that article.cls is not on the system.^^J%
     Either LaTeX has been incorrectly installed, or the
     \noexpand\input@path^^J%
     is incorrect. A correct definition should be added to^^J%
```

```
     texsys.cfg, and the format remade.}
   \pause
   \typeout{^^J%
     Typical definitions of \noexpand\input@path include:^^J^^J%
     \string\let\string\input@path=\noexpand\@undefined
      (the default definition)^^J^^J%
     \string\def\string\input@path{\@percentchar^^J
       {/usr/lib/tex/inputs/} {/usr/local/lib/tex/inputs/} }^^J^^J%
     \string\def\string\input@path{\@percentchar^^J
       {c:/tex/inputs/} {a:/} }^^J^^J%
     \string\def\string\input@path{\@percentchar^^J
       {tex_inputs:} {[SOMEWHERE.TEX.INPUTS]} }^^J}%
   \pause
   \typeout{^^J%
     Note that \noexpand\input@path should be undefined
       unless your^^J%
     TeX installation does not make
       \noexpand\openin and \noexpand\input^^J%
     search the same directories.^^J%
     If \noexpand\input@path is defined, entries should be^^J%
     in the same syntax as \noexpand\@currdir^^J%
     ie full directory names that may be concatenated with the^^J%
     basename (note the final / and ] in the above examples).^^J%
     Some systems may need more complicated settings.^^J%
     See texsys.cfg for more examples.^^J%
    ! BAD \noexpand\input@path!!}
   \@@end}%
\pause
\typeout{^^J%
  Checking the TeX version^^J%
  ========================}
\dimen@\ifx\@TeXversion\@undefined4\else\@TeXversion\fi\p@%
\ifx\noboundary\relax
  \typeout{^^J%
    This is TeX 2. You will not be able to use all the new features^^J%
    of LaTeX2e with such an old TeX.^^J%
    The current version (1994/10/11) is TeX 3.1415.^^J%
    Consider upgrading your TeX.}
  \ifdim\dimen@<3\p@\else
     \errhelp{Check that texsys.cfg has not defined \@TeXversion}
     \def\OldTeX{%
       BAD: \noexpand\@TeXversion is incorrect: \meaning\@TeXversion}
     \errmessage{\OldTeX}
  \fi
\else
    \ifdim\dimen@>3.14\p@
      \typeout{This appears to be a recent version of TeX!^^J%
       If the following `lines' all appear on the same line,^^J%
```

```
      separated by \string^\string^J %
      then there has been an incorrect installation.}
   \else
     \typeout{^^J%
      This appears to be a TeX between 3.0 and 3.14^^J%
      but the current version (1994/10/11) is TeX 3.1415^^J%
      consider upgrading your TeX.^^J%
      The following `lines' will appear on the same line,^^J%
      separated by \string^\string^J;^^J%
      the same problem may affect other messages from LaTeX.}
    \fi
```

```
\message{line1^^Jline2^^Jline3}
\pause
```

```
\fi
```

To check that the LaTeX fonts have been installed, the well known trick of going into `\batchmode`, and testing for `\nullfont` is used. Not all fonts are tested, just a representative sample.

```
\typeout{^^J%
  Checking fonts^^J%
  ====================================}
```

```
\def\checkfont#1{%
  \batchmode
  \font\test=#1\relax
  \errorstopmode
  \ifx\test\nullfont
    \typeout{\@spaces! BAD: #1.tfm not found!}
    \@tempswatrue
  \else
    \typeout{\@spaces OK: #1.tfm found}
  \fi}
```

```
\typeout{^^JChecking Standard TeX fonts...}
\@tempswafalse
\checkfont{cmr10}
\checkfont{cmr12}
\checkfont{cmmi10}
\if@tempswa
  \errhelp{Obtain a complete standard TeX font distribution.}
  \errmessage{BAD: Missing Standard Fonts}
\fi
```

```
\typeout{^^JChecking LaTeX Picture Mode fonts...}
\@tempswafalse
\checkfont{lcircle10}
\checkfont{lcirclew10}
\if@tempswa
  \@tempswafalse
  \checkfont{circle10}
```

```
 \checkfont{circlew10}
 \if@tempswa
   \typeout{^^J! BAD: You do not have the picture mode fonts:^^J%
          lcircle10 and lcirclew10}
 \else
   \typeout{^^J! BAD:%
          You have the picture mode fonts with their old names:^^J%
          circle10 and circlew10 have been renamed to^^J%
          lcircle10 and lcirclew10}
 \fi
 \errhelp{Obtain a complete standard LaTeX font distribution.}
 \errmessage{BAD: Missing LaTeX Fonts}
\fi

\typeout{^^JChecking Extra LaTeX Computer Modern fonts...}
\@tempswafalse
\checkfont{cmmib5}
\checkfont{cmmib7}
\checkfont{cmex7}
\if@tempswa
\typeout{! BAD:^^J%
 LaTeX2e uses a few `extra' Computer Modern fonts produced by^^J%
 The American Mathematical Society.^^J%
 If you install The AMSFONTS font collection, then these, and other,^^J%
 fonts will be available to LaTeX.^^J%
 Although installing AMSFONTS is recommended, LaTeX does not require^^J%
 The full collection; you may obtain a minimal set of extra LaTeX^^J%
 fonts from any ctan archive, in the directory macros/latex/fonts/}
\errhelp{Obtain LaTeX fonts or  the AMSFONTS collection.}
\errmessage{BAD: Missing LaTeX Fonts}
\fi

\@@end
⟨/ltxcheck⟩
```

File b
ltplain.dtx

9 Plain TeX

LaTeX includes almost all of the functionality of Knuth's original 'Basic Macros' That is, the plain TeX format described in Appendix B of the TeXBook.

LaTeX font definitions are done using NFSS2 (file ltfss.dtx) so all of PLAIN's font definitions are removed.

LaTeX has its own tabbing environment, so PLAIN's is disabled.

Macro definitions for math have been moved to some other file so that they are grouped together.

LaTeX uses its own output routine, so most of the plain one was removed.

A lot of individual macros are now moved to the relevant LaTeX files so that this file will gradually vanish.

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

```
\message{Preloading the plain format: codes,}
```

We had to define the `\catcodes` right away, before the message line, since `\message` uses the `{` and `}` characters. When INITEX (the TeX 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:

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

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

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

```
\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.

```
\chardef\@ne=1
\chardef\tw@=2
\chardef\thr@@=3
\chardef\sixt@@n=16
\chardef\@cclv=255
\mathchardef\@cclvi=256
\mathchardef\@m=1000
\mathchardef\@M=10000
\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.

```
\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 ifferent nesting levels. All such registers should be defined through these macros; otherwise conflicts may occur, especially when two or more more macro packages are being used at once.

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.

```
\count10=22 % allocates \count registers 23, 24, ...
\count11=9 % allocates \dimen registers 10, 11, ...
\count12=9 % allocates \skip registers 10, 11, ...
\count13=9 % allocates \muskip registers 10, 11, ...
\count14=9 % allocates \box registers 10, 11, ...
\count15=9 % allocates \toks registers 10, 11, ...
\count16=-1 % allocates input streams 0, 1, ...
\count17=-1 % allocates output streams 0, 1, ...
\count18=3 % allocates math families 4, 5, ...
\count19=0 % allocates \language codes 1, 2, ...
\count20=255 % allocates insertions 254, 253, ...
\countdef\insc@unt=20 % the insertion counter
\countdef\allocationnumber=21 % the most recent allocation
\countdef\m@ne=22 \m@ne=-1 % a handy constant
\def\wlog{\immediate\write\m@ne} % write on log file (only)
```

Here are abbreviations for the names of scratch registers that don't need to be allocated.

```
\countdef\count@=255
\dimendef\dimen@=0
\dimendef\dimen@i=1 % global only
\dimendef\dimen@ii=2
\skipdef\skip@=0
\toksdef\toks@=0
```

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.
To find out which counter `\foo` is, you can look at `\allocationnumber`.
Since there's no `\boxdef` command, `\chardef` is used to define a
`\newbox`, `\newinsert`, `\newfam`, and so on.
LaTeX 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.

```
\def\newcount{\alloc@0\count\countdef\insc@unt}
\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
\def\newskip{\alloc@2\skip\skipdef\insc@unt}
\def\newmuskip{\alloc@3\muskip\muskipdef\@cclvi}
\def\newbox{\alloc@4\box\chardef\insc@unt}
\def\newhelp#1#2{\newtoks#1#1\expandafter{\csname#2\endcsname}}
\def\newtoks{\alloc@5\toks\toksdef\@cclvi}
\def\newread{\alloc@6\read\chardef\sixt@@n}
\def\newwrite{\alloc@7\write\chardef\sixt@@n}
```

LaTeX defines `\newfam` in `ltfss.dtx`.

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

```
\def\newlanguage{\alloc@9\language\chardef\@cclvi}
\def\alloc@#1#2#3#4#5{\global\advance\count1#1by\@ne
  \ch@ck#1#4#2% make sure there's still room
  \allocationnumber=\count1#1%
  \global#3#5=\allocationnumber
  \wlog{\string#5=\string#2\the\allocationnumber}}
\def\newinsert#1{\global\advance\insc@unt by\m@ne
  \ch@ck0\insc@unt\count
  \ch@ck1\insc@unt\dimen
  \ch@ck2\insc@unt\skip
  \ch@ck4\insc@unt\box
  \allocationnumber=\insc@unt
  \global\chardef#1=\allocationnumber
  \wlog{\string#1=\string\insert\the\allocationnumber}}
\def\ch@ck#1#2#3{\ifnum\count1#1<#2%
  \else\errmessage{No room for a new #3}\fi}
```

Here are some examples of allocation.

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

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

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

```
\def\newif#1{%
  \count@\escapechar \escapechar\m@ne
    \let#1\iffalse
    \@if#1\iftrue
    \@if#1\iffalse
  \escapechar\count@}
```

```
\def\@if#1#2{%
  \expandafter\def\csname\expandafter\@gobbletwo\string#1%
                    \expandafter\@gobbletwo\string#2\endcsname
                       {\let#1#2}}
```

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

If this file is used in an old TeX we define the new features of TeX 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).

```
\ifx\undefined\inputlineno
  \newcount\inputlineno
  \inputlineno-1              % this could be used to detect that
                              % an old TeX is in force
  \ifx\undefined\language     % extra test for MLTeX 2, RmS 91/11/07
    \newcount\language
  \fi
  \newcount\lefthyphenmin
  \newcount\righthyphenmin
  \newcount\errorcontextlines
  \newcount\holdinginserts
  \newdimen\emergencystretch
  \newcount\badness
  \let\noboundary\relax
  \newcount\setlanguage
\fi
```

Assign initial values to TeX's parameters

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

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

```
\pretolerance=100
\tolerance=200 % INITEX sets this to 10000
\hbadness=1000
\vbadness=1000
\linepenalty=10
\hyphenpenalty=50
\exhyphenpenalty=50
\binoppenalty=700
```

```
\relpenalty=500
\clubpenalty=150
\widowpenalty=150
\displaywidowpenalty=50
\brokenpenalty=100
\predisplaypenalty=10000
```

`\postdisplaypenalty`=0
`\interlinepenalty`=0
`\floatingpenalty`=0, set during `\insert`
`\outputpenalty`=0, set before TeX enters `\output`

```
\doublehyphendemerits=10000
\finalhyphendemerits=5000
\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

```
\tracinglostchars=1
```

`\tracingcommands`=0
`\tracingrestores`=0
`\language`=0

```
\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

```
\defaulthyphenchar=`\-
\defaultskewchar=-1
```

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

```
\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.

```
\showboxbreadth=-1
\showboxdepth=-1
\errorcontextlines=-1
```

```
\hfuzz=0.1pt
\vfuzz=0.1pt
\overfullrule=5pt
\hsize=6.5in
\vsize=8.9in
\maxdepth=4pt
\splitmaxdepth=\maxdimen
\boxmaxdepth=\maxdimen
```

\lineskiplimit=0pt, changed by \normalbaselines

```
\delimitershortfall=5pt
\nulldelimiterspace=1.2pt
\scriptspace=0.5pt
```

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

```
\parindent=20pt
```

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

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

```
\parskip=0pt plus 1pt
\abovedisplayskip=12pt plus 3pt minus 9pt
\abovedisplayshortskip=0pt plus 3pt
\belowdisplayskip=12pt plus 3pt minus 9pt
\belowdisplayshortskip=7pt plus 3pt minus 4pt
```

\leftskip=0pt
\rightskip=0pt

```
\topskip=10pt
\splittopskip=10pt
```

\tabskip=0pt
\spaceskip=0pt
\xspaceskip=0pt

```
\parfillskip=0pt plus 1fil
```

We also define special registers that function like parameters:

```
\newskip\normalbaselineskip \normalbaselineskip=12pt
\newskip\normallineskip \normallineskip=1pt
\newdimen\normallineskiplimit \normallineskiplimit=0pt
\newcount\interfootnotelinepenalty \interfootnotelinepenalty=100
```

Definitions for preloaded fonts

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

Macros for setting ordinary text

```
\message{macros,}
```

```
\def\frenchspacing{\sfcode`\.\@m \sfcode`\?\@m \sfcode`\!\@m
  \sfcode`\:\@m \sfcode`\;\@m \sfcode`\,\@m}
\def\nonfrenchspacing{\sfcode`\.3000\sfcode`\?3000\sfcode`\!3000%
  \sfcode`\:2000\sfcode`\;1500\sfcode`\,1250 }
```

```
\def\normalbaselines{\lineskip\normallineskip
  \baselineskip\normalbaselineskip \lineskiplimit\normallineskiplimit}
```

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

```
\def\lq{`} \def\rq{'}
\def\lbrack{[} \def\rbrack{]}
```

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

```
\def\space{ }
\def\empty{}
\def\null{\hbox{}}
```

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

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

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

`\loop` `\iterate` `\repeat`

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

```
\long\def \loop #1\repeat{%
  \def\iterate{#1\relax  % Extra \relax
               \expandafter\iterate\fi
               }%
  \iterate
  \let\iterate\relax
}
```

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

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

LaTeX defines `\smallskip`, etc. `ltspace.dtx`.

```
\def\nointerlineskip{\prevdepth-1000\p@}
\def\offinterlineskip{\baselineskip-1000\p@
  \lineskip\z@ \lineskiplimit\maxdimen}

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

LaTeX defines `~` in `ltdefns.dtx`.

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

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

\def\filbreak{\par\vfil\penalty-200\vfilneg}
\def\goodbreak{\par\penalty-500 }
\def\eject{\par\break}
\def\supereject{\par\penalty-\@MM}

\def\removelastskip{\ifdim\lastskip=\z@\else\vskip-\lastskip\fi}
\def\smallbreak{\par\ifdim\lastskip<\smallskipamount
  \removelastskip\penalty-50\smallskip\fi}
```

```
\def\medbreak{\par\ifdim\lastskip<\medskipamount
  \removelastskip\penalty-100\medskip\fi}
\def\bigbreak{\par\ifdim\lastskip<\bigskipamount
  \removelastskip\penalty-200\bigskip\fi}
```

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

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?).

```
%\def\underbar#1{$\setbox\z@\hbox{#1}\dp\z@\z@
%  \m@th \underline{\box\z@}$}
\def\underbar#1{\underline{\setbox\tw@\hbox{#1}\dp\tw@\z@\box\tw@}}
```

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

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

```
\def\hidewidth{\hskip\hideskip} % for alignment entries that
                                % can stick out
\def\ialign{\everycr{}\tabskip\z@skip\halign} % initialized \halign
\newcount\mscount
\def\multispan#1{\omit \mscount#1\relax
  \loop\ifnum\mscount>\@ne \sp@n\repeat}
\def\sp@n{\span\omit\advance\mscount\m@ne}
```

```
\def\item{\par\hang\textindent}
\def\itemitem{\par\indent \hangindent2\parindent \textindent}
```

```
\def\narrower{%
  \advance\leftskip\parindent
  \advance\rightskip\parindent}
```

LaTeX defines `\ae` and similar commands elsewhere.

```
\chardef\%=`\%
\chardef\&=`\&
\chardef\#=`\#
```

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

```
\def\leavevmode{\unhbox\voidb@x} % begins a paragraph, if necessary
```

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

```
\def\oalign#1{\leavevmode\vtop{\baselineskip\z@skip \lineskip.25ex%
  \ialign{##\crcr#1\crcr}}}
\def\o@lign{\lineskiplimit\z@ \oalign}
\def\ooalign{\lineskiplimit-\maxdimen \oalign}
```

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

LaTeX change: `\d`, `\b`, `\c`, `\copyright`, `\TeX` defined elsewhere.

LaTeX change: Make `\t` work in a moving argument. Now defined elsewhere.

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 begining of `\dotfill` and `\hrulefill` so that they work as expected in vertical mode.

```
\def\hrulefill{\leavevmode\leaders\hrule\hfill\kern\z@}
\def\dotfill{\leavevmode\cleaders
     \hbox{$\m@th \mkern1.5mu.\mkern1.5mu$}\hfill\kern\z@}
```

LaTeX change: `\bye` is eliminated.

```
\outer\def\bye{\par\vfill\supereject\end}
```

LaTeX change: `\phantom` moved to ltmath.

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

```
\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\vsize8.9truein\dimen\footins8truein}
```

FMi & RmS 1991/08/26: added `\errorcontextlines=\maxdimen`, suggested by J. Schrod

```
\def\tracingall{\tracingcommands\tw@\tracingstats\tw@
  \tracingpages\@ne\tracingoutput\@ne\tracinglostchars\@ne
  \tracingmacros\tw@\tracingparagraphs\@ne\tracingrestores\@ne
  \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode
  \errorcontextlines\maxdimen\tracingonline\@ne}
```

LaTeX change: `\showhyphens` Defined later.

```
\nonfrenchspacing % punctuation affects the spacing
⟨/2ekernel⟩
```

File c
ltvers.dtx

10 Version Identifictation

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

\fmtname
\fmtversion

```
⟨*2ekernel⟩
\def\fmtname{LaTeX2e}
\edef\fmtversion{1995/06/01}
```

Check that the format being made is not too old.

```
\def\reserved@a#1/#2/#3\@nil{%
  \count@\year
  \advance\count@-#1\relax
  \multiply\count@ by 12\relax
  \advance\count@\month
  \advance\count@-#2\relax}
\expandafter\reserved@a\fmtversion\@nil
```

`\count@` is now the age of this file in months.

```
\ifnum\count@>12
  \typeout{^^J%
*****************************************************************^^J%
*  You are attempting to make a LaTeX format from a source file^^J%
*  That is more than one year old.^^J%
*^^J%
*  If you enter <return> to scroll past this message then the format^^J%
*  will be built, but please consider obtaining newer source files^^J%
*  before continuing to build LaTeX.^^J%
*^^J%
*  LaTeX is re-issued every 6 months June and December.^^J%
*****************************************************************^^J%
}
  \errhelp{To avoid this error message, obtain new LaTeX sources.}
  \errmessage{LaTeX source files more than 1 year old!}
\fi
\let\reserved@a\relax
```

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

```
\everyjob{\typeout{\fmtname
⟨autoload⟩\space(autoload version)%
                                        \space<\fmtversion>}}
\immediate\write16{\fmtname
⟨autoload⟩\space(autoload version)%
```

```
                                    \space<\fmtversion>}
⟨/2ekernel⟩
```

File d

ltalloc.dtx

11 Counters

This section deals with counter and other variable allocation.

```
⟨*2ekernel⟩
\message{counters,}
```

```
***************************************
*            COUNTERS, ETC.           *
***************************************
```

THE FOLLOWING ARE FROM PLAIN:

`\z@`	: A zero dimen or number. It's more efficient to write `\parindent\z@` than `\parindent` 0pt.
`\@ne`	: The number 1.
`\m@ne`	: The number -1.
`\tw@`	: The number 2.
`\sixt@@n`	: The number 16.
`\@m`	: The number 1000.
`\@xxxii`	: The number 32
`\@M`	: The number 10000.
`\@Mi`	: The number 10001.
`\@Mii`	: The number 10002.
`\@Miii`	: The number 10003.
`\@Miv`	: The number 10004.
`\@MM`	: The number 20000.

`\@flushglue` : Glue used for `\right- & \leftskip` to = 0pt plus 1fil

```
\chardef\@xxxii=32
\mathchardef\@Mi=10001
\mathchardef\@Mii=10002
\mathchardef\@Miii=10003
\mathchardef\@Miv=10004
```

Redefine PLAIN.TEX macros not to be `\outer` (no longer needed)

```
\def\newcount{\alloc@0\count\countdef\insc@unt}
\def\newdimen{\alloc@1\dimen\dimendef\insc@unt}
\def\newskip{\alloc@2\skip\skipdef\insc@unt}
\def\newbox{\alloc@4\box\chardef\insc@unt}
\def\newwrite{\alloc@7\write\chardef\sixt@@n}
```

```
\newcount\@tempcnta
\newcount\@tempcntb
\newif\if@tempswa\@tempswatrue
```

```
\newdimen\@tempdima
\newdimen\@tempdimb
\newdimen\@tempdimc
```

```
\newbox\@tempboxa
```

```
\newskip\@flushglue \@flushglue = 0pt plus 1fil
\newskip\@tempskipa
\newskip\@tempskipb
\newtoks\@temptokena
⟨/2ekernel⟩
```

File e
ltdefns.dtx

12 Definitions

This section contains commands used in defining other macros.

```
⟨*2ekernel⟩
\catcode`\~=13 \def~{\penalty\@M \ }
```

12.1 Initex initialisations

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

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

`\typeout` Display something on the terminal.

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

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

```
\newlinechar`\^^J
```

12.2 Saved versions of TEX primitives

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

`\@@par`

```
\let\@@par=\par
%\let\@@input=\input    %%% moved earlier
%\let\@@end=\end        %%%
```

`\@@hyph` `\-` The following was added 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 definition is saved as **`\@@hyph`** just in case anyone needs it.

```
\let\@@hyph=\-       % Original defin
\def\-{\discretionary{-}{}{}}
```

`\@dischyph`

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

`\@@italiccorr` Save the original italic correction.

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

\@height \@depth \@width \@minus \@plus

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

```
\def\@height{height}
\def\@depth{depth}
\def\@width{width}
\def\@minus{minus}
\def\@plus{plus}
```

\hb@xt@

The next one is another 100 tokens worth.

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

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

12.3 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`

```
\def\typein{%
  \let\@typein\relax\@ifnextchar[\@xtypein{\@xtypein[\@typein]}}
\def\@xtypein[#1]#2{%
  \typeout{#2}%
  \advance\endlinechar\@M
  \read\@inputcheck to#1%
  \advance\endlinechar-\@M}
```

`\@namedef`

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

`\@nameuse`

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

`\@cons`

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

`\@car`
`\@cdr`

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

`\@carcube` `\@carcube T1 ... Tn\@nil` = `T1 T2 T3` , $n > 3$

```
\def\@carcube#1#2#3#4\@nil{#1#2#3}
```

`\@star@or@long` Look ahead for a `*`. If present reset `\l@ngrel@x` so that the next definition, `#1`, will be non-long.

```
\def\@star@or@long#1{%
  \@ifstar
   {\let\l@ngrel@x\relax#1}%
   {\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.

```
\let\l@ngrel@x\relax
```

\newcommand User level \newcommand.

```
\def\newcommand{\@star@or@long\new@command}
```

\new@command

```
\def\new@command#1{%
  \@ifnextchar [{\@newcommand#1}{\@newcommand#1[0]}}
```

\@newcommand \@xargdef \@xargdef Handling arguments for \newcommand.

```
\def\@newcommand#1[#2]{%
  \@ifnextchar [{\@xargdef#1[#2]}%
                {\@argdef#1[#2]}}
```

Define #1 if it is definable.

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

Handle the second optional argument.

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

Define the actual command to be:
\def\foo{\@ifnextchar[\\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 exisiting command names. "Default" is the contents of the second optional argument of \(re)newcommand.

```
     \expandafter\def\expandafter#1\expandafter{%
          \expandafter
          \@ifnextchar
          \expandafter
             [%
          \csname\string#1\expandafter\endcsname
          \expandafter
          {\csname\string#1\endcsname[#3]}}%
```

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

```
       \expandafter\@yargdef
          \csname\string#1\endcsname
           \tw@
           {#2}%
           {#4}}}
```

\@yargdef Generate a primitive argument specification, from a LaTeX [⟨*num*⟩] form.

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

definition of `\reserved@a`. LaTeX2.09 goes into an infinite loop with `\renewcommand[1]{\@tempa}{foo}`.
(DPC 6 October 93)

```
\long\def\@yargdef#1#2#3{%
  \@tempcnta#3\relax
  \advance \@tempcnta \@ne
  \let\@hash@\relax
  \edef\reserved@a{\ifx#2\tw@ [\@hash@1]\fi}%
  \@tempcntb #2%
  \@whilenum\@tempcntb <\@tempcnta
     \do{%
         \edef\reserved@a{\reserved@a\@hash@\the\@tempcntb}%
         \advance\@tempcntb \@ne}%
  \let\@hash@##%
  \l@ngrel@x\expandafter\def\expandafter#1\reserved@a}
```

`\@reargdef`

```
\long\def\@reargdef#1[#2]#3{%
  \@yargdef#1\@ne{#2}{#3}}
```

`\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 `\@temp`a–e.)

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

`\renew@command`

```
\def\renew@command#1{%
  {\escapechar\m@ne\xdef\@gtempa{{\string#1}}}%
  \expandafter\@ifundefined\@gtempa
     {\@latex@error{\string#1 undefined}\@ehc}%
     {}%
  \let\@ifdefinable\@rc@ifdefinable
  \new@command#1}
```

`\@ifdefinable`
`\@@ifdefinable`
`\@rc@ifdefinable`

Test is user is allowed to define a command.

```
\long\def\@ifdefinable #1#2{%
      \edef\reserved@a{\expandafter\@gobble\string #1}%
     \@ifundefined\reserved@a
         {\edef\reserved@b{\expandafter\@carcube \reserved@a xxx\@nil}%
          \ifx \reserved@b\@qend \@notdefinable\else
            \ifx \reserved@a\@qrelax \@notdefinable\else
              #2%
            \fi
          \fi}%
         \@notdefinable}
```

Saved definition of `\@ifdefinable`.

```
\let\@@ifdefinable\@ifdefinable
```

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

```
\long\def\@rc@ifdefinable#1#2{%
  \let\@ifdefinable\@@ifdefinable
  #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.

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

`\new@environment`

```
\def\new@environment#1{%
  \@ifnextchar [{\@newenva#1}{\@newenva#1[0]}}
\def\@newenva#1[#2]{%
   \@ifnextchar [{\@newenvb#1[#2]}{\@newenv{#1}{[#2]}}}
\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.

```
\def\renewenvironment{\@star@or@long\renew@environment}
```

`\renew@environment`

```
\def\renew@environment#1{%
  \@ifundefined{#1}%
     {\@latex@error{Environment #1 undefined}\@ehc
     }{}%
  \expandafter\let\csname#1\endcsname\relax
  \expandafter\let\csname end#1\endcsname\relax
  \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 `\par`s)

Make sure that an attempt to define a 'graf' or 'group' environment fails.

```
\long\def\@newenv#1#2#3#4{%
  \@ifundefined{#1}%
    {\expandafter\let\csname#1\expandafter\endcsname
                         \csname end#1\endcsname}%
    \relax
  \expandafter\new@command
     \csname #1\endcsname#2{#3}%
     \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}
```

\providecommand

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

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

\provide@command

```
\def\provide@command#1{%
  {\escapechar\m@ne\xdef\@gtempa{{\string#1}}}%
  \expandafter\@ifundefined\@gtempa
    {\def\reserved@a{\new@command#1}}%
    {\def\reserved@a{\renew@command\reserved@a}}%
   \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 befor a macro package starts redefining things. Currently two macros are considered to have the same definition if they are the same except for different default arguments. That is, if the old definition was: `\newcommand\xxx[2][a]{(#1)(#2)}` then `\CheckCommand\xxx[2][b]{(#1)(#2)}` would *not* generate a warning, but, for instance `\CheckCommand\xxx[2]{(#1)(#2)}` would.

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

\check@command

```
\def\check@command#1#2#{\@check@c#1{#2}}
```

\@check@c

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

```
\long\def\@check@c#1#2#3{%
  \expandafter\let\csname\string\reserved@a\endcsname\relax
  \renew@command\reserved@a#2{#3}%
  \@ifundefined{\string\reserved@a}%
   {\@check@eq#1\reserved@a}%
   {\expandafter\@check@eq
          \csname\string#1\expandafter\endcsname
          \csname\string\reserved@a\endcsname}}
```

\@check@eq

Complain if `#1` and `#2` are not `\ifx` equal.

```
\def\@check@eq#1#2{%
  \ifx#1#2\else
     \@latex@warning@no@line
                {Command \noexpand#1 has
                 changed.\MessageBreak
                 Check if current package is valid}%
  \fi}
```

\@gobble
\@gobbletwo
\@gobblefour

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

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

\@firstofone
\@firstoftwo
\@secondoftwo

Some argument-grabbers.

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

\@iden

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

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

\@expandtwoargs

A macro to totally expand two arguments to another macro

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

\@backslashchar

A category code 12 backslash.

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

12.4 Robust commands and protect

Fragile and robust commands are one of the thornier issues in LaTeX's commands. Whilst typesetting documents, LaTeX makes use of many of TeX's features, such as arithmetic, defining macros, and setting variables. However, there are (at least) three different ocassions when these commands are not safe. These are called 'moving arguments' by LaTeX, 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 LaTeX 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 `\edef`s, `\mark`s 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`.

`\@unexpandable@protect`
`\@unexpandable@noexpand`

These commands are used for setting `\protect` inside `\edef`s.

```
\def\@unexpandable@protect{\noexpand\protect\noexpand}
\def\@unexpandable@noexpand{\noexpand\noexpand\noexpand}
```

`\DeclareRobustCommand`
`\x@protect`
`\@x@protect`
`\@typeset@protect`

This is a package-writers command, which has the same syntax as `\newcommand`, but which declares a protected command. It does this by having
`\DeclareRobustCommand\foo`
define `\foo` to be `\protect\foo<space>`,
and then use `\newcommand\foo<space>`.
Since the internal command is `\foo<space>`, when it is written to an auxiliary file, it will appear as `\foo`.

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

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

which expands to:

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

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

Note: setting `\protect` to any value other than `\relax` whilst in 'typesetting' mode will cause commands to go into an infinite loop! In particular, setting `\relax` to `\@empty` will cause `\_` to loop forever. It will also break lots of other things, such as protected `\ifmmode`s inside `\halign`s. 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`.

```
\def\DeclareRobustCommand{\@star@or@long\declare@robustcommand}
```

```
\def\declare@robustcommand#1{%
   \ifx#1\@undefined\else\ifx#1\relax\else
      \@latex@info{Redefining \string#1}%
   \fi\fi
   \edef\reserved@a{\string#1}%
   \def\reserved@b{#1}%
   \edef\reserved@b{\expandafter\strip@prefix\meaning\reserved@b}%
   \edef#1{%
```

```
      \ifx\reserved@a\reserved@b
         \noexpand\x@protect
         \noexpand#1%
      \fi
      \noexpand\protect
      \expandafter\noexpand\csname
         \expandafter\@gobble\string#1 \endcsname
   }%
   \let\@ifdefinable\@rc@ifdefinable
   \expandafter\new@command\csname
      \expandafter\@gobble\string#1 \endcsname
}
\def\x@protect#1{%
   \ifx\protect\@typeset@protect\else
      \@x@protect#1%
   \fi
}
\def\@x@protect#1\fi#2#3{%
   \fi\protect#1%
}
\let\@typeset@protect\relax
```

`\set@display@protect`
`\set@typeset@protect`

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

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

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

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

```
\def\protected@edef{%
   \let\@@protect\protect
   \let\protect\@unexpandable@protect
   \afterassignment\restore@protect
   \edef
}
\def\protected@xdef{%
   \let\@@protect\protect
   \let\protect\@unexpandable@protect
   \afterassignment\restore@protect
   \xdef
}
\def\unrestored@protected@xdef{%
   \let\protect\@unexpandable@protect
   \xdef
}
\def\restore@protect{\let\protect\@@protect}
```

`\protect`

The normal meaning of `\protect`

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

12.5 Internal defining commands

These commands are used internally to define other LaTeX commands.

\@ifundefined Check if first arg is undefined or `\relax` and execute second or third arg depending,

```
\def\@ifundefined#1{%
  \expandafter\ifx\csname#1\endcsname\relax
    \expandafter\@firstoftwo
  \else
    \expandafter\@secondoftwo
  \fi}
```

\@onlypreamble This macro adds its argument to the list of commands to be disabled after `\begin{document}`.

```
\def\@preamblecmds{}
\def\@onlypreamble#1{%
  \expandafter\gdef\expandafter\@preamblecmds\expandafter{%
       \@preamblecmds\do#1}}
\@onlypreamble\@onlypreamble
```

\@qend
\@qrelax The following define `\@qend` and `\@qrelax` to be the strings '`end`' and '`relax`' with the characters `\catcode`d 12.

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

```
\@ifnextchar X{YES}{NO}
 BEGIN
   \reserved@e := X  % uses \let
   \reserved@a := YES
   \reserved@b := NO
   \futurelet\reserved@c
   \@ifnch
 END

\@ifnch ==
  BEGIN
    if  \reserved@c = blank space
      then  \reserved@d := def(\@xifnch)
      else  if  \reserved@c = \reserved@e
                then  \reserved@d := def(\reserved@a)
                else  \reserved@d := def(\reserved@b)
            fi
    fi
    \reserved@d
  END

\@xifnch ==
```

```
BEGIN
  gobble blanks
  \futurelet\reserved@c
  \@ifnch
END
```

\@ifnextchar

```
\def\@ifnextchar#1#2#3{%
  \let\reserved@e=#1\def\reserved@a{#2}\def\reserved@b{#3}\futurelet
    \reserved@c\@ifnch}
\def\@ifnch{\ifx \reserved@c \@sptoken \let\reserved@d\@xifnch
      \else \ifx \reserved@c \reserved@e\let\reserved@d\reserved@a\else
          \let\reserved@d\reserved@b\fi
      \fi \reserved@d}
```

\makeatletter \makeatother

Make internal control sequences accessible or inaccessible.

```
\def\makeatletter{\catcode`\@=11\relax}
\def\makeatother{\catcode`\@=12\relax}
```

\@sptoken

NOTE: the following hacking must precede the definition of `\:` as math medium space.

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

\@xifnch

```
\def\:{\@xifnch} \expandafter\def\: {\futurelet\reserved@c\@ifnch}
```

\@ifstar

```
\def\@ifstar#1#2{\@ifnextchar *{\def\reserved@a*{#1}\reserved@a}{#2}}
```

\@dblarg \@xdblarg

```
\long\def\@dblarg#1{\@ifnextchar[{#1}{\@xdblarg{#1}}}
\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.

```
\def\@sanitize{\@makeother\ \@makeother\\\@makeother\$\@makeother\&%
\@makeother\#\@makeother\^\@makeother\_\@makeother\%\@makeother\~}
⟨/2ekernel⟩
```

\@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 float specifiers, this should be done when they are defined.

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

File f
ltcntrl.dtx

13 Program control structure

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

```
1 ⟨*2ekernel⟩
2 \message{control,}
```

```
     *********************************************
     *      PROGRAM CONTROL STRUCTURE MACROS        *
     *********************************************

 \@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
\@iwhilenum

```
\let\@whilenoop\@gobble
\def\@whilenum#1\do #2{\ifnum #1\relax #2\relax\@iwhilenum{#1\relax
     #2\relax}\fi}
\def\@iwhilenum#1{\ifnum #1\let\@nextwhile\@iwhilenum
         \else\let\@nextwhile\@whilenoop\fi\@nextwhile{#1}}
```

\@whiledim
\@iwhiledim

```
\def\@whiledim#1\do #2{\ifdim #1\relax#2\@iwhiledim{#1\relax#2}\fi}
\def\@iwhiledim#1{\ifdim #1\let\@nextwhile\@iwhiledim
        \else\let\@nextwhile\@whilenoop\fi\@nextwhile{#1}}
```

\@whileswnoop
\@whilesw
\@iwhilesw

```
\long\def\@whileswnoop#1\fi{}
\long\def\@whilesw#1\fi#2{#1#2\@iwhilesw{#1#2}\fi\fi}
\long\def\@iwhilesw#1\fi{#1\let\@nextwhile\@iwhilesw
         \else\let\@nextwhile\@whileswnoop\fi\@nextwhile{#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

```
\def\@nnil{\@nil}
```

\@empty

```
\def\@empty{}
```

\@fornoop

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

\@for

```
\def\@for#1:=#2\do#3{%
  \expandafter\def\expandafter\@fortmp\expandafter{#2}%
  \ifx\@fortmp\@empty \else
    \expandafter\@forloop#2,\@nil,\@nil\@@#1{#3}\fi}
```

\@forloop

```
\def\@forloop#1,#2,#3\@@#4#5{\def#4{#1}\ifx #4\@nnil \else
       #5\def#4{#2}\ifx #4\@nnil \else#5\@iforloop #3\@@#4{#5}\fi\fi}
```

\@iforloop

```
\def\@iforloop#1,#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
       \let\@nextwhile\@fornoop \else
      #4\relax\let\@nextwhile\@iforloop\fi\@nextwhile#2\@@#3{#4}}
```

\@tfor

```
\def\@tfor#1:={\@tf@r#1 }
\def\@tf@r#1#2\do#3{\def\@fortmp{#2}\ifx\@fortmp\space\else
    \@tforloop#2\@nil\@nil\@@#1{#3}\fi}
\def\@tforloop#1#2\@@#3#4{\def#3{#1}\ifx #3\@nnil
       \let\@nextwhile\@fornoop \else
      #4\relax\let\@nextwhile\@tforloop\fi\@nextwhile#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.

```
\def\@break@tfor#1\@@#2#3{\fi\fi}
```

\@removeelement Removes an element from a comma-separated list and put the result in a control sequence, called as `\@removeelement{`⟨*element*⟩`}{`⟨*list*⟩`}{`⟨*cs*⟩`}`.

Added `\ifx,` to remove extra ,'s.

```
\def\@removeelement#1#2#3{%
  \def\reserved@a##1,#1,##2\reserved@a{##1,##2\reserved@b}%
  \def\reserved@b##1,\reserved@b##2\reserved@b{%
    \ifx,##1\@empty\else##1\fi}%
  \edef#3{%
    \expandafter\reserved@b\reserved@a,#2,\reserved@b,#1,\reserved@a}}
```

```
\let\protect=\relax
```

```
⟨/2ekernel⟩
```

File g
lterror.dtx

14 Error handling

This section defines LaTeX's error commands.

```
⟨*2ekernel⟩
\message{errors,}
```

14.1 General commands

\MessageBreak

This command prints a new-line inside a messaage, 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`.

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

\GenericInfo

This takes two arguments: a continuation and a message, and sends the result to the log file.

```
\DeclareRobustCommand{\GenericInfo}[2]{%
   \begingroup
      \def\MessageBreak{^^J#1}%
      \set@display@protect
      \immediate\write\m@ne{#2\on@line.}%
   \endgroup
}
```

\GenericWarning

This takes two arguments: a continuation and a message, and sends the result to the screen.

```
\DeclareRobustCommand{\GenericWarning}[2]{%
   \begingroup
      \def\MessageBreak{^^J#1}%
      \set@display@protect
      \immediate\write\@unused{^^J#2\on@line.^^J}%
   \endgroup
}
```

\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.

```
\bgroup
\lccode`\@=`\ %
\lccode`\~=`\ %
\lccode`\}=`\ %
```

```
\lccode`\{=`\ %
\lccode`\T=`\T%
\lccode`\H=`\H%
\catcode`\ =11\relax%
\lowercase{%
\egroup%
```

Unfortunately TEX versions older than 3.141 have a bug which means that `^^J` does not force a linebreak in `\message` and `\errmessage` commands. So for these old TEX's we use `\typeout` to produce the message, and then have an empty `\errmessage` command. This causes an extra line of the form

```
! .
```

To appear on the terminal, but if you do not like it, you can always upgrade your TEX! In order for your format to use this version, you must define the macro `\@TeXversion` to be the version number, e.g., 3.14 of the underlying TEX. Se the comments in `ltdircheck.dtx`.

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

First the 'standard case'.

```
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
{{#4}}%
\errhelp
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\let
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\@empty
\def\MessageBreak{^^J#1}%
\def~{\errmessage{%
#2.^^J^^J%
#3^^J%
Type  H <return>  for immediate help%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
}}%
~%
\endgroup}%
```

```
\else%
```

Secondly the version for old TeX's.

```
\DeclareRobustCommand{\GenericError}[4]{%
\begingroup%
\immediate\write\@unused{}%
\def\MessageBreak{^^J}%
\set@display@protect%
\edef%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
{{#4}}%
\errhelp
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\let
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
\errmessage
\def\MessageBreak{^^J#1}%
\def~{\typeout{! %
#2.^^J^^J%
#3^^J%
Type  H <return>  for immediate help.}%
%    %<-------------------do not delete this space!------------------->%
\@err@                                                                 %
{}}%
~%
\endgroup}%
```

```
\fi}%
```

\PackageError
\PackageWarning
\PackageWarningNoLine
\PackageInfo
\ClassError
\ClassWarning
\ClassWarningNoLine
\ClassInfo

These commands are intended for use by package and class writers, to give information to authors. The syntax is:

> `\PackageError{`⟨*package*⟩`}{`⟨*error*⟩`}{`⟨*help*⟩`}`
> `\PackageWarning{`⟨*package*⟩`}{`⟨*warning*⟩`}`
> `\PackageWarningNoLine{`⟨*package*⟩`}{`⟨*warning*⟩`}`
> `\PackageInfo{`⟨*package*⟩`}{`⟨*info*⟩`}`

and similarly for classes. The `Error` commands print the ⟨*error*⟩ message, and present the interactive prompt; if the author types `h`, then the ⟨*help*⟩ information is displayed. The `Warning` commands produce a warning but do not present the interactive prompt. The `WarningNoLine` commands do the same, but don't print the input line number. The `Info` commands write the message to the `log` file. Within the messages, the command `\MessageBreak` can be used to break a line, `\protect` can be used to protect command names, and `\space` is a space, for example:

```
\newcommand{\foo}{FOO}
\PackageWarning{ethel}{%
   Your hovercraft is full of eels,\MessageBreak
```

```
and \protect\foo\space is \foo}
```

produces:

```
Package ethel warning: Your hovercraft is full of eels,
(ethel)                and \foo is FOO on input line 54.
```

```
\def\PackageError#1#2#3{%
   \GenericError{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      Package #1 Error: #2%
   }{%
      See the #1 package documentation for explanation.%
   }{#3}%
}
\def\PackageWarning#1#2{%
   \GenericWarning{%
      (#1)\@spaces\@spaces\@spaces\@spaces
   }{%
      Package #1 Warning: #2%
   }%
}
\def\PackageWarningNoLine#1#2{%
   \PackageWarning{#1}{#2\@gobble}%
}
\def\PackageInfo#1#2{%
   \GenericInfo{%
      (#1) \@spaces\@spaces\@spaces
   }{%
      Package #1 Info: #2%
   }%
}
\def\ClassError#1#2#3{%
   \GenericError{%
      (#1) \space\@spaces\@spaces\@spaces
   }{%
      Class #1 Error: #2%
   }{%
      See the #1 class documentation for explanation.%
   }{#3}%
}
\def\ClassWarning#1#2{%
   \GenericWarning{%
      (#1) \space\@spaces\@spaces\@spaces
   }{%
      Class #1 Warning: #2%
   }%
}
```

```
\def\ClassWarningNoLine#1#2{%
   \ClassWarning{#1}{#2\@gobble}%
}
\def\ClassInfo#1#2{%
   \GenericInfo{%
      (#1) \space\space\@spaces\@spaces
   }{%
      Class #1 Info: #2%
   }%
}
```

\@latex@error
\@latex@warning
\@latex@warning@no@line
\@latex@info
\@latex@info@no@line

Errors and other info, for use in the LaTeX core.

```
\def\@latex@error#1#2{%
   \GenericError{%
      \space\space\space\@spaces\@spaces\@spaces
   }{%
      LaTeX Error: #1%
   }{%
      See the LaTeX manual or LaTeX Companion for explanation.%
   }{#2}%
}
\def\@latex@warning#1{%
   \GenericWarning{%
      \space\space\space\@spaces\@spaces\@spaces
   }{%
      LaTeX Warning: #1%
   }%
}
\def\@latex@warning@no@line#1{%
   \@latex@warning{#1\@gobble}}
\def\@latex@info#1{%
   \GenericInfo{%
      \@spaces\@spaces\@spaces
   }{%
      LaTeX Info: #1%
   }%
}
\def\@latex@info@no@line#1{%
  \@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 be manipulated with `\setcounter` (once it is defined :-)

```
\let\c@errorcontextlines\errorcontextlines
\c@errorcontextlines=-1
```

\on@line — The message ‘ on input line n’, if possible.

```
\ifnum\inputlineno=\m@ne
  \let\on@line\empty
\else
  \def\on@line{ on input line \the\inputlineno}
\fi
```

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

```
\let\@warning\@latex@warning
\let\@@warning\@latex@warning@no@line
\let\@latexerr\@latex@error
```

\@spaces — Four spaces.

```
\def\@spaces{\space\space\space\space}
```

14.2 Specific errors

\@eha \@ehb \@ehc \@ehd — The more common error help messages.

```
\def\@eha{Your command was ignored.\MessageBreak
Type \space I <command> <return> \space to replace it
  with another command,\MessageBreak
or \space <return> \space to continue without it.}
\def\@ehb{You've lost some text. \space \@ehc}
\def\@ehc{Try typing \space <return>
  \space to proceed.\MessageBreak
If that doesn't work, type \space X <return> \space to
  quit.}
\def\@ehd{You're in trouble here.  \space\@ehc}
```

Here most of the error message-generating commands of LaTeX.

`\@notdefinable` : Error message generated in `\@ifdefinable` from calls

by `\newcommand, \newlength, \newtheorem` specifying
an

already-defined command name.

`\@nolnerr` : Generated by `\newline` and `\\` when called in vertical mode.

'\... undefined' : Generated in `\renewcommand.`

`\@nocnterr` : obsolete error message generated in 209 by
`\setcounter, \addtocounter` or `\newcounter`
for undefined counter.
DO NOT use for 2e it MIGHT vanish!
Use `\@nocounterr` CNT instead.

`\@nocounterr` CNT : Generated by `\setcounter, \addtocounter` or `\newcounter` for undefined counter CNT.

`\@ctrerr` : Called when trying to print the value of a counter numbered by letters that's greater than 26.

'Environment — undefined' : Issued by `\begin` for undefined environment.

`\@badend` : Called by `\end` that doesn't match its `\begin.`

`\@badmath` : Called by `\[`, `\]`, `\(` or `\)` when used in wrong mode.

`\@toodeep` : Called by a list environment nested more than six levels deep, or an enumerate or itemize nested more than four levels.

`\@badpoptabs:` Called by `\endtabbing` when not enough `\poptabs` have occurred, or by `\poptabs` when too many have occurred.

`\@badtab` : Called by `\>`, `\+` , `\-` or `\<` when stepping to an undefined tab.

'tab overflow' : Occurs in `\=` when maximum number of tabs exceeded.

'`\<` in mid line' : Occurs in `\<` when it appears in middle of line.

`\@preamerr` : Occurs in array or tabular environment, or in `\multicolumn` command, when error in argument detected.

`\@badlinearg` : Occurs in `\line` and `\vector` command when a bad slope argument is encountered.

`\@parmoderr` : Occurs in a float environment or a `\marginpar` when encountered in inner vertical mode.

`\@fltovf` : Occurs in float environment or `\marginpar` when there are no more free boxes for storing floats.

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

'Float(s) lost' : In output routine, caused by a float environment or `\marginpar` occurring in inner vertical mode.

`\@badcrerr` : A `\\` used where it shouldn't be in a centering or flushing environment.
obsolete; replaced with `\@nolnerr`

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

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

`\@invalidchar` : An invalid character in the input file.

```
\def\@notdefinable{\@latex@error{%
   Command \@backslashchar\reserved@a\space already defined%
}\@eha}
```

```
\def\@nolnerr{\@latex@error{There's no line here to end}\@eha}
```

```
\def\@nocounterr#1{\@latex@error{No counter '#1' defined}\@eha}
\def\@nocnterr{\@nocounterr{??}}
```

```
\def\@ctrerr{\@latex@error{Counter too large}\@ehb}
```

```
\def\@nodocument{\@latex@error{Missing \protect\begin{document}}\@ehd}
```

RmS 1992/08/24: added code to `\@badend` to display position of non-matching `\begin`. FMi 1993/01/14: missing space added.

```
\def\@badend#1{%
  \@latex@error{\protect\begin{\@currenvir}\@currenvline\space ended
                    by \protect\end{#1}}\@eha}
```

```
\def\@badmath{\@latex@error{Bad math environment delimiter}\@eha}
```

```
\def\@toodeep{\@latex@error{Too deeply nested}\@ehd}
```

```
\def\@badpoptabs{%
  \@latex@error{\protect\pushtabs \space and \protect\poptabs
      \space don't match}\@ehd}
```

```
\def\@badtab{\@latex@error{Undefined tab position}\@ehd}
```

This error is special: it appear in places where we normally have to `\protect` expansions. Howwever, 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`.

```
\def\@preamerr#1{%
  \begingroup
    \let\protect\relax
    \@latex@error{\ifcase #1 Illegal character\or
     Missing @-exp\or Missing p-arg\fi\space
     in array arg}\@ehd
  \endgroup}
```

```
\def\@badlinearg{%
  \@latex@error{%
       Bad \protect\line\space or \protect\vector \space argument}\@ehb}
```

```
\def\@parmoderr{\@latex@error{Not in outer par mode}\@ehb}
```

```
\def\@fltovf{\@latex@error{Too many unprocessed floats}\@ehb}
```

```
\def\@latexbug{\@latex@error{This may be a LaTeX bug}{Call for help}}
```

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

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

```
\def\@noitemerr{\@latex@error{Something's wrong--perhaps a missing
\protect\item}\@ehc}
```

```
\def\@notprerr {\@latex@error{Can be used only in preamble}\@eha}
```

`\@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.

```
\def\@inmatherr#1{%
   \relax
   \ifmmode
     \@latex@error{Command \protect#1 invalid in math mode}\@ehc
   \fi}
```

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

```
%\def\@invalidchar{\@latex@error{Invalid character in input}\@ehc}
⟨/2ekernel⟩
```

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.

```
⟨*2ekernel⟩
\message{par,}
```

`\@setpar` `\@par` Initiate a long-term change to `\par`.

```
\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.

```
\def\@par{\let\par\@@par\par}
⟨/2ekernel⟩
```

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

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

File i
ltspace.dtx

16 Spacing

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

```
⟨*2ekernel⟩
\message{spacing,}
```

```
     *********************************************
     *      SPACING / LINE AND PAGE BREAKING          *
     *********************************************


USER COMMANDS:
\nopagebreak[i] : i = 0,...,4.  Default argument = 4.  Puts a penalty
                into the vertical list output as follows:
                  0 : penalty = 0
                  1 : penalty = \@lowpenalty
                  2 : penalty = \@medpenalty
                  3 : penalty = \@highpenalty
                  4 : penalty = 10000
\pagebreak[i]   : same as \nopagebreak except negatives of its penalty
\linebreak[i], \nolinebreak[i] : analogs of the above
\samepage : inhibits page breaking most places by setting the
             following penalties to 10000
                   \interlinepenalty
                   \postdisplaypenalty
                   \interdisplaylinepenalty
                   \@beginparpenalty
                   \@endparpenalty
                   \@itempenalty
                   \@secpenalty
                   \interfootnotelinepenalty

\obeycr     : defines <CR> == \\\relax.
\restorecr : restores <CR> to its usual meaning.

\\           : initially defined to be \newline
\\[LENGTH] : initially defined to be \vspace{LENGTH}\newline
             Note: \\* adds a \vadjust{\penalty 10000}
```

\nopagebreak
\@nopgbk

```
\def\nopagebreak{\@ifnextchar[\@nopgbk{\@nopgbk[4]}}
\def\@nopgbk[#1]{\ifvmode \penalty \@getpen{#1}\else
\@bsphack\vadjust{\penalty \@getpen{#1}}\@esphack\fi}
```

\pagebreak
\@pgbk

```
\def\pagebreak{\@ifnextchar[\@pgbk{\@pgbk[4]}}
\def\@pgbk[#1]{\ifvmode \penalty -\@getpen{#1}\else
\@bsphack\vadjust{\penalty -\@getpen{#1}}\@esphack\fi}
```

\nolinebreak
\@nolnbk

```
\def\nolinebreak{\@ifnextchar[\@nolnbk{\@nolnbk[4]}}
\def\@nolnbk[#1]{\ifvmode \@nolnerr\else \@tempskipa\lastskip
     \unskip \penalty \@getpen{#1}\ifdim \@tempskipa >\z@
     \hskip\@tempskipa\ignorespaces\fi\fi}
```

\linebreak
\@lnbk

```
\def\linebreak{\@ifnextchar[\@lnbk{\@lnbk[4]}}
\def\@lnbk[#1]{\ifvmode \@nolnerr\else
     \unskip\penalty -\@getpen{#1}\fi}
```

\samepage

```
\def\samepage{\interlinepenalty\@M
   \postdisplaypenalty\@M
   \interdisplaylinepenalty\@M
   \@beginparpenalty\@M
   \@endparpenalty\@M
   \@itempenalty\@M
   \@secpenalty\@M
   \interfootnotelinepenalty\@M}
```

\newline

`\nobreak` added to `\newline` to prevent null lines when `\newline` ends an overfull line. Change made 24 May 89 as suggested by Frank Mittelbach and Rainer Schöpf

```
\def\newline{\ifvmode \@nolnerr \else \unskip\nobreak\hfil
  \penalty -\@M\fi}
```

\\
\@normalcr

The internal definition of the 'normal' definition of \\.

```
\DeclareRobustCommand\\{%
  \@ifstar
    {\vadjust{\penalty\@M}\@xnewline}%
    \@xnewline}
\expandafter\let\expandafter\@normalcr
     \csname\expandafter\@gobble\string\\ \endcsname
```

\@xnewline

```
\def\@xnewline{\@ifnextchar[\@newline\newline}
```

\@newline

```
\def\@newline[#1]{\ifhmode\unskip\fi\vspace{#1}\newline}
```

```
\def\@getpen#1{\ifcase #1 0 \or \@lowpenalty\or
        \@medpenalty \or \@highpenalty
        \else \@M \fi}
```

@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.

```
\newif\if@nobreak \@nobreakfalse
```

`\@bsphack` ... `\@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.

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

```
\@bsphack ==
 BEGIN
  if not mmode then                %% Test for math mode added 18 Dec
89
        \dimen\@savsk := \lastskip
      if  hmode  then  \@savsf := \spacefactor  fi
  fi
 END

\@esphack ==
 BEGIN
  if not mmode then                %% Test for math mode added 18 Dec
89
   if  hmode
     then  \spacefactor := \@savsf
           if \dimen\@savsk > 0pt  then  \ignorespaces fi
   fi
  fi
 END

\@Esphack ==
 BEGIN
```

```
if not mmode then
  if  hmode
    then  \spacefactor := \@savsf
          if \dimen\@savsk > 0pt  then  \ignorespaces
                                         \global\@ignoretrue  fi
  fi
fi
END
```

```
\newdimen\@savsk
\newcount\@savsf
```

```
\def\@Esphack{\relax\ifmmode\else\ifhmode\spacefactor\@savsf
     \ifdim \@savsk >\z@ \global\@ignoretrue \ignorespaces
 \fi \fi\fi}
```

\@bsphack
\@esphack

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.

```
\def \@bsphack{%
   \relax \ifhmode
     \@savsk \lastskip
     \@savsf \spacefactor
   \fi
}
\def \@esphack{%
     \relax \ifhmode
       \spacefactor \@savsf
       \ifdim \@savsk>\z@
         \ignorespaces
       \fi
     \fi
}
\def \@Esphack{%
     \relax \ifhmode
       \spacefactor \@savsf
       \ifdim \@savsk>\z@
         \global\@ignoretrue
         \ignorespaces
       \fi
     \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
% }
```

VERTICAL SPACING:

LaTeX supports the PLAIN TeX commands `\smallskip`, `\medskip` and `\bigskip`.
However, it redefines them using `\vspace` instead of `\vskip`.

Extra vertical space is added by the command 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 = T. 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

```
\def\@xaddvskip{\ifdim \lastskip <\@tempskipb\vskip-\lastskip\vskip
             \@tempskipb
       \else  \ifdim \@tempskipb<\z@
                 \ifdim \lastskip <\z@
                    \else \advance\@tempskipb\lastskip
                          \vskip -\lastskip \vskip \@tempskipb
     \fi\fi\fi}
```

\addvspace

```
\def\addvspace#1{%
  \ifvmode
     \if@minipage\else
       \ifdim \lastskip =\z@
         \vskip #1\relax
       \else
       \@tempskipb#1\relax
         \@xaddvskip
       \fi
     \fi
  \else
    \@noitemerr
  \fi
}
```

\addpenalty

```
\def\addpenalty#1{%
  \ifvmode
    \if@minipage
    \else
      \if@nobreak
      \else
        \ifdim\lastskip=\z@
          \penalty#1\relax
        \else
          \@tempskipb\lastskip
          \vskip -\lastskip
          \penalty#1%
          \vskip\@tempskipb
        \fi
      \fi
    \fi
  \else
```

```
    \@noitemerr
  \fi
}
```

\vspace \@vspace \@vspacer

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

- 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.

```
\def\vspace{\@ifstar\@vspacer\@vspace}
\def\@vspace #1{%
  \ifvmode
    \vskip #1
    \vskip\z@skip
   \else
     \@bsphack
     \vadjust{\@restorepar
              \vskip #1
              \vskip\z@skip
              }%
     \@esphack
   \fi
}
\def\@vspacer #1{%
  \ifvmode
    \dimen@\prevdepth
    \hrule \@height\z@
    \nobreak
    \vskip #1
    \vskip\z@skip
    \prevdepth\dimen@
  \else
    \@bsphack
    \vadjust{\@restorepar
             \hrule \@height\z@
             \nobreak
             \vskip #1
             \vskip\z@skip
             }%
    \@esphack
  \fi
}
```

```
\def\smallskip{\vspace\smallskipamount}
\def\medskip{\vspace\medskipamount}
\def\bigskip{\vspace\bigskipamount}
```

```
\newskip\smallskipamount \smallskipamount=3pt plus 1pt minus 1pt
\newskip\medskipamount   \medskipamount  =6pt plus 2pt minus 2pt
\newskip\bigskipamount   \bigskipamount =12pt plus 4pt minus 4pt
```

\endtrivlist
\@endparenv
\@doendpe

HORIZONTAL SPACE

`\,` : 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.”

`\@` : 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.

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

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

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

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

```
\def\@hspacer#1{\vrule \@width\z@\nobreak
                \hskip #1\hskip \z@skip}
                      % extra \hskip 0pt added 12/17/85 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
```

define `\fill` to = 0pt plus 1fill

```
\newskip\fill \fill = 0pt plus 1fill
```

`\stretch{N}` == 0pt plus N fill

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

```
\def\thinspace{\kern .16667em }
\def\negthinspace{\kern-.16667em }
\def\enspace{\kern.5em }
```

```
\def\enskip{\hskip.5em\relax}
\def\quad{\hskip1em\relax}
\def\qquad{\hskip2em\relax}
```

\changes{v1.2g}{1995/06/11}
{(CAR) \cs{relax} added to stop silent eating of *.}

```
{\catcode`\^^M=13 \gdef\obeycr{\catcode`\^^M13 \def^^M{\\\relax}%
    \@gobblecr}%
\gdef\restorecr{\catcode`\^^M5 }}
⟨/2ekernel⟩
```

File j
ltlogos.dtx

17 Logos

Various logos are defined here.

`\TeX` The TeX logo, adjusted so that a full stop after the logo counts as ending a sentence.

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

`\LaTeX` The LaTeX logo.

```
\DeclareRobustCommand{\LaTeX}{L\kern-.36em
        {\setbox0\hbox{T}%
         \vbox to\ht0{\hbox{$\m@th$%
                            \csname S@\f@size\endcsname
                            \fontsize\sf@size\z@
                            \math@fontsfalse\selectfont
                            A}%
                      \vss}%
        }%
        \kern-.15em
        \TeX}
```

`\LaTeXe` The LaTeX 2_ε logo as proposed by A-W designers.

```
\DeclareRobustCommand{\LaTeXe}{\mbox{%
  \if b\expandafter\@car\f@series\@nil\boldmath\fi
  \LaTeX\kern.15em 2${}_{\textstyle\varepsilon}$}}
⟨/2ekernel⟩
```

File k
ltfiles.dtx

18 File Handling

The following user commands are defined in this part:

`\document` (ie `\begin{document}`)
Reads in the .AUX files and `\catcode`'s `@` to 12.

`\nofiles`
Suppresses all file output by setting `\@filesw` false.

`\includeonly` {*NAME1, ... ,NAMEn*}
Causes only parts NAME1, ... ,NAMEn to be read by their `\include` commands. Works by setting partsw true and setting `\@partlist` to NAME1, ... ,NAMEn.

`\include` {*NAME*}
Does an `\input` NAME unless `\partsw` is true and NAME is not in `\@partlist`. If `\@filesw` is true, then it directs .AUX output to NAME.AUX, including a checkpoint at the end.

`\input` {*NAME*}
The same as TeX's `\input`, except it allows optional braces around the file name. In LaTeX 2_ε, it also avoids the primiive 'missing file' error, if the file can not be found.

`\IfFileExists` {*NAME*}{*then*}{*else*}
If the file exists on the system, execute *then* otherwise execute *else*.

`\InputIfFileExists` {*NAME*}{*then*}{*else*}
If the file exists on the system, execute *then* and input *NAME* otherwise execute *else*.

```
1 ⟨*2ekernel⟩
2 \message{files,}
```

```
VARIABLES, SWITCHES AND INTERNAL COMMANDS:
   \@mainaux     : Output file number for main .AUX file.
   \@partaux     : Output file number for current part's .AUX file.
   \@auxout      : Either \@mainout or \@partout, depending on
                    which .AUX file output goes to.
   \@input{foo} : If file foo exists, then \input's it,
                    otherwise types a warning message.
   @filesw       : Switch -- set false if no .AUX, .TOC, .IDX etc
                       files are to be written
   @partsw       : Set true by a \includeonly command.
   \@partlist    : Set to the argument of the \includeonly command.

   \cp@FOO       : The checkpoint for \include'd file FOO.TEX, written
                    by \@writeckpt at the end of file FOO.AUX
```

```
\document ==
  BEGIN
    \endgroup   % cancels \begingroup generated by \begin command
    \@colht := \@colroom := \vsize := \textheight
    \columnwidth := \textwidth
    \@clubpenalty := \clubpenalty        % \@clubpenalty saves
value.
    IF @twocolumn = T
      THEN \columnwidth := (\columnwidth - \columnsep)/2
           @firstcolumn := T
    FI
    \hsize  := \linewidth := \columnwidth
    \begingroup
       \@floatplacement \@dblfloatplacement
       \@input{\jobname.aux}
    \endgroup
    IF \@filesw = T
      THEN  open file \@mainaux for writing
            write ``\relax''on file \@mainaux
    FI
    \csname process@table\endcsname   % execute \process@table
                                      % if it is defined (NFSS)
    \glb@currsize := {} %% Force \baselineskip initialisation. (NFSS)
    \do{COMMAND} == BEGIN \let COMMAND = \@notprerr END
    \@preamblecmds
    \do == \noexpand
    \@normalsize
    \everypar{}
    @noskipsec := F
    G@refundefined := G F
  END

\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@b == FILE
          for \reserved@a := \@partlist
              do if eval(\reserved@a) = eval(\reserved@b)
                   then \@tempswa := T         fi
              od
  fi

  if \@tempswa = T
     then \@auxout := \@partaux
          if \@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 \@unused

Allocate read stream for testing and output stream.

```
\newread\@inputcheck
\newwrite\@unused
```

\@mainaux \@partaux

```
\newwrite\@mainaux
\newwrite\@partaux
```

\if@filesw \if@partsw

```
\newif\if@filesw \@fileswtrue
\newif\if@partsw \@partswfalse
```

\@clubpenalty

```
\newcount\@clubpenalty
```

FMi & RmS 1991/08/26 set `\@noskipsec` switch to true in the preamble and to false by `\begin{document}` to catch lists in the preamble, i.e., to produce a "nodocument" error when things like `\maketitle` appear before `\begin{document}`.

`\@noskipsectrue` set below where switch is defined

1991/03/26 FMi: `\process@table` added to support NFSS. This will also work with old lfonts if no other style defines `\process@table`.

\document

```
\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.

```
  \ifx\@unusedoptionlist\@empty\else
    \@latex@warning@no@line{Unused global option(s):^^J%
             \@spaces[\@unusedoptionlist]}%
  \fi
  \@colht\textheight
  \@colroom\textheight \vsize\textheight
  \columnwidth\textwidth
  \@clubpenalty\clubpenalty
  \if@twocolumn
    \advance\columnwidth -\columnsep
    \divide\columnwidth\tw@ \hsize\columnwidth \@firstcolumntrue
  \fi
  \hsize\columnwidth \linewidth\hsize
  \begingroup\@floatplacement\@dblfloatplacement
   \makeatletter\let\@writefile\@gobbletwo
   \global\@multiplelabelsfalse
   \@input{\jobname.aux}%
  \endgroup
  \if@filesw
    \immediate\openout\@mainaux\jobname.aux
    \immediate\write\@mainaux{\relax}%
```

```
  \fi
  \process@table
  \let\glb@currsize\@empty  %% Force math initialisation.
  \normalsize
  \everypar{}\@noskipsecfalse\G@refundefinedfalse
```

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. 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.

```
  \@begindocumenthook
  \ifdim\topskip< 1sp\topskip 1sp\relax\fi
```

Most of the following assignments will be done globally in case the user adds something like `\begin{multicols}` to the document hook, ie starts are group in `\begin{document}`.

```
  \global\let\@begindocumenthook\@undefined
  \ifx\@listfiles\@undefined
    \global\let\@filelist\relax
    \global\let\@addtofilelist\@gobble
  \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. `\AtBeginDocument`.

```
  \def\do##1{\global\let ##1\@notprerr}%
  \@preamblecmds
```

The next line is a pure safty 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.

```
  \global\let\do\noexpand
}
```

```
\def\nofiles{\@fileswfalse
  \typeout{No auxiliary output files.^^J}%
  \long\def\protected@write##1##2##3{}%
  \let\makeindex\relax
  \let\makeglossary\relax}
\@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`.

```
\long\def \protected@write#1#2#3{%
      \begingroup
       \let\thepage\relax
       #2%
       \let\protect\@unexpandable@protect
```

```
        \edef\reserved@a{\write#1{#3}}%
        \reserved@a
      \endgroup
      \if@nobreak\ifvmode\nobreak\fi\fi
}
```

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

\include

```
\def\includeonly#1{\@partswtrue
                  \edef\@partlist{\zap@space#1 \@empty}}
```

\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.

```
\def\include#1{\relax
  \ifnum\@auxout=\@partaux
    \@latex@error{\string\include\space cannot be nested}\@eha
  \else \@include#1 \fi}
\def\@include#1 {%
  \clearpage
  \if@filesw
    \immediate\write\@mainaux{\string\@input{#1.aux}}%
  \fi
  \@tempswatrue
  \if@partsw
    \@tempswafalse
    \edef\reserved@b{#1}%
    \@for\reserved@a:=\@partlist\do
      {\ifx\reserved@a\reserved@b\@tempswatrue\fi}%
  \fi
  \if@tempswa
    \let\@auxout\@partaux
    \if@filesw
      \immediate\openout\@partaux #1.aux
      \immediate\write\@partaux{\relax}%
    \fi
    \@input@{#1.tex}%
    \clearpage
    \@writeckpt{#1}%
    \if@filesw
      \immediate\closeout\@partaux
    \fi
  \else
```

```
    \@nameuse{cp@#1}%
  \fi
  \let\@auxout\@mainaux}
```

```
\def\@writeckpt#1{\if@filesw
\immediate\write\@partaux{\string\@setckpt{#1}\@charlb}%
{\let\@elt\@wckptelt \cl@@ckpt}\immediate\write\@partaux{\@charrb}\fi}
```

```
\def\@wckptelt#1{\immediate\write\@partaux
{\string\setcounter{#1}{\the\@nameuse{c@#1}}}}
```

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

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

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

```
{\catcode`[=1 \catcode`]=2
\catcode`{=11 \catcode`}=11
\gdef\@charlb[{]
\gdef\@charrb[}]
]% }brace matching
```

18.1 Safe Input Macros

`\IfFileExists`

```
\long\def \IfFileExists#1#2#3{%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck
    \ifx\input@path\@undefined
      \def\reserved@a{#3}%
    \else
      \def\reserved@a{\@iffileonpath{#1}{#2}{#3}}%
    \fi
  \else
    \closein\@inputcheck
    \edef\@filef@und{#1 }%
    \def\reserved@a{#2}%
  \fi
  \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`.

```
\long\def \@iffileonpath#1#2#3{%
  \def\reserved@a{#3}%
  \expandafter\@tfor\expandafter\reserved@b\expandafter
             :\expandafter=\input@path\do{%
    \openin\@inputcheck\reserved@b#1 %
    \ifeof\@inputcheck\else
      \edef\@filef@und{\reserved@b#1 }%
      \def\reserved@a{#2}%
      \closein\@inputcheck
```

```
      \@break@tfor
    \fi}%
  \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`'.

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

\input Input a file: if argument is given in braces use save input macros, otherwise use TeX's primitive `\input` command (which is called `\@@input` in LaTeX).

```
\def\input{\@ifnextchar\bgroup\@iinput\@@input}
```

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

```
\def\@iinput#1{%
  \InputIfFileExists{#1}{}%
  {\filename@parse{#1}%
   \edef\reserved@a{\noexpand\@missingfileerror
     {\filename@area\filename@base}%
     {\ifx\filename@ext\relax tex\else\filename@ext\fi}}%
   \reserved@a}}
```

\@input Define `\@input` in terms of `\IfIfileExists`. 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`.

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

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

```
\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.

```
\def\@missingfileerror#1#2{%
    \typeout{^^J! LaTeX Error: File `#1.#2' not found.^^J^^J%
     Type X to quit or <RETURN> to proceed,^^J%
     or enter new name. (Default extension: #2)^^J}%
    \message{Enter file name: }%
     {\endlinechar\m@ne
      \global\read\m@ne to\@gtempa}%
    \ifx\@gtempa\@empty
    \else
      \def\reserved@a{x}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
      \def\reserved@a{X}\ifx\reserved@a\@gtempa\batchmode\@@end\fi
      \filename@parse\@gtempa
```

```
      \edef\filename@ext{%
        \ifx\filename@ext\relax#2\else\filename@ext\fi}%
     \edef\reserved@a{%
       \noexpand\InputIfFileExists
         {\filename@area\filename@base.\filename@ext}%
         {}%
         {\noexpand\@missingfileerror
            {\filename@area\filename@base}{\filename@ext}}}%
      \reserved@a
    \fi}
```

\@obsoletefile For compatibility with LaTeX 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.

```
\def\@obsoletefile#1#2{%
  \@latex@warning@no@line{inputting `#1' instead of obsolete `#2'}}
\@onlypreamble\@obsoletefile
```

18.2 Listing files

\@filelist A list of files input so far.

```
\let\@filelist\@gobble
```

\@addtofilelist Add to the list of files input so far.

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

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

```
\def\listfiles{%
  \let\listfiles\relax
  \def\@listfiles##1##2##3##4##5##6##7##8##9\@@{%
     \def\reserved@d{\\}%
     \@tfor\reserved@c:=##1##2##3##4##5##6##7##8\do{%
       \ifx\reserved@c\reserved@d
         \edef\filename@area{ \filename@area}%
       \fi}}%

  \def\@dofilelist{%
     \typeout{^^J *File List*}%
     \@for\@currname:=\@filelist\do{%
       \filename@parse\@currname
       \edef\reserved@a{%
          \filename@base.%
          \ifx\filename@ext\relax tex\else\filename@ext\fi}%
       \expandafter\let\expandafter\reserved@b
                              \csname ver@\reserved@a\endcsname
       \expandafter\expandafter\expandafter\@listfiles\expandafter
             \filename@area\filename@base\\\\\\\\\\\\\\\\\\\@@
```

```
      \typeout{%
        \filename@area\reserved@a
        \ifx\reserved@b\relax\else\@spaces\reserved@b\fi}}%
    \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}
```

```
\@onlypreamble\listfiles
```

`\@dofilelist`

```
\let\@dofilelist\relax
```

```
⟨/2ekernel⟩
```

File l
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 the `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 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 `\aa` in the `OT1` encoding is:

```
\DeclareTextCommand{\aa}{OT1}{{\accent23a}}
```

`\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}[1]{{\accent127#1}}
```

have the same effect.

`\DeclareTextComposite{`⟨*command*⟩`}{`⟨*encoding*⟩`}{`⟨*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* should have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

`\DeclareTextCompositeCommand{`⟨*command*⟩`}{`⟨*encoding*⟩`}{`⟨*text*⟩`}`

For example, you can declare that `\'i` is the same as `\'\i` by saying:

```
\DeclareTextCompositeCommand{\'}{OT1}{i}{\'\i}
```

The *command* should have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above commands can be used in two ways. Normally they can 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{\ss}{OT1}` 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.

`\DeclareTextCommandDefault{`⟨*command*⟩`}{`⟨*definition*⟩`}`

For example, the default definition of the command `\textonequarter` (which produces the fraction $\frac{1}{4}$) is built using math mode:

```
\DeclareTextCommandDefault{\textonequarter}{$\frac14$}
```

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 Docstrip modules

This `.dtx` file is be used to generate several related files containing font encoding definitions. The mutually exculsive docstrip options are:

`T1`	generates `T1enc.def` for the Cork encoding,
`OT1`	generates `OT1enc.def` for Knuth's CM encoding,
`OMS`	generates `OMSenc.def` for Knuth's math symbol encoding,
`OML`	generates `OMLenc.def` for Knuth's math letters encoding,
`package`	generates `fontenc.sty` for selecting encodings,
`2ekernel`	for the kernel commands.

19.2 Definitions for the kernel

19.2.1 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.

```
⟨*2ekernel⟩
\message{font encodings,}
```

\DeclareTextCommand
\ProvideTextCommand
\DeclareTextSymbol
\@dec@text@cmd
\@changed@cmd
\@changed@x
\@changed@x@err

If you say:

```
\DeclareTextCommand{\foo}{T1}...
```

then `\foo` is defined to be `\T1-cmd \foo \T1\foo`, where `\T1\foo` is *one* control sequence, not two! We then call `\newcommand` to define `\T1\foo`.

```
\def\DeclareTextCommand{%
   \@dec@text@cmd\newcommand
}
\def\ProvideTextCommand{%
   \@dec@text@cmd\providecommand
}
\def\DeclareTextSymbol#1#2#3{%
   \@dec@text@cmd\chardef#1{#2}#3\relax
}
\def\@dec@text@cmd#1#2#3{%
   \expandafter\def\expandafter#2%
      \expandafter{%
         \csname#3-cmd\expandafter\endcsname
         \expandafter#2%
         \csname#3\string#2\endcsname
      }%
   \let\@ifdefinable\@rc@ifdefinable
   \expandafter#1\csname#3\string#2\endcsname
}
```

The declarations are only available before `\begin{document}`.

```
\@onlypreamble\DeclareTextCommand
\@onlypreamble\DeclareTextSymbol
```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There's 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.

```
\def\@current@cmd#1{%
   \ifx\protect\@typeset@protect\else
      \noexpand#1\expandafter\@gobble
   \fi
}
\def\@changed@cmd#1#2{%
   \ifx\protect\@typeset@protect
      \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
         \expandafter\ifx\csname ?\string#1\endcsname\relax
            \expandafter\def\csname ?\string#1\endcsname{%
               \@changed@x@err{#1}%
            }%
         \fi
         \ifmmode\else
            \expandafter\let
               \csname\cf@encoding \string#1\expandafter\endcsname
               \csname ?\string#1\endcsname
         \fi
         \csname ?\string#1%
            \expandafter\expandafter\expandafter\endcsname
      \else
         \csname\cf@encoding\string#1%
            \expandafter\expandafter\expandafter\endcsname
      \fi
   \else
      \noexpand#1%
   \fi
}
\def\@changed@x@err#1{%
   \@latex@error{%
      Command \protect#1 undefined in encoding \cf@encoding%
   }\@eha
}
```

`\DeclareTextCommandDefault`
`\ProvideTextCommandDefault`

These define commands with encoding `?`. Note that `\DeclareTextCommandDefault` can only be used in the preamble, but that the `\Provide` version is allowed in inputenc `.def` files, so is allowed anywhere.

```
\def\DeclareTextCommandDefault#1{%
```

```
   \DeclareTextCommand#1?%
}
\def\ProvideTextCommandDefault#1{%
   \ProvideTextCommand#1?%
}
\@onlypreamble\DeclareTextCommandDefault
%\@onlypreamble\ProvideTextCommandDefault
```

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

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

`\DeclareTextAccent`

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

```
\def\DeclareTextAccent#1#2#3{%
   \DeclareTextCommand#1{#2}[1]{{\accent#3 ##1}}
}
\@onlypreamble\DeclareTextAccent
```

`\DeclareTextCompositeCommand`
`\DeclareTextComposite`
`\@text@composite`
`\@text@composite@x`
`\@strip@args`

Another amusing game to play with `\expandafter`, `\csname`, and `\string`.

When you say `\DeclareTextCompositeCommand{\foo}{T1}{a}{bar}`, we first check to make sure that `\T1\foo` is a one-argument macro, by making sure that its `\meaning` begins `macro:#1-`, and call an error if it isn't. Assuming all's well, we look to see if the expansion of `\T1\foo` begins with `\@text@composite`, and if it doesn't, we redefine `\T1\foo` to be:

```
#1 -> \@text@composite \T1\foo #1\@empty \@text@composite {...}
```

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

```
\def\DeclareTextCompositeCommand#1#2#3#4{%
   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
   \edef\reserved@b{\string##1}%
   \edef\reserved@c{%
     \expandafter\@strip@args\meaning\reserved@a:-\@strip@args}%
   \ifx\reserved@b\reserved@c
      \expandafter\expandafter\expandafter\ifx
         \expandafter\@car\reserved@a\relax\relax\@nil
         \@text@composite
      \else
         \edef\reserved@b##1{%
            \def\expandafter\noexpand
               \csname#2\string#1\endcsname####1{%
               \noexpand\@text@composite
                  \expandafter\noexpand\csname#2\string#1\endcsname
                  ####1\noexpand\@empty\noexpand\@text@composite
                  {##1}%
            }%
         }%
         \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
```

```
      \fi
      \expandafter\def\csname\expandafter\string\csname
         #2\endcsname\string#1-\string#3\endcsname{#4}%
   \else
      \@latex@error{%
         \string\DeclareTextCompositeCommand\space used on
         inappropriate command \protect#1%
      }\@eha
   \fi
}
\@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 presumbaly 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 'malfeature' 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.

```
\def\@text@composite#1#2#3\@text@composite{%
   \expandafter\@text@composite@x
      \csname\string#1-\string#2\endcsname
}
\def\@text@composite@x#1#2{%
   \ifx#1\relax
      #2%
   \else
      #1%
   \fi
}
```

The `\@strip@args` command just strips the arguments part out of the `\meaning` of a description `macro:...->...`.

```
\def\@strip@args#1:#2-#3\@strip@args{#2}
```

The command `\DeclareTextComposite` uses these commands to declare a command which expands out to a letter.

```
\def\DeclareTextComposite#1#2#3#4{%
   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
   \bgroup
      \lccode`\@=#4%
      \lowercase{%
   \egroup
      \reserved@a @%
   }%
}
\@onlypreamble\DeclareTextComposite
```

\UseTextSymbol
\UseTextAccent
\@use@text@encoding

These fragile commands access glyphs from different encodings. They use grotty low-level calls to the font selection scheme for speed, and in order to make sure that `\UseTextSymbol` doesn't do anything which you're not allowed to do between an `\accent` and its glyph.

```
\def\UseTextSymbol#1#2{%
   \let\@curr@enc\cf@encoding
   \@use@text@encoding{#1}%
   #2%
   \@use@text@encoding\@curr@enc
}
\def\UseTextAccent#1#2#3{%
   \let\@curr@enc\cf@encoding
   \@use@text@encoding{#1}%
   #2{\@use@text@encoding\@curr@enc\selectfont#3}%
   \@use@text@encoding\@curr@enc
}
\def\@use@text@encoding#1{%
   \edef\f@encoding{#1}%
   \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname
   }%
   \pickup@font
   \font@name
   \@@enc@update
}
```

\DeclareTextSymbolDefault
\DeclareTextAccentDefault

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

```
\def\DeclareTextSymbolDefault#1#2{%
   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}%
}
\def\DeclareTextAccentDefault#1#2{%
   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}%
```

```
}
\@onlypreamble\DeclareTextSymbolDefault
\@onlypreamble\DeclareTextAccentDefault
```

19.2.2 Hyphenation

`\patterns` `\@@patterns` `\hyphenation` `\@@hyphenation`

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

```
%\let\@@patterns\patterns
%\let\@@hyphenation\hyphenation
%\def\patterns{%
%   \bgroup
%      \let\protect\@empty
%      \let\@typeset@protect\@empty
%      \let\@changed@x\@changed@x@mouth
%   \afterassignment\egroup
%   \@@patterns
%}
%\def\hyphenation{%
%   \bgroup
%      \let\protect\@empty
%      \let\@typeset@protect\@empty
%      \let\@changed@x\@changed@x@mouth
%   \afterassignment\egroup
%   \@@hyphenation
%}
```

19.2.3 Miscellania

`\a`

The `\a` command is used to access the accent commands even when they have been redefined (for example by the `tabbing` environment).

```
\def\a#1{\expandafter\@changed@cmd\csname#1\endcsname\relax}
```

19.2.4 Default encodings

We define the default encodings for most commands to be either OT1 or OMS.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like `\ae` and `\ss`) or by the TEX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```
\DeclareTextAccentDefault{\"}{OT1}
\DeclareTextAccentDefault{\'}{OT1}
\DeclareTextAccentDefault{\.}{OT1}
\DeclareTextAccentDefault{\=}{OT1}
\DeclareTextAccentDefault{\H}{OT1}
\DeclareTextAccentDefault{\^}{OT1}
\DeclareTextAccentDefault{\`}{OT1}
\DeclareTextAccentDefault{\b}{OT1}
\DeclareTextAccentDefault{\c}{OT1}
\DeclareTextAccentDefault{\d}{OT1}
\DeclareTextAccentDefault{\r}{OT1}
\DeclareTextAccentDefault{\u}{OT1}
\DeclareTextAccentDefault{\v}{OT1}
\DeclareTextAccentDefault{\~}{OT1}
```

Some symbols from OT1:

```
\DeclareTextSymbolDefault{\AA}{OT1}
\DeclareTextSymbolDefault{\AE}{OT1}
\DeclareTextSymbolDefault{\L}{OT1}
\DeclareTextSymbolDefault{\OE}{OT1}
\DeclareTextSymbolDefault{\O}{OT1}
\DeclareTextSymbolDefault{\aa}{OT1}
\DeclareTextSymbolDefault{\ae}{OT1}
\DeclareTextSymbolDefault{\i}{OT1}
\DeclareTextSymbolDefault{\j}{OT1}
\DeclareTextSymbolDefault{\l}{OT1}
\DeclareTextSymbolDefault{\oe}{OT1}
\DeclareTextSymbolDefault{\o}{OT1}
\DeclareTextSymbolDefault{\ss}{OT1}
\DeclareTextSymbolDefault{\textdollar}{OT1}
\DeclareTextSymbolDefault{\textemdash}{OT1}
\DeclareTextSymbolDefault{\textendash}{OT1}
\DeclareTextSymbolDefault{\textexclamdown}{OT1}
%\DeclareTextSymbolDefault{\texthyphenchar}{OT1}
%\DeclareTextSymbolDefault{\texthyphen}{OT1}
\DeclareTextSymbolDefault{\textquestiondown}{OT1}
\DeclareTextSymbolDefault{\textquotedblleft}{OT1}
\DeclareTextSymbolDefault{\textquotedblright}{OT1}
\DeclareTextSymbolDefault{\textquoteleft}{OT1}
\DeclareTextSymbolDefault{\textquoteright}{OT1}
\DeclareTextSymbolDefault{\textsterling}{OT1}
```

Some symbols from OMS (some are commented out to save memory for the moment—they may be added later):

```
%\DeclareTextSymbolDefault{\textbackslash}{OMS}
%\DeclareTextSymbolDefault{\textbar}{OMS}
\DeclareTextSymbolDefault{\textbraceleft}{OMS}
\DeclareTextSymbolDefault{\textbraceright}{OMS}
\DeclareTextSymbolDefault{\textbullet}{OMS}
```

```
\DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
\DeclareTextSymbolDefault{\textdagger}{OMS}
\DeclareTextSymbolDefault{\textparagraph}{OMS}
\DeclareTextSymbolDefault{\textperiodcentered}{OMS}
\DeclareTextSymbolDefault{\textsection}{OMS}
\DeclareTextAccentDefault{\textcircled}{OMS}
```

Some symbols from OML:

```
%\DeclareTextSymbolDefault{\textless}{OML}
%\DeclareTextSymbolDefault{\textgreater}{OML}
\DeclareTextAccentDefault{\t}{OML}
```

Some defaults we can fake:

```
\DeclareTextCommandDefault{\copyright}{{\textcircled{c}}}
%\DeclareTextCommandDefault{\textasciicircum}{\^{}}
%\DeclareTextCommandDefault{\textasciitilde}{\~{}}
\DeclareTextCommandDefault{\textcompwordmark}{\kern\z@}
\DeclareTextCommandDefault{\textunderscore}{%
   \leavevmode\mathunderscore
}
\DeclareTextCommandDefault{\textvisiblespace}{%
   \mbox{\kern.06em\vrule \@height.3ex}%
   \vbox{\hrule \@width.3em}%
   \hbox{\vrule \@height.3ex}%
}
\DeclareTextCommandDefault{\textellipsis}{%
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font
   .\kern\fontdimen3\font
}
```

19.2.5 Math material

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

```
\DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}
\DeclareRobustCommand{\{}{\ifmmode\lbrace\else\textbraceleft\fi}
\DeclareRobustCommand{\}}{\ifmmode\rbrace\else\textbraceright\fi}
\DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
\DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
\DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
\DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}
\DeclareRobustCommand{\_}{%
   \ifmmode\mathunderscore\else\textunderscore\fi
}
\DeclareRobustCommand{\pounds}{%
   \ifmmode\mathsterling\else\textsterling\fi
}
\DeclareRobustCommand{\dots}{%
   \ifmmode\mathellipsis\else\textellipsis\fi
}
```

```
\let\ldots\dots
⟨/2ekernel⟩
```

19.3 The fontenc package

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

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fontenc}[1994/12/14 v1.7f Standard LaTeX package]
\DeclareOption*{%
   \let\encodingdefault\CurrentOption
   \InputIfFileExists{\CurrentOption enc.def}{}{%
      \@warning{Encoding file `\CurrentOption enc.def' not found}%
   }%
}
\ProcessOptions*
\fontencoding\encodingdefault\selectfont
⟨/package⟩
```

19.4 Definitions for the OT1 encoding

The definitions for the 'TEX text' (OT1) encoding.

```
⟨*OT1⟩
\ProvidesFile{OT1enc.def}[1994/12/14 Standard LaTeX file]
```

Declare the encoding.

```
\DeclareFontEncoding{OT1}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\'}{OT1}{19}
\DeclareTextAccent{\.}{OT1}{95}
\DeclareTextAccent{\=}{OT1}{22}
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\`}{OT1}{18}
\DeclareTextAccent{\~}{OT1}{126}
\DeclareTextAccent{\H}{OT1}{125}
\DeclareTextAccent{\u}{OT1}{21}
\DeclareTextAccent{\v}{OT1}{20}
\DeclareTextAccent{\r}{OT1}{23}
```

Some accents have to be built by hand:

```
\DeclareTextCommand{\b}{OT1}[1]
   {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{29}%
     \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}}}
\DeclareTextCommand{\c}{OT1}[1]
```

```
  {\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
   \else{\ooalign{\unhbox\z@\crcr\hidewidth\char24\hidewidth}}\fi}
\DeclareTextCommand{\d}{OT1}[1]
  {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{10}.\hidewidth}}}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AE}{OT1}{29}
\DeclareTextSymbol{\OE}{OT1}{30}
\DeclareTextSymbol{\O}{OT1}{31}
\DeclareTextSymbol{\ae}{OT1}{26}
\DeclareTextSymbol{\i}{OT1}{16}
\DeclareTextSymbol{\j}{OT1}{17}
\DeclareTextSymbol{\oe}{OT1}{27}
\DeclareTextSymbol{\o}{OT1}{28}
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextSymbol{\textemdash}{OT1}{124}
\DeclareTextSymbol{\textendash}{OT1}{123}
\DeclareTextSymbol{\textexclamdown}{OT1}{60}
%\DeclareTextSymbol{\texthyphenchar}{OT1}{`\-}
%\DeclareTextSymbol{\texthyphen}{OT1}{`\-}
\DeclareTextSymbol{\textquestiondown}{OT1}{62}
\DeclareTextSymbol{\textquotedblleft}{OT1}{92}
\DeclareTextSymbol{\textquotedblright}{OT1}{`\"}
\DeclareTextSymbol{\textquoteleft}{OT1}{`\`}
\DeclareTextSymbol{\textquoteright}{OT1}{`\'}
```

Some symbols which are faked from others:

```
\DeclareTextCommand{\aa}{OT1}
  {{\accent23a}}
\DeclareTextCommand{\L}{OT1}
  {\leavevmode\setbox0\hbox{L}\hb@xt@\wd0{\hss\char32L}}
\DeclareTextCommand{\l}{OT1}
  {{\char32l}}
\DeclareTextCommand{\AA}{OT1}
  {\leavevmode\setbox0\hbox{h}\dimen@\ht0\advance\dimen@-1ex%
   \rlap{\raise.67\dimen@\hbox{\char'27}}A}
\DeclareTextCommand{\SS}{OT1}
  {SS}
```

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

```
\DeclareTextCommand{\textdollar}{OT1}{{%
  \ifdim \fontdimen\@ne\font >\z@
     \slshape
  \else
     \upshape
  \fi
  \char`\$}}
\DeclareTextCommand{\textsterling}{OT1}{{%
  \ifdim \fontdimen\@ne\font >\z@
     \itshape
  \else
```

```
      \fontshape{ui}\selectfont
  \fi
  \char`\$}}
⟨/OT1⟩
```

19.5 Definitions for the T1 encoding

The definitions for the 'Extended TEX text' (T1) encoding.

```
⟨*T1⟩
\ProvidesFile{T1enc.def}[1994/12/14 Standard LaTeX file]
```

Declare the encoding.

```
\DeclareFontEncoding{T1}{}{}
```

Declare the accents.

```
\DeclareTextAccent{\`}{T1}{0}
\DeclareTextAccent{\'}{T1}{1}
\DeclareTextAccent{\^}{T1}{2}
\DeclareTextAccent{\~}{T1}{3}
\DeclareTextAccent{\"}{T1}{4}
\DeclareTextAccent{\H}{T1}{5}
\DeclareTextAccent{\r}{T1}{6}
\DeclareTextAccent{\v}{T1}{7}
\DeclareTextAccent{\u}{T1}{8}
\DeclareTextAccent{\=}{T1}{9}
\DeclareTextAccent{\.}{T1}{10}
```

Some accents have to be built by hand:

```
\DeclareTextCommand{\b}{T1}[1]
  {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{29}%
    \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}}}
\DeclareTextCommand{\c}{T1}[1]
  {\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
    \else{\ooalign{\hidewidth\char11\hidewidth
      \crcr\unhbox\z@}}\fi}
\DeclareTextCommand{\d}{T1}[1]
  {{\o@lign{\relax#1\crcr\hidewidth\sh@ft{10}.\hidewidth}}}
\DeclareTextCommand{\k}{T1}[1]
  {\oalign{\null#1\crcr\hidewidth\char12}}
```

Declare the text symbols.

```
\DeclareTextSymbol{\AA}{T1}{197}
\DeclareTextSymbol{\AE}{T1}{198}
\DeclareTextSymbol{\DH}{T1}{208}
\DeclareTextSymbol{\DJ}{T1}{208}
\DeclareTextSymbol{\L}{T1}{138}
\DeclareTextSymbol{\NG}{T1}{141}
\DeclareTextSymbol{\OE}{T1}{215}
\DeclareTextSymbol{\O}{T1}{216}
\DeclareTextSymbol{\SS}{T1}{223}
\DeclareTextSymbol{\TH}{T1}{222}
```

```
\DeclareTextSymbol{\aa}{T1}{229}
\DeclareTextSymbol{\ae}{T1}{230}
\DeclareTextSymbol{\dh}{T1}{240}
\DeclareTextSymbol{\dj}{T1}{158}
\DeclareTextSymbol{\guillemotleft}{T1}{19}
\DeclareTextSymbol{\guillemotright}{T1}{20}
\DeclareTextSymbol{\guilsinglleft}{T1}{14}
\DeclareTextSymbol{\guilsinglright}{T1}{15}
\DeclareTextSymbol{\i}{T1}{25}
\DeclareTextSymbol{\j}{T1}{26}
\DeclareTextSymbol{\l}{T1}{170}
\DeclareTextSymbol{\ng}{T1}{173}
\DeclareTextSymbol{\oe}{T1}{247}
\DeclareTextSymbol{\o}{T1}{248}
\DeclareTextSymbol{\quotedblbase}{T1}{18}
\DeclareTextSymbol{\quotesinglbase}{T1}{13}
\DeclareTextSymbol{\ss}{T1}{255}
%\DeclareTextSymbol{\textasciicircum}{T1}{`\^}
%\DeclareTextSymbol{\textasciitilde}{T1}{`\~}
%\DeclareTextSymbol{\textbackslash}{T1}{`\\}
%\DeclareTextSymbol{\textbar}{T1}{`\|}
\DeclareTextSymbol{\textbraceleft}{T1}{`\{}
\DeclareTextSymbol{\textbraceright}{T1}{`\}}
\DeclareTextSymbol{\textcompwordmark}{T1}{23}
\DeclareTextSymbol{\textdollar}{T1}{`\$}
\DeclareTextSymbol{\textemdash}{T1}{22}
\DeclareTextSymbol{\textendash}{T1}{21}
\DeclareTextSymbol{\textexclamdown}{T1}{189}
%\DeclareTextSymbol{\textgreater}{T1}{`\>}
%\DeclareTextSymbol{\texthyphenchar}{T1}{127}
%\DeclareTextSymbol{\texthyphen}{T1}{`\-}
%\DeclareTextSymbol{\textless}{T1}{`\<}
\DeclareTextSymbol{\textquestiondown}{T1}{190}
\DeclareTextSymbol{\textquotedblleft}{T1}{16}
\DeclareTextSymbol{\textquotedblright}{T1}{17}
\DeclareTextSymbol{\textquotedbl}{T1}{`\"}
\DeclareTextSymbol{\textquoteleft}{T1}{`\`}
\DeclareTextSymbol{\textquoteright}{T1}{`\'}
\DeclareTextSymbol{\textsection}{T1}{159}
\DeclareTextSymbol{\textsterling}{T1}{191}
\DeclareTextSymbol{\textunderscore}{T1}{95}
\DeclareTextSymbol{\textvisiblespace}{T1}{32}
\DeclareTextSymbol{\th}{T1}{254}
```

Declare the composites.

```
\DeclareTextComposite{\.}{T1}{i}{`\i}
\DeclareTextComposite{\u}{T1}{A}{128}
\DeclareTextComposite{\k}{T1}{A}{129}
\DeclareTextComposite{\'}{T1}{C}{130}
\DeclareTextComposite{\v}{T1}{C}{131}
```

```
\DeclareTextComposite{\v}{T1}{D}{132}
\DeclareTextComposite{\v}{T1}{E}{133}
\DeclareTextComposite{\k}{T1}{E}{134}
\DeclareTextComposite{\u}{T1}{G}{135}
\DeclareTextComposite{\'}{T1}{L}{136}
\DeclareTextComposite{\v}{T1}{L}{137}
\DeclareTextComposite{\'}{T1}{N}{139}
\DeclareTextComposite{\v}{T1}{N}{140}
\DeclareTextComposite{\H}{T1}{O}{142}
\DeclareTextComposite{\'}{T1}{R}{143}
\DeclareTextComposite{\v}{T1}{R}{144}
\DeclareTextComposite{\'}{T1}{S}{145}
\DeclareTextComposite{\v}{T1}{S}{146}
\DeclareTextComposite{\c}{T1}{S}{147}
\DeclareTextComposite{\v}{T1}{T}{148}
\DeclareTextComposite{\c}{T1}{T}{149}
\DeclareTextComposite{\H}{T1}{U}{150}
\DeclareTextComposite{\r}{T1}{U}{151}
\DeclareTextComposite{\"}{T1}{Y}{152}
\DeclareTextComposite{\'}{T1}{Z}{153}
\DeclareTextComposite{\v}{T1}{Z}{154}
\DeclareTextComposite{\.}{T1}{Z}{155}
\DeclareTextComposite{\.}{T1}{I}{157}
\DeclareTextComposite{\u}{T1}{a}{160}
\DeclareTextComposite{\k}{T1}{a}{161}
\DeclareTextComposite{\'}{T1}{c}{162}
\DeclareTextComposite{\v}{T1}{c}{163}
\DeclareTextComposite{\v}{T1}{d}{164}
\DeclareTextComposite{\v}{T1}{e}{165}
\DeclareTextComposite{\k}{T1}{e}{166}
\DeclareTextComposite{\u}{T1}{g}{167}
\DeclareTextComposite{\'}{T1}{l}{168}
\DeclareTextComposite{\v}{T1}{l}{169}
\DeclareTextComposite{\'}{T1}{n}{171}
\DeclareTextComposite{\v}{T1}{n}{172}
\DeclareTextComposite{\H}{T1}{o}{174}
\DeclareTextComposite{\'}{T1}{r}{175}
\DeclareTextComposite{\v}{T1}{r}{176}
\DeclareTextComposite{\'}{T1}{s}{177}
\DeclareTextComposite{\v}{T1}{s}{178}
\DeclareTextComposite{\c}{T1}{s}{179}
\DeclareTextComposite{\v}{T1}{t}{180}
\DeclareTextComposite{\c}{T1}{t}{181}
\DeclareTextComposite{\H}{T1}{u}{182}
\DeclareTextComposite{\r}{T1}{u}{183}
\DeclareTextComposite{\"}{T1}{y}{184}
\DeclareTextComposite{\'}{T1}{z}{185}
\DeclareTextComposite{\v}{T1}{z}{186}
\DeclareTextComposite{\.}{T1}{z}{187}
\DeclareTextComposite{\`}{T1}{A}{192}
```

```
\DeclareTextComposite{\'}{T1}{A}{193}
\DeclareTextComposite{\^}{T1}{A}{194}
\DeclareTextComposite{\~}{T1}{A}{195}
\DeclareTextComposite{\"}{T1}{A}{196}
\DeclareTextComposite{\r}{T1}{A}{197}
\DeclareTextComposite{\c}{T1}{C}{199}
\DeclareTextComposite{\`}{T1}{E}{200}
\DeclareTextComposite{\'}{T1}{E}{201}
\DeclareTextComposite{\^}{T1}{E}{202}
\DeclareTextComposite{\"}{T1}{E}{203}
\DeclareTextComposite{\`}{T1}{I}{204}
\DeclareTextComposite{\'}{T1}{I}{205}
\DeclareTextComposite{\^}{T1}{I}{206}
\DeclareTextComposite{\"}{T1}{I}{207}
\DeclareTextComposite{\~}{T1}{N}{209}
\DeclareTextComposite{\`}{T1}{O}{210}
\DeclareTextComposite{\'}{T1}{O}{211}
\DeclareTextComposite{\^}{T1}{O}{212}
\DeclareTextComposite{\~}{T1}{O}{213}
\DeclareTextComposite{\"}{T1}{O}{214}
\DeclareTextComposite{\`}{T1}{U}{217}
\DeclareTextComposite{\'}{T1}{U}{218}
\DeclareTextComposite{\^}{T1}{U}{219}
\DeclareTextComposite{\"}{T1}{U}{220}
\DeclareTextComposite{\'}{T1}{Y}{221}
\DeclareTextComposite{\`}{T1}{a}{224}
\DeclareTextComposite{\'}{T1}{a}{225}
\DeclareTextComposite{\^}{T1}{a}{226}
\DeclareTextComposite{\~}{T1}{a}{227}
\DeclareTextComposite{\"}{T1}{a}{228}
\DeclareTextComposite{\r}{T1}{a}{229}
\DeclareTextComposite{\c}{T1}{c}{231}
\DeclareTextComposite{\`}{T1}{e}{232}
\DeclareTextComposite{\'}{T1}{e}{233}
\DeclareTextComposite{\^}{T1}{e}{234}
\DeclareTextComposite{\"}{T1}{e}{235}
\DeclareTextComposite{\`}{T1}{i}{236}
\DeclareTextComposite{\`}{T1}{\i}{236}
\DeclareTextComposite{\'}{T1}{i}{237}
\DeclareTextComposite{\'}{T1}{\i}{237}
\DeclareTextComposite{\^}{T1}{i}{238}
\DeclareTextComposite{\^}{T1}{\i}{238}
\DeclareTextComposite{\"}{T1}{i}{239}
\DeclareTextComposite{\"}{T1}{\i}{239}
\DeclareTextComposite{\~}{T1}{n}{241}
\DeclareTextComposite{\`}{T1}{o}{242}
\DeclareTextComposite{\'}{T1}{o}{243}
\DeclareTextComposite{\^}{T1}{o}{244}
\DeclareTextComposite{\~}{T1}{o}{245}
\DeclareTextComposite{\"}{T1}{o}{246}
```

```
\DeclareTextComposite{\`}{T1}{u}{249}
\DeclareTextComposite{\'}{T1}{u}{250}
\DeclareTextComposite{\^}{T1}{u}{251}
\DeclareTextComposite{\"}{T1}{u}{252}
\DeclareTextComposite{\'}{T1}{y}{253}
⟨/T1⟩
```

19.6 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 LATEX text symbols.

```
⟨*OMS⟩
\ProvidesFile{OMSenc.def}[1994/12/14 Standard LaTeX file]
```

Declare the encoding.

```
\DeclareFontEncoding{OMS}{}{}
```

Declare the symbols.

```
%\DeclareTextSymbol{\textbackslash}{OMS}{"6E}
%\DeclareTextSymbol{\textbar}{OMS}{"6A}
\DeclareTextSymbol{\textbraceleft}{OMS}{"66}
\DeclareTextSymbol{\textbraceright}{OMS}{"67}
\DeclareTextSymbol{\textbullet}{OMS}{"0F}
\DeclareTextSymbol{\textdaggerdbl}{OMS}{"7A}
\DeclareTextSymbol{\textdagger}{OMS}{"79}
\DeclareTextSymbol{\textparagraph}{OMS}{"7B}
\DeclareTextSymbol{\textperiodcentered}{OMS}{"01}
\DeclareTextSymbol{\textsection}{OMS}{"78}
\DeclareTextCommand{\textcircled}{OMS}[1]{{%
   \ooalign{%
      \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
      \char"0D
   }%
}}
⟨/OMS⟩
```

19.7 Definitions for the OML encoding

The definitions for the 'TEX math italic' (OML) encoding. Even though this is meant to be a math font, it includes some of the standard LATEX text symbols.

```
⟨*OML⟩
\ProvidesFile{OMLenc.def}[1994/12/14 Standard LaTeX file]
```

Declare the encoding.

```
\DeclareFontEncoding{OML}{}{}
```

Declare the symbols.

```
%\DeclareTextSymbol{\textless}{OML}{`\<}
%\DeclareTextSymbol{\textgreater}{OML}{`\>}
```

```
\DeclareTextAccent{\t}{OML}{"7F}
⟨/OML⟩
```

File m

ltfssbas.dtx

This file contains the main implementation of the 'low level' font selection commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of the LaTeX '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.

20 Preliminary macros

We define a number of macros that will be used later.

\@spaces

We define `\@spaces` to be an abbreviation for four space tokens.

```
⟨*!2ekernel⟩
\def\@spaces{\space\space\space\space}
⟨/!2ekernel⟩
```

This is also defined in `latex.tex`, but this code cautiously does not assume that any macros are defined elsewhere (except those in `plain.tex`).

\@empty

The `\@empty` macro expands to nothing and is used to test for empty replacement texts.

```
⟨*!2ekernel⟩
\def\@empty{}
```

\@height
\@depth
\@width

The `\@height`, `\@depth` and `\@width` macros are made to conserve token memory.

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

We will now define a few useful macros from `latex.tex`.

\@namedef

`\@namedef` is used to define a macro with unusual characters in its name.

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

\@ifnextchar
\@ifnch
\@sptoken
\@ifxnch

`\@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.

```
\def\@ifnextchar#1#2#3{%
  \let\reserved@e=#1\def\reserved@a{#2}\def\reserved@b{#3}%
    \futurelet\reserved@c\@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.

```
\def\@ifnch{\ifx \reserved@c \@sptoken \let\reserved@d\@xifnch
      \else \ifx \reserved@c \reserved@e\let\reserved@d\reserved@a
            \else\let\reserved@d\reserved@b\fi \fi \reserved@d}
```

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.

```
\def\:{\let\@sptoken= } \:
```

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

```
\def\:{\@xifnch} \expandafter\def\: {\futurelet\reserved@c\@ifnch}
⟨/!2ekernel⟩
```

`\@nomath`

`\@nomath` is used by most macros that will have no effect in math mode. It issues a warning message.

```
⟨*2ekernel⟩
\def\@nomath#1{\relax\ifmmode
   \@font@warning{Command \noexpand#1invalid in math mode}\fi}
```

`\no@alphabet@error`
`\no@alphabet@help`

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 TEX from scanning too far in certain situations.

```
\def\no@alphabet@error#1{\relax \ifmmode
    \@latex@error{Math alphabet identifier \noexpand#1is undefined
                 in math version `\math@version'}%
        {Your requested math alphabet is undefined in the current
         math version.^^JCheck the spelling or use the
         \noexpand\SetMathAlphabet command.}
     \fi}
```

`\new@mathgroup`
`\mathgroup`

We also give a new name to `\newfam` and `\fam` to avoid verbal confusion (see the introduction).[2]

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

[2]For the same reason it seems advisable to `\let\fam` and `\newfam` equal to `\relax`, but this is commented out to retain compatibility to existing style files.

21 Macros for setting up the tables

\DeclareFontShape

The macro `\DeclareFontShape` takes 6 arguments:

```
\def\DeclareFontShape{\begingroup
```

First we restore the catcodes of all characters used in the syntax.

```
  \nfss@catcodes
```

We use `\expandafter \endgroup` to restore catcode in case something goes wrong with the argument parsing (suggested by Tim Van Zandt)

```
  \expandafter\endgroup
  \DeclareFontShape@}
\def\DeclareFontShape@#1#2#3#4#5#6{%
  \expandafter\ifx\csname #1+#2\endcsname\undefined
    \@latex@error{Font family `#1+#2' unknown}\@eha
  \else
    \expandafter
      \xdef\csname#1/#2/#3/#4\endcsname{\expandafter\noexpand
                                    \csname #5\endcsname}%
    \def\reserved@a{#6}%
    \global
    \expandafter\let\csname#5\expandafter\endcsname
       \ifx\reserved@a\@empty
         \@empty
       \else
         \reserved@a
       \fi
  \fi
  }
```

\DeclareFixedFont

Define a direct font switch that avoids all overhead.

```
\def\DeclareFixedFont#1#2#3#4#5#6{%
  \begingroup
     \math@fontsfalse
     \every@math@size{}%
     \fontsize{#6}\z@
     \usefont{#2}{#3}{#4}{#5}%
     \global\expandafter\let\expandafter#1\the\font
  \endgroup
  }
⟨/2ekernel⟩
```

\@makeother

We need just another macro from `latex.tex`:

```
⟨*!2ekernel⟩
\def\@makeother#1{\catcode`#1=12\relax}
⟨/!2ekernel⟩
```

\dosubst@fontshape

The code could be shortened by supplying the information in a different way. This should be investigated.

```
%\def\dosubst@fontshape#1#2{%
```

```
%        \@font@warning{Font/shape `#1' not available\MessageBreak
%                       Substituting `#2' instead}%
```

We now prepare a macro that will reload the real font, i.e. the un-substituted one after the current group. This will correct the problem that loading a substitution font means renaming the font identifier of record.

```
%        \def\curr@fontshape{#2}%
%        \do@subst@correction
```

We now let the font shape definition locally (i.e. inside the group mentioned above) to the font shape definition of the substitution font. When the font is loaded this will be restored and therefore further substitution will warn again.

```
%        \expandafter\let
%          \csname#1\expandafter\endcsname
%          \csname#2\endcsname}
```

\do@subst@correction

```
⟨*2ekernel⟩
\def\do@subst@correction{%
       \xdef\subst@correction{%
          \font@name
          \global\expandafter\font
            \csname \curr@fontshape/\f@size\endcsname
            \noexpand\fontname\font
           \relax}%
```

Calling `\subst@correction` after the current group means calling it after we have loaded the substitution font which is done inside a group.

```
       \aftergroup\subst@correction
}
```

\extra@def The 'extra' macro is defined as follows: we simply put a '+' character in front of the *family* name.

```
\expandafter\def\csname +cmtt\endcsname#1{%
   \hyphenchar#1\m@ne}
```

We provide an abbreviation for this:

```
 \def\extra@def#1#2{%
   \expandafter\def\csname +#1\endcsname##1{#2}}
```

so that the above definition looks like

```
\extra@def{cmtt}{\hyphenchar#1\m@ne}
```

However, this is inefficient if there is nothing to do (i.e. if the second argument is empty), so we provide a special test for this case. So here is the actual definition:

[definition of `\extra@def` removed]

```
\def\DeclareFontFamily#1#2#3{%
```

If we want fast checking for the encoding scheme we can just check for `\T@..` being defined.

```
% \@tempswafalse
% \def\reserved@b{#1}%
% \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
%       \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
% \cdp@list
% \if@tempswa
 \@ifundefined{T@#1}%
    {%
     \@latex@error{Encoding scheme  `#1' unknown}\@eha
    }%
    {%
```

We store the argument `#3` in a temporary macro `\reserved@a`. This must have one parameter since `#1` is allowed in the third argument of `\extra@def` (otherwise TEX will not accept the definition).

```
      \def\reserved@a{#3}%
```

We compare `\reserved@a` with a macro with one argument and empty replacement text, i.e. with `\@gobble`. If these two are the same we `\let` the 'extra' macro equal to `\@gobble`.

```
      \global
      \expandafter\let\csname #1+#2\expandafter\endcsname
             \ifx \reserved@a\@empty
               \@empty
             \else \reserved@a
             \fi
    }%
}
```

`\DeclareFontEncoding`

```
\let\cdp@list\@empty
\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.

```
  \begingroup
  \nfss@catcodes
  \expandafter\endgroup
  \DeclareFontEncoding@}
```

```
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
     \def\cdp@elt{\noexpand\cdp@elt}%
     \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                    {\default@family}{\default@series}%
                    {\default@shape}}%
```

To support encoding dependent commands (like accents) we initialise the command \⟨*encoding*⟩`-cmd` to be `\@changed@cmd`. (See `ltoutenc.dtx` for details.)

```
     \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
  }

\def\DeclareFontSubstitution#1#2#3#4{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \@latex@error{Encoding scheme  `#1' unknown}\@eha
  \else
    \begingroup
       \def\reserved@a{#1}%
       \toks@{}%
       \def\cdp@elt##1##2##3##4{%
          \def\reserved@b{##1}%
          \ifx\reserved@a\reserved@b
             \addto@hook\toks@{\cdp@elt{#1}{#2}{#3}{#4}}%
          \else
             \addto@hook\toks@{\cdp@elt{##1}{##2}{##3}{##4}}%
          \fi}%
        \cdp@list
        \xdef\cdp@list{\the\toks@}%
    \endgroup
    \global
    \@namedef{D@#1}{%
           \def\default@family{#2}%
           \def\default@series{#3}%
           \def\default@shape{#4}%
           }%
  \fi
 }
\def\DeclareFontEncodingDefaults#1#2{%
  \ifx\relax#1\else
    \ifx\default@T\@empty\else
      \@font@info{Overwriting encoding scheme text defaults}%
    \fi
    \gdef\default@T{#1}%
  \fi
  \ifx\relax#2\else
    \ifx\default@M\@empty\else
      \@font@info{Overwriting encoding scheme math defaults}%
    \fi
    \gdef\default@M{#2}%
```

```
  \fi
}
\let\default@T\@empty
\let\default@M\@empty
```

`\preload@sizes` As we already explained the macro `\preload@sizes` provides a convenient way to specify fonts to be preloaded. It's definition is as follows: it takes four arguments and opens a group to keep everything local except the `\font` assignments which are executed globally.

[old macro removed]

```
\def\DeclarePreloadSizes#1#2#3#4#5{%
 \@ifundefined{T@#1}%
   {\@latex@error{Encoding scheme  `#1' unknown}\@eha}%
   {%
```

Don't know at the moment what this group here does!

```
    \begingroup
```

We define a macro `\reserved@f`[3] that grabs the next *size* and loads the corresponding font. This is done by delimiting `\reserved@f`'s only argument by the token `,` (comma).

```
      \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.)

```
        \if>##1>%
          \let\reserved@f\relax
        \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.

```
          \xdef\font@name{\csname#1/#2/#3/#4/##1\endcsname}%
          \pickup@font
```

Now we forget the name of the font just loaded. More precisely, we set the corresponding control sequence to `\relax`. This means that later on, when the font is first used, the macro `\define@newfont` is called again to execute the 'extra' macro for this font.

```
          \global\expandafter\let\font@name\relax
        \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.

```
        \reserved@f}%
```

[3] We cannot use `\@tempa` since it is needed in `\pickup@font`.

We finish with reinserting the list of sizes after the `\reserved@f` macro and appending an empty element so that the end of the list is recognized properly.

```
    \reserved@f#5,,%
  \endgroup
  }%
}
```

`\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.

```
\newif\ifmath@fonts \math@fontstrue
```

`\DeclareMathSizes` `\DeclareMathSizes*` `\DeclareMathSizes` takes the text size, math text size, math script size, and math scriptscript size as arguments and defines the right `\S@`... macro.

```
\def\DeclareMathSizes{%
  \@ifstar{\@DeclareMathSizes\math@fontsfalse}%
          {\@DeclareMathSizes{}}}
\def\@DeclareMathSizes#1#2#3#4#5{%
    \@defaultunits\dimen@#2pt\relax\@nnil
    \def\reserved@a{#3}%
    \ifx\reserved@a\@empty
      \expandafter \let
        \csname S@\strip@pt\dimen@\endcsname
        \math@fontsfalse
    \else
      \expandafter \gdef
      \csname S@\strip@pt\dimen@\endcsname
           {\gdef\tf@size{#3}\gdef\sf@size{#4}%
                             \gdef\ssf@size{#5}%
            #1%
                             }%
    \fi}
```

22 Selecting a new font

22.1 Macros for the user

`\fontencoding` `\fontfamily` `\fontseries` `\fontshape` As we said in the introduction a font is described by four parameters. We first define macros to specify the wanted *family*, *series*, or *shape*. These are simply recorded in internal macros `\f@family`, `\f@series`, and `\f@shape`, resp. We use `\edef`'s so that the arguments can also be macros.

`\f@encoding` `\f@family` `\f@series` `\f@shape`

```
\DeclareRobustCommand\fontencoding[1]{%
    \expandafter\ifx\csname T@#1\endcsname\relax
      \@latex@error{Encoding scheme `#1' unknown}\@eha
    \else
      \edef\f@encoding{#1}%
```

```
      \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`.

```
        \let\enc@update\relax
      \else
```

If current and new encoding differ we define the macro `\enc@update` to contain all updates necessary at `\selectfont` time.

```
        \let\enc@update\@@enc@update
      \fi
    \fi
}
```

```
\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`.

```
          \expandafter
          \let
            \csname\cf@encoding -cmd\endcsname
            \@changed@cmd
```

Then we turn the one for the new encoding to `\@current@cmd` (see `ltoutenc.dtx` for further explanations).

```
          \expandafter
          \let
            \csname\f@encoding-cmd\endcsname
            \@current@cmd
```

We execute the default settings `\default@T`, followed by the one for the new encoding.

```
        \default@T
        \csname T@\f@encoding\endcsname
```

Finally we change the default substitution values, disable `\enc@update` and make `\f@encoding` officially the current encoding.

```
        \csname D@\f@encoding\endcsname
        \let\enc@update\relax
        \let\cf@encoding\f@encoding
}
```

`\enc@update` The default action in `\selectfont` is to do nothing.

```
\let\enc@update\relax
```

```
\DeclareRobustCommand\fontfamily[1]{\edef\f@family{#1}}
\DeclareRobustCommand\fontseries[1]{\edef\f@series{#1}}
\DeclareRobustCommand\fontshape [1]{\edef\f@shape{#1}}
```

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.

```
\def\usefont#1#2#3#4{\fontencoding{#1}\fontfamily{#2}%
             \fontseries{#3}\fontshape{#4}\selectfont
             \ignorespaces}
```

`\linespread` The comand `\linespread` changes the current `\baselinestretch` by calling `\set@fontsize`. The values for `\f@size` and `\f@baselineskip` will be left unchanged.

```
\DeclareRobustCommand\linespread[1]
   {\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 efectively overwritten by a size change.

```
\DeclareRobustCommand\fontsize[2]
   {\set@fontsize\baselinestretch{#1}{#2}}
```

`\f@linespread` This macro holds the current internal value for `\baselinestretch`.

```
\let\f@family\@empty
\let\f@series\@empty
\let\f@shape\@empty
\let\f@size\@empty
\let\f@baselineskip\@empty
\let\f@linespread\@empty
```

`\cf@encoding`

```
\let\f@encoding\@empty
\let\cf@encoding\@empty
```

`\f@size` `\setnew@baselineskip` `\fontsize` We also define a macro that allows to specify a size. In this case, however, we also need the value of `\baselineskip`. We cannot set `\baselineskip` immediately, so it is recorded in the macro `\setnew@baselineskip`. We use `\edef` here because the second argument (`#2`) might be a macro. Later on (in `\selectfont`) we will check `\f@size` against `\glb@currsize`. In this way we are able to determine with only one test whether one of these parameters has changed.

```
%\DeclareRobustCommand\fontsize[2]{%
%    \@defaultunits\@tempdimb#1pt\relax\@nnil
%    \edef\f@size{\strip@pt\@tempdimb}%
%    \@defaultunits\@tempskipa #2pt\relax\@nnil
%    \edef\f@baselineskip{\the\@tempskipa}}
```

`\@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.

```
\def\@defaultunits{\afterassignment\remove@to@nnil}
```

`\strip@pt` `\rem@pt` This macro strips the characters `pt` produced by using `\the` on a dimen register.

```
\begingroup
  \catcode`P=12
  \catcode`T=12
  \lowercase{
    \def\x{\def\rem@pt##1.##2PT{##1\ifnum##2>\z@.##2\fi}}}
  \expandafter\endgroup\x
\def\strip@pt{\expandafter\rem@pt\the}
```

`\mathversion` `\math@version` `\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.

```
\DeclareRobustCommand\mathversion[1]
          {\@nomath\mathversion
           \expandafter\ifx\csname mv@#1\endcsname\relax
           \@latex@error{Math version `#1' is not defined}\@eha\else
           \edef\math@version{#1}%
```

We need to force a math font setup both now and at the point where we return to the previous math version. Forcing a math font setup can simply be done by setting `\glb@currsize` to an invalid value since this will trigger the setup when the formula starts.

```
           \gdef\glb@currsize{}%
```

When the scope of the current `\mathversion` ends we need to restore the old setup. However this time we need to force it directly at least if we are inside math, otherwise we could wait. Another way to enhance this code here is todo the setting only if the version really has changed after all. This might be interesting in case of `amstext` and `boldsymbol`.

```
           \aftergroup\glb@settings
           \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_ε 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` `\frozen@everydisplay` New internal names for `\everymath` and `\everydisplay`.

```
\let\frozen@everymath\everymath
\let\frozen@everydisplay\everydisplay
```

`\everymath` `\everydisplay` Now we provide now user hooks that will be called in the frozen internals.

```
\newtoks\everymath
\newtoks\everydisplay
```

`\frozen@everymath` Now we define the behaviour of the frozen hooks: first check the math setup then call the user hook.

```
\frozen@everymath = {\check@mathfonts
                     \the\everymath}
```

`\frozen@everydisplay` Ditto for the display hook.

```
\frozen@everydisplay = {\check@mathfonts
                        \the\everydisplay}
```

`\if@inmath` To detect nested formulae we need a switch.

```
\newif\if@inmath
```

`\curr@math@size` This holds locally the current math size.

```
\let\curr@math@size\@empty
```

22.2 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.

```
\def\pickup@font{%
    \expandafter \ifx \font@name \relax
       \define@newfont
    \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`

```
{\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.

```
\gdef\split@name#1/#2/#3/#4/#5\@nil{\def\f@encoding{#1}%
                                    \def\f@family{#2}%
                                    \def\f@series{#3}%
                                    \def\f@shape{#4}%
                                    \def\f@size{#5}}}
```

`\curr@fontshape` Abbreviation which may get removed again for speed.

```
\def\curr@fontshape{\f@encoding/\f@family/\f@series/\f@shape}
```

`\define@newfont` Now we can tackle the problem of defining a new font.

```
\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.

```
  \begingroup
    \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`.

```
    \expandafter\expandafter\expandafter
       \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.

```
%     \expandafter\ifx
%        \csname\curr@fontshape\endcsname \relax
      \try@load@fontshape % try always
%     \fi
    \expandafter\ifx
       \csname\curr@fontshape\endcsname \relax
      \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).

```
%       \csname\curr@fontshape\endcsname
      \extract@font\fi
```

We are nearly finished and must only restore the `\escapechar` by closing the group.

```
  \endgroup}

```

```
\def\try@load@fontshape{%
  \expandafter
  \ifx\csname \f@encoding+\f@family\endcsname\relax
    \@font@info{Try loading font information for
                  \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.

```
    \global\expandafter\let
       \csname\f@encoding+\f@family\endcsname\@empty
     \makeatletter
```

Set the catcodes used in the syntax, but do it only once (this will be restored at the end of the font loading group).

```
     \nfss@catcodes
     \let\nfss@catcodes\relax
     \@input@{\f@encoding\f@family.fd}%
  \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 `\catcode`s 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.

```
\def\nfss@catcodes{%
```

First we start with ignoring all blanks and newlines.

```
     \catcode`\ 9\relax
     \catcode`\^^I9\relax
     \catcode`\^^M9\relax
```

Then we set up `\`, `{`, `}`, and `%` in case an `.fd` file is loaded during a verbatim environment.

```
     \catcode`\\\z@
     \catcode`\{\@ne
     \catcode`\}\tw@
     \catcode`\%14\relax
```

The we make sure that the important syntax parts have the right `\catcode`.

```
  \@makeother\<%
  \@makeother\>%
  \@makeother\*%
  \@makeother\.%
```

```
  \@makeother\-%
  \@makeother\/%
  \@makeother\[%
  \@makeother\]%
}
```

\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!

```
\def\DeclareErrorFont#1#2#3#4#5{%
    \xdef\error@fontshape{%
       \noexpand\expandafter\noexpand\split@name\noexpand\string
       \expandafter\noexpand\csname#1/#2/#3/#4/#5\endcsname
       \noexpand\@nil}%
```

Initialize all those internal variables which may or may not have values in the first seconds of NFSS' bootstraping 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.

```
%     \gdef\f@encoding{#1}%
    \gdef\default@family{#2}%
    \gdef\default@series{#3}%
    \gdef\default@shape{#4}%
    \global\let\f@family\default@family
    \global\let\f@series\default@series
    \global\let\f@shape\default@shape
    \gdef\f@size{#5}%
    \gdef\f@baselineskip{#5pt}%
}
```

\wrong@fontshape

Before we come to the macro `\extract@font` we have to take care of unknown *curr@fontshape* combinations. The general strategy is to issue a warning and to try a default *shape*, then a default *series*, and finally a default *family*. If this last one also fails TeX will go into an infinite loop. But if the defaults are set incorrectly one deserves nothing else!

```
\def\wrong@fontshape{%
   \csname D@\f@encoding\endcsname     % install defaults if in math
```

We remember the wanted *curr@fontshape* combination which we will need in a moment.

```
   \edef\reserved@a{\csname\curr@fontshape\endcsname}%
  \ifx\last@fontshape\reserved@a
     \errmessage{Corrupted NFSS tables}%
     \error@fontshape
  \else
```

Then we warn the user about the mess and set the shape to its default.

```
   \let\f@shape\default@shape
```

If the combination is not known, try the default *series*.

```
    \expandafter\ifx\csname\curr@fontshape\endcsname\relax
       \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!

```
        \expandafter
          \ifx\csname\curr@fontshape\endcsname\relax
           \let\f@family\default@family
        \fi \fi
  \fi
```

At this point a valid *curr@fontshape* combination must have been found. We inform the user about this fact.

```
    \@font@warning{Font shape `\reserved@a' undefined\MessageBreak
                  using `\curr@fontshape' instead}%
    \global\let\last@fontshape\reserved@a
```

We set a flag to give a general warning at `\enddocument`.

```
    \G@defaultsubstrue
```

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.

```
    \global\expandafter\expandafter\expandafter\let
       \expandafter\reserved@a
           \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.

```
    \xdef\font@name{%
      \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.

```
    \pickup@font}
```

`\ifG@defaultsubs`
`\G@defaultsubstrue`

A global flag denoting that some font substiution using internal defaults has happened. To save macro name we do not define it via `\newif` but rather define only those commands that are actually used

```
%\newif\ifG@defaultsubs
%\def\G@defaultsubsfalse{\global\let\ifG@defaultsubs\iffalse}
\def\G@defaultsubstrue{\global\let\ifG@defaultsubs\iftrue}
\let\ifG@defaultsubs\iffalse
```

`\strip@prefix` In `\extract@font` we will need a way to recover the replacement text of a macro. This is done by the primitive `\meaning` together with the macro `\strip@prefix` (for the details see appendix D of the TeXbook, p. 382).

```
\def\strip@prefix#1>{}
```

23 Assigning math fonts to *versions*

`\addstuffto@version` (this macro no longer defined)

`\install@mathalphabet` This is just another name for `\gdef` but we can redefine it if necessary later on.

```
\let\install@mathalphabet\gdef
```

`\math@fonts`

```
\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.

```
%\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.

```
%  \begingroup
```

We set the math fonts for the *family* in question by calling `\getanddefine@fonts` in the correct environment.

```
%      \escapechar\m@ne
%     \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
```

We globally select the math fonts...

```
%     \globaldefs\@ne  \math@fonts
```

... and close the group to restore `\globaldefs` and `\escapechar`.

```
%  \endgroup
```

As long as no *size* or *version* change occurs the ⟨*math alphabet identifier*⟩ should simply switch to the installed *math group* instead of calling `\select@group` unnecessarily. So we globally redefine the first argument (the new ⟨*math alphabet identifier*⟩) to expand into a `\mathgroup` switch and then select this *alphabet*. Note that this redefinition will be overwritten by the next call to a *version* macro.
The original code for the end of `\select@group` was

```
\gdef#1{#3\mathgroup #2}#1\fi}
```

i.e. first redefining the ⟨*math alphabet identifier*⟩ and then calling the new definition to switch to the wanted ⟨*math group*⟩. Now we define the ⟨*math alphabet identifier*⟩ as a call to the `\use@mathgroup` command.

```
%  \xdef#1{\noexpand\use@mathgroup\noexpand#2%
%           {\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 identifer `#1` and the math version macro as arguments.

```
%      \expandafter\extract@alph@from@version
%         \csname mv@\math@version\expandafter\endcsname
%        \expandafter{\number\csname c@mv@\math@version\endcsname}%
%        #1%
%      \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 od `\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.

```
%\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.

```
\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`.

```
    \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`.

```
      \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`.

```
          \def\reserved@c####1{\gdef#1{##1####1##3}}}%
```

Then we execute our auxiliary macro.

```
      \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.

```
      \def\reserved@a\select@group#3##1##2\@nil{%
```

This macro can now directly rebuild the math version definition by calling `\reserved@c`:

```
         \reserved@c{%
            \getanddefine@fonts{#2}##2%
            \install@mathalphabet#3{%
               \relax\ifmmode \else \non@alpherr#3\fi
               \use@mathgroup##1{#2}}}%
```

In addtion it defines the alphabet the way it should be used from now on.

```
        \gdef#3{\relax\ifmmode \else \non@alpherr#3\fi
                \use@mathgroup##1{#2}}}%
```

Finally, we only have to call this macro `\reserved@a` on the old definitions recorded in `\reserved@b`:

```
      \expandafter\reserved@a\reserved@b\@nil
      }
```

`\math@bgroup` `\math@egroup` 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.

```
\let\math@bgroup\bgroup
\def\math@egroup#1{#1\egroup}
```

`\calculate@math@sizes` `\defaultscriptratio` `\defaultscriptscriptratio` Here is the default definition for a more elaborate interface is under testing in mthscale.sty. The default ratio for math sizes is 1 to `\defaultscriptratio` to `\defaultscriptscriptratio`. By default this is 1 to .7 to .5.

```
\def\calculate@math@sizes{%
  \@font@info{Calculating math sizes for size <\f@size>}%
  \dimen@\f@size \p@
  \@tempdimb \defaultscriptratio \dimen@
  \dimen@ \defaultscriptscriptratio \dimen@
  \expandafter\xdef\csname S@\f@size\endcsname{%
    \gdef\noexpand\tf@size{\f@size}%
    \gdef\noexpand\sf@size{\strip@pt\@tempdimb}%
```

```
    \gdef\noexpand\ssf@size{\strip@pt\dimen@}%
    \noexpand\math@fontstrue}}
\def\defaultscriptratio{.7}
\def\defaultscriptscriptratio{.5}
```

\noaccents@ If we don't have a definition for `\noaccents@` we provide a dummy.

```
\ifx\noaccents@\undefined
  \let\noaccents@\@empty
\fi
```

\showhyphens `\showhyphens` must be redefined since the version in `plain.tex` uses `\tenrm`.

```
\def\showhyphens#1{\setbox0\vbox{\parfillskip\z@skip\hsize\maxdimen
  \reset@font
  \pretolerance\m@ne\tolerance\m@ne\hbadness\z@\showboxdepth\z@\ #1}}
```

\addto@hook We need a macro to add tokens to a hook.

```
\long\def\addto@hook#1#2{#1\expandafter{\the#1#2}}
```

\@vpt
\@vipt
\@viipt
\@viiipt
\@ixpt
\@xpt
\@xipt
\@xiipt
\@xivpt
\@xviipt
\@xxpt
\@xxvpt

```
 \def\@vpt{5}
 \def\@vipt{6}
 \def\@viipt{7}
 \def\@viiipt{8}
 \def\@ixpt{9}
 \def\@xpt{10}
 \def\@xipt{10.95}
 \def\@xiipt{12}
 \def\@xivpt{14.4}
 \def\@xviipt{17.28}
 \def\@xxpt{20.74}
 \def\@xxvpt{24.88}
⟨/2ekernel⟩
```

File n
ltfsstrc.dtx

24 Introduction

This package contains the code for tracing font loading and font changes. It basically overlays some of the low-level functions of NFSS with additional code used for tracing.

The package accepts the following options:

errorshow Write all information about font changes etc. only to the transcript file unless an error happens. This means that information about font substitution will not be shown on the terminal.

warningshow Show all NFSS warnings on the terminal. This setting corresponds to the default behaviour of NFSS if the `tracefnt` package is *not* loaded!

infoshow Show all NFSS warning and all NFSS info messages (that are normally only written to the transcript file) also on the terminal. This is the default if the `tracefnt` package is loaded.

debugshow In addition to `infoshow` show also changing of math fonts as far as possible (this option can produce a large amount of output.

loading Show the name of external fonts when they are loaded. This option shows only "newly" loaded fonts not those already preloaded in the format or the class file before the `tracefnt` package became active.

pausing Turn all font warnings into errors so that LaTeX will stop.

25 A driver for this document

The next bit of code contains the documentation driver file for TeX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DocStrip program.

When this file is processed directly by LaTeX this will produce the documentation as well.

```
⟨*driver⟩
\documentclass{ltxdoc}


%\OnlyDescription  % comment out for implementation details

\begin{document}
   \DocInput{ltfsstrc.dtx}
\end{document}
⟨/driver⟩
```

26 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.

```
⟨*package⟩
%\NeedsTeXFormat{LaTeX2e}
%\ProvidesPackage{tracefnt}[??/??/?? v?.??
%                              Standard LaTeX package (font tracing)]
⟨/package⟩
```

The `debug` module makes use of commands contained in a special package file named `trace.sty`.[4]

```
⟨+debug⟩ \input trace.sty
```

27 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.

```
⟨*!trace⟩
\def\tracingfonts{%
  \@font@warning{Command \noexpand\tracingfonts
           not provided.\MessageBreak
           Use the `tracefnt' package.\MessageBreak Command found:}%
       \count@}
⟨/!trace⟩
```

The `\count@` in the line above will remove the number after `\tracingfonts`. Note that this definition will be overwritten be the next line if one of these modules are included.

```
⟨*package, trace, debug⟩
\newcount\tracingfonts
\tracingfonts=0
⟨/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).

[4] This 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.

infoshow is the default for this package here; and **debugshow**, **loading**, and **pausing** extend the amount of information even further.

```
⟨*package⟩
\DeclareOption{errorshow}{%
   \def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{warningshow}{%
   \def\@font@info#1{%
         \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{infoshow}{%
   \def\@font@info#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                     {LaTeX Font Info: \space\space\space#1}}%
    \def\@font@warning#1{%
         \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                        {LaTeX Font Warning: #1}}%
     }
\DeclareOption{loading}{%
    \tracingfonts\tw@
   }
\DeclareOption{debugshow}{%
    \ExecuteOptions{infoshow}%
    \tracingfonts\thr@@
   }
\DeclareOption{pausing}{%
    \def\@font@warning#1{%
      \GenericError
             {(Font)\@spaces\@spaces\@spaces\space\space}%
             {LaTeX Font Warning: #1}%
             {See the LaTeX Companion for details.}%
             {I'll stop for every LaTeX Font Warning because
              you requested\\the `pausing' option
              to the tracefnt package.}}%
   }
```

We make **infoshow** the default, which in turn defines **\font@warning** and **\font@info**.

```
\ExecuteOptions{infoshow}
\ProcessOptions
⟨/package⟩
```

We also need a default definition inside the kernel:

```
⟨*!package⟩
\def\@font@info#1{%
        \GenericInfo{(Font)\@spaces\@spaces\@spaces\space\space}%
                    {LaTeX Font Info: \space\space\space#1}}%
\def\@font@warning#1{%
        \GenericWarning{(Font)\@spaces\@spaces\@spaces\space\space}%
                       {LaTeX Font Warning: #1}}%
⟨/!package⟩
```

28 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 **ltfss.dtx**.

28.1 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.

```
\def\extract@font{%
   \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**).

```
   \global\expandafter\font\font@name\external@font\relax
```

When tracing we typeout the internal and external font name.

```
⟨*trace⟩
   \ifnum \tracingfonts >\@ne
   \@font@info{External font `\external@font'
             loaded as\MessageBreak \font@name}\fi
⟨/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.

```
   \font@name \relax
```

[5]This file is currently not distributed in documented form. Its code is part of ltfss.dtx.

The next two lines execute the "loading actions" for the family and then for the individual font shape.

```
    \csname \f@encoding+\f@family\endcsname
    \csname\curr@fontshape\endcsname
    \relax
      }
```

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`.

```
⟨*!package⟩
\def\get@external@font{%
```

We don't know the external font name at the beginning.

```
   \let\external@font\@empty
   \edef\font@info{\expandafter\expandafter\expandafter\string
        \csname \curr@fontshape \endcsname}%
   \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.

```
   \ifx\external@font\@empty
      \try@size@substitution
      \ifx\external@font\@empty
         \@latex@error{Font \expandafter \string\font@name\space
                   not found}\@eha
         \error@fontshape
         \get@external@font
   \fi\fi
}
⟨/!package⟩
```

`\selectfont` The macro `\selectfont` is called whenever a font change must take place.

```
\DeclareRobustCommand\selectfont
        {%
```

When `debug` is specified we actually want something like 'undebug'. 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.

```
⟨+debug⟩  \pushtracing
⟨+debug⟩  \ifnum\tracingfonts<4 \tracingoff
⟨+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`.

```
    \ifx\f@linespread\baselinestretch \else
      \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 `\twfbf`, etc. We define `\font@name` globally, as always. The reason for this is explained later on.

```
    \xdef\font@name{%
      \csname\curr@fontshape/\f@size\endcsname}%
```

We call the macro `\pickup@font` which will load the font if necessary.

```
    \pickup@font
```

Then we select the font.

```
    \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`.

```
⟨*trace⟩
    \ifnum \tracingfonts>\tw@
      \@font@info{Switching to \font@name}\fi
⟨/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.

```
    \size@update
```

A similar function is called to handle anything related to encoding updates. This one is changed from `\relax` by `\fontencoding`.

```
    \enc@update
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
⟨+debug⟩ \poptracing
    }
```

`\set@fontsize` The macro `\set@fontsize` does the actual work. First it assigns new values to `\f@size`, `\f@baselineskip` and `\f@linespread`.

```
\def\set@fontsize#1#2#3{%
    \@defaultunits\@tempdimb#2pt\relax\@nnil
    \edef\f@size{\strip@pt\@tempdimb}%
    \@defaultunits\@tempskipa#3pt\relax\@nnil
    \edef\f@baselineskip{\the\@tempskipa}%
    \edef\f@linespread{#1}%
```

For backward compatibility and for later testing within `\selectfont` the internal value of `\f@linespread` is passed back to `\baselinestretch`.

```
    \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:

```
      \def\size@update{%
```

First calculate the new `\baselineskip` and also store it in `normalbaselineskip`

```
        \baselineskip\f@baselineskip\relax
        \baselineskip\f@linespread\baselineskip
        \normalbaselineskip\baselineskip
```

then to set up a new `\strutbox`

```
        \setbox\strutbox\hbox{%
          \vrule\@height.7\baselineskip
                \@depth.3\baselineskip
                \@width\z@}%
```

We end with a bit of tracing information.

```
⟨*trace⟩
   \ifnum \tracingfonts>\tw@
      \ifx\f@linespread\@empty
        \let\reserved@a\@empty
      \else
        \def\reserved@a{\f@linespread x}%
      \fi
      \@font@info{Changing size to \f@size/\reserved@a
               \f@baselineskip}%
      \aftergroup\type@restoreinfo \fi
⟨/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.

```
        \let\size@update\relax}%
  }
```

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.

```
\let\size@update\relax
```

`\type@restoreinfo` This macro produces some info when a font size and/or baseline change will get restored.

```
⟨*trace⟩
   \def\type@restoreinfo{%
      \ifx\f@linespread\@empty
        \let\reserved@a\@empty
      \else
        \def\reserved@a{\f@linespread x}%
      \fi
      \@font@info{Restoring size to
               \f@size/\reserved@a\f@baselineskip}}
⟨/trace⟩
```

\glb@settings
\glb@currsize

The macro `\glb@settings` globally selects all math fonts for the current size if necessary.

```
\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.

```
    \expandafter\ifx\csname S@\f@size\endcsname\relax
      \calculate@math@sizes
    \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.

```
    \csname S@\f@size\endcsname
    \ifmath@fonts
⟨*trace⟩
      \ifnum \tracingfonts>\tw@
        \@font@info{Setting up math fonts for
                    \f@size/\f@baselineskip}\fi
⟨/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.

```
      \begingroup
        \escapechar\m@ne
        \csname mv@\math@version \endcsname
```

Then we set `\globaldefs` to 1 so that all following changes are done globally. The math font assignments recorded in `\math@fonts` are executed and `\glb@currsize` is set equal to `\f@size`. This signals that the fonts for math in this size are set up.

```
        \globaldefs\@ne
        \math@fonts
        \let \glb@currsize \f@size
      \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.

```
        \the\every@math@size
```

Otherwise we announce that the math fonts are not set up for this size.

```
⟨*trace⟩
       \else
         \ifnum \tracingfonts>\tw@
           \@font@info{No math setup for
                         \f@size/\f@baselineskip}\fi
⟨/trace⟩
     \fi
}
```

\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).

```
⟨*!package⟩
\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.

```
\newtoks\every@math@size
\every@math@size={}
⟨/!package⟩
```

28.2 Math fonts setup

28.2.1 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 \mbox{ \small for all $b$ and $c\in Z$}$
```

Here the inner formulae `b` and `c\in Z` are typeset in `\small` but we have to return to `\normalsize` before we reach the closing `$` of the outer formula.

This is handled in the following way:

1. At any point in the document the global variable `\gbl@currsize` contains the point size for which the math fonts currently are set up.

2. Whenever we start a formula we compare its value with the local variable `\f@size` that describes the current text font size.

3. If both are the same we assume that we can use the current math font setup without adjustment.

4. If they differ we call `\gbl@settings` which changes the math font setup and updates `\gbl@currsize`.
   (a) If we are recursively inside another formula (`\if@inmath`) we ensure that `\gbl@settings` is executed again in the outer formula, so that the old setup is automatically restored.
   (b) Otherwise, we set the switch `@inmath` locally to `true` so that all nested formulae will be able to detect that they are nested in some outer formula.

The above algorithm has the following features:

- For sizes which are not containing any formula no math setup is done. Compared to the original algorithm of NFSS this results in the following savings:
  - No unnecessary loading of math fonts for sizes that are not used to typeset any math formulae (explicit or implicit ones).
  - No time overhead due to unnecessary changes of the math font setup on entrance and exit of the text font size.
- Math font setup changes for top-level formulae will survive (there is no restoration after the formula) thus any following formula in the same size will be directly typesetable. 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 ($2 \times \langle$non-math levels$\rangle$ per inner formula).

28.2.2 Code for math font size setting

`\check@mathfonts` In the `\check@mathfonts` macros we implement the steps 2 to 4.

```
\def\check@mathfonts{%
  \ifx \glb@currsize \f@size
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: no change \f@size\space
         curr/global (\curr@math@size/\glb@currsize)}\fi
⟨/trace⟩
  \else
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: setting up \f@size\space
         curr/global (\curr@math@size/\glb@currsize)}\fi
⟨/trace⟩
```

```
    \glb@settings
    \if@inmath
      \aftergroup\restglb@settings
⟨*trace⟩
    \else
      \ifnum \tracingfonts>\thr@@
       \@font@info{*** MATH: no resetting (not in
                   nested math)}\fi
⟨/trace⟩
    \fi
  \fi
  \let\curr@math@size\f@size
  \@inmathtrue
}
```

\restglb@settings This macro will be executed the first time after the current formula.

```
\def\restglb@settings{%
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: restoring}\fi
⟨/trace⟩
      \begingroup
        \let\f@size\curr@math@size
        \ifx\glb@currsize \f@size
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: ... already okay (\f@size)}\fi
⟨/trace⟩
        \else
⟨*trace⟩
    \ifnum \tracingfonts>\thr@@
        \@font@info{*** MATH: ... to \f@size}\fi
⟨/trace⟩
          \glb@settings
        \fi
      \endgroup
}
```

28.2.3 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.

```
\def\use@mathgroup#1#2{\relax\ifmmode
```

```
⟨*trace⟩
  \ifnum \tracingfonts>\tw@
```

```
    \count@#2\relax
    \@font@info{Using \noexpand\mathgroup
               (\the\count@) #2}\fi
⟨/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.

```
     \math@bgroup
         \expandafter\ifx\csname M@\f@encoding\endcsname#1\else
         #1\fi
         \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.

```
     \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 $\mathcal{A}\mathcal{M}\mathcal{S}$-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`.

```
⟨*trace⟩
  \ifx\math@bgroup\begingroup
    \def\math@egroup#1{#1\endgroup
      \ifnum \tracingfonts>\tw@
     \@font@info{Restoring \noexpand\mathgroup
        (\ifnum\mathgroup=\m@ne default\else \the\mathgroup \fi)%
        }\fi}
  \fi
⟨/trace⟩
```

`\getanddefine@fonts` `\getanddefine@fonts` has two arguments: the ⟨*math group number*⟩ and the *family/series/shape* name as a control sequence.

```
\def\getanddefine@fonts#1#2{%
```

First we turn of tracing when `\tracingfonts` is less than 4.

```
⟨+debug⟩  \pushtracing
⟨+debug⟩  \ifnum\tracingfonts<4 \tracingoff
```

```
⟨+debug⟩  \else \tracingon\getanddefine@fonts \fi
```

```
⟨*trace⟩
  \ifnum \tracingfonts>\tw@
  \count@#1\relax
    \@font@info{\noexpand\mathgroup (\the\count@) #1 :=\MessageBreak
                \string#2 \tf@size/\sf@size/\ssf@size}\fi
⟨/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`.

```
  \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`.

```
  \pickup@font \let\textfont@name\font@name
```

Same game for `\scriptfont` and `\scriptscriptfont`:

```
  \xdef\font@name{\csname \string#2/\sf@size\endcsname}%
  \pickup@font \let\scriptfont@name\font@name
  \xdef\font@name{\csname \string#2/\ssf@size\endcsname}%
  \pickup@font
```

Then we append the new `\textfont`... assignments to the `\math@fonts`.

```
  \edef\math@fonts{\math@fonts
                \textfont#1\textfont@name
                \scriptfont#1\scriptfont@name
                \scriptscriptfont#1\font@name}%
```

Just before ending this macro we have to pop the tracing stack if it was pushed before.

```
⟨+debug⟩ \poptracing
    }
```

29 Scaled font extraction

`\ifnot@nil` We begin with a simple auxiliary macro. It checks whether its argument is the token `\@nil`. If so, it expands to `\@gobble` which discards the following argument, otherwise it expands to `\@firstofone` which reproduces it argument.

```
⟨*!package⟩
\def\ifnot@nil#1{\def\reserved@a{#1}%
  \ifx\reserved@a\@nnil \expandafter\@gobble
  \else \expandafter\@firstofone\fi}
```

[6] One might ask why this expansion does not generate a macro name that starts with an additional `\` character. The solution is that `\escapechar` is set to -1 before `\getanddefine@fonts` is called.

`\remove@to@nnil` `\remove@angles` `\remove@star` `\gobble@font@spec`

Four 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`. Finally, `\gobble@font@spec` gobbles up everything up to, and including the three tokens `\@nil<\@nnil`.

```
\def\remove@to@nnil#1\@nnil{}
\def\remove@angles#1>{\set@simple@size@args}
\def\remove@star#1*{#1}
\def\gobble@font@spec#1\@nil<\@nnil{}
```

`\extract@sizefn`

This macro takes a size specification and parses it into size function and the optional and mandatory arguments.

```
\def\extract@sizefn#1*#2\@nil{%
  \if>#2>\set@size@funct@args#1\@nil
         \let\sizefn@info\@empty
  \else\expandafter\set@size@funct@args\remove@star#2\@nil
        \def\sizefn@info{#1}\fi
  }
```

`\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 flavour. We will define a macro `\extract@fontinfo` to find the external font name ('cmss12') for us:

```
\def\extract@fontinfo#1<12*#2>#3<#4\@nnil{%
    \set@simple@size@args#3<#4\@nnil
    \execute@size@function{#2}}
```

so that when it gets called via

```
\extract@fontinfo<10*>cmss10<12*>cmss12<17*>cmss17\@nnil
```

`#1` will contain all characters before `<12*>`, `#2` will be empty, `#3` will be exactly `cmss12`, and `#3` will be `17>cmss17`. The expansion is therefore

```
\set@simple@size@args cmss12<17*>cmss17\@nnil
\execute@size@function{}
```

This means: the default (empty) size function will be executed, with its optional argument 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`.

```
%% this could be replaced by \try@size@range making the subst slower!
\def\try@simple@size{%
```

`\reserved@a` is made an abbreviation for the head of the definition of the macro `\extract@fontinfo`.

```
    \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.

```
    \expandafter\reserved@a\expandafter<\f@size>##2<##3\@nnil{%
          \ifnot@nil{##2}%
            {\set@simple@size@args##2<##3\@nnil
```

```
            \execute@size@function\sizefn@info
          }}%
```

Now we call `\extract@fontinfo`. Note the `<\@nil` tokens at the end.

```
    \expandafter\expandafter
    \expandafter\extract@fontinfo\expandafter\font@info
    \expandafter<\f@size>\@nil<\@nnil
}
```

`\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,

```
\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.

```
          \if<#1<%
            \expandafter\remove@angles
          \else
            \extract@sizefn#1*\@nil
            \expandafter\remove@to@nnil
          \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 macors for extraction of size range specification.

`\extract@rangefontinfo`

`\extract@rangefontinfo` goes through a font shape definition in the input until it recognizes the tokens `<\@nil->`. It looks for font ranges with font size functions. 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.

```
\def\extract@rangefontinfo#1<#2>{%
       \is@range#2->\@nil#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 `\@nil` in the call to `\is@range` appear at the beginning of the first argument to `\extract@rangefontinfo` and are hence ignored.

```
\def\is@range#1-#2\@nil{%
  \if>#2\expandafter\check@single\else
     \expandafter\check@range\fi}
```

`\lower@bound`
`\upper@bound`

We use two dimen registers `\lower@bound` and `\upper@bound` to store the lower and upper endpoints of the range we found.

```
\newdimen\lower@bound
\newdimen\upper@bound
```

`\check@range`

`\check@range` takes lower bound as parameter `#1`, upper bound as `#2`, size function as `#3` and the size function's arguments as `#4`. If `#3` is the special token `\@nil` `\font@info` is exhausted and we can stop searching.

```
\def\check@range#1-#2>#3<#4\@nnil{%
 \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.

```
     \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.

```
     \upper@bound0#2\p@
     \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.

```
     \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.

```
        \lower@bound0#1\p@
        \ifdim \f@size \p@<\lower@bound
         \else
```

If both tests are passed we can try executing the size function.

```
           \set@simple@size@args#3<#4\@nnil
           \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.

```
          \ifx\external@font\@empty
          \else
            \let\reserved@f\@empty
```

```
        \fi
      \fi
    \fi
    \reserved@f}}
```

\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).

```
\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.

```
    \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 no good and we have to recurse.

```
    \ifdim \f@size \p@=#1\p@
```

Otherwise if this test is passed we can try executing the size function.

```
      \set@simple@size@args#2<#3\@nnil
      \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.

```
      \ifx\external@font\@empty
      \else
        \let\reserved@f\@empty
      \fi
    \fi
    \reserved@f}
```

\set@size@funct@args
\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`.

```
\def\set@size@funct@args{\@ifnextchar[%
  \set@size@funct@args@{\set@size@funct@args@[]}}
\def\set@size@funct@args@[#1]#2\@nil{%
  \def\mandatory@arg{#2}%
  \def\optional@arg{#1}}
```

\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 `<...>`.

```
\def\DeclareSizeFunction#1#2{\@namedef{s@fct@#1}{#2}}
⟨/!package⟩
```

\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`).

```
\def\execute@size@function#1{%
⟨*trace⟩
        \@ifundefined{s@fct@#1}%
          {\errmessage{Undefined font size function #1}%
           \s@fct@}%
          {\csname s@fct@#1\endcsname}%
⟨/trace⟩
⟨-trace⟩      \csname s@fct@#1\endcsname
}
```

`\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.

```
⟨*!package⟩
\def\try@size@range{%
    \expandafter\extract@rangefontinfo\font@info <-*>\@nil<\@nnil
}
```

`\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.

```
\def\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.

```
  \@tempdimb \maxdimen
  \let \best@size \@empty
```

Now we loop over the specification

```
  \expandafter \try@simples \font@info <\number\@M>\@nil<\@nnil
}
```

`\font@submax` `\fontsubfuzz` The macro `\font@submax` records the maximal deviation from the desired size encountered so far. Its value is used in a warning message at `\end{coument}`. The macro `\fontsubfuzz` contains the amount that will not cause terminal warnings (warnings still go into the transcript file).

```
\def\font@submax{0pt}
\def\fontsubfuzz{.4pt}
⟨/!package⟩
⟨+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.

```
⟨*!package⟩
\def\try@simples#1<#2>{%
  \tryif@simple#2->\tryif@simple}
```

\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.

```
\def\tryif@simple#1-#2\tryif@simple{%
```

Most common case for `\reserved@f` first:

```
  \let \reserved@f \try@simples
  \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.

```
    \dimen@ #1\p@
    \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.

```
      \ifdim \f@size\p@<\dimen@
        \@tempdimc \dimen@
        \advance\@tempdimc -\f@size\p@
      \else
        \@tempdimc \f@size\p@
        \advance\@tempdimc -\dimen@
      \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`.

```
      \ifdim \@tempdimc<\@tempdimb
        \@tempdimb \@tempdimc
        \def \best@size{#1}%
      \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`.

```
    \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.

```
%%\subst@size              %% coded inline
%%\def\subst@size{%
```

```
  \ifx \external@font\@empty
    \ifx \best@size\@empty
    \else
      \ifdim \@tempdimb>\font@submax \relax
        \xdef \font@submax {\the\@tempdimb}%
      \fi
      \let \f@user@size \f@size
      \let \f@size \best@size
      \ifdim \@tempdimb>\fontsubfuzz\relax
        \@font@warning{Font shape `\curr@fontshape'
                      in size <\f@user@size> not
                      available\MessageBreak
                      size <\f@size> substituted}%
      \fi
      \try@simple@size
      \do@subst@correction
    \fi
 \fi
%%}
```

This brings us back into the main part of `\tryif@simple`. Finally we get rid of any rubbish left over on the input stack.

Correct to terminate? could be simpler.

```
      \let \reserved@f \gobble@font@spec
    \fi
  \fi
```

If it's a range iterate also.

```
  \reserved@f}
```

29.1 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, optinal 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.

```
\DeclareSizeFunction{}{\empty@sfcnt\@font@warning}
\DeclareSizeFunction{s}{\empty@sfcnt\@font@info}
```

```
\def\empty@sfcnt#1{%
      \@tempdimb \f@size\p@
      \ifx\optional@arg\@empty
      \else
        \@tempdimb \optional@arg\@tempdimb
        #1{Font shape `\curr@fontshape'
            will be\MessageBreak
            scaled to size \the\@tempdimb}%
      \fi
```

```
    \edef\external@font{\mandatory@arg\space at\the\@tempdimb}}
```

\s@fct@gen \s@fct@sgen This size function generates the external name from the mandatory argument and the requested user size, and thus can be used for external names where the size is encoded in the font name. The optinal 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.

```
\DeclareSizeFunction{gen}{\gen@sfcnt\@font@warning}
\DeclareSizeFunction{sgen}{\gen@sfcnt\@font@info}

\def\gen@sfcnt{%
    \edef\mandatory@arg{\mandatory@arg\f@size}%
    \empty@sfcnt}
```

\s@fct@sub This size function handles font substitution. The mandatory argument is a family/series/shape combination, the optional argument (if present) is ignored. The font encoding scheme cannot be changed. Therefore, the first thing we do is to prepend the encoding scheme.

```
\DeclareSizeFunction{sub}{\sub@sfcnt\@font@warning}
\DeclareSizeFunction{ssub}{\sub@sfcnt\@font@info}
\def\sub@sfcnt#1{%
   \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.

```
   \begingroup
    \expandafter\split@name\mandatory@arg/\@nil
    \try@load@fontshape
   \endgroup
```

Then we record the current `\f@size` since it may get clobbered.

```
   \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`.

```
   \expandafter
   \ifx\csname\mandatory@arg\endcsname\relax
     \errmessage{No declaration for shape \mandatory@arg}%
     \error@fontshape
   \else
```

Otherwise we warn the user about the substitution taking place.

```
     #1{Font shape `\curr@fontshape' in size <\f@size> not
                    available\MessageBreak
                    Font shape `\mandatory@arg' tried instead}%
     \expandafter\split@name\mandatory@arg/\@nil
   \fi
```

Then we restart the font specification scan by calling `\get@external@font`.

```
   \edef\f@size{\f@user@size}%
   \get@external@font
```

Finally `\do@subst@correction` is called to get the font name right.

```
    \do@subst@correction
}
```

`\s@fct@subf` The `subf` size function allows substitution of another font. The mandatory argument is the external name of the font to be substituted, the optional argument a size scaling factor like in the default size function. The main difference to the default size function is the warning message.

```
\DeclareSizeFunction{subf}{\subf@sfcnt\@font@warning}
\DeclareSizeFunction{ssubf}{\subf@sfcnt\@font@info}
\def\subf@sfcnt#1{%
      #1{Font shape `\curr@fontshape'
                    in size \f@size\space
                    not available\MessageBreak
                    external font `\mandatory@arg'
                    used}%
      \empty@sfcnt#1%
      }
```

`\s@fct@fixed` The `fixed` size function is for using a font at a different size than requested. A warning message is printed, and the external font to be used is taken from the mandatory argument. If an optional argument is present it is used as the 'at' size for the font. Otherwise the font is loaded at its design size.

```
\DeclareSizeFunction{fixed}{\fixed@sfcnt\@font@warning}
\DeclareSizeFunction{sfixed}{\fixed@sfcnt\@font@info}
\def\fixed@sfcnt#1{%
  \ifx\optional@arg\@empty
    \let\external@font\mandatory@arg
  \else
    \edef\external@font{\mandatory@arg\space at\optional@arg pt}%
  \fi
  #1{External font `\external@font' loaded for size\MessageBreak
    <\f@size>}%
}
⟨/!package⟩
```

File o
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.

30 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]

`\new@fontshape`
`\subst@fontshape`
`\extra@def`
`\default@mextra`
`\define@mathalphabet`
`\define@mathgroup`

These macros are the interfaces in NFSS1 which shouldn't be used any longer. We all define them to call the macro `\scan@fontshape` 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.

```
⟨*autoload⟩
\def\new@fontshape{\scan@fontshape\new@fontshape}
\def\subst@fontshape{\scan@fontshape\subst@fontshape}
\def\extra@def{\scan@fontshape\extra@def}
\def\default@mextra{\scan@fontshape\default@mextra}
\def\define@mathalphabet{\scan@fontshape\define@mathalphabet}
\def\define@mathgroup{\scan@fontshape\define@mathgroup}
```

`\scan@fontshape`

Here is the kernel definition for `\scan@fontshape` which loads the actual definitions from the file `nfsscmp.def`.

```
\def\scan@fontshape{\input{nfsscmp.def}}
```

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.

```
\begingroup
\nfss@catcodes
\catcode`\ =10\relax
⟨/autoload⟩
⟨*compat⟩
```

`\new@fontshape` The interface is now `\DeclareFontShape`.

```
\gdef\new@fontshape#1#2#3#4{%
     \warn@rel@i\new@fontshape\DeclareFontShape
     \expandafter\scan@fontshape\@gobble#4<\@nil><<%
     \DeclareFontShape U{#1}{#2}{#3}\reserved@f}
```

`\warn@rel@i` The warning message used above.

```
\gdef\warn@rel@i#1#2{%
 \@font@warning{***  NFSS release 1 command
                \noexpand#1found\MessageBreak
   ***  Update by using release 2 command
        \string#2.\MessageBreak
   ***  Recovery is probably possible}%
}
```

`\scan@fontshape` `\scan@@fontshape` This will scan the old font shape definition syntax.

```
\gdef\scan@fontshape{%
  \let\reserved@f\@empty
  \let\reserved@e\@empty %        holds last info
  \scan@@fontshape
}
\gdef\scan@@fontshape#1>#2#3<{%
  \ifx\@nil#1%
    \edef\reserved@f{\reserved@f\reserved@e}%
  \else
    \def\reserved@b{#1}%       nick names
    \def\reserved@c{#3}%
    \in@{ at}{#3}%
    \ifin@
      \in@{pt}{#3}%  not a proof but a good chance
      \ifin@
```

We grap also everything after pt and discard it if people have forgotten to place a percent sign there.

```
        \def\reserved@a##1 at##2pt##3\@nil{%
           \def\reserved@b{##2}%
           \def\reserved@c{##1}%
           }%
        \reserved@a#3\@nil
      \fi
    \fi
    \ifnum 0<0#2
      \edef\reserved@d{subf*\reserved@c}%
      \ifcase #2\or
```

```
      \or
      \else
        \errmessage{*** What's this? NFSS release 0? ***}%
      \fi
    \else
      \edef\reserved@d{#2\reserved@c}%
    \fi
    \ifx\reserved@d\reserved@e
      \edef\reserved@f{\reserved@f<\reserved@b>}%
    \else
      \edef\reserved@f{\reserved@f\reserved@e<\reserved@b>}%add old info
      \let\reserved@e\reserved@d
    \fi
    \expandafter\scan@@fontshape
  \fi
}
```

\subst@fontshape This is now also handled by the extend syntax of `\DeclareFontShape`.

```
\gdef\subst@fontshape#1#2#3#4#5#6{%
     \warn@rel@i\subst@fontshape\DeclareFontShape
     \DeclareFontShape{U}{#1}{#2}{#3}{<->sub*#4/#5/#6}{}}
```

\extra@def This was replaced by `\DeclareFontFamily`.

```
\gdef\extra@def#1#2#3{%
     \warn@rel@i\extra@def\DeclareFontFamily
     \DeclareFontFamily{U}{#1}{}%
}
```

\default@mextra The new name is `\DeclareFontEncodingDefaults` but in this case we don't feel comfortable with this either.

```
\gdef\default@mextra{%
  \warn@rel@i\default@mextra\DeclareFontEncodingDefaults
```

We pick up the argument to `\default@mextra` implicitly as the second argument of `\DeclareFontEncodingDefaults`.

```
  \DeclareFontEncodingDefaults\relax
}
```

\preload@sizes The new interface is `\DeclarePreloadSizes`.

```
\gdef\preload@sizes{%
     \warn@rel@i\preload@sizes\DeclarePreloadSizes
     \DeclarePreloadSizes U%
}
```

\err@rel@i This macro is used in cases where emulation with NFSS2 features is not really possible.

```
\gdef\err@rel@i#1#2{%
  \@latex@error{***  NFSS release 1 command \noexpand#1found%
           ^^J***  Recovery not possible. Use \string#2}%
```

```
      {The new release of NFSS doesn't support the
       \noexpand#1command^^Jany longer.
       Please upgrade your file to the syntax of NFSS
       release 2^^Jusing the \noexpand#2command.}%
```

Let's die.

```
  \batchmode\input.\relax
}
```

\define@mathalphabet

This is a case where dying is best.

```
\gdef\define@mathalphabet{%
      \err@rel@i\define@mathalphabet\DeclareMathAlphabet
}
```

\define@mathgroup

And here is another one

```
\gdef\define@mathgroup{%
      \err@rel@i\define@mathgroup\DeclareSymbolFont
}
⟨/compat⟩
```

That finishes the definitions for the old interfaces — but first we better finish the group.

```
⟨*autoload⟩
\endgroup
⟨/autoload⟩
```

File p

ltfssdcl.dtx

This file contains the main implementation of the font selection scheme commands. See other parts of the LaTeX distribution, or *The LaTeX 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.

31 Interface Commands

`\if@tempswa` We define here a new `\if...` macro. It is also used (and defined) in `latex.tex` but this file isn't loaded when initex processes these definitions.

```
⟨*!2ekernel⟩
\newif\if@tempswa
⟨/!2ekernel⟩
```

`\in@` `\ifin@` `\@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.

```
\def\in@#1#2{%
 \def\in@@##1#1##2##3\in@@{%
  \ifx\in@##2\in@false\else\in@true\fi}%
 \in@@#2#1\in@\in@@}
\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`⟨*version*$_1$⟩`\version@elt`⟨*version*$_2$⟩...
  `\version@elt`⟨*version*$_n$⟩

- 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

  `{{`⟨*family*⟩`}{`⟨*series*⟩`}{`⟨*shape*⟩`}}`

`\version@list` We initialize the version list to be empty.

```
\let\version@list=\@empty
```

`\new@mathversion` The macro `\new@mathversion` is called with the version control sequence as its argument.

```
%\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.

```
%  \expandafter\in@\expandafter#1\expandafter{\version@list}%
%  \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.

```
%    \@latex@error{Math version
%                 `\expandafter\@gobblefour\string#1'
%                 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`.

```
%  \else
%     \global\expandafter\newcount\csname c@\expandafter
%                                 \@gobble\string#1\endcsname
%     \global\csname c@\expandafter
%                                 \@gobble\string#1\endcsname\@ne
%     \def\version@elt{\noexpand\version@elt\noexpand}%
%     \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.

```
%     \def\reserved@c{\noexpand\reserved@c\noexpand}%
%     \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).

```
%     \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.

```
%         \edef##1{\expandafter\remove@nil##1%
%                  \reserved@c
%                  #1%
%                  \reserved@e
%                  \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.

```
%     \alpha@list
%  \fi}
```

\newmathalphabet
\newmathalphabet@@
\newmathalphabet@@@

`\newmathalphabet` is the old form.

```
⟨*compat⟩
\let\if@no@font@opt\iftrue
\def\@no@font@optfalse{\let\if@no@font@opt\iffalse}
\def\newmathalphabet{%
  \if@no@font@opt
    \@latex@error{*** NFSS release 1 command
                  \noexpand\newmathalphabet found%
     ^^J \space*** Automatic recovery not possible.%
     ^^J \space*** TYPE H for Help%
             }%
      {Please look at the file usrguide.tex for hints on
       how to resolve this problem.}%
  \else
     \warn@rel@i\newmathalphabet\DeclareMathAlphabet
  \fi
  \@ifstar\newmathalphabet@@@
          \newmathalphabet@@}
\def\newmathalphabet@@#1{\DeclareMathAlphabet#1{U}{}{}{}}
```

```
\def\newmathalphabet@@@#1#2#3#4{%
      \DeclareMathAlphabet{#1}{U}{#2}{#3}{#4}}
⟨/compat⟩
```

`\alpha@list` As we explained above every entry in `\alpha@list` has the form
`\group@elt`
⟨*alphabet identifier*⟩⟨*internal group number*⟩⟨*default font assignments*⟩...
We initialize it to `\@empty`.

```
\let\alpha@list\@empty
```

`\addtoversion` `\addtoversion` is the old form.

```
⟨*compat⟩
\def\addtoversion#1#2{%
  \warn@rel@i\addtoversion\SetMathAlphabet
  \SetMathAlphabet#2{#1}{U}}
⟨/compat⟩
```

`new stuff`

```
% start alloc at 0
\count18=-1
```

another begingroup bug:

```
\def\stepcounter #1{\global\advance\csname c@#1\endcsname\@ne
  \begingroup\let\@elt\@stpelt \csname cl@#1\endcsname\endgroup }
```

```
\newif\ifrestore@version \restore@versiontrue
```

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.

```
\def\select@group#1#2#3#4{%
\ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
{%
\ifmmode
 \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
    \begingroup
      \escapechar\m@ne
      \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
      \globaldefs\@ne  \math@fonts
    \endgroup
    \ifrestore@version
       \global\restore@versionfalse
       \xdef\restore@mathversion
            {\expandafter\noexpand\csname mv@\math@version\endcsname
             \global\csname c@mv@\math@version\endcsname
             \number\csname c@mv@\math@version\endcsname\relax}%
       \aftergroup\dorestore@version
    \fi
    \xdef#1{\noexpand\use@mathgroup\noexpand#2%
```

```
           {\number\csname c@mv@\math@version\endcsname}}%
    \stepcounter{mv@\math@version}%
  \else
    \let#1\relax
    \@latex@error{Too many math alphabets used in
                  version \math@version}%
       \@eha
  \fi
\else \expandafter\non@alpherr\fi
#1{#4}%
}%
}
\def\non@alpherr#1{\@latex@error{%
```

Since the argument is the internal alphabet name, we have to get rid of the `@` in its name. The trick here is to use `\@firstoftwo` which is a LaTeX macro which discards the second of two arguments.

```
    \expandafter\@firstoftwo\string#1 allowed only in math mode}\@ehd}
```

```
\def\dorestore@version
 {\ifmmode
    \aftergroup\dorestore@version
  \else
    \global\restore@versiontrue
    \begingroup
      \let\getanddefine@fonts\@gobbletwo
      \restore@mathversion
    \endgroup
  \fi}%
```

We surround `\select@group` with braces so that functions using it can be used directly after `_` or `^`.

```
\def\document@select@group#1#2#3#4{%
\ifx\math@bgroup\bgroup\else\relax\expandafter\@firstofone\fi
{%
\ifmmode
  \ifnum\csname c@mv@\math@version\endcsname<\sixt@@n
    \begingroup
      \escapechar\m@ne
      \getanddefine@fonts{\csname c@mv@\math@version\endcsname}#3%
      \globaldefs\@ne  \math@fonts
    \endgroup
    \expandafter\extract@alph@from@version
        \csname mv@\math@version\expandafter\endcsname
        \expandafter{\number\csname
                        c@mv@\math@version\endcsname}%
         #1%
    \stepcounter{mv@\math@version}%
  \else
    \let#1\relax
```

```
     \@latex@error{Too many math alphabets used
                   in version \math@version}%
        \@eha
  \fi
 \else \expandafter\non@alpherr\fi
 #1{#4}%
 }%
}
```

```
\def\process@table{%
    \def\cdp@elt##1##2##3##4{%
        \@font@info{Checking defaults for
                  ##1/##2/##3/##4}%
        \expandafter
        \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.

```
          \begingroup
           \def\f@encoding{##1}\def\f@family{##2}%
           \try@load@fontshape
          \endgroup
        \fi
        \expandafter
        \ifx\csname##1/##2/##3/##4\endcsname\relax
             \@latex@error{This NFSS system isn't set up properly}%
                       {For encoding scheme ##1 the defaults
                        ##2/##3/##4 do not form a valid font shape}%
        \else
             \@font@info{... okay}%
        \fi}%
    \cdp@list
```

Now we make sure that `\error@fontshape` is okay.

```
    \begingroup
       \escapechar\m@ne
       \error@fontshape
       \expandafter\ifx\csname \curr@fontshape\endcsname\relax
          \begingroup
            \try@load@fontshape
           \endgroup
       \fi
       \expandafter\ifx\csname \curr@fontshape\endcsname\relax
         \@latex@error{This NFSS system isn't set up properly}%
            {The system maintainer forgot to specify a suitable
             substitution
             font shape using the \noexpand\DeclareErrorFont
             command}%
       \fi
```

```
    \endgroup
```

Set `\select@group` to its meaning used within the document body.

```
    \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`.

```
    \fontencoding{\encodingdefault}%
    \fontfamily{\familydefault}%
    \fontseries{\seriesdefault}%
    \fontshape{\shapedefault}%
```

kill all macros not longer needed. we need to add many more!!!!!!

```
\everyjob{}%
}

\@onlypreamble\@xDeclareMathDelimiter
\@onlypreamble\@DeclareMathDelimiter
\@onlypreamble\DeclareFontEncoding
\@onlypreamble\DeclareFontEncoding@
\@onlypreamble\DeclareFontEncodingDefaults
\@onlypreamble\DeclareFontSubstitution
\@onlypreamble\DeclareErrorFont
\@onlypreamble\DeclareMathAccent
\@onlypreamble\DeclareMathAlphabet
\@onlypreamble\DeclareMathDelimiter
\@onlypreamble\DeclareMathRadical
\@onlypreamble\DeclareMathSizes
\@onlypreamble\DeclareMathSymbol
\@onlypreamble\DeclareMathVersion
\@onlypreamble\DeclarePreloadSizes
\@onlypreamble\DeclareSizeFunction
\@onlypreamble\DeclareSymbolFont
\@onlypreamble\DeclareSymbolFontAlphabet
\@onlypreamble\SetMathAlphabet
\@onlypreamble\SetMathAlphabet@
\@onlypreamble\SetSymbolFont
\@onlypreamble\SetSymbolFont@
\@onlypreamble\alpha@elt
\@onlypreamble\alpha@list
\@onlypreamble\cdp@elt
\@onlypreamble\cdp@list
\@onlypreamble\dorestore@version
\@onlypreamble\get@cdp
\@onlypreamble\group@elt
\@onlypreamble\group@list
\@onlypreamble\ifrestore@version
\@onlypreamble\mathchar@type
\@onlypreamble\new@mathalphabet
\@onlypreamble\new@mathgroup
\@onlypreamble\new@mathversion
```

```
\@onlypreamble\new@symbolfont
\@onlypreamble\process@table
\@onlypreamble\restore@mathversion
\@onlypreamble\restore@versionfalse
\@onlypreamble\restore@versiontrue
\@onlypreamble\set@@mathdelimiter
\@onlypreamble\set@mathaccent
\@onlypreamble\set@mathchar
\@onlypreamble\set@mathchar
\@onlypreamble\set@mathdelimiter
%\@onlypreamble\set@mathradical
\@onlypreamble\set@mathsymbol
\@onlypreamble\set@mathsymbol
\@onlypreamble\version@elt
\@onlypreamble\version@list
⟨*compat⟩
\@onlypreamble\addtoversion
\@onlypreamble\default@mextra
\@onlypreamble\define@mathalphabet
\@onlypreamble\define@mathgroup
\@onlypreamble\extra@def
\@onlypreamble\new@fontshape
\@onlypreamble\newmathalphabet
\@onlypreamble\newmathalphabet@@
\@onlypreamble\newmathalphabet@@@
\@onlypreamble\preload@sizes
\@onlypreamble\scan@fontshape
\@onlypreamble\scan@@fontshape
\@onlypreamble\subst@fontshape
\@onlypreamble\warn@rel@i
\@onlypreamble\err@rel@i
⟨/compat⟩
```

```
\def\DeclareMathVersion#1{%
  \expandafter\new@mathversion\csname mv@#1\endcsname}
```

```
\def\new@mathversion#1{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \@font@info{Redeclaring math version
                `\expandafter\@gobblefour\string#1'}%
  \else
    \global\expandafter\newcount\csname c@\expandafter
                                 \@gobble\string#1\endcsname
    \def\version@elt{\noexpand\version@elt\noexpand}%
    \edef\version@list{\version@list\version@elt#1}%
  \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.

```
  \toks@{}%
```

```
  \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.

```
  \def\group@elt##1##2{%
       \advance\count@\@ne
       \addto@hook\toks@{\getanddefine@fonts##1##2}%
       }%
  \group@list
```

We set the counter for this math version to the number of math groups found in `\group@list`.

```
  \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 distinugish the case that an alphabet by default should produce an error in new versions.

```
  \def\alpha@elt##1##2##3{%
       \ifx##2\no@alphabet@error
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\no@alphabet@error##1}}%
       \else
         \toks@\expandafter{\the\toks@\install@mathalphabet##1%
             {\select@group##1##2##3}}%
       \fi
          }%
  \alpha@list
```

Finally we define the math version to expand to the contents of `\toks@`.

```
  \xdef#1{\the\toks@}%
}
\def\DeclareSymbolFont#1#2#3#4#5{%
 \@tempswafalse
 \edef\reserved@b{#2}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
   \@ifundefined{sym#1}{%
      \expandafter\new@mathgroup\csname sym#1\endcsname
      \expandafter\new@symbolfont\csname sym#1\endcsname
                        {#2}{#3}{#4}{#5}}%
     {%
      \@font@info{Redeclaring symbol font `#1'}%
      \def\group@elt##1##2{%
          \expandafter\ifx\csname sym#1\endcsname##1%
               \expandafter\addto@hook\expandafter
                  \toks@\expandafter{\expandafter\group@elt
                  \expandafter##1\csname#2/#3/#4/#5\endcsname}%
             \else
               \addto@hook\toks@{\group@elt##1##2}%
```

```
            \fi
         }%
      \toks@{}%
      \group@list
      \def\version@elt##1{%
         \expandafter
         \SetSymbolFont@\expandafter##1\csname#2/#3/#4/#5\expandafter
             \endcsname \csname sym#1\endcsname
         }%
      \version@list
     }%
  \else
    \@latex@error{Encoding scheme  `#2' unknown}\@eha
  \fi
  }
\let\group@list\@empty
```

```
\def\new@symbolfont#1#2#3#4#5{%
    \toks@\expandafter{\group@list}%
    \edef\group@list{\the\toks@\noexpand\group@elt\noexpand#1%
                    \expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
    \def\version@elt##1{\toks@\expandafter{##1}%
                  \edef##1{\the\toks@\noexpand\getanddefine@fonts
                  #1\expandafter\noexpand\csname#2/#3/#4/#5\endcsname}%
                 \global\advance\csname c@\expandafter
                                \@gobble\string##1\endcsname\@ne
                }%
    \version@list
}
```

```
\def\SetSymbolFont#1#2#3#4#5#6{%
 \@tempswafalse
 \edef\reserved@b{#3}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
  \expandafter\SetSymbolFont@
    \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
    \endcsname \csname sym#1\endcsname
 \else
  \@latex@error{Encoding scheme  `#3' unknown}\@eha
 \fi
}
```

```
\def\SetSymbolFont@#1#2#3{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \expandafter\in@\expandafter#3\expandafter{\group@list}%
    \ifin@
      \begingroup
        \expandafter\get@cdp\string#2\@nil\reserved@a
```

```
        \toks@{}%
        \def\install@mathalphabet##1##2{%
             \addto@hook\toks@{\install@mathalphabet##1{##2}}%
            }%
        \def\getanddefine@fonts##1##2{%
          \ifnum##1=#3%
             \addto@hook\toks@{\getanddefine@fonts#3#2}%
             \expandafter\get@cdp\string##2\@nil\reserved@b
             \ifx\reserved@a\reserved@b\else
                \@font@warning{Encoding `\reserved@b' has changed
                    to `\reserved@a' for symbol font\MessageBreak
                   `\expandafter\@gobblefour\string#3' in the
                    math version `\expandafter
                    \@gobblefour\string#1'}%
             \fi
             \@font@info{%
                Overwriting symbol font
                `\expandafter\@gobblefour\string#3' in
                 version `\expandafter
                \@gobblefour\string#1'\MessageBreak
                \@spaces \expandafter\@gobble\string##2 -->
                         \expandafter\@gobble\string#2}%
          \else
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
          \fi}%
         #1%
         \xdef#1{\the\toks@}%
      \endgroup
    \else
       \@latex@error{Symbol font `\expandafter\@gobblefour\string#3'
                  not defined}\@eha
    \fi
  \else
    \@latex@error{Math version `\expandafter\@gobblefour\string#1'
       is not
       defined}{You probably mispelled the name of the math
       version.^^JOr you have to specify an additional package.}%
  \fi
}
\def\get@cdp#1#2/#3\@nil#4{\def#4{#2}}
\def\DeclareMathAlphabet#1#2#3#4#5{%
 \@tempswafalse
 \edef\reserved@b{#2}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
   \expandafter\ifx
   \csname\expandafter\@gobble\string#1\endcsname
```

```
  \relax
     \new@mathalphabet#1{#2}{#3}{#4}{#5}%
  \else
```

Check if it is already a math alphabet.

```
    \edef\reserved@a{\noexpand\in@{\string\select@group}%
        {\expandafter\meaning\csname @\expandafter
         \@gobble\string#1\endcsname}}%
    \reserved@a
    \ifin@
      \@font@info{Redeclaring math alphabet \string#1}%
      \def\version@elt##1{%
        \expandafter\SetMathAlphabet@\expandafter
           ##1\csname#2/#3/#4/#5\expandafter\endcsname

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

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

```
      \edef\reserved@a{\noexpand\in@{\string\use@mathgroup}%
        {\expandafter\meaning\csname @\expandafter
         \@gobble\string#1\endcsname}}%
      \reserved@a
      \ifin@
```

In that case overwriting is simple since there is nothing inserted in the math version macros.

```
        \@font@info{Redeclaring math alphabet \string#1}%
        \new@mathalphabet#1{#2}{#3}{#4}{#5}%
```

Otherwise panic.

```
      \else
        \@latex@error{Command `\string#1' already defined}\@eha
      \fi
    \fi
  \fi
 \else
  \@latex@error{Encoding scheme  `#2' unknown}\@eha
 \fi
  }
\def\new@mathalphabet#1#2#3#4#5{%
    \toks@\expandafter{\alpha@list}%
    \edef#1{\expandafter\noexpand\csname @\expandafter
            \@gobble\string#1\endcsname
            \if/#5/%
               \noexpand\no@alphabet@error
               \noexpand\no@alphabet@error
            \else
               \expandafter\noexpand\csname M@#2\endcsname
```

```
               \expandafter\noexpand\csname#2/#3/#4/#5\endcsname
            \fi
           }%
    \toks2\expandafter{#1}%
    \edef\alpha@list{\the\toks@\noexpand\alpha@elt\the\toks2}%
    \def\version@elt##1{\toks@\expandafter{##1}%
                   \edef##1{\the\toks@\install@mathalphabet
                            \expandafter\noexpand
                            \csname @\expandafter\@gobble
                               \string#1\endcsname
                           {\if/#5/%
                             \noexpand\no@alphabet@error
                             \noexpand#1%
                            \else
                             \noexpand\select@group\the\toks2
                            \fi}}%
                  }%
    \version@list
    \expandafter\edef\csname @\expandafter\@gobble
               \string#1\endcsname{\if/#5/%
              \noexpand\no@alphabet@error
              \noexpand#1%
            \else
              \noexpand\select@group\the\toks2
            \fi}%
    \edef#1{\noexpand\protect
            \expandafter\noexpand\csname @\expandafter
            \@gobble\string#1\endcsname}%
}
\let\alpha@list\@empty
\def\SetMathAlphabet#1#2#3#4#5#6{%
 \@tempswafalse
 \edef\reserved@b{#3}%
 \def\cdp@elt##1##2##3##4{\def\reserved@c{##1}%
      \ifx\reserved@b\reserved@c \@tempswatrue\fi}%
 \cdp@list
 \if@tempswa
  \expandafter\SetMathAlphabet@
    \csname mv@#2\expandafter\endcsname\csname#3/#4/#5/#6\expandafter
    \endcsname \csname M@#3\expandafter\endcsname
    \csname @\expandafter\@gobble\string#1\endcsname#1%
 \else
  \@latex@error{Encoding scheme  `#3' unknown}\@eha
 \fi
}
\def\SetMathAlphabet@#1#2#3#4#5{%
  \expandafter\in@\expandafter#1\expandafter{\version@list}%
  \ifin@
    \expandafter\in@\expandafter#4\expandafter{\alpha@list}%
```

```
  \ifin@
    \begingroup
      \toks@{}%
      \def\getanddefine@fonts##1##2{%
           \addto@hook\toks@{\getanddefine@fonts##1##2}%
          }%
      \def\reserved@c##1##2##3##4{%                % for message below
          \expandafter\@gobble\string##4}%
      \def\install@mathalphabet##1##2{%
        \ifx##1#4%
           \addto@hook\toks@
              {\install@mathalphabet#4{\select@group#4#3#2}}%
           \@font@info{Overwriting math alphabet
              `\string#5' in version `\expandafter
               \@gobblefour\string#1'\MessageBreak
               \@spaces \reserved@c##2 -->
                      \expandafter\@gobble\string#2}%
        \else
           \addto@hook\toks@{\install@mathalphabet##1{##2}}%
        \fi
        }%
      #1%
      \xdef#1{\the\toks@}%
    \endgroup
  \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.

```
      \edef\reserved@a{%
        \noexpand\in@{\string\use@mathgroup}{\meaning#4}}%
      \reserved@a
      \ifin@
        \def\reserved@b##1\use@mathgroup##2##3{%
            \def\reserved@b{##3}\def\reserved@c{##2}}%
        \expandafter\reserved@b#4%
        \begingroup
          \def\install@mathalphabet##1##2{%
              \addto@hook\toks@{\install@mathalphabet##1{##2}}%
             }%
           \def\getanddefine@fonts##1##2{%
             \addto@hook\toks@{\getanddefine@fonts##1##2}%
             \ifnum##1=\reserved@b
                \expandafter
                \addto@hook\expandafter\toks@
                \expandafter{\expandafter\install@mathalphabet
                \expandafter#4\expandafter
                      {\expandafter\select@group\expandafter
                        #4\reserved@c##2}}%
             \fi
```

```
                      }%
           \def\version@elt##1{%
               \toks@{}%
               ##1%
               \xdef##1{\the\toks@}%
              }%
           \version@list
         \endgroup
```

Put it into the `\alpha@list` with default 'error'

```
         \expandafter\gdef\expandafter\alpha@list\expandafter
             {\alpha@list
              \alpha@elt #4\no@alphabet@error \no@alphabet@error}%
         \gdef#4{\no@alphabet@error #5}% fake things :-)
```

Then call the internal setting routine again:

```
         \SetMathAlphabet@{#1}{#2}{#3}#4#5%
       \else
         \@latex@error{Command `\string#5' not defined as a
                       math alphabet}%
            {Use \noexpand\DeclareMathAlphabet to define it.}%
       \fi
    \fi
  \else
    \@latex@error{Math version `\expandafter\@gobblefour\string#1'
       is not
       defined}{You probably mispelled the name of the math
       version.^^JOr you have to specify an additional package.}%
  \fi
}
```

could do with more checks like allowing single number in `#4` lowercase in `#4` etc

```
\def\DeclareMathAccent#1#2#3#4{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \begingroup
      \count\z@=#4\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \if\relax\noexpand#1% is command?
        \edef\reserved@a{\noexpand\in@{\string\mathaccent}{\meaning#1}}%
        \reserved@a
        \ifin@
          \expandafter\set@mathaccent
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
```

```
        \@font@info{Redeclaring math accent \string#1}%
      \else
        \expandafter\ifx
        \csname\expandafter\@gobble\string#1\endcsname
        \relax
          \expandafter\set@mathaccent
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
        \else
          \@latex@error{Command `\string#1' already defined}\@eha
        \fi
      \fi
    \else
     \@latex@error{Not a command name: `\noexpand#1'}\@eha
    \fi
  \endgroup
 \else
  \@latex@error{Symbol font `#3' is not defined}\@eha
 \fi
}
```

```
\def\set@mathaccent#1#2#3#4{%
 \xdef#2{\mathaccent"\mathchar@type#3\hexnumber@#1#4\relax}}
```

```
\def\DeclareMathSymbol#1#2#3#4{%
 \expandafter\in@\csname sym#3\expandafter\endcsname
    \expandafter{\group@list}%
 \ifin@
  \begingroup
    \count\z@=#4\relax
    \count\tw@\count\z@
    \divide\count\z@\sixt@@n
    \count@\count\z@
    \multiply\count@\sixt@@n
    \advance\count\tw@-\count@
    \if\relax\noexpand#1% is command?
      \edef\reserved@a{\noexpand\in@{\string\mathchar}{\meaning#1}}%
      \reserved@a
      \ifin@
        \expandafter\set@mathsymbol
           \csname sym#3\endcsname#1#2%
           {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
        \@font@info{Redeclaring math symbol \string#1}%
      \else
          \expandafter\ifx
          \csname\expandafter\@gobble\string#1\endcsname
          \relax
          \expandafter\set@mathsymbol
             \csname sym#3\endcsname#1#2%
             {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
        \else
```

```
            \@latex@error{Command `\string#1' already defined}\@eha
          \fi
        \fi
      \else
        \expandafter\set@mathchar
          \csname sym#3\endcsname#1#2
          {\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      \fi
    \endgroup
  \else
    \@latex@error{Symbol font `#3' is not defined}\@eha
  \fi
}
\def\set@mathchar#1#2#3#4{%
  \global\mathcode`#2="\mathchar@type#3\hexnumber@#1#4\relax}
\def\set@mathsymbol#1#2#3#4{%
  \global\mathchardef#2"\mathchar@type#3\hexnumber@#1#4\relax}
%\def\mathsymbol#1#2#3{%
%  \@tempcnta=#3\relax
%  \@tempcntb\@tempcnta
%  \divide\@tempcnta\sixt@@n
%  \count@\@tempcnta
%  \multiply\count@\sixt@@n
%  \advance\@tempcntb-\count@
%  \mathchar"\mathchar@type#1\hexnumber@#2%
%             \hexnumber@\@tempcnta\hexnumber@\@tempcntb\relax}


%
%\def\DeclareMathAlphabetCharacter#1#2#3{%
%  \DeclareMathSymbol{#1}7{#2}{#3}}

\def\DeclareMathDelimiter#1{%
  \if\relax\noexpand#1%
    \expandafter\@DeclareMathDelimiter
  \else
    \expandafter\@xDeclareMathDelimiter
  \fi
  #1}
\def\@DeclareMathDelimiter#1#2#3#4#5#6{%
  \expandafter\in@\csname sym#3\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
    \expandafter\in@\csname sym#5\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \begingroup
        \count\z@=#4\relax
```

```
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
    %
      \count\z@=#6\relax
      \count\tw@\count\z@
      \divide\count\z@\sixt@@n
      \count@\count\z@
      \multiply\count@\sixt@@n
      \advance\count\tw@-\count@
      \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
    %
      \edef\reserved@a{\noexpand\in@{\string\delimiter}{\meaning#1}}%
      \reserved@a
      \ifin@
        \expandafter\set@mathdelimiter
           \csname sym#3\expandafter\endcsname
           \csname sym#5\endcsname#1#2%
           \reserved@c\reserved@d
        \@font@info{Redeclaring math delimiter \string#1}%
      \else
          \expandafter\ifx
          \csname\expandafter\@gobble\string#1\endcsname
          \relax
          \expandafter\set@mathdelimiter
            \csname sym#3\expandafter\endcsname
            \csname sym#5\endcsname#1#2%
            \reserved@c\reserved@d
        \else
          \@latex@error{Command `\string#1' already defined}\@eha
        \fi
      \fi
    \endgroup
  \else
    \@latex@error{Symbol font `#5' is not defined}\@eha
  \fi
\else
  \@latex@error{Symbol font `#3' is not defined}\@eha
\fi
}
\def\@xDeclareMathDelimiter#1#2#3#4#5{%
\expandafter\in@\csname sym#2\expandafter\endcsname
   \expandafter{\group@list}%
\ifin@
  \expandafter\in@\csname sym#4\expandafter\endcsname
     \expandafter{\group@list}%
```

```
    \ifin@
      \begingroup
        \count\z@=#3\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@c{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
      %
        \count\z@=#5\relax
        \count\tw@\count\z@
        \divide\count\z@\sixt@@n
        \count@\count\z@
        \multiply\count@\sixt@@n
        \advance\count\tw@-\count@
        \edef\reserved@d{\hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
        \expandafter\set@@mathdelimiter
           \csname sym#2\expandafter\endcsname\csname sym#4\endcsname#1%
           \reserved@c\reserved@d
      \endgroup
    \else
      \@latex@error{Symbol font `#4' is not defined}\@eha
    \fi
  \else
    \@latex@error{Symbol font `#2' is not defined}\@eha
  \fi
}
```

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 contructs involving `\abovewithdelims` will prematurely end (pr/1329)

```
\def\set@mathdelimiter#1#2#3#4#5#6{%
  \xdef#3{\delimiter"\mathchar@type#4\hexnumber@#1#5%
                                     \hexnumber@#2#6 }}
```

```
\def\set@@mathdelimiter#1#2#3#4#5{%
  \global\delcode`#3="\hexnumber@#1#4\hexnumber@#2#5\relax}
```

```
\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!

```
  \expandafter\ifx
       \csname\expandafter\@gobble\string#1\endcsname
       \relax
     \let#1\radical
  \fi
  \edef\reserved@a{\noexpand\in@{\string\radical}{\meaning#1}}%
  \reserved@a
  \ifin@
```

```
    \expandafter\in@\csname sym#2\expandafter\endcsname
       \expandafter{\group@list}%
    \ifin@
      \expandafter\in@\csname sym#4\expandafter\endcsname
         \expandafter{\group@list}%
      \ifin@
        \begingroup
          \count\z@=#3\relax
          \count\tw@\count\z@
          \divide\count\z@\sixt@@n
          \count@\count\z@
          \multiply\count@\sixt@@n
          \advance\count\tw@-\count@
          \edef\reserved@c{%
            \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
          \count\z@=#5\relax
          \count\tw@\count\z@
          \divide\count\z@\sixt@@n
          \count@\count\z@
          \multiply\count@\sixt@@n
          \advance\count\tw@-\count@
          \edef\reserved@d{%
            \hexnumber@{\count\z@}\hexnumber@{\count\tw@}}%
```

Coded inline instead of using `\set@mathradical`

```
%          \expandafter\set@mathradical
%             \csname sym#2\expandafter\endcsname
%             \csname sym#4\endcsname#1%
%             \reserved@c\reserved@d
          \xdef#1{\radical"\expandafter\hexnumber@
                               \csname sym#2\endcsname\reserved@c
                            \expandafter\hexnumber@
                               \csname sym#4\endcsname\reserved@d
                  \relax}%
        \endgroup
      \else
        \@latex@error{Symbol font `#4' is not defined}\@eha
      \fi
    \else
      \@latex@error{Symbol font `#2' is not defined}\@eha
    \fi
  \else
    \@latex@error{Command `\string#1' already defined}\@eha
  \fi
}
```

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}}
```

```
\let\mathalpha\relax % just a dummy currently
```

```
\def\mathchar@type#1{%
  \ifodd 2#11 #1\else            % is this non-negative number?
    \ifx#1\mathord 0\else
     \ifx#1\mathop   1\else
      \ifx#1\mathbin 2\else
       \ifx#1\mathrel 3\else
        \ifx#1\mathopen 4\else
         \ifx#1\mathclose 5\else
          \ifx#1\mathpunct 6\else
              7%            % anything else is variable ord
          \fi
         \fi
        \fi
       \fi
      \fi
     \fi
    \fi
  \fi}
\def\DeclareSymbolFontAlphabet#1#2{%
  \expandafter\DeclareSymbolFontAlphabet@
    \csname @\expandafter\@gobble\string#1\endcsname{#2}#1}
\def\DeclareSymbolFontAlphabet@#1#2#3{%
```

We use the switch `\if@tempswa` to decide if we can declare this symbol font alphabet.

```
    \@tempswatrue
```

First check if `#2` is known to be a symbol font

```
  \expandafter\in@\csname sym#2\expandafter\endcsname
     \expandafter{\group@list}%
  \ifin@
```

Check if `#1` is defined as a math alphabet defined via `\DeclareMathAlphabet`:

```
    \expandafter\in@\expandafter#1\expandafter{\alpha@list}%
    \ifin@
```

If so remove it from the `\alpha@list` and from all math version macros.

```
      \@font@info{Redeclaring math alphabet \string#3}%
      \toks@{}%
      \def\alpha@elt##1##2##3{%
          \ifx##1#1\else\addto@hook\toks@{\alpha@elt##1##2##3}\fi}%
      \alpha@list
      \xdef\alpha@list{\the\toks@}%
```

Now we loop over all versions and remove the math alphabet:

```
      \def\version@elt##1{%
          \begingroup
            \toks@{}%
            \def\getanddefine@fonts####1####2{%
               \addto@hook\toks@{\getanddefine@fonts####1####2}}%
            \def\install@mathalphabet####1####2{%
```

```
              \ifx####1#1\else
                \addto@hook\toks@{\install@mathalphabet
                                   ####1{####2}}\fi}%
          ##1%
          \xdef##1{\the\toks@}%
        \endgroup
        }%
     \version@list
   \else
```

If #3 is not defined as a math alphabet check if it is defined at all:

```
     \expandafter\ifx
     \csname\expandafter\@gobble\string#1\endcsname
     \relax
```

If it is undefined, fine otherwise check if it is a math alphabet defined via `\DeclareSymbolFontAlphabet`:

```
     \else
       \edef\reserved@a{%
         \noexpand\in@{\string\use@mathgroup}{\meaning#1}}%
       \reserved@a
       \ifin@
         \@font@info{Redeclaring math alphabet \string#3}%
       \else
```

Since the command #3 is defined to be something which is not a math alphabet we have to skip redefining it.

```
         \@tempswafalse
         \@latex@error{Command `\string#3' already defined}\@eha
       \fi
     \fi
   \fi
  \else
```

Since the symbol font is not known we better skip defining this alphabet.

```
    \@tempswafalse
    \@latex@error{Unknown symbol font `#2'}\@eha
  \fi
  \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.

```
    \def\group@elt##1##2{%
       \expandafter\ifx\csname sym#2\endcsname##1%
       \expandafter\reserved@a\string##2\@nil
```

```
       \fi}%
    \def\reserved@a##1##2/##3\@nil{%
       \def\reserved@a{##2}}%
    \group@list
    \toks@{\relax\ifmmode \else \non@alpherr#1\fi}%
    \edef#1{\the\toks@
            \noexpand\use@mathgroup
            \expandafter\noexpand\csname M@\reserved@a\endcsname
            \csname sym#2\endcsname}%
    \def#3{\protect#1}%
  \fi
}
```

File q
ltfssini.dtx

This file contains the top level LaTeX interface to the font selection scheme commands. See other parts of the LaTeX distribution, or *The LaTeX Companion* for higher level documentation of these commands.

32 NFSS Initialisation

Finally, there are six commands that are to be used in LaTeX and that we will therefore protect against expansion at the wrong point: `\fontfamily`, `\fontseries`, `\fontshape`, `\fontsize`, `\selectfont`, and `\mathversion`.

32.1 Providing math *versions*

LaTeX provides two *versions*. We call them normal and bold, respectively.

```
\DeclareMathVersion{normal}
\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*:

```
\DeclareRobustCommand\rmfamily
        {\not@math@alphabet\rmfamily\mathrm
         \fontfamily\rmdefault\selectfont}
\DeclareRobustCommand\sffamily
        {\not@math@alphabet\sffamily\mathsf
         \fontfamily\sfdefault\selectfont}
\DeclareRobustCommand\ttfamily
        {\not@math@alphabet\ttfamily\mathtt
         \fontfamily\ttdefault\selectfont}
```

Then the commands changing the *series*:

```
\DeclareRobustCommand\bfseries
        {\not@math@alphabet\bfseries\mathbf
         \fontseries\bfdefault\selectfont}
\DeclareRobustCommand\mdseries
        {\not@math@alphabet\mdseries\relax
         \fontseries\mddefault\selectfont}
\DeclareRobustCommand\upshape
        {\not@math@alphabet\upshape\relax
         \fontshape\updefault\selectfont}
```

Then the commands changing the *shape*:

```
\DeclareRobustCommand\slshape
        {\not@math@alphabet\slshape\relax
         \fontshape\sldefault\selectfont}
```

```
\DeclareRobustCommand\scshape
        {\not@math@alphabet\scshape\relax
         \fontshape\scdefault\selectfont}
\DeclareRobustCommand\itshape
        {\not@math@alphabet\itshape\mathit
         \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`.

```
\DeclareRobustCommand\em
        {\@nomath\em \ifdim \fontdimen\@ne\font >\z@
                       \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`.

```
\def\not@math@alphabet#1#2{%
   \relax
   \ifmmode
     \@latex@error{Command \noexpand#1invalid in math mode}%
        {%
         Please
         \ifx#2\relax
            define a new math alphabet^^J%
            if you want to use a special font in math mode%
          \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 :-).

```
            use the math alphabet \noexpand#2instead of
            the #1command%
         \fi
         .
        }%
   \fi}
```

Finally we provide two abbreviations to switch to the LaTeX *versions*.

```
\def\boldmath{\@nomath\boldmath
             \mathversion{bold}}
\def\unboldmath{\@nomath\unboldmath
             \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`.

```
\def\math@version{normal}
```

32.2 Miscellaneous

\newfont \symbol We start by defining a few macros that are part of standard LaTeX's user interface. The use of these functions is not encouraged, but they will allow to process older documents without changes to the source.

```
\def\newfont#1#2{\@ifdefinable#1{\font#1=#2\relax}}
\def\symbol#1{\char #1\relax}
```

\@setfontsize \@setsize This abbreviation is used by LaTeX's user level size changing commands, such as `\large`.

```
\def\@setfontsize#1#2#3{\@nomath#1%
```

For the benefit of people relying on keeping the name of the current font command saved in `\@currsize` we define it. To ensure that `\@setfontsize` keeps being robust we omit this assignment during times where `\protect` differs from `\@typeset@protect`.

```
    \ifx\protect\@typeset@protect
      \let\@currsize#1%
    \fi
    \fontsize{#2}{#3}\selectfont}
```

For compatibility we also define `\@setsize` the 209 command

```
⟨*compat⟩
\def\@setsize#1#2#3#4{\@setfontsize#1{#4}{#2}}
⟨/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.

```
\def\oldstylenums#1{%
   \begingroup
```

Provide spacing using the interword space of the current font.

```
    \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 (For this there must be a substitution of `OML/cmm/bx/it` into `OML/cmm/b/it` in the corresponding `.fd` files.

```
    \usefont{OML}{cmm}{\f@series}{it}%
    \mathgroup\symletters #1%
   \endgroup
}
```

\hexnumber@ To set up LaTeX's special math character definitions we first provide a macro to generate hexadecimal numbers. It is a rather simple `\ifcase`.

```
\def\hexnumber@#1{\ifcase\number#1
 0\or 1\or 2\or 3\or 4\or 5\or 6\or 7\or 8\or
 9\or A\or B\or C\or D\or E\or F\fi}
```

\nfss@text In it 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.

```
\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.

```
%\DeclareRobustCommand\copyright
%    {{\ooalign{\hfil
%     \raise.07ex\hbox{\mdseries\upshape c}\hfil\crcr
%     \mathhexbox20D}}}
```

\normalfont
\reset@font
\p@reset@font

The macro `\reset@font` is used in LaTeX to switch to a standard font, in order to initialize the current font in situations where typesetting is done in a new visual context (e.g. in a footnote). We define it here to allow the test for the new LaTeX version above but nevertheless are able to run all kind of mixtures.

The user interface name for `\reset@font` is `\normalfont`:

```
\DeclareRobustCommand\normalfont
                 {\usefont\encodingdefault
                          \familydefault
                          \seriesdefault
                          \shapedefault}
\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.

```
\def\not@base#1{\@latex@error
  {Command \noexpand#1not provided in base LaTeX2e}%
  {Load the latexsym or the amsfonts package to
   define this symbol}}
\def\mho{\not@base\mho}
\def\Join{\not@base\Join}
\def\Box{\not@base\Box}
\def\Diamond{\not@base\Diamond}
\def\leadsto{\not@base\leadsto}
\def\sqsubset{\not@base\sqsubset}
\def\sqsupset{\not@base\sqsupset}
\def\lhd{\not@base\lhd}
```

```
\def\unlhd{\not@base\unlhd}
\def\rhd{\not@base\rhd}
\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.

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}  %%% don't modify this setting
                                       %%% overwrite it in fontdef.cfg
                                       %%% if necessary
```

We now load the customizable parts of NFSS.

```
\InputIfFileExists{fonttext.cfg}
           {\typeout{*************************************^^J%
                     *^^J%
                     * Local config file fonttext.cfg used^^J%
                     *^^J%
                     *************************************}%
            }
           {\input{fonttext.ltx}}
```

Ditto for math although I don't that we will get a lot of customisation :-)

```
\InputIfFileExists{fontmath.cfg}
           {\typeout{*************************************^^J%
                     *^^J%
                     * Local config file fontmath.cfg used^^J%
                     *^^J%
                     *************************************}%
            }
           {\input{fontmath.ltx}}
```

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`.

```
\InputIfFileExists{preload.cfg}
           {\typeout{*************************************^^J%
                     *^^J%
                     * Local config file preload.cfg used^^J%
                     *^^J%
                     *************************************}%
            }
           {\input{preload.ltx}}
```

`\@acci` `\@accii` `\@acciii`

We also save the values of some accents in `\@acci`, `\@accii` and `\@acciii` so they can be restored by a `minipage` inside a `tabbing` environment.

```
\let\@acci\' \let\@accii\` \let\@acciii\=
```

`\cal` `\mit`

Here were the two old ⟨*alphabet identifiers*⟩.

File r
fontdef.dtx

33 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.

34 Customization

You are not allowed to change this source file! If you want to change the default encodings and/or the fonts shape groups preloaded you should 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 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, that 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 encodings are 'old' in the sense that we hope that they will be superseeded soon by encoding standards defined by the TeX user's community. Therefore this set of default encodings may change in the future.

The first candidate is `OT1` which will soon be replaced by `T1` the official TeX text encoding.

> **Warning:** If you add additional encodings to this file there is no guarantee any longer that the files processable at your installation will be processable at other installations too. Thus, if you make use of such an encoding in your document, e.g., if you intend to typeset in Cyrillic (`OT2` encoding), you need to specify this encoding in the preamble of your document prior to sending it to another installation. Once the encoding is specified in that place in your document, the document is processable at all LaTeX installations.
>
> 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.

35 The docstrip modules

The following modules are used to direct `docstrip` in generating external files:

driver	produce a documentation driver file
text	produce the file `fonttext.ltx`
math	produce the file `fonttext.ltx`

A typical `docstrip` command file would then have entries like:

```
\generateFile{fonttext.ltx}{t}{\from{fontdef.dtx}{text}}
```

36 A driver for this document

The next bit of code contains the documentation driver file for TeX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass{ltxdoc}
\GetFileInfo{fontdef.dtx}
```

```
\begin{document}
   \DocInput{fontdef.dtx}
\end{document}
⟨/driver⟩
```

37 The `fonttext.ltx` file

The idendification is done earlier on with a `\ProvidesFile` declaration.

```
⟨*text⟩
\typeout{*** Don't modify this file, use a .cfg file instead ***}
```

37.1 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, eg 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`, eg `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.

```
\input {OMLenc.def}
\input  {T1enc.def}
\input {OT1enc.def}        % <- should come after T1 for speed
\input {OMSenc.def}
```

We then set 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.

```
\fontencoding{OT1}
```

If different encodings for text fonts are in use one could place the common setups into `\DeclareFontEncodingDefaults`. There is now a better mechanism so using this interface is discouraged!

```
\DeclareFontEncodingDefaults{}{}
```

Then we define the default substitution for every encoding. This release of LaTeX 2_ε assumes that the DC 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.

```
\DeclareFontSubstitution{T1}{cmr}{m}{n}
\DeclareFontSubstitution{OT1}{cmr}{m}{n}
```

For every encoding declaration, LaTeX 2_ε will try to verify that the given substitution information makes sense, i.e., that it is impossible to go into an endless loop if font substitution happens. This is done at the moment the `\begin{document}` is encountered. LaTeX 2_ε will then check that for every encoding the substitution defaults form a valid font shape group, which means that it will look 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 LaTeX 2_ε will not read `.fd` files if it already knows about the encoding/family combination.

```
\input  {T1cmr.fd}
\input  {OT1cmr.fd}
```

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 on (nearly) every TeX installation, while the amount of main memory is not a limiting factor on most installations.)

```
\input {OT1cmss.fd}
\input {OT1cmtt.fd}
```

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 for such cases NFSS has a very low-level fallback font that is installed with the following line.

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

This means, "if everything else fails use Computer Modern Roman normal shape at 10pt in the old encoding. You can change the font used but the encoding should be the same as the one specified with `\fontencoding` above.

37.2 Defaults

To allow the use of `\rmfamily`, `\sffamily`, etc. in documents even if nonstandard 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`
`\sfdefault`
`\ttdefault`

The following three definitions set up the meaning for `\rmfamily`, `\sffamily`, and `\ttfamily`.

```
\newcommand\rmdefault{cmr}
\newcommand\sfdefault{cmss}
\newcommand\ttdefault{cmtt}
```

`\bfdefault` `\mddefault`

Series changing commands are influenced by the following hooks.

```
\newcommand\bfdefault{bx}
\newcommand\mddefault{m}
```

`\itdefault` `\sldefault` `\scdefault` `\updefault`

Shape changing commands use the following hooks.

```
\newcommand\itdefault{it}
\newcommand\sldefault{sl}
\newcommand\scdefault{sc}
\newcommand\updefault{n}
```

`\encodingdefault` `\familydefault` `\seriesdefault` `\shapedefault`

Finally we have the hooks that describe the behaviour of the`\normalfont` command. To stay portable, the definition of `\encodingdefault` should *not* be changed and should match the setting above for `\fontencoding`. All other values can be set according to your taste.

```
\newcommand\encodingdefault{OT1}
\newcommand\familydefault{\rmdefault}
\newcommand\seriesdefault{\mddefault}
\newcommand\shapedefault{\updefault}
```

This finishes the low-level setup in `fonttext.ltx`.

```
⟨/text⟩
```

38 The `fontmath.ltx` file

The idendification is done earlier on with a `\ProvidesFile` declaration.

```
⟨*math⟩
\typeout{*** Don't modify this file, use a .cfg file instead ***}
```

38.1 The fontencodings used

```
\DeclareFontEncoding{OML}{}{}
\DeclareFontEncoding{OMS}{}{}
\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 fetched from some other font that has them.

```
\DeclareFontEncoding{U}{}{\noaccents@}
```

The encodings for math are next:

```
\DeclareFontSubstitution{OML}{cmm}{m}{it}
\DeclareFontSubstitution{OMS}{cmsy}{m}{n}
```

```
\DeclareFontSubstitution{OMX}{cmex}{m}{n}
\DeclareFontSubstitution{U}{cmr}{m}{n}
```

```
\input  {OMLcmm.fd}
\input  {OMScmsy.fd}
\input  {OMXcmex.fd}
\input  {Ucmr.fd}
```

38.1.1 Symbolfont and Alphabet declarations

We now define the basic symbol fonts used by LaTeX. 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.

```
\DeclareSymbolFont{operators}   {OT1}{cmr} {m}{n}
\DeclareSymbolFont{letters}     {OML}{cmm} {m}{it}
\DeclareSymbolFont{symbols}     {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}

\SetSymbolFont{operators}{bold}{OT1}{cmr} {bx}{n}
\SetSymbolFont{letters}  {bold}{OML}{cmm} {b}{it}
\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.

```
\DeclareSymbolFontAlphabet{\mathrm}    {operators}
\DeclareSymbolFontAlphabet{\mathnormal}{letters}
\DeclareSymbolFontAlphabet{\mathcal}   {symbols}
\DeclareMathAlphabet      {\mathbf}{OT1}{cmr}{bx}{n}
\DeclareMathAlphabet      {\mathsf}{OT1}{cmss}{m}{n}
\DeclareMathAlphabet      {\mathit}{OT1}{cmr}{m}{it}
\DeclareMathAlphabet      {\mathtt}{OT1}{cmtt}{m}{n}
```

Given the currently available fonts we cannot bolden `\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.

```
\SetMathAlphabet\mathsf{bold}{OT1}{cmss}{bx}{n}
\SetMathAlphabet\mathit{bold}{OT1}{cmr}{bx}{it}
```

38.2 Math font sizes

The declarations below declare the text, script and scriptscript size to be used for each text font size.

All occurences of sizes longer than a single character are replaced with the macro name that holds them, saving a number of tokens (but loose a bit of speed, so this may not stay that way).

```
\DeclareMathSizes{5}{5}{5}{5}
\DeclareMathSizes{6}{6}{5}{5}
\DeclareMathSizes{7}{7}{5}{5}
\DeclareMathSizes{8}{8}{6}{5}
\DeclareMathSizes{9}{9}{6}{5}
\DeclareMathSizes{\@xpt}{\@xpt}{7}{5}
\DeclareMathSizes{\@xipt}{\@xipt}{8}{6}
\DeclareMathSizes{\@xiipt}{\@xiipt}{8}{6}
\DeclareMathSizes{\@xivpt}{\@xivpt}{\@xpt}{7}
\DeclareMathSizes{\@xviipt}{\@xviipt}{\@xiipt}{\@xpt}
\DeclareMathSizes{\@xxpt}{\@xxpt}{\@xivpt}{\@xiipt}
\DeclareMathSizes{\@xxvpt}{\@xxvpt}{\@xxpt}{\@xviipt}
```

38.3 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 IniTEX. However, we repeat them here to have a complete setup which can be exchanged with another if desired.

38.3.1 The letters

```
\DeclareMathSymbol{a}{\mathalpha}{letters}{`a}
\DeclareMathSymbol{b}{\mathalpha}{letters}{`b}
\DeclareMathSymbol{c}{\mathalpha}{letters}{`c}
\DeclareMathSymbol{d}{\mathalpha}{letters}{`d}
\DeclareMathSymbol{e}{\mathalpha}{letters}{`e}
\DeclareMathSymbol{f}{\mathalpha}{letters}{`f}
\DeclareMathSymbol{g}{\mathalpha}{letters}{`g}
\DeclareMathSymbol{h}{\mathalpha}{letters}{`h}
\DeclareMathSymbol{i}{\mathalpha}{letters}{`i}
\DeclareMathSymbol{j}{\mathalpha}{letters}{`j}
\DeclareMathSymbol{k}{\mathalpha}{letters}{`k}
\DeclareMathSymbol{l}{\mathalpha}{letters}{`l}
\DeclareMathSymbol{m}{\mathalpha}{letters}{`m}
\DeclareMathSymbol{n}{\mathalpha}{letters}{`n}
\DeclareMathSymbol{o}{\mathalpha}{letters}{`o}
\DeclareMathSymbol{p}{\mathalpha}{letters}{`p}
\DeclareMathSymbol{q}{\mathalpha}{letters}{`q}
\DeclareMathSymbol{r}{\mathalpha}{letters}{`r}
\DeclareMathSymbol{s}{\mathalpha}{letters}{`s}
\DeclareMathSymbol{t}{\mathalpha}{letters}{`t}
\DeclareMathSymbol{u}{\mathalpha}{letters}{`u}
\DeclareMathSymbol{v}{\mathalpha}{letters}{`v}
\DeclareMathSymbol{w}{\mathalpha}{letters}{`w}
\DeclareMathSymbol{x}{\mathalpha}{letters}{`x}
```

```
\DeclareMathSymbol{y}{\mathalpha}{letters}{`y}
\DeclareMathSymbol{z}{\mathalpha}{letters}{`z}

\DeclareMathSymbol{A}{\mathalpha}{letters}{`A}
\DeclareMathSymbol{B}{\mathalpha}{letters}{`B}
\DeclareMathSymbol{C}{\mathalpha}{letters}{`C}
\DeclareMathSymbol{D}{\mathalpha}{letters}{`D}
\DeclareMathSymbol{E}{\mathalpha}{letters}{`E}
\DeclareMathSymbol{F}{\mathalpha}{letters}{`F}
\DeclareMathSymbol{G}{\mathalpha}{letters}{`G}
\DeclareMathSymbol{H}{\mathalpha}{letters}{`H}
\DeclareMathSymbol{I}{\mathalpha}{letters}{`I}
\DeclareMathSymbol{J}{\mathalpha}{letters}{`J}
\DeclareMathSymbol{K}{\mathalpha}{letters}{`K}
\DeclareMathSymbol{L}{\mathalpha}{letters}{`L}
\DeclareMathSymbol{M}{\mathalpha}{letters}{`M}
\DeclareMathSymbol{N}{\mathalpha}{letters}{`N}
\DeclareMathSymbol{O}{\mathalpha}{letters}{`O}
\DeclareMathSymbol{P}{\mathalpha}{letters}{`P}
\DeclareMathSymbol{Q}{\mathalpha}{letters}{`Q}
\DeclareMathSymbol{R}{\mathalpha}{letters}{`R}
\DeclareMathSymbol{S}{\mathalpha}{letters}{`S}
\DeclareMathSymbol{T}{\mathalpha}{letters}{`T}
\DeclareMathSymbol{U}{\mathalpha}{letters}{`U}
\DeclareMathSymbol{V}{\mathalpha}{letters}{`V}
\DeclareMathSymbol{W}{\mathalpha}{letters}{`W}
\DeclareMathSymbol{X}{\mathalpha}{letters}{`X}
\DeclareMathSymbol{Y}{\mathalpha}{letters}{`Y}
\DeclareMathSymbol{Z}{\mathalpha}{letters}{`Z}
```

38.3.2 The digits

```
\DeclareMathSymbol{0}{\mathalpha}{operators}{`0}
\DeclareMathSymbol{1}{\mathalpha}{operators}{`1}
\DeclareMathSymbol{2}{\mathalpha}{operators}{`2}
\DeclareMathSymbol{3}{\mathalpha}{operators}{`3}
\DeclareMathSymbol{4}{\mathalpha}{operators}{`4}
\DeclareMathSymbol{5}{\mathalpha}{operators}{`5}
\DeclareMathSymbol{6}{\mathalpha}{operators}{`6}
\DeclareMathSymbol{7}{\mathalpha}{operators}{`7}
\DeclareMathSymbol{8}{\mathalpha}{operators}{`8}
\DeclareMathSymbol{9}{\mathalpha}{operators}{`9}
```

38.3.3 Punctuation, brace, etc. keys

```
\DeclareMathSymbol{!}{\mathclose}{operators}{"21}
\DeclareMathSymbol{(}{\mathopen}{operators}{"28}
\DeclareMathSymbol{)}{\mathclose}{operators}{"29}
\DeclareMathSymbol{*}{\mathbin}{symbols}{"03} % \ast
\DeclareMathSymbol{+}{\mathbin}{operators}{"2B}
```

```
\DeclareMathSymbol{,}{\mathpunct}{letters}{"3B}
\DeclareMathSymbol{-}{\mathbin}{symbols}{"00}
\DeclareMathSymbol{.}{\mathord}{letters}{"3A}
\DeclareMathSymbol{/}{\mathord}{letters}{"3D}
\DeclareMathSymbol{:}{\mathrel}{operators}{"3A}
\DeclareMathSymbol{;}{\mathpunct}{operators}{"3B}
\DeclareMathSymbol{<}{\mathrel}{letters}{"3C}
\DeclareMathSymbol{=}{\mathrel}{operators}{"3D}
\DeclareMathSymbol{>}{\mathrel}{letters}{"3E}
\DeclareMathSymbol{?}{\mathclose}{operators}{"3F}
\DeclareMathSymbol{[}{\mathopen}{operators}{"5B}
\DeclareMathSymbol{]}{\mathclose}{operators}{"5D}
\DeclareMathSymbol{|}{\mathord}{symbols}{"6A}

```

Should all of the following being activated by default? Probably not.

```
%\DeclareMathSymbol{`\{}{\mathopen}{symbols}{"66}
%\DeclareMathSymbol{`\}}{\mathclose}{symbols}{"67}
%\DeclareMathSymbol{`\\}{\mathord}{symbols}{"6E} % \backslash
\mathcode`\ ="8000 % \space
\mathcode`\'="8000 % ^\prime
\mathcode`\_="8000 % \_
```

38.3.4 Delimitercodes for characters

[to be completed]

Finally, IniTEX sets all `\delcode` values to -1, except `` \delcode`.=0 ``

```
\delcode`\(="028300
\delcode`\)="029301
\delcode`\[="05B302
\delcode`\]="05D303
\delcode`\<="26830A
\delcode`\>="26930B
\delcode`\/="02F30E
\delcode`\|="26A30C
\delcode`\\="26E30F
```

N.B. `{` and `}` should NOT get delcodes; otherwise parameter grouping fails!

38.4 Symbols accessed via control sequences

38.4.1 Greek letters

```
\DeclareMathSymbol{\alpha}{\mathord}{letters}{"0B}
\DeclareMathSymbol{\beta}{\mathord}{letters}{"0C}
\DeclareMathSymbol{\gamma}{\mathord}{letters}{"0D}
\DeclareMathSymbol{\delta}{\mathord}{letters}{"0E}
\DeclareMathSymbol{\epsilon}{\mathord}{letters}{"0F}
\DeclareMathSymbol{\zeta}{\mathord}{letters}{"10}
\DeclareMathSymbol{\eta}{\mathord}{letters}{"11}
```

```
\DeclareMathSymbol{\theta}{\mathord}{letters}{"12}
\DeclareMathSymbol{\iota}{\mathord}{letters}{"13}
\DeclareMathSymbol{\kappa}{\mathord}{letters}{"14}
\DeclareMathSymbol{\lambda}{\mathord}{letters}{"15}
\DeclareMathSymbol{\mu}{\mathord}{letters}{"16}
\DeclareMathSymbol{\nu}{\mathord}{letters}{"17}
\DeclareMathSymbol{\xi}{\mathord}{letters}{"18}
\DeclareMathSymbol{\pi}{\mathord}{letters}{"19}
\DeclareMathSymbol{\rho}{\mathord}{letters}{"1A}
\DeclareMathSymbol{\sigma}{\mathord}{letters}{"1B}
\DeclareMathSymbol{\tau}{\mathord}{letters}{"1C}
\DeclareMathSymbol{\upsilon}{\mathord}{letters}{"1D}
\DeclareMathSymbol{\phi}{\mathord}{letters}{"1E}
\DeclareMathSymbol{\chi}{\mathord}{letters}{"1F}
\DeclareMathSymbol{\psi}{\mathord}{letters}{"20}
\DeclareMathSymbol{\omega}{\mathord}{letters}{"21}
\DeclareMathSymbol{\varepsilon}{\mathord}{letters}{"22}
\DeclareMathSymbol{\vartheta}{\mathord}{letters}{"23}
\DeclareMathSymbol{\varpi}{\mathord}{letters}{"24}
\DeclareMathSymbol{\varrho}{\mathord}{letters}{"25}
\DeclareMathSymbol{\varsigma}{\mathord}{letters}{"26}
\DeclareMathSymbol{\varphi}{\mathord}{letters}{"27}
\DeclareMathSymbol{\Gamma}{\mathalpha}{operators}{"00}
\DeclareMathSymbol{\Delta}{\mathalpha}{operators}{"01}
\DeclareMathSymbol{\Theta}{\mathalpha}{operators}{"02}
\DeclareMathSymbol{\Lambda}{\mathalpha}{operators}{"03}
\DeclareMathSymbol{\Xi}{\mathalpha}{operators}{"04}
\DeclareMathSymbol{\Pi}{\mathalpha}{operators}{"05}
\DeclareMathSymbol{\Sigma}{\mathalpha}{operators}{"06}
\DeclareMathSymbol{\Upsilon}{\mathalpha}{operators}{"07}
\DeclareMathSymbol{\Phi}{\mathalpha}{operators}{"08}
\DeclareMathSymbol{\Psi}{\mathalpha}{operators}{"09}
\DeclareMathSymbol{\Omega}{\mathalpha}{operators}{"0A}
```

38.4.2 Ordinary symbols

```
\DeclareMathSymbol{\aleph}{\mathord}{symbols}{"40}
\def\hbar{{\mathchar'26\mkern-9muh}}
\DeclareMathSymbol{\imath}{\mathord}{letters}{"7B}
\DeclareMathSymbol{\jmath}{\mathord}{letters}{"7C}
\DeclareMathSymbol{\ell}{\mathord}{letters}{"60}
\DeclareMathSymbol{\wp}{\mathord}{letters}{"7D}
\DeclareMathSymbol{\Re}{\mathord}{symbols}{"3C}
\DeclareMathSymbol{\Im}{\mathord}{symbols}{"3D}
\DeclareMathSymbol{\partial}{\mathord}{letters}{"40}
\DeclareMathSymbol{\infty}{\mathord}{symbols}{"31}
\DeclareMathSymbol{\prime}{\mathord}{symbols}{"30}
\DeclareMathSymbol{\emptyset}{\mathord}{symbols}{"3B}
\DeclareMathSymbol{\nabla}{\mathord}{symbols}{"72}
\def\surd{{\mathchar"1270}}
```

```
\DeclareMathSymbol{\top}{\mathord}{symbols}{"3E}
\DeclareMathSymbol{\bot}{\mathord}{symbols}{"3F}
\def\angle{{\vbox{\ialign{$\m@th\scriptstyle##$\crcr
      \not\mathrel{\mkern14mu}\crcr
      \noalign{\nointerlineskip}
      \mkern2.5mu\leaders\hrule \@height.34pt\hfill\mkern2.5mu\crcr}}}}
\DeclareMathSymbol{\triangle}{\mathord}{symbols}{"34}
\DeclareMathSymbol{\forall}{\mathord}{symbols}{"38}
\DeclareMathSymbol{\exists}{\mathord}{symbols}{"39}
\DeclareMathSymbol{\neg}{\mathord}{symbols}{"3A}
    \let\lnot=\neg
\DeclareMathSymbol{\flat}{\mathord}{letters}{"5B}
\DeclareMathSymbol{\natural}{\mathord}{letters}{"5C}
\DeclareMathSymbol{\sharp}{\mathord}{letters}{"5D}
\DeclareMathSymbol{\clubsuit}{\mathord}{symbols}{"7C}
\DeclareMathSymbol{\diamondsuit}{\mathord}{symbols}{"7D}
\DeclareMathSymbol{\heartsuit}{\mathord}{symbols}{"7E}
\DeclareMathSymbol{\spadesuit}{\mathord}{symbols}{"7F}
```

38.4.3 Large Operators

```
\DeclareMathSymbol{\coprod}{\mathop}{largesymbols}{"60}
\DeclareMathSymbol{\bigvee}{\mathop}{largesymbols}{"57}
\DeclareMathSymbol{\bigwedge}{\mathop}{largesymbols}{"56}
\DeclareMathSymbol{\biguplus}{\mathop}{largesymbols}{"55}
\DeclareMathSymbol{\bigcap}{\mathop}{largesymbols}{"54}
\DeclareMathSymbol{\bigcup}{\mathop}{largesymbols}{"53}
\DeclareMathSymbol{\intop}{\mathop}{largesymbols}{"52}
    \def\int{\intop\nolimits}
\DeclareMathSymbol{\prod}{\mathop}{largesymbols}{"51}
\DeclareMathSymbol{\sum}{\mathop}{largesymbols}{"50}
\DeclareMathSymbol{\bigotimes}{\mathop}{largesymbols}{"4E}
\DeclareMathSymbol{\bigoplus}{\mathop}{largesymbols}{"4C}
\DeclareMathSymbol{\bigodot}{\mathop}{largesymbols}{"4A}
\DeclareMathSymbol{\ointop}{\mathop}{largesymbols}{"48}
    \def\oint{\ointop\nolimits}
\DeclareMathSymbol{\bigsqcup}{\mathop}{largesymbols}{"46}
\DeclareMathSymbol{\smallint}{\mathop}{symbols}{"73}
```

38.4.4 Binary symbols

```
\DeclareMathSymbol{\triangleleft}{\mathbin}{letters}{"2F}
\DeclareMathSymbol{\triangleright}{\mathbin}{letters}{"2E}
\DeclareMathSymbol{\bigtriangleup}{\mathbin}{symbols}{"34}
\DeclareMathSymbol{\bigtriangledown}{\mathbin}{symbols}{"35}
\DeclareMathSymbol{\wedge}{\mathbin}{symbols}{"5E}
   \let\land=\wedge
\DeclareMathSymbol{\vee}{\mathbin}{symbols}{"5F}
   \let\lor=\vee
\DeclareMathSymbol{\cap}{\mathbin}{symbols}{"5C}
```

```
\DeclareMathSymbol{\cup}{\mathbin}{symbols}{"5B}
\DeclareMathSymbol{\ddagger}{\mathbin}{symbols}{"7A}
\DeclareMathSymbol{\dagger}{\mathbin}{symbols}{"79}
\DeclareMathSymbol{\sqcap}{\mathbin}{symbols}{"75}
\DeclareMathSymbol{\sqcup}{\mathbin}{symbols}{"74}
\DeclareMathSymbol{\uplus}{\mathbin}{symbols}{"5D}
\DeclareMathSymbol{\amalg}{\mathbin}{symbols}{"71}
\DeclareMathSymbol{\diamond}{\mathbin}{symbols}{"05}
\DeclareMathSymbol{\bullet}{\mathbin}{symbols}{"0F}
\DeclareMathSymbol{\wr}{\mathbin}{symbols}{"6F}
\DeclareMathSymbol{\div}{\mathbin}{symbols}{"04}
\DeclareMathSymbol{\odot}{\mathbin}{symbols}{"0C}
\DeclareMathSymbol{\oslash}{\mathbin}{symbols}{"0B}
\DeclareMathSymbol{\otimes}{\mathbin}{symbols}{"0A}
\DeclareMathSymbol{\ominus}{\mathbin}{symbols}{"09}
\DeclareMathSymbol{\oplus}{\mathbin}{symbols}{"08}
\DeclareMathSymbol{\mp}{\mathbin}{symbols}{"07}
\DeclareMathSymbol{\pm}{\mathbin}{symbols}{"06}
\DeclareMathSymbol{\circ}{\mathbin}{symbols}{"0E}
\DeclareMathSymbol{\bigcirc}{\mathbin}{symbols}{"0D}
\DeclareMathSymbol{\setminus}{\mathbin}{symbols}{"6E}
\DeclareMathSymbol{\cdot}{\mathbin}{symbols}{"01}
\DeclareMathSymbol{\ast}{\mathbin}{symbols}{"03}
\DeclareMathSymbol{\times}{\mathbin}{symbols}{"02}
\DeclareMathSymbol{\star}{\mathbin}{letters}{"3F}
```

38.4.5 Relations

```
\DeclareMathSymbol{\propto}{\mathrel}{symbols}{"2F}
\DeclareMathSymbol{\sqsubseteq}{\mathrel}{symbols}{"76}
\DeclareMathSymbol{\sqsupseteq}{\mathrel}{symbols}{"77}
\DeclareMathSymbol{\parallel}{\mathrel}{symbols}{"6B}
\DeclareMathSymbol{\mid}{\mathrel}{symbols}{"6A}
\DeclareMathSymbol{\dashv}{\mathrel}{symbols}{"61}
\DeclareMathSymbol{\vdash}{\mathrel}{symbols}{"60}
\DeclareMathSymbol{\nearrow}{\mathrel}{symbols}{"25}
\DeclareMathSymbol{\searrow}{\mathrel}{symbols}{"26}
\DeclareMathSymbol{\nwarrow}{\mathrel}{symbols}{"2D}
\DeclareMathSymbol{\swarrow}{\mathrel}{symbols}{"2E}
\DeclareMathSymbol{\Leftrightarrow}{\mathrel}{symbols}{"2C}
\DeclareMathSymbol{\Leftarrow}{\mathrel}{symbols}{"28}
\DeclareMathSymbol{\Rightarrow}{\mathrel}{symbols}{"29}
\def\neq{\not=} \let\ne=\neq
\DeclareMathSymbol{\leq}{\mathrel}{symbols}{"14}
   \let\le=\leq
\DeclareMathSymbol{\geq}{\mathrel}{symbols}{"15}
   \let\ge=\geq
\DeclareMathSymbol{\succ}{\mathrel}{symbols}{"1F}
\DeclareMathSymbol{\prec}{\mathrel}{symbols}{"1E}
\DeclareMathSymbol{\approx}{\mathrel}{symbols}{"19}
```

```
\DeclareMathSymbol{\succeq}{\mathrel}{symbols}{"17}
\DeclareMathSymbol{\preceq}{\mathrel}{symbols}{"16}
\DeclareMathSymbol{\supset}{\mathrel}{symbols}{"1B}
\DeclareMathSymbol{\subset}{\mathrel}{symbols}{"1A}
\DeclareMathSymbol{\supseteq}{\mathrel}{symbols}{"13}
\DeclareMathSymbol{\subseteq}{\mathrel}{symbols}{"12}
\DeclareMathSymbol{\in}{\mathrel}{symbols}{"32}
\DeclareMathSymbol{\ni}{\mathrel}{symbols}{"33}
    \let\owns=\ni
\DeclareMathSymbol{\gg}{\mathrel}{symbols}{"1D}
\DeclareMathSymbol{\ll}{\mathrel}{symbols}{"1C}
\DeclareMathSymbol{\not}{\mathrel}{symbols}{"36}
\DeclareMathSymbol{\leftrightarrow}{\mathrel}{symbols}{"24}
\DeclareMathSymbol{\leftarrow}{\mathrel}{symbols}{"20}
   \let\gets=\leftarrow
\DeclareMathSymbol{\rightarrow}{\mathrel}{symbols}{"21}
   \let\to=\rightarrow
\DeclareMathSymbol{\mapstochar}{\mathrel}{symbols}{"37}
   \def\mapsto{\mapstochar\rightarrow}
\DeclareMathSymbol{\sim}{\mathrel}{symbols}{"18}
\DeclareMathSymbol{\simeq}{\mathrel}{symbols}{"27}
\DeclareMathSymbol{\perp}{\mathrel}{symbols}{"3F}
\DeclareMathSymbol{\equiv}{\mathrel}{symbols}{"11}
\DeclareMathSymbol{\asymp}{\mathrel}{symbols}{"10}
\DeclareMathSymbol{\smile}{\mathrel}{letters}{"5E}
\DeclareMathSymbol{\frown}{\mathrel}{letters}{"5F}
\DeclareMathSymbol{\leftharpoonup}{\mathrel}{letters}{"28}
\DeclareMathSymbol{\leftharpoondown}{\mathrel}{letters}{"29}
\DeclareMathSymbol{\rightharpoonup}{\mathrel}{letters}{"2A}
\DeclareMathSymbol{\rightharpoondown}{\mathrel}{letters}{"2B}
```

```
\def\cong{\mathrel{\mathpalette\@vereq\sim}} % congruence sign
\def\@vereq#1#2{\lower.5\p@\vbox{\lineskiplimit\maxdimen\lineskip-.5\p@
    \ialign{$\m@th#1\hfil##\hfil$\crcr#2\crcr=\crcr}}}
\def\notin{\mathrel{\m@th\mathpalette\c@ncel\in}}
\def\c@ncel#1#2{\m@th\ooalign{$\hfil#1\mkern1mu/\hfil$\crcr$#1#2$}}
\def\rightleftharpoons{\mathrel{\mathpalette\rlh@{}}}
\def\rlh@#1{\vcenter{\m@th\hbox{\ooalign{\raise2pt
         \hbox{$#1\rightharpoonup$}\crcr
       $#1\leftharpoondown$}}}}
\def\doteq{\buildrel\textstyle.\over=}
```

38.4.6 Arrows

```
\def\joinrel{\mathrel{\mkern-3mu}}
\def\relbar{\mathrel{\smash-}} % \smash, because -
                               % has the same height as +
\def\Relbar{\mathrel=}
\DeclareMathSymbol{\lhook}{\mathrel}{letters}{"2C}
   \def\hookrightarrow{\lhook\joinrel\rightarrow}
```

```
\DeclareMathSymbol{\rhook}{\mathrel}{letters}{"2D}
  \def\hookleftarrow{\leftarrow\joinrel\rhook}
\def\bowtie{\mathrel\triangleright\joinrel\mathrel\triangleleft}
\def\models{\mathrel|\joinrel=}
\def\Longrightarrow{\Relbar\joinrel\Rightarrow}
```

LaTeX Change: `\longrightarrow` and `\longleftarrow` redefined to make then robust.

```
\DeclareRobustCommand\longrightarrow
     {\relbar\joinrel\rightarrow}
\DeclareRobustCommand\longleftarrow
     {\leftarrow\joinrel\relbar}
```

```
\def\Longleftarrow{\Leftarrow\joinrel\Relbar}
\def\longmapsto{\mapstochar\longrightarrow}
\def\longleftrightarrow{\leftarrow\joinrel\rightarrow}
\def\Longleftrightarrow{\Leftarrow\joinrel\Rightarrow}
\def\iff{\;\Longleftrightarrow\;}
```

38.4.7 Punctuation symbols

```
\DeclareMathSymbol{\ldotp}{\mathpunct}{letters}{"3A}
\DeclareMathSymbol{\cdotp}{\mathpunct}{symbols}{"01}
\DeclareMathSymbol{\colon}{\mathpunct}{operators}{"3A}
```

LaTeX change: make … work outside math mode too.

This is commented out, since … is now defined in ltoutenc.dtx.

```
%\def\@ldots{\mathinner{\ldotp\ldotp\ldotp}}
%\DeclareRobustCommand\ldots
%          {\relax\ifmmode\@ldots\else\mbox{$\m@th\@ldots\,$}\fi}
```

End of LaTeX change to …

```
\def\cdots{\mathinner{\cdotp\cdotp\cdotp}}
\def\vdots{\vbox{\baselineskip4\p@ \lineskiplimit\z@
    \kern6\p@\hbox{.}\hbox{.}\hbox{.}}}
\def\ddots{\mathinner{\mkern1mu\raise7\p@
    \vbox{\kern7\p@\hbox{.}}\mkern2mu
    \raise4\p@\hbox{.}\mkern2mu\raise\p@\hbox{.}\mkern1mu}}
```

38.4.8 Math accents

```
\DeclareMathAccent{\acute}{\mathalpha}{operators}{"13}
\DeclareMathAccent{\grave}{\mathalpha}{operators}{"12}
\DeclareMathAccent{\ddot}{\mathalpha}{operators}{"7F}
\DeclareMathAccent{\tilde}{\mathalpha}{operators}{"7E}
\DeclareMathAccent{\bar}{\mathalpha}{operators}{"16}
\DeclareMathAccent{\breve}{\mathalpha}{operators}{"15}
\DeclareMathAccent{\check}{\mathalpha}{operators}{"14}
\DeclareMathAccent{\hat}{\mathalpha}{operators}{"5E}
\DeclareMathAccent{\vec}{\mathord}{letters}{"7E}
\DeclareMathAccent{\dot}{\mathalpha}{operators}{"5F}
```

```
\DeclareMathAccent{\widetilde}{\mathord}{largesymbols}{"65}
\DeclareMathAccent{\widehat}{\mathord}{largesymbols}{"62}

```

38.4.9 Radicals

```
\DeclareMathRadical{\sqrtsign}{symbols}{"70}{largesymbols}{"70}
```

38.4.10 Over and under something, etc

```
\def\overrightarrow#1{\vbox{\m@th\ialign{##\crcr
      \rightarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}
      $\hfil\displaystyle{#1}\hfil$\crcr}}}
\def\overleftarrow#1{\vbox{\m@th\ialign{##\crcr
      \leftarrowfill\crcr\noalign{\kern-\p@\nointerlineskip}%
      $\hfil\displaystyle{#1}\hfil$\crcr}}}
\def\overbrace#1{\mathop{\vbox{\m@th\ialign{##\crcr\noalign{\kern3\p@}%
      \downbracefill\crcr\noalign{\kern3\p@\nointerlineskip}%
      $\hfil\displaystyle{#1}\hfil$\crcr}}}\limits}
\def\underbrace#1{\mathop{\vtop{\m@th\ialign{##\crcr
   $\hfil\displaystyle{#1}\hfil$\crcr\noalign{\kern3\p@\nointerlineskip}%
   \upbracefill\crcr\noalign{\kern3\p@}}}}\limits}
```

(quite a waste of tokens, IMHO — Frank)

```
\def\skew#1#2#3{{\muskip\z@#1mu\divide\muskip\z@\tw@ \mkern\muskip\z@
    #2{\mkern-\muskip\z@{#3}\mkern\muskip\z@}\mkern-\muskip\z@}{}}
```

```
\def\rightarrowfill{$\m@th\smash-\mkern-6mu%
  \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
  \mkern-6mu\mathord\rightarrow$}
\def\leftarrowfill{$\m@th\mathord\leftarrow\mkern-6mu%
  \cleaders\hbox{$\mkern-2mu\smash-\mkern-2mu$}\hfill
  \mkern-6mu\smash-$}
\DeclareMathSymbol{\braceld}{\mathord}{largesymbols}{"7A}
\DeclareMathSymbol{\bracerd}{\mathord}{largesymbols}{"7B}
\DeclareMathSymbol{\bracelu}{\mathord}{largesymbols}{"7C}
\DeclareMathSymbol{\braceru}{\mathord}{largesymbols}{"7D}
\def\downbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
  \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru
  \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd$}
\def\upbracefill{$\m@th \setbox\z@\hbox{$\braceld$}%
  \bracelu\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\bracerd
  \braceld\leaders\vrule \@height\ht\z@ \@depth\z@\hfill\braceru$}
```

38.4.11 Delimiters

```
\DeclareMathDelimiter{\lmoustache}   % top from (, bottom from )
   {\mathopen}{largesymbols}{"7A}{largesymbols}{"40}
\DeclareMathDelimiter{\rmoustache}   % top from ), bottom from (
   {\mathclose}{largesymbols}{"7B}{largesymbols}{"41}
\DeclareMathDelimiter{\arrowvert}    % arrow without arrowheads
```

```
   {\mathord}{symbols}{"6A}{largesymbols}{"3C}
\DeclareMathDelimiter{\Arrowvert}    % double arrow without arrowheads
   {\mathord}{symbols}{"6B}{largesymbols}{"3D}
\DeclareMathDelimiter{\Vert}
   {\mathord}{symbols}{"6B}{largesymbols}{"0D}
\let\|=\Vert
\DeclareMathDelimiter{\vert}
   {\mathord}{symbols}{"6A}{largesymbols}{"0C}
\DeclareMathDelimiter{\uparrow}
   {\mathrel}{symbols}{"22}{largesymbols}{"78}
\DeclareMathDelimiter{\downarrow}
   {\mathrel}{symbols}{"23}{largesymbols}{"79}
\DeclareMathDelimiter{\updownarrow}
   {\mathrel}{symbols}{"6C}{largesymbols}{"3F}
\DeclareMathDelimiter{\Uparrow}
   {\mathrel}{symbols}{"2A}{largesymbols}{"7E}
\DeclareMathDelimiter{\Downarrow}
   {\mathrel}{symbols}{"2B}{largesymbols}{"7F}
\DeclareMathDelimiter{\Updownarrow}
   {\mathrel}{symbols}{"6D}{largesymbols}{"77}
\DeclareMathDelimiter{\backslash}    % for double coset G\backslash H
   {\mathord}{symbols}{"6E}{largesymbols}{"0F}
\DeclareMathDelimiter{\rangle}
   {\mathclose}{symbols}{"69}{largesymbols}{"0B}
\DeclareMathDelimiter{\langle}
   {\mathopen}{symbols}{"68}{largesymbols}{"0A}
\DeclareMathDelimiter{\rbrace}
   {\mathclose}{symbols}{"67}{largesymbols}{"09}
\DeclareMathDelimiter{\lbrace}
   {\mathopen}{symbols}{"66}{largesymbols}{"08}
\DeclareMathDelimiter{\rceil}
   {\mathclose}{symbols}{"65}{largesymbols}{"07}
\DeclareMathDelimiter{\lceil}
   {\mathopen}{symbols}{"64}{largesymbols}{"06}
\DeclareMathDelimiter{\rfloor}
   {\mathclose}{symbols}{"63}{largesymbols}{"05}
\DeclareMathDelimiter{\lfloor}
   {\mathopen}{symbols}{"62}{largesymbols}{"04}
```

\lgroup \rgroup \bracevert

There are three plain TEX delimiters which are not fully supported by NFSS, since they partly point into a bold cmr font. Allocating a full symbol font, just to have three delimiters seems a but 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 a symbol fonts first) and modify the delimiter declarations to point to with their small variant to those symbol fonts. (This is done in `oldlfont.dtx` so look there for examples.)

```
\DeclareMathDelimiter{\lgroup} % extensible ( with sharper tips
     {\mathopen}{largesymbols}{"3A}{largesymbols}{"3A}
```

```
\DeclareMathDelimiter{\rgroup} % extensible ) with sharper tips
     {\mathclose}{largesymbols}{"3B}{largesymbols}{"3B}
\DeclareMathDelimiter{\bracevert} % the vertical bar that extends braces
     {\mathord}{largesymbols}{"3E}{largesymbols}{"3E}
```

38.5 Math versions of text commands

These symbols are the math versions of text commands such as `\P`, `\$`, etc.

`\mathparagraph` `\mathsection` `\mathdollar` `\mathsterling` `\mathunderscore`

These math symbols are not in plain TeX.

```
\mathchardef\mathparagraph"027B
\mathchardef\mathsection"0278
\mathchardef\mathdollar"0024
\def\mathsterling{\mathit{\mathchar"7024}}
\def\mathunderscore{\kern0.06em \vbox{\hrule\@width.3em}}
```

`\mathellipsis`

This is plain TeX's `\ldots`.

```
\def\mathellipsis{\mathinner{\ldotp\ldotp\ldotp}}%
```

38.6 Other special functions and parameters

38.6.1 Biggggg

```
\def\big#1{{\hbox{$\left#1\vbox to8.5\p@{}\right.\n@space$}}}
\def\Big#1{{\hbox{$\left#1\vbox to11.5\p@{}\right.\n@space$}}}
\def\bigg#1{{\hbox{$\left#1\vbox to14.5\p@{}\right.\n@space$}}}
\def\Bigg#1{{\hbox{$\left#1\vbox to17.5\p@{}\right.\n@space$}}}
\def\n@space{\nulldelimiterspace\z@ \m@th}
```

38.6.2 The log-like functions

`\operator@font`

The `\operator@font` determines the symbol font used for log-like functions.

```
\def\operator@font{\mathgroup\symoperators}
```

38.6.3 Parameters

```
\thinmuskip=3mu
\medmuskip=4mu plus 2mu minus 4mu
\thickmuskip=5mu plus 5mu
```

This finishes the low-level setup in `fontmath.ltx`.

```
⟨/math⟩
```

File s

preload.dtx

39 Overview

This file contains an number of possible settings for preloading fonts during installation of NFSS2 (which is used by $\LaTeX\,2_\varepsilon$). 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 `lfonts.tex`
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.

40 Customization

You can customize the preloaded fonts in your $\LaTeX\,2_\varepsilon$ 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* $\LaTeX\,2_\varepsilon$ sytems.

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 `fontdef.dtx`). Adding an extra encoding to the format might produce non-portable documents, thus this should be avoided if possible.

41 Module switches for the DOCSTRIP program

The DOCSTRIP will generate the above file from this source using the following module directives:

driver	produce a documentation driver file
preload	produce a preload... file
cm	for OT1 encoded Computer Modern
dc	for T1 encoded Computer Modern
min	produce minimal subset
xpt	produce 10pt preloads
xipt	produce 11pt preloads
xiipt	produce 12pt preloads
ori	produce preloads similar to old `lfonts.tex`
tex	produce preload.ltx

A typical DOCSTRIP command file would then have entries like:

```
\generateFile{preload.min}{t}{\from{preload.dtx}{preload,min}}
```

for generating preload files.

42 A driver for this document

The next bit of code contains the documentation driver file for TEX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the DOCSTRIP program.

```
⟨*driver⟩
\documentclass{ltxdoc}
%\OnlyDescription  % comment out for implementation details
\begin{document}
   \DocInput{preload.dtx}
\end{document}
⟨/driver⟩
```

43 The code

We begin by loading the math extension font (cmex10) and the LATEX line and circle fonts. It is necessary to do this explicitly since these are used by `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.

```
\expandafter\font\csname OMX/cmex/m/n/10\endcsname=cmex10\relax
\font\tenln  =line10   \font\tenlnw  =linew10\relax
\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.

```
⟨−tex⟩%*******************************************
⟨−tex⟩% Start any modification below this point **
⟨−tex⟩%*******************************************
⟨−tex⟩
%%
%% Computer Modern Roman:
%%---------------------
⟨*ori⟩
\DeclarePreloadSizes{OT1}{cmr}{m}{n}
        {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74,24.88}
\DeclarePreloadSizes{OT1}{cmr}{bx}{n}{9,10,10.95,12,14.4,17.28}
\DeclarePreloadSizes{OT1}{cmr}{m}{sl}{10,10.95,12}
\DeclarePreloadSizes{OT1}{cmr}{m}{it}{7,8,9,10,10.95,12}
⟨/ori⟩
⟨+xpt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{5,7,10}
⟨+xpt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{5,7,10}
⟨+xipt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,10.95}
⟨+xipt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,10.95}
⟨+xiipt & cm⟩ \DeclarePreloadSizes{OT1}{cmr}{m}{n}{6,8,12}
⟨+xiipt & dc⟩ \DeclarePreloadSizes{T1}{cmr}{m}{n}{6,8,12}
%%
%% Computer Modern Sans:
%%--------------------
⟨+ori⟩ \DeclarePreloadSizes{OT1}{cmss}{m}{n}{10,10.95,12}
%%
%% Computer Modern Typewriter:
%%--------------------------
⟨+ori⟩ \DeclarePreloadSizes{OT1}{cmtt}{m}{n}{9,10,10.95,12}
%%
%% Computer Modern Math:
%%--------------------
⟨*ori⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}
         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}
         {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
⟨/ori⟩
```

The math fonts are the same for both DC and CM fonts. So far there isn't an agreed on standard.

```
⟨*xpt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{5,7,10}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{5,7,10}
⟨/xpt⟩
⟨*xipt⟩
\DeclarePreloadSizes{OML}{cmm}{m}{it}{6,8,10.95}
\DeclarePreloadSizes{OMS}{cmsy}{m}{n}{6,8,10.95}
⟨/xipt⟩
```

```
%%
%% LaTeX symbol fonts:
%%--------------------
⟨*ori⟩
\DeclarePreloadSizes{U}{lasy}{m}{n}
          {5,6,7,8,9,10,10.95,12,14.4,17.28,20.74}
⟨/ori⟩
⟨/preload⟩
```

File t

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.

44 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 LaTeX3 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 puttiing 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.

[7] Any package that changes the `\catcode` of a character inside `\nocorrlist` must then explicitly reset the list. Otherwise the changed character will no longer be recognized by the suppression algorithm.

Command	*Corresponds to*	*Action*
`\textrm{..}`	`\rmfamily`	Typeset argument in roman family
`\textsf{..}`	`\sffamily`	Typeset argument in sans serif family
`\texttt{..}`	`\ttfamily`	Typeset argument in `typewriter` family
`\textmd{..}`	`\mdseries`	Typeset argument in medium series
`\textbf{..}`	`\bfseries`	Typeset argument in **bold** series
`\textup{..}`	`\upshape`	Typeset argument in normal shape
`\textit{..}`	`\itshape`	Typeset argument in *italic* shape
`\textsl{..}`	`\slshape`	Typeset argument in *slanted* shape
`\textsc{..}`	`\scshape`	Typeset argument in SMALL CAPS shape
`\emph{..}`	`\em`	Typeset argument *emphasized*

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.

```
\newcommand \nocorrlist {,.}
```

It is best to declare the most often used characters first, because this will make the processing slightly faster. For example,

```
\emph{When using the \NFSS{} high-level commands,
the \emph{proper} use of italic corrections is
automatically taken care of}. Only
\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 LaTeX 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 LaTeX'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 `\/`.

45 The implementation

`\DeclareTextFontCommand` This is the creator function for `\text..` commands. It gives a warning if `\foo` or `\fragfoo` is already defined.

In math mode it simply puts the font declaration and text into a box (possibly an automagically sized one).

Otherwise it first scans the text to see where `\nocorr` occurs within it. This sets the `\check@ic` commands to do what is necessary concerning the italic correction at both ends.

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.

This now checks for empty contents of text command and optimises this case.

```
⟨*2ekernel⟩
\def \DeclareTextFontCommand #1#2{%
  \DeclareRobustCommand#1[1]{%
    \ifmmode
      \nfss@text{#2##1}%
    \else
      \leavevmode
```

```
      {\text@command{##1}%
       #2\check@icl ##1\check@icr
       \expandafter}%
    \fi
                            }%
}
```

\textrm \textsf \texttt \textnormal

Now we define the `\text`⟨*family*⟩ commands in terms of the above; `\texttt` does not look very nice!

```
\DeclareTextFontCommand{\textrm}{\rmfamily}
\DeclareTextFontCommand{\textsf}{\sffamily}
\DeclareTextFontCommand{\texttt}{\ttfamily}
\DeclareTextFontCommand{\textnormal}{\normalfont}
```

\textbf \textmd

For the series attribute:

```
\DeclareTextFontCommand{\textbf}{\bfseries}
\DeclareTextFontCommand{\textmd}{\mdseries}
```

\textit \textsl \textsc \textup

And for the shapes:

```
\DeclareTextFontCommand{\textit}{\itshape}
\DeclareTextFontCommand{\textsl}{\slshape}
\DeclareTextFontCommand{\textsc}{\scshape}
\DeclareTextFontCommand{\textup}{\upshape}
```

\emph

Finally we have the `\em` font change declaration of LaTeX. The corresponding definition with argument is

```
\DeclareTextFontCommand{\emph}{\em}
```

\nocorr

This is just a label, so it does nothing; it should also be unexpandable.

```
\let \nocorr \relax
```

\text@command \check@nocorr@ \check@icl \check@icr

This checks for a `\nocorr` as the first token in its argument and also for one in any other position not protected within braces (the latter is treated as if it were at the end of the argument).

Is this the correct action in the 'empty' case? It is efficient but typographically it is, strictly, incorrect!

```
\def \text@command #1{%
  \def \reserved@a {#1}%
  \ifx \reserved@a \@empty
    \let \check@icl \@empty
    \let \check@icr \@empty
  \else
```

`\space` is a reserved word in LaTeX or actually already in plain TeX. 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.

```
%     \def \reserved@b { }%
%     \ifx \reserved@a \reserved@b
    \ifx \reserved@a \space
```

```
      \let \check@icl \@empty
      \let \check@icr \@empty
    \else
      \check@nocorr@ #1\nocorr\@nil
    \fi
  \fi
}
\def \check@nocorr@ #1#2\nocorr#3\@nil {%
```

The two checks are initialised here to their values in the normal case.

```
  \let \check@icl \maybe@ic
  \def \check@icr {\aftergroup \maybe@ic}%
  \def \reserved@a {\nocorr}%
  \def \reserved@b {#1}%
  \def \reserved@c {#3}%
  \ifx \reserved@a \reserved@b
    \ifx \reserved@c \@empty
```

In this case there is a `\nocorr` at the start but not at the end, so `\check@icl` should be empty.

```
      \let \check@icl \@empty
    \else
```

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

```
      \let \check@icl \@empty
      \let \check@icr \@empty
    \fi
  \else
    \ifx \reserved@c \@empty
```

In this case there is no `\nocorr` anywhere, so we need to check for an italic corection at both the beginning and the end.

```
    \else
```

In this case there is no `\nocorr` at the start but there is one elsewhere, so no `\aftergroup` is needed.

```
      \let \check@icr \@empty
    \fi
  \fi
}
```

`\maybe@ic`
`\maybe@ic@`

These macros implement the italic correction.

```
\def \maybe@ic {\futurelet\@let@token\maybe@ic@}
\def \maybe@ic@ {%
```

We first check to see if the current font is also sloped. (But do not forget the message Rainer sent about an upright font with non-zero slope!)

```
  \ifdim \fontdimen\@ne\font>\z@
  \else
    \@tempswatrue
```

It would be possible, but probably not worthwhile, to continue the forward scan beyond any closing braces.

```
    \expandafter\@tfor\expandafter\reserved@a\expandafter:\expandafter=%
        \nocorrlist
```

We have to hide the `\@let@token` in the macro `\test@next` rather than testing it directly in the loop since it might be `\let` to a `\fi` or `\else`, which would result in chaos.

```
    \do \test@next
```

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`.

```
    \if@tempswa \sw@slant \fi
  \fi
}
```

`\test@next`

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.

```
\def \test@next {%
  \expandafter\let\expandafter\reserved@b\expandafter=\reserved@a\relax
  \ifx\reserved@b\@let@token
```

If they are the same we jump out of the loop.

```
    \@tempswafalse
    \let\@nextwhile\@fornoop
    \expandafter\@gobblefour
  \fi
}
```

`\sw@slant`
`\fix@penalty`

The definition of the mysterious `\sw@slant` command is as follows.

```
\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 sloped 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.

```
  \ifdim \lastskip=\z@
    \fix@penalty
  \else
    \skip@ \lastskip
    \unskip
```

```
    \fix@penalty
    \hskip \skip@
  \fi
}
```

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 LaTeX code?

```
\def \fix@penalty {%
  \ifnum \lastpenalty=\z@
    \@@italiccorr
  \else
    \count@ \lastpenalty
    \unpenalty
    \@@italiccorr
    \penalty \count@
  \fi
}
```

`\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.

```
\def \nocorrlist {,.}
```

`\nfss@text` This command will by default behave like a LaTeX `\mbox` but may be redefined by packages such as `amstext.sty` to be a bit cleverer.

```
\ifx \nfss@text\undefined
  \def \nfss@text {\leavevmode\hbox}
\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}
```

```
\def \DeclareOldFontCommand #1#2#3{%
  \DeclareRobustCommand #1{\@fontswitch {#2}{#3}}%
}
```

\@fontswitch
\@@math@egroup
\@@math@egroup

These two commands actually do the necessary tests and declarative font- or alphabet-changing.

```
\def \@fontswitch #1#2{%
  \ifmmode
     \let \math@bgroup \relax
     \def \math@egroup {\let \math@bgroup \@@math@bgroup
                        \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).

```
     #2\relax
  \else
     #1%
  \fi
}
\let \@@math@bgroup \math@bgroup
\let \@@math@egroup \math@egroup
```

These commands are available only in the preamble.

```
\@onlypreamble \DeclareTextFontCommand
\@onlypreamble \DeclareOldFontCommand
```

46 Initialization

\normalsize

This is defined to produce an error.

```
\def\normalsize{%
  \@latex@error {The font size command \protect\normalsize\space
                 is not defined:\MessageBreak
                 there is probably something wrong with
                 the class file}\@eha
}
⟨/2ekernel⟩
```

File u
ltcounts.dtx

47 Counters and Lengths

Commands for defining and using counters. This file defines:

\newcounter To define a new counter.

\setcounter To set the value of counters.

\addtotcounter Increase the counter #1 by the number #2.

\steptcounter Increase a counter by one.

\refsteptcounter Increase a counter by one, also setting the value used by `\label`.

\value For accessing the value of the counter as a TeX number (as opposed to `\the`⟨*counter*⟩ which expands to the *printed* representation of ⟨*counter*⟩)

\arabic `\arabic{`*counter*`}`: 1, 2, 3, ...

\roman `\roman{`*counter*`}`: i, ii, ii, ...

\Roman `\Roman{`*counter*`}`: I, II, II, ...

\alph `\alph{`*counter*`}`: a, b, c, ...

\Alph `\Alph{`*counter*`}`: A, B, C, ...

\fnsymbol `\fnsymbol{`*counter*`}`: $*$, †, ‡, ...

```
⟨*2ekernel⟩
\message{env. counters,}
```

`\subsection{`Environment Counter Macros`}`

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

`\c@foo` : Contains the counter's numerical value.
It is defined by `\newcount\foocounter.`

`\thefoo` : Macro that expands to the printed value of `\foocounter.`
For example, if sections are numbered within chapters, and section headings look like
Section II-3. The Nature of Counters
then `\thesection` might be defined by:

```
\def\thesection
   {\@Roman{\c@chapter}-\@arabic{\c@section}}
```

`\p@foo` : Macro that expands to a printed 'reference prefix' of counter foo. Any `\ref` to a value created by counter foo will produce the expansion of `\p@foo\thefoo` when the the `\label` command is executed.

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.

`\cl@foo` : List of counters to be reset when foo stepped.
Has format
`\@elt`{countera}`\@elt`{counterb}`\@elt`{counterc}.

The following commands are used to define and modify counters.

`\setcounter`{FOO}{VAL} : Globally sets `\foocounter` equal to VAL.

`\addtocounter`{FOO}{VAL}: Globally increments `\foocounter` by VAL.

`\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.

`\value`{CTR} :
produces the value of counter CTR, for use with
a `\setcounter` or `\addtocounter` command.

`\stepcounter`{FOO} : Globally increments counter `\c@FOO`
and resets all subsidiary counters.

`\refstepcounter`{FOO} : Same a `\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.

`\@addtoreset`{FOO}{BAR} :
Adds counter FOO to the list of counters `\cl@BAR` to be reset
when counter bar is stepped.

NUMBERING MACROS:

`\arabic`{COUNTER} :
Representation of COUNTER as arabic numerals.
Changed 29 Apr 86 to make it print the obvious thing
it COUNTER not positive.

`\roman{COUNTER}` : Representation of COUNTER as lower-case Roman numerals.
`\Roman{COUNTER}` : Representation of COUNTER as upper-case Roman numerals.
`\alph{COUNTER}` : Representation of COUNTER as a lower-case letter: 1 = a, 2 = b, etc.
`\Alph{COUNTER}` : Representation of COUNTER as an upper-case letter: 1 = A, 2 = B, etc.
`\fnsymbol{COUNTER}` : Representation of COUNTER as a footnote symbol: 1 = *, 2 = `\dagger`, etc.

THE ABOVE ARE IMPLEMENTED IN TERMS OF THE FOLLOWING:

`\@arabic\FOOcounter` : Representation of `\FOOcounter` as arabic numerals.
`\@roman\FOOcounter` : Representation of `\FOOcounter` as lower-case Roman numerals.
`\@Roman\FOOcounter` : Representation of `\FOOcounter` as upper-case Roman numerals.
`\@alph\FOOcounter` : Representation of `\FOOcounter` as a lower-case letter: 1 = a, 2 = b, etc.
`\@Alph\FOOcounter` : Representation of `\FOOcounter` as an upper-case letter: 1 = A, 2 = B, etc.
`\@fnsymbol\FOOcounter` : Representation of `\FOOcounter` as a footnote symbol.

```
\def\setcounter#1#2{\@ifundefined{c@#1}{\@nocounterr{#1}}%
{\global\csname c@#1\endcsname#2\relax}}

\def\addtocounter#1#2{\@ifundefined{c@#1}{\@nocounterr{#1}}%
{\global\advance\csname c@#1\endcsname #2\relax}}

\def\newcounter#1{\expandafter\@ifdefinable \csname c@#1\endcsname
    {\@definecounter{#1}}\@ifnextchar[{\@newctr{#1}}{}}

\def\value#1{\csname c@#1\endcsname}

\def\@newctr#1[#2]{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}{\@addtoreset{#1}{#2}}}

\def\stepcounter#1{\addtocounter{#1}\@ne
    \begingroup\let\@elt\@stpelt \csname cl@#1\endcsname\endgroup}
```

```
\def\@stpelt#1{\global\csname c@#1\endcsname \z@}
```

```
\def\cl@@ckpt{\@elt{page}}
```

\@definecounter

```
\def\@definecounter#1{\expandafter\newcount\csname c@#1\endcsname
     \setcounter{#1}\z@
     \global\expandafter\let\csname cl@#1\endcsname\@empty
     \@addtoreset{#1}{@ckpt}%
     \global\expandafter\let\csname p@#1\endcsname\@empty
     \expandafter
     \gdef\csname the#1\expandafter\endcsname\expandafter
          {\expandafter\@arabic\csname c@#1\endcsname}}
```

```
\def\@addtoreset#1#2{\expandafter\@cons\csname cl@#2\endcsname {{#1}}}
```

\arabic \roman \Roman \alph \Alph \fnsymbol

Numbering commands for definitions of `\theCOUNTER` and `\list` arguments.

`\fnsymbol` produces the standard footnoting symbols: asterisk, dagger, etc.

All commands can now be used in text and math mode.

```
\def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
\def\roman#1{\expandafter\@roman\csname c@#1\endcsname}
\def\Roman#1{\expandafter\@Roman\csname c@#1\endcsname}
\def\alph#1{\expandafter\@alph\csname c@#1\endcsname}
\def\Alph#1{\expandafter\@Alph\csname c@#1\endcsname}
\def\fnsymbol#1{\expandafter\@fnsymbol\csname c@#1\endcsname}
```

\@arabic \@roman \@Roman \@alph \@Alph

Internal implementation of the above commands.

```
\def\@arabic#1{\number #1}  %% changed 29 Apr 86
\def\@roman#1{\romannumeral #1}
\def\@Roman#1{\expandafter\uppercase\expandafter{\romannumeral #1}}
\def\@alph#1{%
  \ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or
   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
    y\or z\else\@ctrerr\fi}
\def\@Alph#1{%
  \ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or
   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
    Y\or Z\else\@ctrerr\fi}
```

\@fnsymbol

Typesetting old fashioned footnote symbols. This can be done both in text or math mode now.

```
\def\@fnsymbol#1{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or
   \mathchar "278\or \mathchar "27B\or \|\or **\or \dagger\dagger
   \or \ddagger\ddagger \else\@ctrerr\fi}}
```

⟨/2ekernel⟩

File v
ltpageno.dtx

48 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.

```
⟨*2ekernel⟩
\message{page nos.,}

\countdef\c@page=0 \c@page=1
\def\cl@page{}
\def\pagenumbering#1{%
  \global\c@page \@ne \gdef\thepage{\csname @#1\endcsname
   \c@page}}

⟨/2ekernel⟩
```

File w

ltxref.dtx

49 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.

49.1 Cross Referencing

```
⟨*2ekernel⟩
\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
```

```
\newif\ifG@refundefined
\def\G@refundefinedtrue{\global\let\ifG@refundefined\iftrue}
\def\G@refundefinedfalse{\global\let\ifG@refundefined\iffalse}
```

RmS 91/10/25: added a few extra `\reset@font`,
as suggested by Bernd Raichle
RmS 92/08/14: made `\ref` and `\pageref` robust
RmS 93/09/08: Added setting of @refundefined switch.

`\ref` `\pageref` `\@setref` Referencing a `\label`.

```
\def\@setref#1#2#3{%
  \ifx#1\relax
   \protect\G@refundefinedtrue
   \nfss@text{\reset@font\bfseries ??}%
   \@latex@warning{Reference `#3' on page \thepage \space
             undefined}%
  \else
   \expandafter#2#1\null
  \fi}
\def\ref#1{\expandafter\@setref\csname r@#1\endcsname\@firstoftwo{#1}}
\def\pageref#1{\expandafter\@setref\csname r@#1\endcsname
                                   \@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`.

```
\def\@newl@bel#1#2#3{%
  \@ifundefined{#1@#2}%
    \relax
    {\global\@multiplelabelstrue
     \@latex@warning@no@line{Label `#2' multiply defined}}%
  \global\@namedef{#1@#2}{#3}}
```

```
\def\newlabel{\@newl@bel r}
```

```
\@onlypreamble\newlabel
\@onlypreamble\@newl@bel
```

`\if@multiplelabels` This switch is set true if at least on label is defined more than once. It will be check by the `\enddocument` command.

```
\newif\if@multiplelabels
```

`\label` and `\refstepcounter` changed to allow `\protect`'ed commands to work properly. For example,

```
\def\thechapter{\protect\foo{\arabic{chapter}.\roman{section}}}
```

will cause a `\label{bar}` command to define `\ref{bar}` to expand to something like `\foo{4.d}`. Change made 20 Jul 88.

```
\def\label#1{\@bsphack
  \protected@write\@auxout{}%
         {\string\newlabel{#1}{{\@currentlabel}{\thepage}}}%
  \@esphack}
```

```
\def\refstepcounter#1{\stepcounter{#1}%
    \protected@edef\@currentlabel
       {\csname p@#1\endcsname\csname the#1\endcsname}%
}
```

For `\label` commands that come before any environment

```
\def\@currentlabel{}
```

⟨/2ekernel⟩

File x

ltlength.dtx

50 Lengths

\newlength \setlength \addtolength \settowidth \settoheight \settodepth

Declare `#1` to be a new length command.

Set the length command, `#1`, to the value `#2`.

Increase the value of the length command, `#1`, by the value `#2`.

Set the length, `#1` to the width of a box containing `#2`.

Set the length, `#1` to the height of a box containing `#2`.

Set the length, `#1` to the depth of a box containing `#2`.

```
⟨*2ekernel⟩
\message{lengths,}
```

\newlenth

```
\def\newlength#1{\@ifdefinable#1{\newskip#1}}
```

\setlenth

```
\def\setlength#1#2{#1#2\relax}
```

\addtolenth

`\relax` added 24 Mar 86

```
\def\addtolength#1#2{\advance#1 #2\relax}
```

\settoheight \settodepth \settowidth \@settodim

The obvious analogs of `\settowidth`.

```
\def\@settodim#1#2#3{\setbox\@tempboxa\hbox{{#3}}#2#1\@tempboxa
```

Clear the memory afterwards (which might be a lot).

```
       \setbox\@tempboxa\box\voidb@x}
\def\settoheight{\@settodim\ht}
\def\settodepth {\@settodim\dp}
\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 calulating a dimension.

```
\def\@settopoint#1{\divide#1\p@\multiply#1\p@}
⟨/2ekernel⟩
```

File y
ltmiscen.dtx

51 Miscellaneous Environments

This section implements the basic environment mechanism, and also a few specific environments including `document`, The math environments and related commands, the 'flushing' environments, (`center`, `flushleft`, `flushright`), and `verbatim`.

```
1 ⟨*2ekernel⟩
2 \message{environments,}
```

51.1 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 \global\@ignoretrue.

\@currenvir : the name of the current environment.  Initialized to
              'document' to make \end{document} work right.

\@preamblecmds : a list of commands that can be used only in the
                 preamble (before the \begin{document}), in the
                 form  \do \CMDA \do \CMDB ... .  These commands
                 are redefined to \@notprerr by \begin{document}
                 to save space.  They include the following:
                    \document \documentstyle \@documentstyle
                    \@options \@preamblecmds \@optionlist
                    \@optionfiles \nofiles \includeonly
\makeindex
                    \makeglossary
                 The document style can add any other commands to
                 this list by
                   \def\do{\noexpand\do\noexpand}
                   \edef\@preamblecmds{\@preamblecmds \do ...}

NOTE: \@@end is defined to be the \end command of TeX82.

\enddocument is the user's command for ending the manuscript file.
```

`\stop` is a panic button – to end TeX in the middle.

```
\enddocument ==
  BEGIN
   \@checkend{document}    %% checks for unmatched \begin
   \clearpage
   \begingroup
     if @filesw = true
       then   close file @mainaux
              if G@refundefined = true
               then LaTeX Warning: 'There are undefined references.' fi
              if @multiplelabels = true
                then LaTeX Warning:
                    'One or more label(s) multiply defined.'
                else
                \@setckpt {ARG1}{ARG2} == null
                \newlabel{LABEL}{VAL} ==
                    BEGIN
                      \reserved@a == VAL
                      if def(\reserved@a) = def(\r@LABEL)
                        else @tempswa := true          fi
                    END
                \bibcite{LABEL}{VAL} == null
                    BEGIN
                      \reserved@a == VAL
                      if def(\reserved@a) = def(\g@LABEL)
                        else @tempswa := true          fi
                    END
                @tempswa := false
                make @ a letter
                \input \jobname.AUX
                if @tempswa = true
                  then LaTeX Warning: 'Label may have changed.
                                 Rerun to get cross-references right.'
     fi     fi     fi
   \endgroup
   finish up
  END

 \@writefile{EXT}{ENTRY} ==
     if tf@EXT undefined
       else \write\tf@EXT{ENTRY}
     fi
```

```
\def\@currenvir{document}
```

```
\@onlypreamble\document
\@onlypreamble\documentstyle
\@onlypreamble\@documentstyle

\@onlypreamble\@options
\@onlypreamble\@preamblecmds
\@onlypreamble\@optionlist
\@onlypreamble\@optionfiles

\@onlypreamble\nofiles
\@onlypreamble\includeonly
\@onlypreamble\makeindex
\@onlypreamble\makeglossary
```

```
\newif\if@ignore
```

\enddocument

```
\def\enddocument{%
```

The `\end{document}` hook is executed first. If necessary it can contain a `\clearpage` to output dangling floats first. In this position it can also contain something like `\end{foo}` so that the whole document effectively starts and ends with some special environment. However, this must be used with care, eg if two applications would use this without knowledge of each other the order of the environments will be wrong after all.

```
   \@enddocumenthook
   \@checkend{document}%
   \clearpage
   \begingroup
     \if@filesw
       \immediate\closeout\@mainaux
       \def\@setckpt##1##2{}%
       \def\newlabel{\@testdef r}%
       \def\bibcite{\@testdef b}%
       \@tempswafalse
       \makeatletter \input\jobname.aux
     \fi
```

```
     \@dofilelist
```

First we check for font size substitution bigger than `\fontsubfuzz`. The `\relax` is necessary because this is a macro not a register.

```
     \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.

```
       \@font@warning{Size substitutions with differences\MessageBreak
                  up to \font@submax\space have occured.\@gobbletwo}%
     \fi
```

```
    \ifG@defaultsubs
      \@font@warning{Some font shapes were not available, defaults
                     substituted.\@gobbletwo}%
    \fi
    \ifG@refundefined
      \@latex@warning@no@line{There were undefined references}%
    \fi
```

If a label is defined more than once, `\@tempswa` will always be true and thus produce a "Label(s) may ..." warning. But since a rerun will not solve that problem (unless one uses a package like `varioref` that generates labels on the fly), we suppress this message.

```
    \if@filesw
      \if@multiplelabels
        \@latex@warning@no@line{There were multiply-defined labels}%
      \else
        \if@tempswa
          \@latex@warning@no@line{Label(s) may have changed.
              Rerun to get cross-references right}%
        \fi
      \fi
    \fi
  \endgroup
  \deadcycles\z@\@@end}
```

```
\def\@testdef #1#2#3{%
  \def\reserved@a{#3}\expandafter \ifx \csname #1@#2\endcsname
 \reserved@a  \else \@tempswatrue \fi}
```

`\changes`{v1.0l}{1994/05/20}{Added correct setting of `\cs`{protect}.}
`\changes`{v1.0t}{1994/11/04}{Removed setting of `\cs`{protect}. ASAJ.}

```
\long\def\@writefile#1#2{%
  \@ifundefined{tf@#1}{}%
    {\@temptokena{#2}
     \immediate\write\csname tf@#1\endcsname{\the\@temptokena}%
    }%
}
% \long added 8 Feb 90, as suggested by Chris Rowley
```

```
\def\stop{\clearpage\deadcycles\z@\let\par\@@par\@@end}
```

```
\everypar{\@nodocument} %% To get an error if text appears before the
\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
   FI                           %% immediately following text
   IF @ignore = T
     THEN @ignore :=G F
          \ignorespaces
   FI
  END

 \@checkend{NAME} ==
  BEGIN
   IF \@currenvir = NAME
     ELSE \@badend{NAME}
   FI
  END
```

RmS 92/03/18: changed `\@ignoretrue` to `\@ignorefalse` (as documented)
RmS 92/08/24: added code to `\begin` to remember line number. Used by `\@badend` to display position of non-matching `\begin`.

```
\def\begin#1{%
  \@ifundefined{#1}%
    {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
    {\def\reserved@a{\def\@currenvir{#1}%
     \edef\@currenvline{\on@line}%
     \csname #1\endcsname}}%
  \global\@ignorefalse %% \global...added 2 May 90
```

```
  \begingroup\@endpefalse\reserved@a}
\def\end#1{%
  \csname end#1\endcsname\@checkend{#1}%
  \expandafter\endgroup\if@endpe\@doendpe\fi
  \if@ignore\global\@ignorefalse\ignorespaces\fi}
\def\@checkend#1{\def\reserved@a{#1}\ifx
      \reserved@a\@currenvir \else\@badend{#1}\fi}
```

`\@currenvline` We do need a default value for `\@currenvline` on top-level since the document environment cancels the brace group. This means that a missmatch with `\begin {document}` will not produce a line number. Thus the outer default must be `\@empty` or we will end up with two spaces.

```
\let\@currenvline\empty
```

51.2 Center, Flushright, Flushleft

```
\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` := flushglue. Every environment, like the list
environments,
that set `\rightskip` to its 'normal' value set it to `\@rightskip`
`\changes{v1.0h}{1994/05/03}{\cs{@badcrerr} replaced by \cs{@nolnerr}}`

```
\def\@centercr{\ifhmode \unskip\else \@nolnerr\fi
       \par\@ifstar{\penalty \@M\@xcentercr}\@xcentercr}
\def\@xcentercr{\addvspace{-\parskip}\@ifnextchar
    [\@icentercr\ignorespaces}
```

```
\def\@icentercr[#1]{\vskip #1\ignorespaces}
```

center We use `\relax` to prevent `\item` scanning too far.

```
\def\center{\trivlist \centering\item\relax}
\def\centering{\let\\\@centercr\rightskip\@flushglue\leftskip\@flushglue
\parindent\z@\parfillskip\z@skip}
\def\endcenter{\endtrivlist}
```

```
\newskip\@rightskip \@rightskip \z@skip
```

flushleft We use `\relax` to prevent `\item` scanning too far.

```
\def\flushleft{\trivlist \raggedright\item\relax}
\def\raggedright{%
  \let\\\@centercr\@rightskip\@flushglue \rightskip\@rightskip
  \leftskip\z@skip
  \parindent\z@}
\def\endflushleft{\endtrivlist}
```

flushright We use `\relax` to prevent `\item` scanning too far.

```
\def\flushright{\trivlist \raggedleft\item\relax}
\def\raggedleft{\let\\\@centercr\rightskip\z@skip\leftskip\@flushglue
  \parindent\z@\parfillskip\z@skip}
\def\endflushright{\endtrivlist}
```

```
\message{verbatim,}
```

51.3 Verbatim

The verbatim environment uses the fixed-width `\ttfamily` font, turns blanks into spaces, starts a new line for each carrige 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 is the same, except that spaces print as the TeXbook's space character instead of as blank spaces.

```
{\catcode`\^^M=13 \gdef\@gobblecr{\@ifnextchar
\@gobble\ignorespaces}}
```

```
{\catcode`\ =\active%
\gdef\@vobeyspaces{\catcode`\ \active\let \@xobeysp}}
```

Definition of `\@xobeysp` chaned on 19 Nov 86 from
`\def\@xobeysp{\leavevmode{} }`
to prevent line breaks at spaces. Change suggested by
Nelson Beebe

```
\def\@xobeysp{\leavevmode\penalty10000\ }
```

```
\begingroup \catcode `|=0 \catcode `[= 1
\catcode`]=2 \catcode `\{=12 \catcode `\}=12
\catcode`\\=12 |gdef|@xverbatim#1\end{verbatim}[#1|end[verbatim]]
|gdef|@sxverbatim#1\end{verbatim*}[#1|end[verbatim*]]
|endgroup
```

`\@verbatim` Real start of verbatim environment We use `\relax` to prevent `\item` scanning too far.

```
\def\@verbatim{\trivlist \item\relax
  \if@minipage\else\vskip\parskip\fi
  \leftskip\@totalleftmargin\rightskip\z@skip
  \parindent\z@\parfillskip\@flushglue\parskip\z@skip
```

Added `\@@par` to clear possible `\parshape` definition from a surrounding list (the verbatim guru says).

```
  \@@par
  \@tempswafalse
  \def\par{%
    \if@tempswa
```

A `\leavevmode` added: needed if, for example, a blank verbatim line is the first thing in a list item (wow!).

```
      \leavevmode \null \@@par\penalty\interlinepenalty
    \else
      \@tempswatrue
      \ifhmode\@@par\penalty\interlinepenalty\fi
    \fi}%
```

To allow customization we hide the font used in a separate macro.

```
  \obeylines \verbatim@font \@noligs
  \let\do\@makeother \dospecials
```

To avoid a breakpoint after the labels box, we remove the penalty put there by the list macros: another use of `\unpenalty`!

```
  \everypar \expandafter{\the\everypar \unpenalty}%
}
```

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

```
\def\verbatim{\@verbatim \frenchspacing\@vobeyspaces \@xverbatim}
\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.

```
\def\verbatim@font{\normalfont\ttfamily}
```

```
\@namedef{verbatim*}{\@verbatim\@sxverbatim}
\expandafter\let\csname endverbatim*\endcsname =\endverbatim
```

```
\def\@makeother#1{\catcode`#1=12\relax}
```

```
\let\verb@balance@group\@empty

\def\verb@egroup{\global\let\verb@balance@group\@empty\egroup}

\begingroup
  \obeylines%
  \gdef\verb@eol@error{\obeylines%
    \def^^M{\verb@egroup\@latex@error{%
             \noexpand\verb ended by end of line}\@ehc}}%
\endgroup
```

\verb Typesetting a small piece verbatim.

```
\def\verb{\relax\ifmmode\hbox\else\leavevmode\null\fi
  \bgroup \@noligs
    \verbatim@font \verb@eol@error \let\do\@makeother \dospecials
    \@ifstar\@sverb\@verb}
```

Definitions of `\@sverb` and `\@verb` changed so `\verb+` foo+ does not lose leading blanks when it comes at the beginning of a line. Change made 24 May 89. Suggested by Frank Mittelbach and Rainer Sch\"opf.

```
\def\@sverb#1{%
  \catcode`#1\active
  \lccode`\~`#1%
  \gdef\verb@balance@group{\verb@egroup
     \@latex@error{\noexpand\verb illegal in command argument}\@ehc}%
  \aftergroup\verb@balance@group
  \lowercase{\let~\verb@egroup}}%
```

```
\def\@verb{\@vobeyspaces \frenchspacing \@sverb}
```

\verbatim@nolig@list

```
\def\verbatim@nolig@list{\do\`\do\<\do\>\do\,\do\'\do\-}
```

\do@noligs

```
\def\do@noligs#1{%
  \catcode`#1\active
  \begingroup
     \lccode`\~=`#1\relax
     \lowercase{\endgroup\def~{\leavevmode\kern\z@\char`#1}}}
```

\@noligs To stay compatible with packages that use `\@noligs` we keep it.

```
\def\@noligs{\let\do\do@noligs \verbatim@nolig@list}

⟨/2ekernel⟩
```

File z
ltmath.dtx

52 Math setup

This file contains a lot of the original plain TEX code, as well as the LaTeX environments for math. It still needs sorting out.

```
⟨*2ekernel⟩
\message{math definitions,}
```

52.1 Math commands based on plain TEX

52.1.1 The log-like functions

The standard operators:

```
\def\log{\mathop{\operator@font log}\nolimits}
\def\lg{\mathop{\operator@font lg}\nolimits}
\def\ln{\mathop{\operator@font ln}\nolimits}
\def\lim{\mathop{\operator@font lim}}
\def\limsup{\mathop{\operator@font lim\,sup}}
\def\liminf{\mathop{\operator@font lim\,inf}}
\def\sin{\mathop{\operator@font sin}\nolimits}
\def\arcsin{\mathop{\operator@font arcsin}\nolimits}
\def\sinh{\mathop{\operator@font sinh}\nolimits}
\def\cos{\mathop{\operator@font cos}\nolimits}
\def\arccos{\mathop{\operator@font arccos}\nolimits}
\def\cosh{\mathop{\operator@font cosh}\nolimits}
\def\tan{\mathop{\operator@font tan}\nolimits}
\def\arctan{\mathop{\operator@font arctan}\nolimits}
\def\tanh{\mathop{\operator@font tanh}\nolimits}
\def\cot{\mathop{\operator@font cot}\nolimits}
\def\coth{\mathop{\operator@font coth}\nolimits}
\def\sec{\mathop{\operator@font sec}\nolimits}
\def\csc{\mathop{\operator@font csc}\nolimits}
\def\max{\mathop{\operator@font max}}
\def\min{\mathop{\operator@font min}}
\def\sup{\mathop{\operator@font sup}}
\def\inf{\mathop{\operator@font inf}}
\def\arg{\mathop{\operator@font arg}\nolimits}
\def\ker{\mathop{\operator@font ker}\nolimits}
\def\dim{\mathop{\operator@font dim}\nolimits}
\def\hom{\mathop{\operator@font hom}\nolimits}
\def\det{\mathop{\operator@font det}}
\def\exp{\mathop{\operator@font exp}\nolimits}
\def\Pr{\mathop{\operator@font Pr}}
\def\gcd{\mathop{\operator@font gcd}}
\def\deg{\mathop{\operator@font deg}\nolimits}
```

And some operators have to be done by hand:

```
\def\bmod{\nonscript\mskip-\medmuskip\mkern5mu
  \mathbin{\operator@font mod}\penalty900\mkern5mu%
  \nonscript\mskip-\medmuskip}
\def\pmod#1{\allowbreak\mkern18mu({\operator@font mod}\,\,#1)}
```

52.1.2 Bigggggg

Variants on `\big` and friends for use with delimiters:

```
\def\bigl{\mathopen\big}
\def\bigm{\mathrel\big}
\def\bigr{\mathclose\big}
\def\Bigl{\mathopen\Big}
\def\Bigm{\mathrel\Big}
\def\Bigr{\mathclose\Big}
\def\biggl{\mathopen\bigg}
\def\biggm{\mathrel\bigg}
\def\biggr{\mathclose\bigg}
\def\Biggl{\mathopen\Bigg}
\def\Biggm{\mathrel\Bigg}
\def\Biggr{\mathclose\Bigg}
```

52.1.3 The UNSORTED Rest

The other math commands are lifted from plain TEX.

```
\newdimen\jot \jot=3pt
\newcount\interdisplaylinepenalty \interdisplaylinepenalty=100
```

```
\def\choose{\atopwithdelims()}
\def\brack{\atopwithdelims[]}
\def\brace{\atopwithdelims\{\}}
```

```
\def\mathpalette#1#2{\mathchoice{#1\displaystyle{#2}}%
  {#1\textstyle{#2}}{#1\scriptstyle{#2}}{#1\scriptscriptstyle{#2}}}
\newbox\rootbox
\def\root#1\of{\setbox\rootbox\hbox{$\m@th\scriptscriptstyle{#1}$}
  \mathpalette\r@@t}
\def\r@@t#1#2{\setbox\z@\hbox{$\m@th#1\sqrtsign{#2}$}
  \dimen@\ht\z@ \advance\dimen@-\dp\z@
  \mkern5mu\raise.6\dimen@\copy\rootbox \mkern-10mu \box\z@}
```

`\phantom`
`\mathstrut`

```
\newif\ifv@ \newif\ifh@
\def\vphantom{\v@true\h@false\ph@nt}
\def\hphantom{\v@false\h@true\ph@nt}
\def\phantom{\v@true\h@true\ph@nt}
```

```
\def\ph@nt{%
  \ifmmode
    \expandafter\mathpalette\expandafter\mathph@nt
```

```
  \else
    \expandafter\makeph@nt
  \fi}
\def\makeph@nt#1{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finph@nt}
\def\mathph@nt#1#2{%
  \setbox\z@\hbox{$\m@th#1{#2}$}\finph@nt}
\def\finph@nt{%
  \setbox\tw@\null
  \ifv@ \ht\tw@\ht\z@ \dp\tw@\dp\z@\fi
  \ifh@ \wd\tw@\wd\z@\fi \box\tw@}
\def\mathstrut{\vphantom(}
```

\smash

```
\def\smash{%
  \relax % \relax, in case this comes first in \halign
  \ifmmode
    \expandafter\mathpalette\expandafter\mathsm@sh
  \else
    \expandafter\makesm@sh
  \fi}
\def\makesm@sh#1{%
  \setbox\z@\hbox{\color@begingroup#1\color@endgroup}\finsm@sh}
\def\mathsm@sh#1#2{%
  \setbox\z@\hbox{$\m@th#1{#2}$}\finsm@sh}
\def\finsm@sh{\ht\z@\z@ \dp\z@\z@ \box\z@}
```

\buildrel

```
\def\buildrel#1\over#2{\mathrel{\mathop{\kern\z@#2}\limits^{#1}}}
```

\cases

```
\def\cases#1{\left\{\,\vcenter{\normalbaselines\m@th
    \ialign{$##\hfil$&\quad{##}\hfil\crcr#1\crcr}}\right.}
\def\matrix#1{\null\,\vcenter{\normalbaselines\m@th
    \ialign{\hfil$##$\hfil&&\quad\hfil$##$\hfil\crcr
      \mathstrut\crcr\noalign{\kern-\baselineskip}
      #1\crcr\mathstrut\crcr\noalign{\kern-\baselineskip}}}\,}
\def\pmatrix#1{\left(\matrix{#1}\right)}
```

\bordermatrix

```
\def\bordermatrix#1{\begingroup \m@th
  \@tempdima 8.75\p@
  \setbox\z@\vbox{%
    \def\cr{\crcr\noalign{\kern2\p@\global\let\cr\endline}}%
    \ialign{$##$\hfil\kern2\p@\kern\@tempdima&\thinspace\hfil$##$\hfil
      &&\quad\hfil$##$\hfil\crcr
      \omit\strut\hfil\crcr\noalign{\kern-\baselineskip}%
```

```
      #1\crcr\omit\strut\cr}}%
  \setbox\tw@\vbox{\unvcopy\z@\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{\unhbox\@ne\unskip\global\setbox\@ne\lastbox}%
  \setbox\tw@\hbox{$\kern\wd\@ne\kern-\@tempdima\left(\kern-\wd\@ne
    \global\setbox\@ne\vbox{\box\@ne\kern2\p@}%
    \vcenter{\kern-\ht\@ne\unvbox\z@\kern-\baselineskip}\,\right)$}%
  \null\;\vbox{\kern\ht\@ne\box\tw@}\endgroup}
```

```
\def\openup{\afterassignment\@penup\dimen@=}
\def\@penup{\advance\lineskip\dimen@
  \advance\baselineskip\dimen@
  \advance\lineskiplimit\dimen@}
```

```
\newif\ifdt@p
\def\displ@y{\global\dt@ptrue\openup\jot\m@th
  \everycr{\noalign{\ifdt@p \global\dt@pfalse \ifdim\prevdepth>-1000\p@
      \vskip-\lineskiplimit \vskip\normallineskiplimit \fi
      \else \penalty\interdisplaylinepenalty \fi}}}
\def\@lign{\tabskip\z@skip\everycr{}} % restore inside \displ@y
\def\displaylines#1{\displ@y \tabskip\z@skip
  \halign{\hb@xt@\displaywidth{$\@lign\hfil\displaystyle##\hfil$}\crcr
    #1\crcr}}
```

```
\let\sp=^ \let\sb=_
%\def\,{\mskip\thinmuskip}      % already defined in ltspace
\def\>{\mskip\medmuskip}
\def\;{\mskip\thickmuskip}
\def\!{\mskip-\thinmuskip}
\def\*{\discretionary{\thinspace\the\textfont2\char2}{}{}}
```

\: Nickname for the medium space since `\>` is not available inside `tabbing`.

```
\let\:=\>
```

\prime@s

```
{\catcode`\'=\active \gdef'{^\bgroup\prim@s}}
\def\prim@s{%
  \prime\futurelet\@let@token\pr@m@s}
```

```
\def\pr@m@s{%
  \ifx'\@let@token
    \expandafter\pr@@@s
  \else
    \ifx^\@let@token
      \expandafter\expandafter\expandafter\pr@@@t
    \else
      \egroup
    \fi
  \fi}
```

```
\def\pr@@@s#1{\prim@s} \def\pr@@@t#1#2{#2\egroup}
```

```
{\catcode`\_=\active \gdef_{\_}} % _ in math is
                                 % either subscript or \_
```

52.2 Math Environments

\(\)

Produces `$...$` with checks that `\(` isn't used in math mode, and that `\)` is only used in math mode begun with `\(`.

```
\def\({\relax\ifmmode\@badmath\else$\fi}
\def\){\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 `\]`.

```
\def\[{%
   \relax\ifmmode
      \@badmath
   \else
      \ifvmode
         \nointerlineskip
         \makebox[.6\linewidth]%
      \fi
      $$%%$$ BRACE MATCH HACK
   \fi
}
\def\]{%
   \relax\ifmmode
      \ifinner
         \@badmath
      \else
         $$%%$$ BRACE MATCH HACK
      \fi
   \else
      \@badmath
   \fi
   \ignorespaces
}
```

math displaymath

Disguises for `\(...\)` and `\[...\]`.

```
\let\math=\(
\let\endmath=\)
\def\displaymath{\[}
\def\enddisplaymath{\]\global\@ignoretrue}
```

equation \c@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 flushleft. (See comment on the `\def` of `\@eqnnum`.)

```
\@definecounter{equation}
\def\equation{$$\refstepcounter{equation}}
\def\endequation{\eqno \hbox{\@eqnnum}$$\global\@ignoretrue}
```

\@eqnnum Produces the equation number for equation and eqnarray environments. The following definition is for flushright numbers; for flushleft numbers, see leqno.clo. The equation number is set in black roman type even if an eqnarray environment appears in an italic environment.

```
\def\@eqnnum{{\normalfont \normalcolor (\theequation)}}
```

\stackrel A disguise for plain TEX's buildrel.

```
\def\stackrel#1#2{\mathrel{\mathop{#2}\limits^{#1}}}
```

\frac A disguise for plain TEX's `\over`.

```
\def\frac#1#2{{\begingroup#1\endgroup\over#2}}
```

\sqrt \@sqrt Add an optional argument to plain's `\sqrt` to give the nth root of an expression $\sqrt[n]{e}$.

```
\def\sqrt{\@ifnextchar[\@sqrt\sqrtsign}
\def\@sqrt[#1]{\root #1\of}
```

eqnarray \@eqcnt \@eqpen \if@eqnsw \@eqnsel Here's the eqnarray environment: Default is for left-hand side of equations to be flushright. To make them flushleft, `\let\@eqnsel = \hfil`.

```
\newcount\@eqcnt
\newcount\@eqpen
\newif\if@eqnsw\@eqnswtrue
\newskip\@centering
\@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.

```
\def\eqnarray{%
   \stepcounter{equation}%
   \def\@currentlabel{\p@equation\theequation}%
   \global\@eqnswtrue
   \m@th
   \global\@eqcnt\z@
   \tabskip\@centering
   \let\\\@eqncr
   $$\everycr{}\halign to\displaywidth\bgroup
       \hskip\@centering$\displaystyle\tabskip\z@skip{##}$\@eqnsel
      &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}$\hfil
      &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep
         $\displaystyle{##}$\hfil\tabskip\@centering
      &\global\@eqcnt\thr@@ \hb@xt@\z@\bgroup\hss##\egroup
         \tabskip\z@skip
      \cr
}
\def\endeqnarray{%
      \@@eqncr
```

```
      \egroup
      \global\advance\c@equation\m@ne
   $$\global\@ignoretrue
}
\let\@eqnsel=\relax
```

\nonumber Switches off equation numbering.

```
\def\nonumber{\global\@eqnswfalse}
```

\@eqncr \@xeqncr \@yeqncr

```
\def\@eqncr{%
   {\ifnum0=`}\fi
   \@ifstar{%
      \global\@eqpen\@M\@yeqncr
   }{%
      \global\@eqpen\interdisplaylinepenalty \@yeqncr
   }%
}
\def\@yeqncr{%
   \@ifnextchar [%
      \@xeqncr
      {\@xeqncr[\z@skip]}%
}
\def\@xeqncr[#1]{%
   \ifnum0=`{\fi}%
   \@@eqncr
   \noalign{\penalty\@eqpen\vskip\jot\vskip #1\relax}%
}
```

\@@eqncr

```
\def\@@eqncr{\let\reserved@a\relax
    \ifcase\@eqcnt \def\reserved@a{& & &}\or \def\reserved@a{& &}%
     \or \def\reserved@a{&}\else
       \let\reserved@a\@empty
       \@latex@error{Too many columns in eqnarray environment}\@ehc\fi
     \reserved@a \if@eqnsw\@eqnnum\stepcounter{equation}\fi
     \global\@eqnswtrue\global\@eqcnt\z@\cr}
```

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

```
\let\@seqncr=\@eqncr
\@namedef{eqnarray*}{\def\@eqncr{\nonumber\@seqncr}\eqnarray}
\@namedef{endeqnarray*}{\nonumber\endeqnarray}
```

\lefteqn `\lefteqn{FORMULA}` typesets `FORMULA` in display math style flushleft in a box of width zero.

```
\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`.

```
\DeclareRobustCommand\ensuremath[1]{\ifmmode #1\else $#1$\fi}
⟨/2ekernel⟩
```

File A
ltlists.dtx

53 List, and related environments

The generic commands for creating an indented environment – `enumerate`, `itemize`, `quote`, etc – are:

`\list{`*LABEL*`}{`*COMMANDS*`}` ... `\endlist`

which can be invoked by the user as the list environment. The LABEL argument specifies item labeling. COMMANDS contains commands for changing the horizontal and vertical spacing parameters.

Each item of the environment is begun by the command `\item[ITEMLABEL]` which produces an item labeled by ITEMLABEL. If the argument is missing, then the LABEL argument of the `\list` command is used as the item label.

The label is formed by putting `\makelabel{`*ITEMLABEL*`}` in an hbox whose width is either its natural width or else `\labelwidth`, whichever is larger. The `\list` command defines `\makelabel` to have the default definition:

`\makelabel{`*ARG*`}` == BEGIN `\hfil` ARG END

which, for a label of width less than `\labelwidth`, puts the label flushright, `\labelsep` to the left of the item's text. However, `\makelabel` can be `\let` to another command by the `\list`'s COMMANDS argument.

A `\usecounter{`*foo*`}` command in the second argument causes the counter *foo* to be initialized to zero, and stepped by every `\item` command without an argument. (`\label` commands within the list refer to this counter.)

When you leave a list environment, returning either to an enclosing list or normal text mode, LaTeX begins a new paragraph if and only if you leave a blank line after the `\end` command. This is accomplished by the `\@endparenv` command.

Blank lines are ignored every other reasonable place–i.e.:

- Between the `\begin{list}` and the first `\item`,
- Between the `\item` and the text of that item.
- Between the end of the last item and the `\end{list}`.

For an environment like quotation, in which items are not labeled, the entire environment is a single item. It is defined by letting `\quotation` == `\list{}{...}\item\relax`. (Note the `\relax`, there in case the first character in the environment is a '['.) The spacing parameters provide a great deal of flexability 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.

53.1 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.

`@noparlist` A switch set true for a list that begins an item. No `\topsep` space is added before or after `\item`'s such a list.

`@newlist` Set true by `\list`, set false by the first text (by `\everypar`).

`@noitemarg` Set true when executing an `\item` with no explicit argument. Used to save space. To save time, make two separate `\@item` commands.

`@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 LATEX 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.

53.2 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.

53.3 Penalties

`\@beginparpenalty:` put at the beginning of a list

`\@endparpenalty:` put at end of list

`\@itempenalty:` put between items.

53.4 Horizontal Spacing (dimens)

`\leftmargin:` space between left margin of enclosing environment (or of page if top level list) and left margin of this list. Must be nonnegative.

`\rightmargin:` analogous.

`\listparindent:` extra indentation at beginning of every paragraph of a list except the one started by the `\item` command. May be negative! Usually, labeled lists have `\listparindent` equal to zero.

`\itemindent:` extra indentation added right BEFORE an item label.

`\labelwidth`: nominal width of box that contains the label. If the natural width of the label <= `\labelwidth`, then the label is flushed right inside a box of width `\labelwidth` (with an `\hfil`). Otherwise, a box of the natural width is employed, which causes an indentation of the text on that line.

`\labelsep`: space between end of label box and text of first item.

53.5 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
\itemsep
\parsep
\@topsep
\@topsepadd
\outerparskip

```
⟨*2ekernel⟩
\newskip\topsep
\newskip\partopsep
\newskip\itemsep
\newskip\parsep
\newskip\@topsep
\newskip\@topsepadd
\newskip\@outerparskip
```

\leftmargin
\rightmargin
\listparindent
\itemindent
\labelwidth
\labelsep
\@totalleftmargin

```
\newdimen\leftmargin
\newdimen\rightmargin
\newdimen\listparindent
\newdimen\itemindent
\newdimen\labelwidth
\newdimen\labelsep
\newdimen\linewidth
\newdimen\@totalleftmargin \@totalleftmargin=\z@
```

\leftmargini
\leftmarginii
\leftmarginiii
\leftmarginiv
\leftmarginv
\leftmarginvi

```
\newdimen\leftmargini
\newdimen\leftmarginii
\newdimen\leftmarginiii
\newdimen\leftmarginiv
\newdimen\leftmarginv
\newdimen\leftmarginvi
```

\@listdepth
\@itempenalty
\@beginparpenalty
\@endparpenalty

```
\newcount\@listdepth \@listdepth=0
\newcount\@itempenalty
\newcount\@beginparpenalty
\newcount\@endparpenalty
```

\@labels

```
\newbox\@labels
```

\if@inlabel
\@inlabelfalse
\@inlabeltrue

```
\newif\if@inlabel \@inlabelfalse
```

\if@newlist
\@newlistfalse
\@newlisttrue

```
\newif\if@newlist   \@newlistfalse
```

\if@noparitem
\@noparitemfalse
\@noparitemtrue

```
\newif\if@noparitem \@noparitemfalse
```

\if@noparlist
\@noparlistfalse
\@noparlisttrue

```
\newif\if@noparlist \@noparlistfalse
```

\if@noitemarg
\@noitemargfalse
\@noitemargtrue

```
\newif\if@noitemarg \@noitemargfalse
```

\if@newlist
\@newlistfalse
\@newlisttrue

```
\newif\if@nmbrlist  \@nmbrlistfalse
```

\list

```
\def\list#1#2{\ifnum \@listdepth >5\relax \@toodeep
     \else \global\advance\@listdepth\@ne \fi
  \rightmargin \z@ \listparindent\z@ \itemindent\z@
  \csname @list\romannumeral\the\@listdepth\endcsname
  \def\@itemlabel{#1}\let\makelabel\@mklab \@nmbrlistfalse #2\relax
  \@trivlist
  \parskip\parsep \parindent\listparindent
  \advance\linewidth -\rightmargin \advance\linewidth -\leftmargin
  \advance\@totalleftmargin \leftmargin
  \parshape \@ne \@totalleftmargin \linewidth
  \ignorespaces}
```

\par@deathcycles
\@trivlist

```
\newcount\par@deathcycles

\def\@trivlist{\@topsepadd\topsep
  \if@noskipsec \leavevmode \fi
  \ifvmode \advance\@topsepadd\partopsep \else \unskip\par\fi
  \if@inlabel \@noparitemtrue \@noparlisttrue
    \else \@noparlistfalse \@topsep\@topsepadd \fi
    \advance\@topsep \parskip
  \leftskip\z@skip\rightskip\@rightskip \parfillskip\@flushglue
  \par@deathcycles\z@
  \@setpar{\if@newlist
             \advance\par@deathcycles\@ne
             \ifnum\par@deathcycles>\@m
```

```
            \@noitemerr{\@@par}\fi
        \else{\@@par}\fi}%
  \global\@newlisttrue \@outerparskip\parskip}
```

\trivlist

```
\def\trivlist{\parsep\parskip\@nmbrlistfalse
  \@trivlist \labelwidth\z@ \leftmargin\z@
  \itemindent\z@
```

We initialise `\@itemlabel` so that a `trivlist` with an `\item` not having an optional argument doesn't produce an error message.

```
  \let\@itemlabel\@empty
  \def\makelabel##1{##1}}
```

\endlist

```
\def\endlist{\global\advance\@listdepth\m@ne
    \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
\@endparenv
\@doendpe

```
\def\endtrivlist{%
  \if@inlabel\indent\fi
  \if@newlist\@noitemerr\fi
  \ifhmode\unskip \par\fi
  \if@noparlist \else
    \ifdim\lastskip >\z@
      \@tempskipa\lastskip \vskip -\lastskip
      \advance\@tempskipa\parskip \advance\@tempskipa -\@outerparskip
      \vskip\@tempskipa
    \fi
    \@endparenv
  \fi
}
```

CHANGES TO `\cs`{@endparenv}:
Changed |`\hskip -\parindent`| to |`\setbox0=\lastbox`|
so that a `\cs`{noindent} becomes a no-op when used before
a line immediately following a list environment(23 Oct 86).

To suppress the paragraph indentation in text immediately following
a paragraph-making environment, `\cs`{everypar} is changed to remove the
space, and `\cs`{par} is redefined to restore `\cs`{everypar}. Instead of
redefining `\cs`{par} and `\cs`{everpar}, `\cs`{@endparenv} was changed to
set the @endpe switch, letting `\cs`{end} redefine `\cs`{par} and
`\cs`{everypar}.

This allows paragraph-making environments to work right when called by other environments. (Changed 27 Oct 86)

```
\def\@endparenv{%
  \addpenalty\@endparpenalty\addvspace\@topsepadd\@endpetrue}
\def\@doendpe{\@endpetrue
     \def\par{\@restorepar\everypar{}\par\@endpefalse}\everypar
                {\setbox\z@\lastbox\everypar{}\@endpefalse}}
```

\if@endpe
\@endpefalse
\@endpeltrue

```
\newif\if@endpe
\@endpefalse
```

\@mklab

```
\def\@mklab#1{\hfil #1}
```

\item

```
\def\item{\@inmatherr\item
  \@ifnextchar [\@item{\@noitemargtrue \@item[\@itemlabel]}}
```

\@donoparitem

```
\def\@donoparitem{\@noparitemfalse
   \global\setbox\@labels\hbox{\hskip -\leftmargin
                                \unhbox\@labels
                                 \hskip \leftmargin}\if@minipage\else
  \@tempskipa\lastskip
  \vskip -\lastskip \advance\@tempskipa\@outerparskip
  \advance\@tempskipa -\parskip \vskip\@tempskipa\fi}
```

\@item

```
\def\@item[#1]{\if@noparitem \@donoparitem
  \else \if@inlabel \indent \par \fi
         \ifhmode \unskip\unskip \par \fi
         \if@newlist \if@nobreak \@nbitem \else
                        \addpenalty\@beginparpenalty
                        \addvspace\@topsep \addvspace{-\parskip}\fi
           \else \addpenalty\@itempenalty \addvspace\itemsep
          \fi
    \global\@inlabeltrue
\fi
\everypar{\global\@minipagefalse\global\@newlistfalse
          \if@inlabel
            \global\@inlabelfalse
```

This next command was changed from an hskip to a kern to avoid a break point after the parindent box: the skip can cause a line-break if a very long label occurs in raggedright setting.

```
%           \hskip -\parindent
```

```
            \kern -\parindent
            \box\@labels
            \penalty\z@
          \fi
          \everypar{}}%
\global\@nobreakfalse
\if@noitemarg \@noitemargfalse \if@nmbrlist

      \refstepcounter\@listctr\fi \fi
```

We use `\sbox` to support colour commands.

```
\sbox\@tempboxa{\makelabel{#1}}%
\global\setbox\@labels
 \hbox{\unhbox\@labels \hskip \itemindent
       \hskip -\labelwidth \hskip -\labelsep
       \ifdim \wd\@tempboxa >\labelwidth
                \box\@tempboxa

          \else \hbox to\labelwidth {\unhbox\@tempboxa}\fi
       \hskip \labelsep}\ignorespaces}
```

`\makelabel`

```
\def\makelabel#1{\@latex@error{Lonely \string\item--perhaps a missing
        list environment}\@ehc}
```

`\@nbitem`

```
\def\@nbitem{\@tempskipa\@outerparskip \advance\@tempskipa -\parskip
               \addvspace\@tempskipa}
```

`\usecounter`

```
\def\usecounter#1{\@nmbrlisttrue\def\@listctr{#1}\setcounter{#1}\z@}
```

53.6 Itemize and Enumerate

Enumeration is done with four counters: `enumi`, `enumii`, `enumiii` and `enumiv`, where `enum`N 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@enum`N`\theenum`N 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 •. `\@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`

```
\newcount\@enumdepth \@enumdepth = 0
```

`\c@enumi`
`\c@enumii`
`\c@enumii`
`\c@enumiv`

```
\@definecounter{enumi}
\@definecounter{enumii}
\@definecounter{enumiii}
\@definecounter{enumiv}
```

`enumerate`

```
\def\enumerate{%
  \ifnum \@enumdepth >\thr@@\@toodeep\else
    \advance\@enumdepth\@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
      \list
        {\csname label\@enumctr\endcsname}%
        {\usecounter\@enumctr\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\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

```
\newcount\@itemdepth \@itemdepth = 0
```

itemize

```
\def\itemize{%
  \ifnum \@itemdepth >\thr@@\@toodeep\else
    \advance\@itemdepth\@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \list
      {\csname\@itemitem\endcsname}%
      {\def\makelabel##1{\hss\llap{##1}}}%
  \fi}
\let\enditemize =\endlist
⟨/2ekernel⟩
```

File B
ltboxes.dtx

54 LaTeX Box commands

`\makebox` `\makebox[`⟨*wid*⟩`][`⟨*pos*⟩`]{`⟨*obj*⟩`}`
Puts ⟨*obj*⟩ in an `\hbox` of width ⟨*wid*⟩, positioned by ⟨*pos*⟩.

The possible ⟨*pos*⟩ are:

`s` stretched,
`l` flushleft,
`r` flushright,
`c` (default) centred.

If ⟨*wid*⟩ is missing, then ⟨*pos*⟩ is also missing and ⟨*obj*⟩ is put in an `\hbox` of its natural width.

`\makebox(`⟨*x*⟩`,`⟨*y*⟩`)[`⟨*pos*⟩`]{`⟨*obj*⟩`}`
Puts ⟨*obj*⟩ in an `\hbox` of width $x *$ `\unitlength` and height $y *$ `\unitlength`. ⟨*pos*⟩ arguments are `s`, `l`, `r` or `c` (default) for stretched, flushleft, flushright or centred, and `t` or `b` for top, bottom – or combinations like `tr` or `rb`. Default for horizontal and vertical are centered.

`\mbox` `\mbox{`⟨*obj*⟩`}` The same as `\makebox{`⟨*obj*⟩`}`, but is more efficient as no checking for optional arguments is done.

`\newsavebox` `\newsavebox{\cmd}` : If `\cmd` is undefined, then defines it to be a TeX box register.

`\savebox` `\savebox{\cmd}` ... : `\cmd` is defined to be a TeX box register, and the '...' are any `\makebox` arguments. It is like `\makebox`, except it doesn't produce text but saves the value in `\box \cmd`.

`\sbox` `\sbox{`⟨*cmd*⟩`}{`⟨*obj*⟩`}` is an efficient abbreviation for
`\savebox{`⟨*cmd*⟩`}{`⟨*obj*⟩`}`.

`lrbox` `\begin{lrbox}{`⟨*cmd*⟩`}`⟨*text*⟩`\end{lrbox}` is equivalent to
`\sbox{`⟨*cmd*⟩`}{`⟨*text*⟩`}`
except that any white space at the beginning and end of ⟨*text*⟩ is ignored.

`\framebox` `\framebox` ... : like `\makebox`, except it puts a 'frame' around the box. The frame is made of lines of thickness `\fboxrule`, separated by space `\fboxsep` from the text – except for `\framebox(X,Y)` ... , where the thickness of the lines is as for the picture environment, and there is no separation added.

`\fbox` `\fbox{`⟨*obj*⟩`}` is an abbreviation for `\framebox{`⟨*obj*⟩`}`.

`\parbox` `\parbox[`⟨*pos*⟩`][`⟨*height*⟩`][`⟨*inner-pos*⟩`]{`⟨*width*⟩`}{`⟨*text*⟩`}` : Makes a box with `\hsize` ⟨*width*⟩, positioned by ⟨*pos*⟩ as follows: c : `\vcenter` (placed in `$...$` if not in math mode) b : `\vbox` t : `\vtop` default value is c. Sets `\hsize` := ⟨*width*⟩ and calls `\@parboxrestore`, which does the following: Restores the original definitions of:

\par
\\
\-
\'
\`
\=

Resets the following parameters:

\parindent	=	0pt	
\parskip	=	0pt	added 20 Jan 87
\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 restore \\.

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.

```
⟨*2ekernel⟩
\message{boxes,}
```

\makebox

\makebox User level command just looks for optional [or (.

```
\def\makebox{%
  \leavevmode
  \@ifnextchar(%)
    \@makepicbox
    {\@ifnextchar[\@makebox\mbox}}
```

\mbox

The basic horizontal box command for LaTeX.

```
\long\def\mbox#1{\leavevmode\hbox{#1}}
```

\@makebox

Look for a possible second optional argument (defaults to c).

```
\def\@makebox[#1]{%
  \@ifnextchar [{\@imakebox[#1]}{\@imakebox[#1][c]}}
```

\@begin@tempboxa

Helper macro for supporting \height, \width etc. Grab #1 into \@tempboxa and measure it.

```
\long\def\@begin@tempboxa#1#2{%
   \begingroup
     \setbox\@tempboxa#1{\color@begingroup#2\color@endgroup}%
     \def\width{\wd\@tempboxa}%
     \def\height{\ht\@tempboxa}%
     \def\depth{\dp\@tempboxa}%
     \let\totalheight\@ovri
     \totalheight\height
     \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.

```
\let\@end@tempboxa\endgroup
```

\bm@c \bm@l \bm@r \bm@s \bm@t \bm@b

Set up spacing.

```
\def\bm@c{\hss\unhbox\@tempboxa\hss}
\def\bm@l{\unhbox\@tempboxa\hss}\let\bm@t\bm@l
\def\bm@r{\hss\unhbox\@tempboxa}\let\bm@b\bm@r
\def\bm@s{\unhbox\@tempboxa}
```

\@imakebox

Internal form of \makebox.

```
\long\def\@imakebox[#1][#2]#3{%
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%        support calc
    \hb@xt@\@tempdima{\csname bm@#2\endcsname}%
  \@end@tempboxa}
```

\@makepicbox

Picture mode form of \makebox.

```
\def\@makepicbox(#1,#2){%
  \@ifnextchar[{\@imakepicbox(#1,#2)}{\@imakepicbox(#1,#2)[]}}
```

\@imakepicbox — picture mode version

```
\long\def\@imakepicbox(#1,#2)[#3]#4{%
  \vbox to#2\unitlength
   {\let\mb@b\vss \let\mb@l\hss\let\mb@r\hss
    \let\mb@t\vss
    \@tfor\reserved@a :=#3\do{%
      \if s\reserved@a
        \let\mb@l\relax\let\mb@r\relax
      \else
        \expandafter\let\csname mb@\reserved@a\endcsname\relax
      \fi}%
    \mb@t
    \hb@xt@ #1\unitlength{\mb@l #4\mb@r}%
    \mb@b
    \kern\z@}}
```

\set@color — This macro is initialy a no-op, but the colour package will redefine it to insert a `\special`.

```
\let\set@color\relax
```

\color@begingroup
\color@endgroup
\color@setgroup
\normalcolor
\color@hbox
\color@vbox
\color@endbox

These macros are initialy a no-op, but the colour package will redefine them to be `\begingroup`, `\endgroup`, `\begingroup\set@color`, `\hbox\bgroup\color@begingroup`, `\color@endgroup\egroup`. and ⟨*set to main document colour*⟩ respectively.

```
\let\color@begingroup\relax
\let\color@endgroup\relax
\let\color@setgroup\relax
\let\normalcolor\relax
\let\color@hbox\relax
\let\color@vbox\relax
\let\color@endbox\relax
```

\newsavebox — Allocate a new ‘savebox’.

```
\def\newsavebox#1{\@ifdefinable{#1}{\newbox#1}}
```

\savebox — Save `#1` in a box register.

```
\def\savebox#1{%
  \@ifnextchar(%)
    {\@savepicbox#1}{\@ifnextchar[{\@savebox#1}{\sbox#1}}}
```

\sbox — Save `#1` in a box register.

```
\long\def\sbox#1#2{\setbox#1\hbox{%
  \color@setgroup#2\color@endgroup}}
```

\@savebox — Look for second optional argument.

```
\def\@savebox#1[#2]{%
  \@ifnextchar [{\@isavebox#1[#2]}{\@isavebox#1[#2][c]}}
```

\@isavebox

```
\long\def\@isavebox#1[#2][#3]#4{%
  \setbox#1\hbox{%
    \color@setgroup
      \@imakebox[#2][#3]{#4}%
    \color@endgroup}}
```

\@savepicbox Picture mode version of `\savebox`.

```
\def\@savepicbox#1(#2,#3){%
  \@ifnextchar[%]
    {\@isavepicbox#1(#2,#3)}{\@isavepicbox#1(#2,#3)[]}}
```

\@isavepicbox Picture mode version of `\savebox`.

```
\long\def\@isavepicbox#1(#2,#3)[#4]#5{%
  \setbox#1\hbox{%
    \color@setgroup
      \@imakepicbox(#2,#3)[#4]{#5}%
    \color@endgroup}}
```

\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.

```
\def\lrbox#1{%
  \edef\reserved@a{%
    \endgroup
    \setbox#1\hbox{%
      \begingroup\aftergroup}%
        \def\noexpand\@currenvir{\@currenvir}%
        \def\noexpand\@currenvline{\on@line}}%
  \reserved@a
    \@endpefalse
    \color@setgroup
      \ignorespaces}
```

\endlrbox End the `lrbox` environment.

```
\def\endlrbox{\unskip\color@endgroup}
```

\usebox unchanged

```
\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 occurence of `\@halfwidth` by `\@wholewidth`

```
\long\def\frame#1{%
  \leavevmode
  \hbox{%
    \hskip-\@wholewidth
    \vbox{%
```

```
      \vskip-\@wholewidth
      \hrule \@height\@wholewidth
      \hbox{%
        \vrule\@width\@wholewidth
        #1%
        \vrule\@width\@wholewidth}%
      \hrule\@height\@wholewidth
      \vskip-\@wholewidth}%
    \hskip-\@wholewidth}}
```

`\fboxrule` `\fboxsep` user level parameters,

```
\newdimen\fboxrule
\newdimen\fboxsep
```

`\fbox` Abbreviated framed box command.

```
\long\def\fbox#1{%
  \leavevmode
  \setbox\@tempboxa\hbox{%
    \color@begingroup
      \kern\fboxsep{#1}\kern\fboxsep
    \color@endgroup}%
  \@frameb@x\relax}
```

`\framebox` Framed version of `\makebox`.

```
\def\framebox{%
  \@ifnextchar(%)
    \@framepicbox{\@ifnextchar[\@framebox\fbox}}
```

`\@framebox` Deal with optional arguments.

```
\def\@framebox[#1]{%
  \@ifnextchar[%]
    {\@iframebox[#1]}%
    {\@iframebox[#1][c]}}
```

`\@iframebox` The handling the optional arguments. In order to set the whole box, including the frame to the specified dimension, we first determine that dimension from the natural size of the text, `#3`. calculated width.

```
\long\def\@iframebox[#1][#2]#3{%
  \leavevmode
  \@begin@tempboxa\hbox{#3}%
    \setlength\@tempdima{#1}%
    \setbox\@tempboxa\hb@xt@\@tempdima
         {\kern\fboxsep\csname bm@#2\endcsname\kern\fboxsep}%
    \@frameb@x{\kern-\fboxrule}%
  \@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.

```
\def\@frameb@x#1{%
```

```
  \@tempdima\fboxrule
  \advance\@tempdima\fboxsep
  \advance\@tempdima\dp\@tempboxa
  \hbox{%
    \lower\@tempdima\hbox{%
      \vbox{%
        \hrule\@height\fboxrule
        \hbox{%
          \vrule\@width\fboxrule
          #1%
          \vbox{%
            \vskip\fboxsep
            \box\@tempboxa
            \vskip\fboxsep}%
          #1%
          \vrule\@width\fboxrule}%
        \hrule\@height\fboxrule}%
                          }%
        }%
}
```

`\@framepicbox` Picture mode version.

```
\def\@framepicbox(#1,#2){%
  \@ifnextchar[{\@iframepicbox(#1,#2)}{\@iframepicbox(#1,#2)[]}}
```

`\@iframepicbox` Picture mode version.

```
\long\def\@iframepicbox(#1,#2)[#3]#4{%
  \frame{\@imakepicbox(#1,#2)[#3]{#4}}}
```

`\parbox` The main vertical-box command for LaTeX.

```
\def\parbox{%
  \@ifnextchar[%]
    \@iparbox
    {\@iiiparbox c\@empty[s]}}
```

`\@iparbox` Optional argument handling.

```
\def\@iparbox[#1]{%
  \@ifnextchar[%]
    {\@iiparbox{#1}}%
    {\@iiiparbox{#1}\@empty[s]}}
```

`\@iiparbox` Optional argument handling.

```
\def\@iiparbox#1[#2]{%
  \@ifnextchar[%]
    {\@iiiparbox{#1}{#2}}%
    {\@iiiparbox{#1}{#2}[#1]}}
```

`\@iiiparbox` The internal version of `\parbox`.

`\@parboxto`

```
\let\@parboxto\@empty
```

```
\long\def\@iiiparbox#1#2[#3]#4#5{%
  \leavevmode
  \@pboxswfalse
  \setlength\@tempdima{#4}%
  \@begin@tempboxa\vbox{\hsize\@tempdima\@parboxrestore#5\endgraf}%
    \ifx\@empty#2\else
      \setlength\@tempdimb{#2}%
      \def\@parboxto{to\@tempdimb}%
    \fi
    \if#1b\vbox
    \else\if #1t\vtop
    \else\ifmmode\vcenter
    \else\@pboxswtrue $\vcenter
    \fi\fi\fi
    \@parboxto{\let\hss\vss\let\unhbox\unvbox
       \csname bm@#3\endcsname}%
    \if@pboxsw \m@th$\fi
  \@end@tempboxa}
```

`\@arrayparboxrestore` Restore various paragraph parameters.

```
\def\@arrayparboxrestore{%
  \let\par\@@par
  \let\-\@dischyph
```

Redefined accents to allow changes in font encoding

```
  \let\'\@acci\let\`\@accii\let\=\@acciii
  \parindent\z@ \parskip\z@skip
  \everypar{}%
  \linewidth\hsize
  \@totalleftmargin\z@
  \leftskip\z@skip \rightskip\z@skip \@rightskip\z@skip
  \parfillskip\@flushglue \lineskip\normallineskip
  \baselineskip\normalbaselineskip
  \sloppy}
```

`\parboxrestore` Restore various paragraph parameters, and also \\.

```
\def\@parboxrestore{\@arrayparboxrestore\let\\\@normalcr}
```

`\if@minipage` Switch that is true at the start of a minipage.

```
\newif\if@minipage \@minipagefalse
```

`\minipage` Essentially an environment form of `\parbox`.

```
\def\minipage{%
  \@ifnextchar[%]
    \@iminipage
    {\@iiiminipage c\@empty[s]}}
```

`\@iminipage` Optional argument handling.

```
\def\@iminipage[#1]{%
```

```
  \@ifnextchar[%]
    {\@iiminipage{#1}}%
    {\@iiiminipage{#1}\@empty[s]}}
```

\@iiminipage Optional argument handling.

```
\def\@iiminipage#1[#2]{%
  \@ifnextchar[%]
    {\@iiiminipage{#1}{#2}}%
    {\@iiiminipage{#1}{#2}[#1]}}
```

\@iiiminipage Internal form of `minipage`.

```
\def\@iiiminipage#1#2[#3]#4{%
  \leavevmode
  \@pboxswfalse
  \setlength\@tempdima{#4}%
  \def\@mpargs{{#1}{#2}[#3]{#4}}%
  \setbox\@tempboxa\vbox\bgroup
    \color@begingroup
      \hsize\@tempdima
      \textwidth\hsize \columnwidth\hsize
      \@parboxrestore
      \def\@mpfn{mpfootnote}\def\thempfn{\thempfootnote}\c@mpfootnote\z@
      \let\@footnotetext\@mpfootnotetext
      \let\@listdepth\@mplistdepth \@mplistdepth\z@
      \@minipagerestore\global\@minipagetrue %% \global added 24 May 89
      \everypar{\global\@minipagefalse\everypar{}}}
```

\@minipagerestore Hook so that other styles can reset other commands in a minipage.

```
  \let\@minipagerestore=\relax
```

\endminipage

```
\def\endminipage{%
    \par
    \unskip
    \ifvoid\@mpfootins\else
      \vskip\skip\@mpfootins
      \normalcolor
      \footnoterule
      \unvbox\@mpfootins
    \fi
    \global\@minipagefalse   %% added 24 May 89
  \color@endgroup
  \egroup
  \expandafter\@iiiparbox\@mpargs{\unvbox\@tempboxa}}
```

\@mplistdepth \@mpfootins Versions of `\@listdepth` and `\footins` local to minipage.

```
\newcount\@mplistdepth
\newinsert\@mpfootins
```

`\@mpfootnotetext` Minipage version of `\@footnotetext`.

Final `\strut` added 27 Mar 89, on suggestion by Don Hosek

```
\long\def\@mpfootnotetext#1{%
  \global\setbox\@mpfootins\vbox{%
    \unvbox\@mpfootins
    \reset@font\footnotesize
    \hsize\columnwidth
    \@parboxrestore
    \protected@edef\@currentlabel
         {\csname p@mpfootnote\endcsname\@thefnmark}%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}%
    \color@endgroup}}
```

```
\newif\if@pboxsw
```

`\rule` Draw a rule of the specified size.

```
\def\rule{\@ifnextchar[\@rule{\@rule[\z@]}}
```

`\@rule` internal form of `\rule`.

```
\def\@rule[#1]#2#3{%
  \@tempdima#3\advance\@tempdima #1%
  \leavevmode
  \hbox{\vrule\@width#2\@height\@tempdima\@depth-#1}}
```

`\@@underline` Saved primitive `\underline`.

```
\let\@@underline\underline
```

`\underline` LaTeX version works outside math.

```
\def\underline#1{%
  \relax
  \ifmmode\@@underline{#1}%
  \else $\@@underline{\hbox{#1}}\m@th$\relax\fi}
```

`\raisebox` Raise a box, and change its vertical dimensions.

```
\def\raisebox#1{%
  \leavevmode
  \@ifnextchar[{\@rsbox{#1}}{\@irsbox{#1}[]}}
```

`\@rsbox` Optional argument handling.

```
\def\@rsbox#1[#2]{%
  \@ifnextchar[{\@iirsbox{#1}[#2]}{\@irsbox{#1}[#2]}}
```

`\@argrsbox` ...

`\@irsbox` Internal version of `\raisebox` (less than two optional args).

```
\long\def\@irsbox#1[#2]#3{%
  \@begin@tempboxa\hbox{#3}%
```

```
    \setlength\@tempdima{#1}%
    \ifx\\#2\\\else\setlength\@tempdimb{#2}\fi
    \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
    \ifx\\#2\\\else\ht\@tempboxa\@tempdimb\fi
    \box\@tempboxa
  \@end@tempboxa}
```

`\@iirsbox` Internal version of `\raisebox` (two optional args).

```
\long\def\@iirsbox#1[#2][#3]#4{%
  \@begin@tempboxa\hbox{#4}%
    \setlength\@tempdima{#1}%
    \setlength\@tempdimb{#2}%
    \setlength\dimen@{#3}%
    \setbox\@tempboxa\hbox{\raise\@tempdima\box\@tempboxa}%
    \ht\@tempboxa\@tempdimb
    \dp\@tempboxa\dimen@
    \box\@tempboxa
  \@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. .

```
\def\@finalstrut#1{%
  \unskip\vrule\@width\z@\@height\z@\@depth\dp#1}
```

54.1 Some low-level constructs

The following commands are basically inherited from plain TEX.

`\leftline` `\rightline` `\centerline` `\@@line` These macros place text on a full line either centred or left or right adjusted.

```
\def\@@line{\hb@xt@\hsize}
\def\leftline#1{\@@line{#1\hss}}
\def\rightline#1{\@@line{\hss#1}}
\def\centerline#1{\@@line{\hss#1\hss}}
```

`\rlap` `\llap` These macros place text to the left or right of the current reference point without taking up space.

⟨/2ekernel⟩

File C
lttab.dtx

55 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 LaTeX 2.09 version, not the extended version described in *The LaTeX Companion*. Use the `array` package to obtain the extended version.

55.1 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 :=G \@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 untabs"        fi
    else error message "Left tab in middle of line"
  fi
 END

\@tabplus ==
 BEGIN
      if  \@nxttabmar < \@hightab
```

```
          then \@nxttabmar :=G \@nxttabmar+1
          else error message "Undefined tab"
       fi
 END

\@tabminus ==
 BEGIN
       if \@nxttabmar > \@firsttab
          then \@nxttabmar :=G \@nxttabmar-1
          else error message "Too many untabs"
       fi
 END

\@tabrj ==
 BEGIN \@stopfield
       \@addfield
       @rjfield :=G T
       \@startfield
 END

\@tablab ==
 BEGIN \@stopfield
       \box\@curline G:= \hbox{ \box\@curline %% 'G' added 17 Jun
86
                                \hskip - width of \box\@curfield
                                \hskip -\tabbingsep
                                \box\@curfield
                                \hskip \tabbingsep }
       \@startfield
 END

\pushtabs ==
  BEGIN
    \@stopfield
    \@tabpush :=G \@tabpush + 1
    \begingroup
    \@contfield
  END

\poptabs ==
 BEGIN
   \@stopfield
   if \@tabpush > 0
     then \endgroup
          \@tabpush :=G \@tabpush - 1
     else error message: "Too many \poptabs''
```

```
  fi
  \@contfield
END
```

\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.

```
⟨defx⟩\begingroup
⟨defx⟩\makeatletter
⟨defx⟩\nfss@catcodes
⟨2ekernel⟩\expandafter\let\csname ver@autotabg.sty\endcsname\fmtversion
```

```
⟨*2ekernel | autoload⟩
\newdimen\@gtempa
\chardef\@firsttab=\the\allocationnumber
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa\newdimen\@gtempa
\newdimen\@gtempa
\chardef\@maxtab=\the\allocationnumber
\dimen\@firsttab=0pt
\newcount\@nxttabmar
\newcount\@curtabmar
\newcount\@curtab
\newcount\@hightab
\newcount\@tabpush
\newbox\@curline
\newbox\@curfield
\newbox\@tabfbox
⟨/2ekernel | autoload⟩
```

```
⟨*2ekernel | def⟩
\newif\if@rjfield
⟨def⟩\global\let\if@rjfield\if@rjfield
⟨def⟩\global\let\@rjfieldtrue\@rjfieldtrue
⟨def⟩\global\let\@rjfieldfalse\@rjfieldfalse
```

```
\gdef\@startline{\global\@curtabmar\@nxttabmar
   \global\@curtab\@curtabmar\global\setbox\@curline\hbox
%                                 missing \global added 17 Jun 86
    {}\@startfield\strut}
```

```
\gdef\@stopline{\unskip\@stopfield\if@rjfield \global\@rjfieldfalse
   \@tempdima\@totalleftmargin \advance\@tempdima\linewidth
 \hb@xt@\@tempdima{\@itemfudge\hskip\dimen\@curtabmar
   \box\@curline\hfil\box\@curfield}\else\@addfield
   \hbox{\@itemfudge\hskip\dimen\@curtabmar\box\@curline}\fi}
```

\@startfield

```
\gdef\@startfield{%
  \global\setbox\@curfield\hbox\bgroup\color@begingroup}
```

\@stopfield

```
\gdef\@stopfield{%
  \color@endgroup\egroup}
```

\@contfield

```
\gdef\@contfield{%
  \global\setbox\@curfield\hbox\bgroup\color@begingroup
  \unhbox\@curfield}
```

```
\gdef\@addfield{\global\setbox\@curline\hbox{\unhbox
     \@curline\unhbox\@curfield}}
\gdef\@ifatmargin{\ifdim \wd\@curline =\z@}
```

```
\gdef\@tabcr{\@stopline \@ifstar{\penalty \@M \@xtabcr}\@xtabcr}
```

```
\gdef\@xtabcr{\@ifnextchar[\@itabcr{\@startline\ignorespaces}}
```

```
\gdef\@itabcr[#1]{\vskip #1\@startline\ignorespaces}
```

```
\gdef\kill{\@stopfield\@startline\ignorespaces}
```

We use `\relax` to prevent `\item` to scan too far.

```
\gdef\tabbing{\lineskip \z@skip\let\>\@rtab\let\<\@ltab\let\=\@settab
     \let\+\@tabplus\let\-\@tabminus\let\`\@tabrj\let\'\@tablab
     \let\\=\@tabcr
     \global\@hightab\@firsttab
     \global\@nxttabmar\@firsttab
     \dimen\@firsttab\@totalleftmargin
     \global\@tabpush\z@ \global\@rjfieldfalse
     \trivlist \item\relax
     \if@minipage\else\vskip\parskip\fi
     \setbox\@tabfbox\hbox{\rlap{\indent\hskip\@totalleftmargin
       \the\everypar}}\def\@itemfudge{\box\@tabfbox}%
     \@startline\ignorespaces}
```

```
\gdef\endtabbing{%
  \@stopline\ifnum\@tabpush >\z@ \@badpoptabs \fi\endtrivlist}
```

```
\gdef\@rtab{\@stopfield\@addfield\ifnum \@curtab<\@hightab
      \global\advance\@curtab \@ne \else\@badtab\fi
      \@tempdima\dimen\@curtab
      \advance\@tempdima -\dimen\@curtabmar
      \advance\@tempdima -\wd\@curline
      \global\setbox\@curline\hbox{\unhbox\@curline\hskip\@tempdima}%
      \@startfield\ignorespaces}
```

Omitted `\global` added to `\@rtab` 17 Jun 86

```
\gdef\@settab{\@stopfield\@addfield\ifnum \@curtab <\@maxtab
      \global\advance\@curtab \@ne \else
         \@latex@error{Tab overflow}\@ehd\fi
      \ifnum\@curtab >\@hightab
      \@hightab\@curtab\fi
      \dimen\@curtab\dimen\@curtabmar
      \advance\dimen\@curtab \wd\@curline\@startfield\ignorespaces}
\gdef\@ltab{\@ifatmargin\ifnum\@curtabmar >\@firsttab
      \global\advance\@curtab \m@ne \global\advance\@curtabmar\m@ne\else
      \@badtab\fi\else
      \@latex@error{\string\<\space in mid line}\@ehd\fi\ignorespaces}
\gdef\@tabplus {\ifnum \@nxttabmar <\@hightab
      \global\advance\@nxttabmar \@ne \else
      \@badtab\fi\ignorespaces}
\gdef\@tabminus{\ifnum\@nxttabmar >\@firsttab
      \global\advance\@nxttabmar \m@ne \else
      \@badtab\fi\ignorespaces}
\gdef\@tabrj{%
  \@stopfield\@addfield\global\@rjfieldtrue\@startfield\ignorespaces}
```

```
\gdef\@tablab{\@stopfield\global\setbox\@curline\hbox{\box\@curline
     \hskip -\wd\@curfield \hskip -\tabbingsep \box\@curfield
      \hskip \tabbingsep}\@startfield\ignorespaces}
```

`\setbox\@curline` made `\global` in `\@tablab`. 17 Jun 86

```
\gdef\pushtabs{%
  \@stopfield\@addfield\global\advance\@tabpush \@ne \begingroup
       \@contfield}
\gdef\poptabs{\@stopfield\@addfield\ifnum\@tabpush >\z@ \endgroup
     \global\advance\@tabpush \m@ne \else
     \@badpoptabs\fi\@contfield}
⟨/2ekernel | def⟩
```

```
⟨*2ekernel | autoload⟩
\newdimen\tabbingsep
⟨/2ekernel | autoload⟩
```

```
⟨*def⟩
\endgroup
⟨/def⟩
```

```
⟨*autoload⟩
\def\tabbing{\input{autotabg.sty}\tabbing}
⟨/autoload⟩
```

55.2 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` commmand.

PREAMBLE:
The PREAMBLE argument of an array or tabular environment can contain
the following:
l,r,c : indicate where entry is to be placed.
| : for vertical rule
@{EXP} : inserts the text EXP in every column.
`\arraycolsep` or `\tabcolsep` spacing is suppressed.
*{N}{PRE} : equivalent to writing N copies of PRE in the preamble.
PRE may contain *{N'}{EXP'} expressions.
p{LEN} : makes entry in parbox of width LEN.

SPECIAL ARRAY COMMANDS:
`\multicolumn`{N}{FORMAT}{ITEM} : replaces the next N column items by
ITEM, formatted according to FORMAT. FORMAT should contain at
most one l,r or c. If it contains none, then ITEM is ignored.

`\vline` : draws a vertical line the height of the current row. May appear in an array element entry.
`\hline` : draws a horizontal line between rows. Must appear either before the first entry (to appear above the first row) or right after a \\ command. If followed by another `\hline`, then adds a `\vskip` of `\doublerulesep`.

`\cline`[i-j] : draws horizontal lines between rows covering columns

```
                  i through j, inclusive.  Multiple commands may follow
                  one another to provide lines covering several disjoint
                  columns
   \extracolsep{WIDTH} : for use inside an @ in the preamble.  Causes
a
                 WIDTH space to be added between columns for the rest
of
                 the columns.  This is in addition to the ordinary
                 intercolumn space.

  \array ==
    BEGIN
      \@acol    == \@arrayacol
      \@classz  == \@arrayclassz
      \@classiv == \@arrayclassiv
      \\        == \@arraycr
      \@halignto == NULL
      \@tabarray
    END

  \endarray{NAME} ==  BEGIN  \crcr }}  END

  \tabular  ==
    BEGIN
      \@halignto == NULL
      \@tabular
    END

  \tabular*{WIDTH} ==
    BEGIN
      \@halignto == to WIDTH
      \@tabular
    END

  \@tabular ==
    BEGIN
      \leavevmode
      \hbox { $
         \@acol    == \@tabacol
         \@classz  == \@tabclassz
         \@classiv == \@tabclassiv
         \\        == \@tabularcr
         \@tabarray
    END

  \endtabular == BEGIN \crcr}} $} END
```

```
\@tabarray == if next char = [ then \@array else \@array[c] fi

\@array[POS]{PREAMBLE} ==
  BEGIN
    define \@arstrutbox to make \@arstrut produce strut of height
      and depth \arraystretch times the height and
      depth of a normal strut.
    \@mkpream{PREAMBLE}
    \@preamble == \halign \@halignto {\tabskip=0pt\@arstrut
                              eval{\@preamble}\tabskip = 0pt\cr  %%
}
    \@startpbox == \@@startpbox
    \@endpbox == \@@endpbox
    if POS = t then \vtop
              else if POS = b then \vbox
                              else \vcenter
    fi              fi
   {
    \par           ==L {} % changed 92/09/18
    \@sharp        == #
    \protect       == \relax
    \lineskip      :=L 0pt
    \baselineskip  :=L 0pt
    \@preamble
  END


\@arraycr ==
 BEGIN
   $               %% Prevents extra space at end of row's last entry.
   if next char = [
    then  \@argarraycr
    else  $ \cr          %% Needed to balance $
 END

\@argarraycr[LENGTH] ==
 BEGIN
   $                      %% Needed to balance $ of \@arraycr
   if LENGTH > 0
     then  \@tempdima := depth of \@arstrutbox + LENGTH
           \vrule height 0pt width 0pt depth \@tempdima
           \cr
     else  \cr \noalign{\vskip LENGTH}
 END
```

`\@tabularcr` and `\@argtabularcr` same as `\@arraycr` and `\@argarraycr`
except without the extra `$`'s.

```
⟨*2ekernel | autoload⟩
\def\extracolsep#1{\tabskip #1\relax}

\def\array{\let\@acol\@arrayacol \let\@classz\@arrayclassz
 \let\@classiv\@arrayclassiv
 \let\\\@arraycr\let\@halignto\@empty\@tabarray}

\def\endarray{\crcr\egroup\egroup}
\def\endtabular{\crcr\egroup\egroup $\egroup}
\expandafter \let \csname endtabular*\endcsname = \endtabular

\def\tabular{\let\@halignto\@empty\@tabular}
```

RmS 93/08/05: Replaced `\expandafter\def`... by `\@namedef`

```
\@namedef{tabular*}#1{\def\@halignto{to#1}\@tabular}

\def\@tabular{\leavevmode \hbox \bgroup $\let\@acol\@tabacol
   \let\@classz\@tabclassz
   \let\@classiv\@tabclassiv \let\\\@tabularcr\@tabarray}
```

RmS 91/11/04 added `\m@th`

```
\def\@tabarray{\m@th\@ifnextchar[\@array{\@array[c]}}
```

RmS 93/11/03 changed `\halign` to `\ialign` and removed superfluous `\tabskip` assignment

`\array`

```
\def\@array[#1]#2{\setbox\@arstrutbox\hbox{\vrule
     \@height\arraystretch \ht\strutbox
     \@depth\arraystretch \dp\strutbox
     \@width\z@}\@mkpream{#2}\edef\@preamble{\ialign \noexpand\@halignto
      \bgroup \@arstrut \@preamble \tabskip\z@skip \cr}%
      \let\@startpbox\@@startpbox \let\@endpbox\@@endpbox

      \let\tabularnewline\\%
      \if #1t\vtop \else \if#1b\vbox \else \vcenter \fi\fi

      \bgroup \let\par\@empty
       \let\@sharp##\let\protect\relax
       \lineskip\z@skip\baselineskip\z@skip\@preamble}
```

`\@arraycr` array version of `\\`.

```
\def\@arraycr{%
  ${\ifnum0=`}\fi\@ifstar\@xarraycr\@xarraycr}
```

\@arraycr

```
\def\@xarraycr{\@ifnextchar[\@argarraycr{\ifnum0=`{\fi}${}\cr}}
```

\@argarraycr

```
\def\@argarraycr[#1]{%
  \ifnum0=`{\fi}${}\ifdim #1>\z@ \@xargarraycr{#1}\else
   \@yargarraycr{#1}\fi}
```

\tabularnewline tabular version of \\.

\@tabularcr

```
\def\@tabularcr{%
  {\ifnum0=`}\fi\@ifstar\@xtabularcr\@xtabularcr}
\let\tabularnewline\relax
```

```
\def\@xtabularcr{\@ifnextchar[\@argtabularcr{\ifnum0=`{\fi}\cr}}
```

```
\def\@argtabularcr[#1]{\ifnum0=`{\fi}\ifdim #1>\z@
   \unskip\@xargarraycr{#1}\else \@yargarraycr{#1}\fi}
```

```
\def\@xargarraycr#1{\@tempdima #1\advance\@tempdima \dp \@arstrutbox
   \vrule \@height\z@ \@depth\@tempdima \@width\z@ \cr}
```

```
\def\@yargarraycr#1{\cr\noalign{\vskip #1}}
```

\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`

```
\def\multicolumn#1#2#3{\multispan{#1}\begingroup
  \@mkpream{#2}%
  \def\@sharp{#3}\set@typeset@protect
  \let\@startpbox\@@startpbox\let\@endpbox\@@endpbox
  \@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
|	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 ->                                   % |
         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 \@arrayclassv except without the $ ... $
```

```
\@classv ==
  BEGIN
   \@preamble :=
       \@preamble * \@startpbox{\@nextchar}\ignorespaces\@sharp
                               \@endpbox
  END
```

`\@expast{S}`: Sets `\reserved@a` := S with all instances of *{N}{STRING}
replaced by N copies of STRING, where N > 0. An * appearing inside braces is ignored, but *-expressions inside STRING are expanded, so nested *-expressions are handled properly.

```
\@expast{S} == BEGIN  \@xexpast S *0x\@@  END

\@xexpast S1 *{N}{S2} S3 \@@ ==
 BEGIN
   \reserved@a   := S1
   \@tempcnta := N
   if \@tempcnta > 0
     then  while \@tempcnta > 0 do \reserved@a   := \reserved@a S2
                                   \@tempcnta := \@tempcnta - 1 od
           \reserved@b == \@xexpast
     else  \reserved@b == \@xexnoop
   fi
   \expandafter \reserved@b \reserved@a S3 \@@
 END
```

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

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

\def\@xexpast#1*#2#3#4\@@{\edef\reserved@a{#1}\@tempcnta#2\relax
   \ifnum\@tempcnta >\z@ \@whilenum\@tempcnta >\z@\do
      {\edef\reserved@a{\reserved@a#3}\advance\@tempcnta \m@ne}%
      \let\reserved@b\@xexpast
     \else \let\reserved@b\@xexnoop\fi
   \expandafter\reserved@b \reserved@a #4\@@}

\newif\if@firstamp
\def\@addamp{\if@firstamp \@firstampfalse \else
   \edef\@preamble{\@preamble &}\fi}
\def\@arrayacol{\edef\@preamble{\@preamble \hskip \arraycolsep}}
\def\@tabacol{\edef\@preamble{\@preamble \hskip \tabcolsep}}
\def\@ampacol{\@addamp \@acol}
\def\@acolampacol{\@acol\@addamp\@acol}
```

\changes{v1.0h}{1994/11/04}{(ASAJ) Added \cs{@unexpandable@protect} to \cs{@mkpream}.}

```
\def\@mkpream#1{\@firstamptrue\@lastchclass6
  \let\@preamble\@empty
  \let\protect\@unexpandable@protect
  \let\@sharp\relax
  \let\@startpbox\relax\let\@endpbox\relax
  \@expast{#1}%
  \expandafter\@tfor \expandafter
    \@nextchar \expandafter:\expandafter=\reserved@a\do
       {\@testpach\@nextchar
    \ifcase \@chclass \@classz \or \@classi \or \@classii \or \@classiii
      \or \@classiv \or\@classv \fi\@lastchclass\@chclass}%
  \ifcase \@lastchclass \@acol
      \or \or \@preamerr \@ne\or \@preamerr \tw@\or \or \@acol \fi}
```

```
\def\@arrayclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
   \or \or \@addamp \or
   \@acolampacol \or \@firstampfalse \@acol \fi
\edef\@preamble{\@preamble
  \ifcase \@chnum
     \hfil$\relax\@sharp$\hfil \or $\relax\@sharp$\hfil
    \or \hfil$\relax\@sharp$\fi}}
```

RmS 91/08/14 inserted extra braces around entry for NFSS

```
\def\@tabclassz{\ifcase \@lastchclass \@acolampacol \or \@ampacol \or
   \or \or \@addamp \or
   \@acolampacol \or \@firstampfalse \@acol \fi
\edef\@preamble{\@preamble{%
  \ifcase \@chnum
     \hfil\ignorespaces\@sharp\unskip\hfil
     \or \ignorespaces\@sharp\unskip\hfil
     \or \hfil\hskip\z@skip \ignorespaces\@sharp\unskip\fi}}}
```

```
\def\@classi{\ifcase \@lastchclass \@acol \@arrayrule \or
   \@addtopreamble{\hskip \doublerulesep}\@arrayrule\or
   \or \or \@arrayrule \or
   \@acol \@arrayrule \or \@arrayrule \fi}
```

```
\def\@classii{\ifcase \@lastchclass \or
   \@addtopreamble{\hskip .5\arrayrulewidth}\fi}
```

```
\def\@classiii{\ifcase \@lastchclass \@acolampacol \or
   \@addamp\@acol \or
   \or \or \@addamp \or
   \@acolampacol \or \@ampacol \fi}
```

```
\def\@tabclassiv{\@addtopreamble\@nextchar}
```

```
\def\@arrayclassiv{\@addtopreamble{$\@nextchar$}}
```

```
\def\@classv{\@addtopreamble{\@startpbox{\@nextchar}\ignorespaces
\@sharp\@endpbox}}

\def\@addtopreamble#1{\edef\@preamble{\@preamble #1}}

\newcount\@chclass
\newcount\@lastchclass
\newcount\@chnum

\newdimen\arraycolsep
\newdimen\tabcolsep
\newdimen\arrayrulewidth
\newdimen\doublerulesep

\def\arraystretch{1}    % Default value.

\newbox\@arstrutbox
\def\@arstrut{%
  \relax\ifmmode\copy\@arstrutbox\else\unhcopy\@arstrutbox\fi}

\def\@arrayrule{\@addtopreamble{\hskip -.5\arrayrulewidth
   \vrule \@width \arrayrulewidth\hskip -.5\arrayrulewidth}}

\def\@testpach#1{\@chclass \ifnum \@lastchclass=\tw@ 4 \else
    \ifnum \@lastchclass=3 5 \else
      \z@ \if #1c\@chnum \z@ \else
                              \if #1l\@chnum \@ne \else
                              \if #1r\@chnum \tw@ \else
          \@chclass \if #1|\@ne \else
                    \if #1@\tw@ \else
                    \if #1p3 \else \z@ \@preamerr 0\fi
  \fi  \fi  \fi  \fi  \fi  \fi
\fi}

\def\hline{%
  \noalign{\ifnum0=`}\fi\hrule \@height \arrayrulewidth \futurelet
   \reserved@a\@xhline}

\def\@xhline{\ifx\reserved@a\hline
               \vskip\doublerulesep
```

Measure from the middle of the rules.

```
               \vskip-\arrayrulewidth
             \fi
      \ifnum0=`{\fi}}

\def\vline{\vrule \@width \arrayrulewidth}

\newcount\@cla
\newcount\@clb
```

```
\def\cline#1{\@cline[#1]}
\def\@cline[#1-#2]{\noalign{\global\@cla#1\relax
\global\advance\@cla\m@ne
\ifnum\@cla>\z@\global\let\@gtempa\@clinea\else
  \global\let\@gtempa\@clineb\fi
\global\@clb#2\relax
\global\advance\@clb-\@cla}\@gtempa
\noalign{\vskip-\arrayrulewidth}}
```

```
\def\@clinea{\multispan\@cla&\multispan\@clb
\unskip\leaders\hrule \@height \arrayrulewidth \hfill
\cr}
```

```
\def\@clineb{\multispan\@clb
\unskip\leaders\hrule \@height \arrayrulewidth \hfill
\cr}
```

`\@startpbox` `\@endpbox`

Helper macros for 'p' columns.

`\@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)

```
\def\@startpbox#1{\vtop\bgroup \hsize #1\@arrayparboxrestore}
\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.

```
\let\@@startpbox=\@startpbox
\let\@@endpbox=\@endpbox
```

```
⟨/2ekernel | autoload⟩
```

File D
ltpictur.dtx

56 Picture Mode

Picture mode commands. In addition to the commands available in LaTeX2.09, This section adds the new `\qbezier` command for drawing curves.

\qbezier `\qbezier[`N`](`AX,AY`)(`BX,BY`)(`CX,CY`)`

plots a quadratic Bezier curve from (AX,AY) to (CX,CY), with (BX,BY) as the third Bezier point, using $N+1$ points equally spaced parametrically. If $N = 0$ (the default value), then a sufficient number of points are used to draw a connected curve–except that at most `\qbeziermax` $+ 1$ points are drawn. A "point" is a square of side `\@wholewidth`.

\bezier In addition, to be compatible with the old `bezier` package, a variant of this command, `\bezier`, is defined, in which the first argument is not optional.

```
\unitlength      = value of dimension argument
\@wholewidth     = current line width
\@halfwidth      = half of current line width
\@linefnt        = font for drawing lines
\@circlefnt      = font for drawing circles

\linethickness{DIM} : Sets the width of horizontal and vertical lines
    in a picture to DIM.  Does not change width of slanted lines
    or circles.   Width of all lines reset by \thinlines and
    \thicklines

\picture(XSIZE,YSIZE)(XORG,YORG)
  BEGIN
    \@picht :=L YSIZE * \unitlength
    box \@picbox :=
          \hb@xt@ XSIZE * \unitlength
            {\hskip -XORG * \unitlength
             \lower YORG * \unitlength
             \hbox{
             \ignorespaces    %% added 13 June 89
  END

\endpicture ==
  BEGIN
                  } \hss }
                  heigth 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 'defx' code below would ensure that the autoload file is read in with sane catcodes. Actually this code is placed earlier in the file, before the `\ProvidesFile` command.

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.

```
⟨defx⟩\begingroup
⟨defx⟩\makeatletter
⟨defx⟩\nfss@catcodes
⟨2ekernel⟩\expandafter\let\csname ver@autopict.sty\endcsname\fmtversion
```

`\@wholewidth`
`\@halfwidth`

```
⟨*2ekernel | autoload⟩
\newdimen\@wholewidth
\newdimen\@halfwidth
```

\unitlength

```
\newdimen\unitlength \unitlength =1pt
```

\@picbox
\@picht

```
\newbox\@picbox
\newdimen\@picht
⟨/2ekernel | autoload⟩
```

\picture `#1` should be white space.

\pictur@ `#1` should be a ((eating any white space before the bracket),

```
⟨*2ekernel | def⟩
\long\gdef\picture#1{\pictur@#1}
\gdef\pictur@(#1){%
  \@ifnextchar({\@picture(#1)}{\@picture(#1)(0,0)}}
⟨/2ekernel | def⟩
⟨*autoload⟩
\def\picture{\pictur@\picture}
\def\pictur@{\input{autopict.sty}}
⟨/autoload⟩
```

\@picture

```
⟨*2ekernel | def⟩
\gdef\@picture(#1,#2)(#3,#4){%
  \@picht#2\unitlength
  \setbox\@picbox\hb@xt@#1\unitlength\bgroup
    \hskip -#3\unitlength
    \lower #4\unitlength\hbox\bgroup
      \ignorespaces}
```

\endpicture

```
\gdef\endpicture{%
  \egroup\hss\egroup
    \ht\@picbox\@picht\dp\@picbox\z@
    \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).

```
\long\gdef\put(#1,#2)#3{%
  \@killglue\raise#2\unitlength
  \hb@xt@\z@{\kern#1\unitlength #3\hss}%
  \ignorespaces}
```

\multiput `#3` had better be a (.

```
\gdef\multiput(#1,#2)#3{%
  \@xdim #1\unitlength
  \@ydim #2\unitlength
   \@multiput()}
```

\multiput

```
\long\gdef\@multiput(#1,#2)#3#4{%
  \@killglue\@multicnt #3\relax
  \@whilenum \@multicnt >\z@\do
    {\raise\@ydim\hb@xt@\z@{\kern\@xdim #4\hss}%
     \advance\@multicnt\m@ne
     \advance\@xdim#1\unitlength\advance\@ydim#2\unitlength}%
  \ignorespaces}
```

\@killglue

```
\gdef\@killglue{\unskip\@whiledim \lastskip >\z@\do{\unskip}}
⟨/2ekernel | def⟩
```

\thinlines
\thicklines

```
⟨*2ekernel | def⟩
\gdef\thinlines{\let\@linefnt\tenln \let\@circlefnt\tencirc
  \@wholewidth\fontdimen8\tenln \@halfwidth .5\@wholewidth}
\gdef\thicklines{\let\@linefnt\tenlnw \let\@circlefnt\tencircw
  \@wholewidth\fontdimen8\tenlnw \@halfwidth .5\@wholewidth}
⟨/2ekernel | def⟩
⟨*autoload⟩
\def\thinlines{\pictur@\thinlines}
\def\thicklines{\pictur@\thinlines}
⟨/autoload⟩
```

\linethickness

```
⟨*2ekernel | def⟩
\gdef\linethickness#1{\@wholewidth #1\relax \@halfwidth .5\@wholewidth}
⟨/2ekernel | def⟩
⟨*autoload⟩
\def\linethickness{\pictur@\linethickness}
⟨/autoload⟩
```

\ishortstack

```
⟨*2ekernel | def⟩
\gdef\shortstack{\@ifnextchar[\@shortstack{\@shortstack[c]}}
```

\@ishortstack

```
\gdef\@shortstack[#1]{%
  \leavevmode
  \vbox\bgroup
    \baselineskip-\p@\lineskip 3\p@
    \let\mb@l\hss\let\mb@r\hss
    \expandafter\let\csname mb@#1\endcsname\relax
    \let\\\@stackcr
    \@ishortstack}
```

\@ishortstack

```
\gdef\@ishortstack#1{\ialign{\mb@l {##}\unskip\mb@r\cr #1\crcr}\egroup}
```

\@stackcr
\@ixstackcr

```
\gdef\@stackcr{\@ifstar\@ixstackcr\@ixstackcr}
\gdef\@ixstackcr{\@ifnextchar[\@istackcr{\cr\ignorespaces}}
```

\@istackcr

```
\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
 if
END

\@sline ==
 BEGIN
   if \@xarg < 0
     then @negarg := T
          \@xarg  := -\@xarg
          \@yyarg := -\@yarg
     else @negarg := F
          \@yyarg := \@yarg
   fi
   \@tempcnta := |\@yyarg|
   if \@tempcnta > 6
     then error: 'LATEX ERROR: Illegal \line or \vector argument.'
          \@tempcnta := 0
   fi
   \box\@linechar := \hbox{\@linefnt \@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
         if
 if
END

\@hvector ==
  BEGIN
    \@hline
    {\@linefnt 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
```

```
\newif\if@negarg
<def>\global\let\if@negarg\if@negarg
<def>\global\let\@negargtrue\@negargtrue
<def>\global\let\@negargfalse\@negargfalse
```

```
\gdef\line(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@linelen #3\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vline
      \else \ifnum\@yarg =\z@ \@hline \else \@sline\fi
    \fi
  \fi}
```

```
\gdef\@sline{%
  \ifnum\@xarg<\z@ \@negargtrue \@xarg -\@xarg \@yyarg -\@yarg
  \else \@negargfalse \@yyarg \@yarg \fi
\ifnum \@yyarg >\z@ \@tempcnta\@yyarg \else \@tempcnta -\@yyarg \fi
\ifnum\@tempcnta>6 \@badlinearg\@tempcnta\z@ \fi
\ifnum\@xarg>6 \@badlinearg\@xarg \@ne \fi
\setbox\@linechar\hbox{\@linefnt\@getlinechar(\@xarg,\@yyarg)}%
\ifnum \@yarg >\z@ \let\@upordown\raise \@clnht\z@
   \else\let\@upordown\lower \@clnht \ht\@linechar\fi
\@clnwd \wd\@linechar
\if@negarg
  \hskip -\wd\@linechar \def\reserved@a{\hskip -2\wd\@linechar}%
\else
     \let\reserved@a\relax
\fi
\@whiledim \@clnwd <\@linelen \do
  {\@upordown\@clnht\copy\@linechar
   \reserved@a
   \advance\@clnht \ht\@linechar
   \advance\@clnwd \wd\@linechar}%
\advance\@clnht -\ht\@linechar
\advance\@clnwd -\wd\@linechar
\@tempdima\@linelen\advance\@tempdima -\@clnwd
\@tempdimb\@tempdima\advance\@tempdimb -\wd\@linechar
\if@negarg \hskip -\@tempdimb \else \hskip \@tempdimb \fi
\multiply\@tempdima \@m
\@tempcnta \@tempdima
\@tempdima \wd\@linechar \divide\@tempcnta \@tempdima
\@tempdima \ht\@linechar \multiply\@tempdima \@tempcnta
\divide\@tempdima \@m
\advance\@clnht \@tempdima
\ifdim \@linelen <\wd\@linechar
   \hskip \wd\@linechar
  \else\@upordown\@clnht\copy\@linechar\fi}
```

```
\gdef\@hline{\ifnum \@xarg <\z@ \hskip -\@linelen \fi
\vrule \@height \@halfwidth \@depth \@halfwidth \@width \@linelen
\ifnum \@xarg <\z@ \hskip -\@linelen \fi}
```

```
\gdef\@getlinechar(#1,#2){\@tempcnta#1\relax\multiply\@tempcnta 8%
  \advance\@tempcnta -9\ifnum #2>\z@ \advance\@tempcnta #2\relax\else
  \advance\@tempcnta -#2\relax\advance\@tempcnta 64 \fi
  \char\@tempcnta}
```

```
\gdef\vector(#1,#2)#3{\@xarg #1\relax \@yarg #2\relax
  \@tempcnta \ifnum\@xarg<\z@ -\@xarg\else\@xarg\fi
  \ifnum\@tempcnta<5\relax
  \@linelen #3\unitlength
  \ifdim\@linelen<\z@\@badlinearg\else
    \ifnum\@xarg =\z@ \@vvector
```

```
      \else \ifnum\@yarg =\z@ \@hvector \else \@svector\fi
    \fi
  \fi
  \else\@badlinearg\fi}
\gdef\@hvector{\@hline\hb@xt@\z@{\@linefnt
 \ifnum \@xarg <\z@ \@getlarrow(1,0)\hss\else
    \hss\@getrarrow(1,0)\fi}}
\gdef\@vvector{\ifnum \@yarg <\z@ \@downvector \else \@upvector \fi}
\gdef\@svector{\@sline
  \@tempcnta\@yarg \ifnum\@tempcnta <\z@ \@tempcnta -\@tempcnta\fi
  \ifnum\@tempcnta <5%
    \hskip -\wd\@linechar
    \@upordown\@clnht \hbox{\@linefnt  \if@negarg
    \@getlarrow(\@xarg,\@yyarg)\else \@getrarrow(\@xarg,\@yyarg)\fi}%
  \else\@badlinearg\fi}
\gdef\@getlarrow(#1,#2){\ifnum #2=\z@ \@tempcnta'33 \else
  \@tempcnta #1\relax\multiply\@tempcnta \sixt@@n \advance\@tempcnta
  -9 \@tempcntb #2\relax\multiply\@tempcntb \tw@
  \ifnum \@tempcntb >\z@ \advance\@tempcnta \@tempcntb
  \else\advance\@tempcnta -\@tempcntb\advance\@tempcnta 64
  \fi\fi\char\@tempcnta}
\gdef\@getrarrow(#1,#2){\@tempcntb #2\relax
\ifnum\@tempcntb <\z@ \@tempcntb -\@tempcntb\relax\fi
\ifcase \@tempcntb\relax \@tempcnta'55 \or
\ifnum #1<\thr@@ \@tempcnta #1\relax\multiply\@tempcnta
24\advance\@tempcnta -6 \else \ifnum #1=\thr@@ \@tempcnta 49
\else\@tempcnta 58 \fi\fi\or
\ifnum #1<\thr@@ \@tempcnta=#1\relax\multiply\@tempcnta
24\advance\@tempcnta -\thr@@ \else \@tempcnta 51 \fi\or
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta -\tw@ \else
\@tempcnta #1\relax\multiply\@tempcnta
\sixt@@n \advance\@tempcnta 7 \fi\ifnum #2<\z@ \advance\@tempcnta 64 \fi
\char\@tempcnta}
\gdef\@vline{\ifnum \@yarg <\z@ \@downline \else \@upline\fi}
\gdef\@upline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \@linelen \@depth \z@\hss}}
\gdef\@downline{%
  \hb@xt@\z@{\hskip -\@halfwidth \vrule \@width \@wholewidth
   \@height \z@ \@depth \@linelen \hss}}
\gdef\@upvector{\@upline\setbox\@tempboxa\hbox{\@linefnt\char'66}\raise
     \@linelen \hb@xt@\z@{\lower \ht\@tempboxa\box\@tempboxa\hss}}
```

```
\gdef\@downvector{\@downline\lower \@linelen
      \hb@xt@\z@{\@linefnt\char'77\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 < \@dascnt
                  do \copy\@dashbox
                     \@tempcnta := \@tempcnta + 1
                  od
               }
      \@tempcnta := 0
      put(0,Y){\hskip \@dashdim
                while \@tempcnta < \@dascnt
                  do \copy\@dashbox
                     \@tempcnta := \@tempcnta + 1
                  od
               }
```

```
%% vertical dashes
      \@dashdim := Y * \unitlength
      \@dashcnt := \@dashdim + 200 % to prevent roundoff error
      \@dashdim := D * \unitlength
      \@dashcnt := \@dashcnt / \@dashdim
      if \@dashcnt is odd
        then \@dashdim := 0pt
             \@dashcnt := (\@dashcnt + 1) / 2
        else \@dashdim := \@dashdim / 2
             \@dashcnt := \@dashcnt / 2 - 1
             \box\@dashbox   := \hbox{\hskip -\@halfwidth
                                      \vrule width \@wholewidth
                                             height \@dashdim  }
             \put(0,0){\copy\@dashbox}
             \put(X,0){\copy\@dashbox}
             \put(0,Y){\lower\@dashdim\copy\@dashbox}
             \put(X,Y){\lower\@dashdim\copy\@dashbox}
             \@dashdim := 3 * \@dashdim
      fi
      \box\@dashbox := \hbox{\vrule width \@wholewidth
                                   height D * \unitlength       }
      \@tempcnta := 0
      put(0,0){\hskip -\halfwidth
               \vbox{while \@tempcnta < \@dashcnt
                       do \vskip D*\unitlength
                          \copy\@dashbox
                          \@tempcnta := \@tempcnta + 1
                       od
                      \vskip \@dashdim
                     } }
      \@tempcnta := 0
      put(X,0){\hskip -\halfwidth
               \vbox{while \@tempcnta < \@dashcnt
                       do \vskip D*\unitlength
                          \copy\@dashbox
                          \@tempcnta := \@tempcnta + 1
                       od
                      \vskip \@dashdim
                     }
               }
   }     % END DASHES

 \@imakepicbox(X,Y)
END
```

```
\gdef\dashbox#1(#2,#3){\leavevmode\hb@xt@\z@{\baselineskip \z@skip
\lineskip \z@skip
\@dashdim #2\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
\ifodd\@dashcnt\@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else \divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
\@width \@dashdim}\put(0,0){\copy\@dashbox}%
\put(0,#3){\copy\@dashbox}%
\put(#2,0){\hskip-\@dashdim\copy\@dashbox}%
\put(#2,#3){\hskip-\@dashdim\box\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox \hbox{\vrule \@height \@halfwidth \@depth \@halfwidth
\@width #1\unitlength\hskip #1\unitlength}\@tempcnta\z@
\put(0,0){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}\@tempcnta\z@
\put(0,#3){\hskip\@dashdim \@whilenum \@tempcnta <\@dashcnt
\do{\copy\@dashbox\advance\@tempcnta \@ne }}%
\@dashdim #3\unitlength
\@dashcnt \@dashdim \advance\@dashcnt 200
\@dashdim #1\unitlength\divide\@dashcnt \@dashdim
\ifodd\@dashcnt \@dashdim \z@
\advance\@dashcnt \@ne \divide\@dashcnt \tw@
\else
\divide\@dashdim \tw@ \divide\@dashcnt \tw@
\advance\@dashcnt \m@ne
\setbox\@dashbox\hbox{\hskip -\@halfwidth
\vrule \@width \@wholewidth
\@height \@dashdim}\put(0,0){\copy\@dashbox}%
\put(#2,0){\copy\@dashbox}%
\put(0,#3){\lower\@dashdim\copy\@dashbox}%
\put(#2,#3){\lower\@dashdim\copy\@dashbox}%
\multiply\@dashdim \thr@@
\fi
\setbox\@dashbox\hbox{\vrule \@width \@wholewidth
\@height #1\unitlength}\@tempcnta\z@
\put(0,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta <\@dashcnt
\do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}\@tempcnta\z@
\put(#2,0){\hskip -\@halfwidth \vbox{\@whilenum \@tempcnta<\@dashcnt
\do{\vskip #1\unitlength\copy\@dashbox\advance\@tempcnta \@ne }%
\vskip\@dashdim}}\@makepicbox(#2,#3)}
```

CIRCLES AND OVALS

```
USER COMMANDS:

 \circle{D} : Produces the circle with the diameter as close as
              possible to D * \unitlength.  \put(X,Y){\circle{D}}
              puts the circle with its center at (X,Y).

 \oval(X,Y) : Makes an oval as round as possible that fits in the
              rectangle of width X * \unitlength and height
              Y * \unitlength. The reference point is the center.

\oval(X,Y)[POS] : Save as \oval(X,Y) except it draws only the
                  half or quadrant of the oval indicated by POS.
                  E.G., \oval(X,Y)[t] draws just the top half
                  and \oval(X,Y)[br] draws just the bottom right
                  quadrant.  In all cases, the reference point is
                  the same as the unqualified \oval(X,Y) command.

\@ovvert {DELTA1} {DELTA2} : Makes a vbox containing either the left
    side or the right side of the oval being constructed.  The baseline
    will coincide with the outside bottom edge of the oval; the left
    side of the box will coincide with the left edge of the vertical
    rule.  The width of the box will be \@tempdima.
    DELTA1 and DELTA2 are added to the character number in
\@tempcnta
    to get the characters for the top and bottom quarter circle pieces.

\@ovhorz : Makes an hbox containing the straight rule for either the
         top or the bottom of the oval being constructed.  The baseline
         will coincide with bottom edge of the rule; the left side of
         the box will coincide with the left side of the oval.
         The width of the box will be \@ovxx.

\@getcirc {DIAM} : Sets \@tempcnta to the character number
              of the top-right quarter circle with the largest
              diameter less than or equal to DIAM.
              Sets \@tempboxa to an hbox containing that character.
              Sets \@tempdima to \wd \@tempboxa, which is the distance
              from the circle's left outside edge to its right
              inside edge.
              (These characters are like those described in the
              TeXbook, pp. 389-90.)

\@getcirc {DIAM} ==
  BEGIN
    \@tempcnta        := integer coercion of (DIAM + 2pt)
```

```
                                      + 2pt added 1 Nov
88
     \@tempcnta        := \@tempcnta / integer coercion of 4pt
     if \@tempcnta > 10
       then \@tempcnta := 10 fi
     if \@tempcnta > 0
       then \@tempcnta := \@tempcnta-1
       else LaTeX Warning: Oval too small.
     fi
     \@tempcnta        := 4 * \@tempcnta
     \@tempboxa        := \hbox{\@circlefnt \char \@tempcnta}
     \@tempdima        := \wd \@tempboxa
   END

 \@put{X}{Y}{OBJ} ==
   BEGIN
     \raise Y \hb@xt@ 0pt{\hskip X OBJ \hss}
   END

 \@oval(X,Y)[POS] ==
   BEGIN
     \begingroup
       \boxmaxdepth := \maxdimen
       @ovt := @ovb := @ovl := @ovr := true
       for all E in POS
         do  @ovE := false od
       \@ovxx      := X * \unitlength
       \@ovyy      := Y * \unitlength
       \@tempdimb := min(\@ovxx,\@ovyy)
       \@getcirc{\@tempdimb-2pt}   %% "-2pt" added 7 Dec 89
       \@ovro     := \ht \@tempboxa
       \@ovri     := \dp \@tempboxa
       \@ovdx     := \@ovxx - \@tempdima
       \@ovdx     := \@ovdx/2
       \@ovdy     := \@ovyy - \@tempdima
       \@ovdy     := \@ovyy/2
       \@circlefnt
       \@tempboxa :=
           \hbox{
                 if @ovr
                   then \@ovvert{3}{2} \kern -\@tempdima
                 fi
                 if @ovl
                   then \kern \@ovxx \@ovvert{0}{1} \kern
-\@tempdima
                        \kern -\@ovxx
```

```
                fi
                if @ovt
                  then \@ovhorz \kern -\@ovxx
                fi
                if @ovb
                  then \raise \@ovyy \@ovhorz
                fi
               }
      \@ovdx    := \@ovdx + \@ovro
      \@ovdy    := \@ovdy + \@ovro
     \ht\@tempboxa := \dp\@tempboxa := 0
      \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}
   \endgroup
  END

\@ovvert {DELTA1} {DELTA2} ==
  BEGIN
     \vbox to \@ovyy {
                     if @ovb
                       then \@tempcntb := \@tempcnta + DELTA1
                            \kern -\@ovro
                            \hbox { \char \@tempcntb }
                            \nointerlineskip
                       else \kern \@ovri \kern \@ovdy
                     fi
                     \leaders \vrule width \@wholewidth \vfil
                     \nointerlineskip
                     if @ovt
                       then \@tempcntb := \@tempcnta + DELTA2
                            \hbox { \char \@tempcntb }
                       else \kern \@ovdy \kern \@ovro
                     fi
                    }
  END

\@ovhorz ==
  BEGIN
   \hb@xt@ \@ovxx{
                  \kern \@ovro
                  if @ovr
                    then
                    else \kern \@ovdx
                  fi
                  \leaders \hrule height \@wholewidth \hfil
                  if @ovl
                    then
```

```
                  else \kern \@ovdx
                 fi
                 \kern \@ovri
                }
  END

\circle{DIAM} ==
  BEGIN
   \begingroup
   \boxmaxdepth := maxdimen
   \@tempdimb := DIAM *\unitlength
   if \@tempdimb > 15.5pt
     then \@getcirc{\@tempdimb}
          \@ovro := \ht \@tempboxa
          \@tempboxa := \hbox{
                  \@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
```

```
\newif\if@ovt
⟨def⟩\global\let\if@ovt
⟨def⟩\global\let\@ovttrue\@ovttrue
⟨def⟩\global\let\@ovtfalse\@ovtfalse
\newif\if@ovb
⟨def⟩\global\let\if@ovb
⟨def⟩\global\let\@ovbtrue\@ovbtrue
⟨def⟩\global\let\@ovbfalse\@ovbfalse
\newif\if@ovl
⟨def⟩\global\let\if@ovl
⟨def⟩\global\let\@ovltrue\@ovltrue
⟨def⟩\global\let\@ovlfalse\@ovlfalse
\newif\if@ovr
⟨def⟩\global\let\if@ovr
⟨def⟩\global\let\@ovrtrue\@ovrtrue
⟨def⟩\global\let\@ovbfalse\@ovbfalse
⟨/2ekernel | def⟩
⟨*2ekernel | autoload⟩
\newdimen\@ovxx
\newdimen\@ovyy
\newdimen\@ovdx
\newdimen\@ovdy
\newdimen\@ovro
\newdimen\@ovri
⟨/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.

```
⟨*2ekernel | def⟩
\gdef\@getcirc#1{\@tempdima #1\relax \advance\@tempdima 2\p@
  \@tempcnta\@tempdima
  \@tempdima 4\p@ \divide\@tempcnta\@tempdima
  \ifnum \@tempcnta >10\relax \@tempcnta 10\relax\fi
  \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne
    \else \@latex@warning{Oval too small}\fi
  \multiply\@tempcnta 4\relax
  \setbox \@tempboxa \hbox{\@circlefnt
  \char \@tempcnta}\@tempdima \wd \@tempboxa}

\gdef\@put#1#2#3{\raise #2\hb@xt@\z@{\hskip #1#3\hss}}

\gdef\oval(#1,#2){\@ifnextchar[{\@oval(#1,#2)}{\@oval(#1,#2)[]}}

\gdef\@oval(#1,#2)[#3]{\begingroup\boxmaxdepth \maxdimen
  \@ovttrue \@ovbtrue \@ovltrue \@ovrtrue
  \@tfor\reserved@a :=#3\do{\csname @ov\reserved@a false\endcsname}%
  \@ovxx
  #1\unitlength \@ovyy #2\unitlength
```

```
  \@tempdimb \ifdim \@ovyy >\@ovxx \@ovxx\else \@ovyy \fi
  \advance \@tempdimb -2\p@
  \@getcirc \@tempdimb
  \@ovro \ht\@tempboxa \@ovri \dp\@tempboxa
  \@ovdx\@ovxx \advance\@ovdx -\@tempdima \divide\@ovdx \tw@
  \@ovdy\@ovyy \advance\@ovdy -\@tempdima \divide\@ovdy \tw@
  \@circlefnt \setbox\@tempboxa
  \hbox{\if@ovr \@ovvert32\kern -\@tempdima \fi
  \if@ovl \kern \@ovxx \@ovvert01\kern -\@tempdima \kern -\@ovxx \fi
  \if@ovt \@ovhorz \kern -\@ovxx \fi
  \if@ovb \raise \@ovyy \@ovhorz \fi}\advance\@ovdx\@ovro
  \advance\@ovdy\@ovro \ht\@tempboxa\z@ \dp\@tempboxa\z@
  \@put{-\@ovdx}{-\@ovdy}{\box\@tempboxa}%
  \endgroup}
\gdef\@ovvert#1#2{\vbox to\@ovyy{%
    \if@ovb \@tempcntb \@tempcnta \advance \@tempcntb #1\relax
      \kern -\@ovro \hbox{\char \@tempcntb}\nointerlineskip
    \else \kern \@ovri \kern \@ovdy \fi
    \leaders\vrule \@width \@wholewidth\vfil \nointerlineskip
    \if@ovt \@tempcntb \@tempcnta \advance \@tempcntb #2\relax
      \hbox{\char \@tempcntb}%
    \else \kern \@ovdy \kern \@ovro \fi}}
\gdef\@ovhorz{\hb@xt@\@ovxx{\kern \@ovro
    \if@ovr \else \kern \@ovdx \fi
    \leaders \hrule \@height \@wholewidth \hfil
    \if@ovl \else \kern \@ovdx \fi
    \kern \@ovri}}
\gdef\circle{\@inmatherr\circle\@ifstar\@dot\@circle}
\gdef\@circle#1{%
  \begingroup \boxmaxdepth \maxdimen \@tempdimb #1\unitlength
   \ifdim \@tempdimb >15.5\p@ \@getcirc\@tempdimb
      \@ovro\ht\@tempboxa
     \setbox\@tempboxa\hbox{\@circlefnt
      \advance\@tempcnta\tw@ \char \@tempcnta
      \advance\@tempcnta\m@ne \char \@tempcnta \kern -2\@tempdima
      \advance\@tempcnta\tw@
      \raise \@tempdima \hbox{\char\@tempcnta}\raise \@tempdima
        \box\@tempboxa}\ht\@tempboxa\z@ \dp\@tempboxa\z@
      \@put{-\@ovro}{-\@ovro}{\box\@tempboxa}%
   \else  \@circ\@tempdimb{96}\fi\endgroup}
\gdef\@dot#1{\@tempdimb #1\unitlength \@circ\@tempdimb{112}}
\gdef\@circ#1#2{\@tempdima #1\relax \advance\@tempdima .5\p@
   \@tempcnta\@tempdima \@tempdima \p@
   \divide\@tempcnta\@tempdima
   \ifnum\@tempcnta >15\relax \@tempcnta 15\relax \fi
   \ifnum \@tempcnta >\z@ \advance\@tempcnta\m@ne\fi
   \advance\@tempcnta #2\relax
   \@circlefnt \char\@tempcnta}
```

INITIALIZATION

```
%\thinlines
\global\let\@linefnt\tenln \global\let\@circlefnt\tencirc
\global\@wholewidth\fontdimen8\tenln \global\@halfwidth .5\@wholewidth
⟨/2ekernel | def⟩
⟨*2ekernel | autoload⟩
\newcount\@xarg
\newcount\@yarg
\newcount\@yyarg
\newcount\@multicnt
\newdimen\@xdim
\newdimen\@ydim
\newbox\@linechar
\newdimen\@linelen
\newdimen\@clnwd
\newdimen\@clnht
\newdimen\@dashdim
\newbox\@dashbox
\newcount\@dashcnt
⟨/2ekernel | autoload⟩
```

56.1 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.

```
⟨*2ekernel | def⟩
⟨def⟩\ifx\qbeziermax\@undefined
\gdef\qbeziermax{500}
⟨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 `(`.

```
\newcommand\qbezier[2][0]{\bezier{#1}#2}
⟨def⟩\global\let\qbezier\qbezier
⟨def⟩\global\expandafter\let\csname\string\qbezier
⟨def⟩            \expandafter\endcsname\csname\string\qbezier\endcsname
```

`\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 `(`.

```
\gdef\bezier#1)#2(#3)#4({\@bezier#1)(#3)(}
```

`\@bezier`

```
\gdef\@bezier#1(#2,#3)(#4,#5)(#6,#7){%
 \ifnum #1=\z@
     \@ovxx #4\unitlength
       \advance\@ovxx -#2\unitlength
       \ifdim \@ovxx<\z@ \@ovxx -\@ovxx \fi
     \@ovdx #6\unitlength
```

```
        \advance\@ovdx -#4\unitlength
        \ifdim \@ovdx<\z@ \@ovdx -\@ovdx \fi
        \ifdim \@ovxx<\@ovdx \@ovxx \@ovdx \fi
      \@ovyy #5\unitlength
        \advance\@ovyy -#3\unitlength
        \ifdim \@ovyy<\z@ \@ovyy -\@ovyy \fi
      \@ovdy #7\unitlength
        \advance\@ovdy -#5\unitlength
        \ifdim \@ovdy<\z@  \@ovdy -\@ovdy \fi
        \ifdim \@ovyy<\@ovdy \@ovyy  \@ovdy \fi
      \@multicnt
         \ifdim \@ovxx>\@ovyy \@ovxx \else \@ovyy \fi
      \@ovxx .5\@halfwidth \divide\@multicnt\@ovxx
      \ifnum \qbeziermax<\@multicnt \@multicnt\qbeziermax\relax \fi
  \else \@multicnt#1\relax \fi
  \@tempcnta\@multicnt \advance\@tempcnta\@ne
  \@ovdx #4\unitlength \advance\@ovdx -#2\unitlength
      \multiply\@ovdx \tw@
  \@ovxx #6\unitlength \advance\@ovxx -#2\unitlength
      \advance\@ovxx -\@ovdx \divide\@ovxx\@multicnt
  \@ovdy #5\unitlength \advance\@ovdy -#3\unitlength
       \multiply\@ovdy \tw@
  \@ovyy #7\unitlength \advance\@ovyy -#3\unitlength
      \advance\@ovyy -\@ovdy \divide\@ovyy\@multicnt
  \setbox\@tempboxa\hbox{\vrule \@height\@halfwidth
                               \@depth \@halfwidth
                               \@width \@wholewidth}%
   \put(#2,#3){%
     \count@\z@
     \@whilenum{\count@<\@tempcnta}\do
        {\@xdim\count@\@ovxx
           \advance\@xdim\@ovdx
           \divide\@xdim\@multicnt
           \multiply\@xdim\count@
         \@ydim\count@\@ovyy
            \advance\@ydim\@ovdy
            \divide\@ydim\@multicnt
            \multiply\@ydim\count@
         \raise \@ydim
            \hb@xt@\z@{\kern\@xdim
                        \unhcopy\@tempboxa\hss}%
         \advance\count@\@ne}}}
⟨/2ekernel | def⟩
⟨def⟩\endgroup
```

File E
ltthm.dtx

57 Theorem Environments

The user creates his own theorem-like environments with the command
`\newtheorem{`*name*`}{`*text*`}[`*counter*`]` or
`\newtheorem{`*name*`}[`*oldname*`]{`*text*`}`
This defines the environment ⟨*name*⟩ to be just as one would expect a theorem environment to be, except that it prints ⟨*text*⟩ instead of "Theorem".

If ⟨*oldname*⟩ is given, then environments ⟨*name*⟩ and ⟨*oldname*⟩ use the same counter, so using a ⟨*name*⟩ environment advances the number of the next ⟨*name*⟩ environment, and vice-versa.

If ⟨*counter*⟩ is given, then environment ⟨*name*⟩ is numbered within ⟨*counter*⟩.

E.g., if ⟨*counter*⟩ = `subsection`, then the first ⟨*name*⟩ in subsection 7.2 is numbered ⟨*text*⟩ 7.2.1.

The way ⟨*name*⟩ environments are numbered can be changed by redefining `\the`⟨*name*⟩.

DOCUMENT STYLE PARAMETERS

`\@thmcounter{`COUNTER`}` : A command such that
`\edef\theCOUNTER{\@thmcounter{`COUNTER`}}`
defines `\theCOUNTER` to produce a number for a theorem environment.
The default is:
BEGIN `\noexpand\arabic{`COUNTER`}` END

`\@thmcountersep` : A separator placed between a theorem number and
the number of the counter within which it is numbered.
E.g., to make the third theorem of section 7.2 be numbered
7.2-3, `\@thmcountersep` should be `\def`'ed to '-'. Its
default is '.'.

`\@begintheorem{`NAME`}{`NUMBER`}` : A command that begins a
theorem
environment for a 'theorem' named 'NAME NUMBER' –
e.g., `\@begintheorem{`Lemma`}{`3.7`}` starts Lemma 3.7.

`\@opargbegintheorem{`NAME`}{`NUMBER`}{`OPARG`}` :
A command that begins a theorem
environment for a 'theorem' named 'NAME NUMBER' with
optional
argument OPARG – e.g., `\@begintheorem{`Lemma`}{`3.7`}{`Jones`}`
starts 'Lemma 3.7 (Jones):'.

\@endtheorem : A command that ends a theorem environment.

```
\newtheorem{NAME}{TEXT}[COUNTER] ==
  BEGIN
    if \NAME is definable
      then \@definecounter{NAME}
           if COUNTER present
             then \@newctr{NAME}[COUNTER] fi
                  \theNAME ==  BEGIN \theCOUNTER \@thmcountersep
                                    eval\@thmcounter{NAME}
END
             else \theNAME ==  BEGIN eval\@thmcounter{NAME} END
           \NAME == \@thm{NAME}{TEXT}
           \endNAME == \@endtheorem
      else  error
    fi
  END

\newtheorem{NAME}[OLDNAME]{TEXT}==
  BEGIN
    if counter OLDNAME 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
```

```
⟨*2ekernel⟩
\def\newtheorem#1{\@ifnextchar[{\@othm{#1}}{\@nthm{#1}}}

\def\@nthm#1#2{%
\@ifnextchar[{\@xnthm{#1}{#2}}{\@ynthm{#1}{#2}}}
```

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`)

```
\def\@xnthm#1#2[#3]{\expandafter\@ifdefinable\csname #1\endcsname
{\@definecounter{#1}\@newctr{#1}[#3]%
\expandafter\xdef\csname the#1\endcsname{\expandafter\noexpand
  \csname the#3\endcsname \@thmcountersep \@thmcounter{#1}}%
\global\@namedef{#1}{%
  \@thm{#1}{#2}}\global\@namedef{end#1}{\@endtheorem}}}

\def\@ynthm#1#2{\expandafter\@ifdefinable\csname #1\endcsname
{\@definecounter{#1}%
\expandafter\xdef\csname the#1\endcsname{\@thmcounter{#1}}%
\global\@namedef{#1}{%
  \@thm{#1}{#2}}\global\@namedef{end#1}{\@endtheorem}}}
```

`\@othm`

```
\def\@othm#1[#2]#3{%
  \@ifundefined{c@#2}{\@nocounterr{#2}}%
  {\expandafter\@ifdefinable\csname #1\endcsname
  {\global\@namedef{the#1}{\@nameuse{the#2}}%
\global\@namedef{#1}{\@thm{#2}{#3}}%
\global\@namedef{end#1}{\@endtheorem}}}}

\def\@thm#1#2{\refstepcounter
    {#1}\@ifnextchar[{\@ythm{#1}{#2}}{\@xthm{#1}{#2}}}

\def\@xthm#1#2{\@begintheorem{#2}{\csname the#1\endcsname}\ignorespaces}
\def\@ythm#1#2[#3]{\@opargbegintheorem{#2}{\csname
       the#1\endcsname}{#3}\ignorespaces}
```

DEFAULT VALUES

```
\def\@thmcounter#1{\noexpand\arabic{#1}}
\def\@thmcountersep{.}
```

\@begintheorem
\@opargbegintheorem
\@endtheorem

Providing theorem defaults.

```
\def\@begintheorem#1#2{\trivlist
   \item[\hskip \labelsep{\bfseries #1\ #2}]\itshape}
\def\@opargbegintheorem#1#2#3{\trivlist
      \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\itshape}
\def\@endtheorem{\endtrivlist}
⟨/2ekernel⟩
```

File F

ltsect.dtx

58 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` LaTeX 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.

```
⟨*2ekernel⟩
\message{title,}
```

58.1 The Title

`\title` `\author` `\date` `\thanks` `\and` `\maketitle`

The user defines the title, author, date by the declarations `\title{`*name*`}`, `\author{`*name*`}` and `{`*date*`}`. Inside these, he can use the `\thanks{`*footnote text*`}` command to make acknowledgements, notice of address, etc. in a footnote. If there are multiple authors, they have to be separated with the `\and` command.

And finally, the `\maketitle` command produces the actual title, using the information prevoisly saved with the other commands.

`\title` `\@title`

`\title` for use in `\maketitle`. If not given `\maketitle` will produce an error message.

```
\def\title#1{\gdef\@title{#1}}
\def\@title{\@latex@error{No \noexpand\title given}\@ehc}
```

`\author` `\@author`

`\author` for use in `\maketitle`. If not given `\maketitle` will produce a warning message.

```
\def\author#1{\gdef\@author{#1}}
\def\@author{\@latex@warning@no@line{No \noexpand\author given}}
```

`\date` `\@date`

`\date` for use in `\maketitle`. If not given `\maketitle` will produce `\today` as the default.

```
\def\date#1{\gdef\@date{#1}}
\gdef\@date{\today}
```

`\thanks`

```
\def\thanks#1{\footnotemark
    \protected@xdef\@thanks{\@thanks
        \protect\footnotetext[\the\c@footnote]{#1}}%
}
```

`\@thanks`

```
\def\@thanks{}
```

\and

```
\def\and{%%                           % \begin{tabular}
\end{tabular}\hskip 1em \@plus.17fil%
\begin{tabular}[t]{c}%% \end{tabular}
}
```

```
\message{sectioning,}
```

58.2 Sectioning

\@secpenalty

```
\newcount\@secpenalty
\@secpenalty = -300
```

\if@noskipsec
\@noskipsectrue

```
\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

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
```

```
\def\@startsection#1#2#3#4#5#6{\if@noskipsec \leavevmode \fi
  \par \@tempskipa #4\relax
  \@afterindenttrue
  \ifdim \@tempskipa <\z@
    \@tempskipa -\@tempskipa \@afterindentfalse
  \fi
  \if@nobreak \everypar{}\else
    \addpenalty\@secpenalty\addvspace\@tempskipa
  \fi
  \@ifstar
    {\@ssect{#3}{#4}{#5}{#6}}%
    {\@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 \begingroup
              STYLE
              \@hangfrom{\hskip INDENT\@svsec}
              {\interlinepenalty 10000 ARG2\par}
           \endgroup
           \NAMEmark{ARG1}
           \addcontentsline{toc}{NAME}
              { IF  LEVEL > \c@secnumdepth
                  ELSE \protect\numberline{\theNAME}  FI
                ARG1 }
      ELSE \@svsechd == BEGIN  STYLE
                              \hskip INDENT\@svsec
                              ARG2
                              \NAMEmark{ARG1}
                              \addcontentsline{toc}{NAME}
                                 { IF  LEVEL > \c@secnumdepth
                                     ELSE

\protect\numberline{\theNAME}
                                     FI
                                   ARG1 }
                         END
    FI
    \@xsect{AFTERSKIP}
 END
```

```
\def\@sect#1#2#3#4#5#6[#7]#8{%
    \ifnum #2>\c@secnumdepth
      \let\@svsec\@empty\else
      \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.

```
      \protected@edef\@svsec{\@seccntformat{#1}\relax}%
    \fi
    \@tempskipa #5\relax
     \ifdim \@tempskipa>\z@
       \begingroup #6\relax
         \@hangfrom{\hskip #3\relax\@svsec}%
                   {\interlinepenalty \@M #8\par}%
       \endgroup
      \csname #1mark\endcsname{#7}\addcontentsline
        {toc}{#1}{\ifnum #2>\c@secnumdepth \else
                      \protect\numberline{\csname the#1\endcsname}\fi
```

```
                   #7}\else
        \def\@svsechd{#6\hskip #3\relax  %% \relax added 2 May 90
                   \@svsec #8\csname #1mark\endcsname
                      {#7}\addcontentsline
                           {toc}{#1}{\ifnum #2>\c@secnumdepth \else
                           \protect\numberline{\csname the#1\endcsname}%
                                     \fi
                       #7}}\fi
     \@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

   END
```

```
\def\@xsect#1{\@tempskipa #1\relax
      \ifdim \@tempskipa>\z@
       \par \nobreak
       \vskip \@tempskipa
       \@afterheading
    \else \global\@nobreakfalse \global\@noskipsectrue
       \everypar{\if@noskipsec \global\@noskipsecfalse
                  \clubpenalty\@M \hskip -\parindent
                  \begingroup \@svsechd \endgroup \unskip
                  \@tempskipa #1\relax
                  \hskip -\@tempskipa
```

```
                   \else \clubpenalty \@clubpenalty
                     \everypar{}\fi}\fi\ignorespaces}
```

\@seccntformat This command formats the section number including the space following it.

```
\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
                       FI
                END
 END
```

\@ssect

```
\def\@ssect#1#2#3#4#5{\@tempskipa #3\relax
  \ifdim \@tempskipa>\z@
    \begingroup #4\@hangfrom{\hskip #1}{\interlinepenalty \@M #5\par}%
    \endgroup
  \else \def\@svsechd{#4\hskip #1\relax #5}\fi
   \@xsect{#3}}
```

\if@afterindent
\@afterindenttrue

```
\newif\if@afterindent \@afterindenttrue
```

\@afterheading

```
\def\@afterheading{\global\@nobreaktrue
      \everypar{\if@nobreak
                  \global\@nobreakfalse
                  \clubpenalty \@M
                  \if@afterindent \else {\setbox\z@\lastbox}\fi
                \else \clubpenalty \@clubpenalty
                   \everypar{}\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.

```
\def\@hangfrom#1{\setbox\@tempboxa\hbox{{#1}}%
      \hangindent \wd\@tempboxa\noindent\box\@tempboxa}
```

\c@secnumdepth
\c@tocdepth

```
\newcount\c@secnumdepth
\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 \`⟨*starcmd*⟩ `\`⟨*unstarcmd*⟩ `}`
2. `\def\`⟨*starcmd*⟩`[#1]#2{` ... `}` `%` Command to define `\chapter[`...`]{`...`}`
3. `\def\`⟨*unstarcmd*⟩`#1{` ... `}` `%` Command to define `\chapter*{`...`}`

```
\def\secdef#1#2{\@ifstar{#2}{\@dblarg{#1}}}
```

58.2.1 Initializations

\sectionmark
\subsectionmark
\subsubsectionmark
\paragraphmark
\subparagraphmark

```
\def\sectionmark#1{}
\def\subsectionmark#1{}
\def\subsubsectionmark#1{}
\def\paragraphmark#1{}
\def\subparagraphmark#1{}
```

```
\message{contents,}
```

58.3 Table of Contents etc.

58.3.1 Convention

`\tf@`⟨*foo*⟩ = file number for output for table foo. The file is opened only if `@filesw` = `true`.

58.3.2 Commands

A `\l@`⟨*type*⟩`{`⟨*entry*⟩`}{`⟨*page*⟩`}` Macro needs to defined by document style for making an entry of type ⟨*type*⟩ in a table of contents, etc. E.g., the document style should define `\l@chapter`, `\l@section`, etc.

Note: When the `\protect` command is used in the ⟨*entry*⟩ or ⟨*text*⟩ of one of the commands below, it causes the following control sequence to be written on the file without being expanded. The sequence will be expanded when the table of contents entry is processed.

Surprise: Inside an `\addcontentsline` or `\addtocontents` command argument, the commands: `\index`, `\glossary`, and `\label` are no-ops . This could cause a problem if the user puts an `\index` or `\label` into one of the commands he writes, or into the optional 'short version' argument of a `\section` or `\caption` command.

`\@starttoc` The `\@starttoc{`*ext*`}` command is used to define the commands: `\tableofcontents`, `\listoffigures`, etc.

For example: `\@starttoc{lof}` is used in `\listoffigures`. This command reads the `.`⟨*ext*⟩ file and sets up to write the new `.`⟨*ext*⟩ file.

```
\@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
```

```
\def\@starttoc#1{\begingroup
  \makeatletter
  \@input{\jobname.#1}\if@filesw
  \expandafter\newwrite\csname tf@#1\endcsname
            \immediate\openout
                \csname tf@#1\endcsname \jobname.#1\relax
  \fi \global\@nobreakfalse \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 aplication of `\addtocontents`. Note that `\thepage` is not expandable during `\protected@write` therefore one gets the page number at the time of the `\shipout`.

```
\def\addcontentsline#1#2#3{%
  \addtocontents{#1}{\protect\contentsline{#2}{#3}{\thepage}}%
}
```

\addtocontents The \addtocontents{⟨*table*⟩}{⟨*text*⟩} command adds ⟨*text*⟩ to the .⟨*table*⟩ file, with no page number.

```
\long\def\addtocontents#1#2{%
  \protected@write\@auxout
      {\let\label\@gobble \let\index\@gobble \let\glossary\@gobble}%
      {\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.

```
\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

```
\def\@dottedtocline#1#2#3#4#5{\ifnum #1>\c@tocdepth \else
  \vskip \z@ \@plus.2\p@
  {\leftskip #2\relax \rightskip \@tocrmarg \parfillskip -\rightskip
```

```
    \parindent #2\relax\@afterindenttrue
   \interlinepenalty\@M
   \leavevmode
   \@tempdima #3\relax
   \advance\leftskip \@tempdima \null\hskip -\leftskip
    {#4}\nobreak\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.

```
        \mkern \@dotsep mu\hbox{.}\mkern \@dotsep
        mu$}\hfill \nobreak
            \hb@xt@\@pnumwidth{%
              \hfil\normalfont \normalcolor #5}\par}\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*⟩ flushleft in a box of width `\@tempdima` (Before 25 Jan 88 change, it also added `\@tempdima` to the hanging indentation.)

```
\def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
⟨/2ekernel⟩
```

File G
ltfloat.dtx

59 Floats

The different types of floats are identified by a ⟨*type*⟩ name, which is the name of the counter for that kind of float. For example, figures are of type 'figure' and tables are of type 'table'. Each ⟨*type*⟩ has associated a positive ⟨*type number*⟩, which is a power of two. E.g.,
figures might be have type number 1, tables type number 2, programs type number 4, etc.

The locations where a float can go are specified by a ⟨*placement specifier*⟩, which is a list of the possible locations, each denoted by a letter as follows:

h :	here	— at the current location in the text.
t :	top	— at the top of a text page.
b :	bottom	— at the bottom of a text page.
p :	page	— on a separate float page

In addition, in conjunction with these, you can use '!' which means that the current values of the float positioning parameters are ignored for this float. (Has no effect on 'p', float page positioning.) For example, 'pht' specifies that the float can appear in any of three locations: page, here or top.

59.1 Floating Environments

```
⟨*2ekernel⟩
\message{floats,}
```

Where floats may appear on a page, and how many may appear there are specified by the following float placement parameters. The numbers are named like counters so the user can set them with the ordinary counter-setting commands.

```
\c@topnumber       : Number of floats allowed at the top of a column.
\topfraction       : Fraction of column that can be devoted to floats.
\c@dbltopnumber, \dbltopfraction
                   : Same as above, but for double-column floats.
\c@bottomnumber, \bottomfraction
                   : Same as above for bottom of page.
\c@totalnumber     : Number of floats allowed in a single column,
                           including in-text floats.
\textfraction      :Minimum fraction of column that must contain text.
\floatpagefraction: Minimum fraction of page that must be taken
                           up by float page.
```

\dblfloatpagefraction
: Same as above, for double-column floats.

The document style must define the following.

\fps@TYPE : The default placement specifier for floats of type TYPE.

\ftype@TYPE : The type number for floats of type TYPE.

\ext@TYPE : The file extension indicating the file on which the contents list for float type TYPE is stored.
For example, \ext@figure = 'lof'.

\fnum@TYPE : A macro to generate the figure number for a caption.
For example, \fnum@TYPE == Figure \thefigure.

\@makecaption{NUM}{TEXT} :
A macro to make a caption, with NUM the value
produced by \fnum@... and TEXT the text of the caption.
It can assume it's in a \parbox of the appropriate width.

\@float{TYPE}[PLACEMENT] : This macro begins a float environment for a
single-column float of type TYPE with PLACEMENT as the placement
specifier. The default value of PLACEMENT is defined by
\fps@TYPE. The environment is ended by \end@float.
E.g., \figure == \@float{figure}, \endfigure == \end@float.

```
\@float{TYPE}[PLACEMENT] ==
 BEGIN
   if hmode then \@bsphack
                 \@floatpenalty := -10002
            else \@floatpenalty := -10003
   fi
   \@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 floats'
            fi
     fi
     \@currbox :=G   \color@vbox
                      \normalcolor
                        \vbox{
                         %% 15 Dec 87 –
                         %% removed \boxmaxdepth :=L 0pt
                         %% that made box 0 depth because it screwed
                         %% things up. Instead, added \vskip0pt at
end
                              \hsize = \columnwidth
                              \@parboxrestore
                              \@floatboxreset
   END

  \caption ==
    BEGIN
     \refstepcounter{\@captype}
     \@dblarg{\@caption{\@captype}}
    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.

```
\newcount\@floatpenalty
```

```
\def\caption{\refstepcounter\@captype \@dblarg{\@caption\@captype}}
```

\@caption

```
\long\def\@caption#1[#2]#3{\par\addcontentsline{\csname
  ext@#1\endcsname}{#1}{\protect\numberline{\csname
  the#1\endcsname}{\ignorespaces #2}}\begingroup
    \@parboxrestore
    \normalsize
    \@makecaption{\csname fnum@#1\endcsname}{\ignorespaces #3}\par
  \endgroup}
```

\@float
\@dblflset

```
\def\@float#1{%
  \@ifnextchar[{\@xfloat{#1}}{\edef\reserved@a{\noexpand\@xfloat
    {#1}[\csname fps@#1\endcsname]}\reserved@a}}
```

\@dblfloat

```
\def\@dblfloat{%
  \if@twocolumn\let\reserved@a\@dbflt\else\let\reserved@a\@float\fi
  \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.

```
\def\@xfloat #1[#2]{%
   \def \@captype {#1}%
   \def \@fps {#2}%
   \@onelevel@sanitize \@fps
   \def \reserved@b {!}%
   \ifx \reserved@b \@fps
     \@fpsadddefault
   \else
     \ifx \@fps \@empty
       \@fpsadddefault
     \fi
   \fi
   \ifhmode
     \@bsphack
     \@floatpenalty -\@Mii
   \else
     \@floatpenalty-\@Miii
   \fi
  \ifinner
     \@parmoderr\@floatpenalty\z@
  \else
    \@next\@currbox\@freelist
      {%
       \@tempcnta \sixt@@n
       \expandafter \@tfor \expandafter \reserved@a
         \expandafter :\expandafter =\@fps
         \do
          {%
           \if \reserved@a h%
             \ifodd \@tempcnta
             \else
               \advance \@tempcnta \@ne
             \fi
           \fi
           \if \reserved@a t%
             \@setfpsbit \tw@
           \fi
           \if \reserved@a b%
             \@setfpsbit 4%
           \fi
           \if \reserved@a p%
             \@setfpsbit 8%
           \fi
           \if \reserved@a !%
             \ifnum \@tempcnta>15
```

```
              \advance\@tempcnta -\sixt@@n\relax
            \fi
          \fi
          }%
      \@tempcntb \csname ftype@\@captype \endcsname
      \multiply \@tempcntb \@xxxii
      \advance \@tempcnta \@tempcntb
      \global \count\@currbox \@tempcnta
      }%
    \@fltovf
  \fi
```

The remainder sets up the box in which the float is typeset, and the typesetting enevironment to be used. It is essential to have the extra box to avoid the unwanted space that would otherwise often be put at the top of the float.

It ends with a hook; not sure how useful this is but it is needed at present to deal with double-column floats.

```
  \global \setbox\@currbox
    \color@vbox
      \normalcolor
      \vbox \bgroup
        \hsize\columnwidth \@parboxrestore
        \@floatboxreset
}
```

`\@floatboxreset` The default value of this is empty, for one-column floats.

```
\def \@floatboxreset {%
        \reset@font
        \normalsize
        \@setnobreak
        \@setminipage
}
```

`\@setnobreak`

```
\def \@setnobreak{%
  \if@nobreak
    \def\outer@nobreak{\global\@nobreaktrue}%
    \global\@nobreakfalse
  \fi
}
```

`\@setminipage`

```
\def \@setminipage{%
  \global\@minipagetrue
  \everypar{\global\@minipagefalse\everypar{}}%
}
```

`\end@float`

```
\def\end@float{%
        \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87
```

```
      \global\@minipagefalse
      \outer@nobreak
    \egroup                  %% end of vbox
  \color@endbox
  \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed `\textheight`, otherwise float will never get typeset (91/03/15 FMi).

```
    \@largefloatcheck
    \@cons\@currlist\@currbox
    \ifnum\@floatpenalty <-\@Mii
      \penalty -\@Miv
      \@tempdima\prevdepth    %% saving and restoring \prevdepth added
      \vbox{}%                %% 26 May 87 to prevent extra vertical
      \prevdepth \@tempdima   %% space when used in vertical mode
      \penalty\@floatpenalty

    \else
      \vadjust{\penalty -\@Miv \vbox{}\penalty\@floatpenalty}\@Esphack
    \fi
  \fi
}
```

`\end@dblfloat`

```
\def\end@dblfloat{%
\if@twocolumn
      \par\vskip\z@skip      %% \par\vskip\z@ added 15 Dec 87

    \egroup                  %% end of vbox
    \outer@nobreak
  \color@endbox
  \ifnum\@floatpenalty <\z@
```

We make sure that we never exceed `\textheight`, otherwise float will never get typeset (91/03/15 FMi).

```
    \@largefloatcheck
    \@cons\@dbldeferlist\@currbox
  \fi
```

RmS 92/03/18 changed `\@esphack` to `\@Esphack`.

```
    \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
\else
  \end@float
\fi
}
```

`\outer@nobreak`

```
\let\outer@nobreak\@empty
```

`\@largefloatcheck` This calculates by how much a float is oversize for the page and prints this in a warning message.

```
\def \@largefloatcheck{%
  \ifdim \ht\@currbox>\textheight
    \@tempdima -\textheight
    \advance \@tempdima \ht\@currbox

    \@latex@warning {Float too large for page by \the\@tempdima}%
    \ht\@currbox \textheight
  \fi
}
```

`\@dbflt`
`\@xdblfloat`

```
\def\@dbflt#1{\@ifnextchar[{\@xdblfloat{#1}}{\@xdblfloat{#1}[tp]}}
\def\@xdblfloat#1[#2]{%
  \@xfloat{#1}[#2]\hsize\textwidth\linewidth\textwidth}
```

Moved to ltoutput 93/12/16

```
%\newcount\c@topnumber
%\newcount\c@dbltopnumber
%\newcount\c@bottomnumber
%\newcount\c@totalnumber
```

An analysis of `\@floatplacement`:
This should be called whenever `\@colht` has been set.

```
\def\@floatplacement{\global\@topnum\c@topnumber
    % Textpage bit, global:
   \global\@toproom \topfraction\@colht
   \global\@botnum  \c@bottomnumber
   \global\@botroom \bottomfraction\@colht
   \global\@colnum  \c@totalnumber
    % Floatpage bit, local:
   \@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.

```
\def \@dblfloatplacement {%
```

Textpage bit: global, but need not be.

```
  \global \@dbltopnum \c@dbltopnumber
  \global \@dbltoproom \dbltopfraction\@colht
```

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

```
  \@textmin \@colht
  \advance \@textmin -\@dbltoproom
```

Floatpage bit: must be local.

```
  \@fpmin \dblfloatpagefraction\textheight
  \@fptop \@dblfptop
```

```
  \@fpsep \@dblfpsep
  \@fpbot \@dblfpbot
}
```

MARGINAL NOTES:

Marginal notes use the same mechanism as floats to communicate with the `\output` routine. Marginal notes are distinguished from floats by having a negative placement specification. The command `\marginpar` [LTEXT]{RTEXT} generates a marginal note in a parbox, using LTEXT if it's on the left and RTEXT if it's on the right. (Default is RTEXT = LTEXT.) It uses the following parameters.

```
\marginparwidth : Width of marginal notes.
\marginparsep   : Distance between marginal note and text.
     the page layout to determine how to move the marginal
     note into the margin.   E.g., \@leftmarginskip ==
     \hskip -\marginparwidth \hskip -\marginparsep .
\marginparpush  :  Minimum vertical separation between \marginpar's
```

Marginal notes are normally put on the outside of the page if @mparswitch = true, and on the right if @mparswitch = false. The command `\reversemarginpar` reverses the side where they are put. `\normalmarginpar` undoes `\reversemarginpar`. These commands have no effect for two-column output.

SURPRISE: if two marginal notes appear on the same line of text, then the second one could appear on the next page, in a funny position.

```
\marginpar [LTEXT]{RTEXT} ==
 BEGIN
   if hmode then \@bsphack
                 \@floatpenalty := -10002
            else \@floatpenalty := -10003
   fi
   if inner
      then LaTeX Error: 'Not in outer paragraph mode.'
           \@floatpenalty := 0
      else if \@freelist has two elements:
              then get \@marbox, \@currbox  from \@freelist
                   \count\@marbox :=G -1
              else \@floatpenalty := 0
                   LaTeX Error: 'Too many unprocessed floats'
                   \@currbox, \@marbox := \@tempboxa    %%use \def
```

```
          fi
      fi
      if optional argument
        then %% \@xmpar ==
             \@savemarbox\@marbox{LTEXT}
             \@savemarbox\@currbox{RTEXT}
        else %% \@ympar ==
             \@savemarbox\@marbox{RTEXT}
             \box\@currbox :=G \box\@marbox
     fi
     \@xympar
   END

\reversemarginpar == BEGIN \@mparbottom    :=G 0
                           @reversemargin  :=G true
                     END

\normalmarginpar  == BEGIN \@mparbottom    :=G 0
                           @reversemargin  :=G false
                     END
```

\marginpar

```
\def\marginpar{\ifhmode \@bsphack\@floatpenalty -\@Mii\else
   \@floatpenalty-\@Miii\fi\ifinner
      \@parmoderr\@floatpenalty\z@
    \else\@next\@currbox\@freelist{}{}\@next\@marbox\@freelist{\global
      \count\@marbox\m@ne}{\@floatpenalty\z@ \@fltovf
         \def\@currbox{\@tempboxa}\def\@marbox{\@tempboxa}}\fi
     \@ifnextchar [\@xmpar\@ympar}
```

\@xmpar

```
\long\def\@xmpar[#1]#2{\@savemarbox\@marbox{#1}\@savemarbox\@currbox
   {#2}\@xympar}
```

\@ympar

```
\long\def\@ympar#1{\@savemarbox\@marbox{#1}\global\setbox\@currbox
     \copy\@marbox\@xympar}
```

\@savemarbox

```
\long\def \@savemarbox #1#2{%
  \global\setbox #1%
    \color@vbox
      \vtop{%
        \hsize\marginparwidth
        \@parboxrestore
        \@marginparreset
```

```
        #2%
        \global\@minipagefalse
        \outer@nobreak
        }%
    \color@endbox
}
```

\@marginparreset

```
\def \@marginparreset {%
        \reset@font
        \normalsize
        \@setnobreak
        \@setminipage
}
```

\@xympar

Setting the box here is done only because the code uses `\end@float`; it will be empty and gets discarded.

```
\def \@xympar{%
  \ifnum\@floatpenalty <\z@\@cons\@currlist\@marbox\fi
  \setbox\@tempboxa
    \color@vbox
      \vbox \bgroup
  \end@float
  \global \@ignorefalse \@esphack
}
```

\reversemarginpar
\normalmarginpar

```
\def\reversemarginpar{\global\@mparbottom\z@ \@reversemargintrue}
\def\normalmarginpar{\global\@mparbottom\z@ \@reversemarginfalse}
```

```
\message{footnotes,}
```

59.2 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
              \textsuperscript{\@thefnmark}

  \@makefntext{NOTE} :
       Must produce the actual footnote, using \@thefnmark as the mark
       of the footnote and NOTE as the text.  It is called when
       effectively  inside a \parbox, with \hsize = \columnwidth.
         For example, it might be as simple as
             $^{\@thefnmark}$  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
```

LaTeX does use the same insert for footnotes as PLAIN.

```
\newinsert\footins
```

LaTeX leaves these initializations for the `\footins` insert.

```
\skip\footins=\bigskipamount % space added when footnote is present
\count\footins=1000 % footnote magnification factor (1 to 1)
\dimen\footins=8in % maximum footnotes per page
```

LaTeX keeps PLAIN TeX's `\footnoterule` as the default.

```
\def\footnoterule{\kern-3\p@
  \hrule \@width 2in \kern 2.6\p@} % the \hrule is .4pt high
```

`\thefootnote`

```
\@definecounter{footnote}
\def\thefootnote{\@arabic\c@footnote}
```

`\thempfootnote`

```
\@definecounter{mpfootnote}
\def\thempfootnote{\itshape\@alph\c@mpfootnote}
```

`\@makefnmark` Default definition.

```
%\def\@makefnmark{\hbox{$^{\@thefnmark}\m@th$}}
\def\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

`\textsuperscript` This command provides superscript characters in the current text font. It's implementation might change!!!

```
\DeclareRobustCommand*\textsuperscript[1]{%
  \@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.

```
\def\@textsuperscript#1{%
  $\m@th^{\mbox{\fontsize\sf@size\z@#1}}$}
```

`\footnotesep`

```
\newdimen\footnotesep
```

`\footnote`

```
\def\footnote{\@ifnextchar[\@xfootnote{\stepcounter\@mpfn
     \protected@xdef\@thefnmark{\thempfn}%
     \@footnotemark\@footnotetext}}
```

`\@xfootnote`

```
\def\@xfootnote[#1]{%
   \begingroup
     \csname c@\@mpfn\endcsname #1\relax
     \unrestored@protected@xdef\@thefnmark{\thempfn}%
   \endgroup
   \@footnotemark\@footnotetext}
```

`\@footnotetext`

```
\long\def\@footnotetext#1{\insert\footins{%
    \reset@font\footnotesize
    \interlinepenalty\interfootnotelinepenalty
```

```
    \splittopskip\footnotesep
    \splitmaxdepth \dp\strutbox \floatingpenalty \@MM
    \hsize\columnwidth \@parboxrestore
    \protected@edef\@currentlabel{%
       \csname p@footnote\endcsname\@thefnmark
    }%
    \color@begingroup
      \@makefntext{%
        \rule\z@\footnotesep\ignorespaces#1\@finalstrut\strutbox}
    \color@endgroup}}
```

\footnotemark

```
\def\footnotemark{%
   \@ifnextchar[\@xfootnotemark
     {\stepcounter{footnote}%
      \protected@xdef\@thefnmark{\thefootnote}%
      \@footnotemark}}
```

\@xfootnotemark

```
\def\@xfootnotemark[#1]{%
   \begingroup
      \c@footnote #1\relax
      \unrestored@protected@xdef\@thefnmark{\thefootnote}%
   \endgroup
   \@footnotemark}
```

\@footnotemark

```
\def\@footnotemark{%
  \leavevmode
  \ifhmode\edef\@x@sf{\the\spacefactor}\nobreak\fi
  \@makefnmark
  \ifhmode\spacefactor\@x@sf\fi
  \relax}
```

\footnotetext

```
\def\footnotetext{%
     \@ifnextchar [\@xfootnotenext
       {\protected@xdef\@thefnmark{\thempfn}%
    \@footnotetext}}
```

\@xfootnotenext

```
\def\@xfootnotenext[#1]{%
  \begingroup
     \csname c@\@mpfn\endcsname #1\relax
     \unrestored@protected@xdef\@thefnmark{\thempfn}%
  \endgroup
  \@footnotetext}
```

\thempfn
\@mpfn

```
\def\@mpfn{footnote}
\def\thempfn{\thefootnote}
⟨/2ekernel⟩
```

File H
ltidxglo.dtx

60 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
   if \@filesw = T
     then  open file \jobname.IDX as \@indexfile
              \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
   fi
  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.

```
⟨*2ekernel⟩
\message{index,}

\def\makeindex{%
  \newwrite\@indexfile
  \immediate\openout\@indexfile=\jobname.idx
  \def\index{\@bsphack\begingroup
             \@sanitize
             \@wrindex}\typeout
    {Writing index file \jobname.idx}%
}

\def\@wrindex#1{%
   \protected@write\@indexfile{}%
      {\string\indexentry{#1}{\thepage}}%
 \endgroup
 \@esphack
}

\def\index{\@bsphack\begingroup \@sanitize\@index}

\def\@index#1{\endgroup\@esphack}

\def\makeglossary{%
  \newwrite\@glossaryfile
  \immediate\openout\@glossaryfile=\jobname.glo
  \def\glossary{\@bsphack\begingroup
                \@sanitize
                \@wrglossary}\typeout
   {Writing glossary file \jobname.glo }%
}

\def\@wrglossary#1{%
   \protected@write\@glossaryfile{}%
      {\string\glossaryentry{#1}{\thepage}}%
 \endgroup
 \@esphack
}

\def\glossary{\@bsphack\begingroup\@sanitize\@index}

⟨/2ekernel⟩
```

File I
ltbibl.dtx

61 Bibliography Generation

A bibliography is created by the `thebibliography` environment, which generates a title such as "References", and a list of entries. The BIBTEX program will create a file containing such an environment, which will be read in by the `\bibliography` command. With BIBTEX, the following commands will be used.

`\bibliography` `\bibliography{`*file1,file2, ... ,filen*`}` : specifies the bibdata files. Writes a `\bibdata` entry on the `.aux` file and tries to read in `mainfile.bbl`.

`\bibliographystyle` `\bibliographystyle{`*style*`}` : Writes a `\bibstyle` entry on the `.aux` file.

`thebibliography` The `thebibliography` environment is a list environment. To save the use of an extra counter, it should use `enumiv` as the item counter. Instead of using `\item`, items in the bibliography are produced by the following commands:
`\bibitem{`*name*`}` : Produces a numbered entry cited as ⟨*name*⟩.
`\bibitem[`*label*`]{`*name*`}` : Produces an entry labeled by ⟨*Label*⟩ and cited by ⟨*name*⟩.

The former is used for bibliographies with citations like [1], [2], etc.; the latter is used for citations like [Knuth82].

The document class must define the thebibliography environment. This environment has a single argument, which is the widest bibliography label– e.g., if the [Knuth67] is the widest entry, then this argument will be Knuth67. The `\thebibliography` command must begin a list environment, which the `\endthebibliography` command ends.

`\cite` Entries are cited by the command `\cite{`*name*`}`.

`\nocite` `\nocite{`*citations*`}` puts information on the `.aux` file that causes BIBTEX to include the `{`*citations*`}` list in the bibliography, but puts nothing in the text.

`\nocite{*}` is special: it tells BIBTEX to put the whole of a collection of references into the bibiography.

```
1 ⟨*2ekernel⟩
2 \message{bibliography,}
```

PARAMETERS

```
   \@cite   : A macro such that \@cite{LABEL1,LABEL2}{NOTE}
              produces the output for a \cite[NOTE]{FOO1,FOO2}
command,
              where entry FOOi is defined by \bibitem[LABELi]{FOOi}.
              The switch @tempswa is true if the optional NOTE
argument
              is present.
              The default definition is :
                \@cite{LABELS}{NOTE} ==
```

```
BEGIN [LABELS
      IF @tempswa = T THEN , NOTE FI
      ]
END
```

`\@biblabel` : A macro to produce the label in the bibliography entry. For `\bibitem`[LABEL]{NAME}, the label is generated by `\@biblabel`{LABEL}. It has the default definition `\@biblabel`{LABEL} -> [LABEL].

CONVENTION

`\b@FOO` : The name or number of the reference created by `\cite`{FOO} E.g., if `\cite`{FOO} -> [17] , then `\b@FOO` -> 17.

`\changes`{v1.1g}{1995/05/08}{Removed unnecessary braces}

```
\def\bibitem{\@ifnextchar[\@lbibitem\@bibitem}
```

RmS 92/02/26: Added `\hfill` to restore left-alignment of
bibliography labels in alpha style

```
\def\@lbibitem[#1]#2{\item[\@biblabel{#1}\hfill]\if@filesw
      {\let\protect\noexpand
       \immediate
       \write\@auxout{\string\bibcite{#2}{#1}}}\fi\ignorespaces}
```

Placement of '}' in def of `\@lbibitem` corrected 29 Apr 87
(Error found by Arthur Ogawa.)

RmS 91/11/13: Changed counter enumi to enumiv,
as it says in the comment above
RmS 92/01/10: Changed `\c@enumiv` to `\value{\@listctr}`.

`\@bibitem`

```
\def\@bibitem#1{\item\if@filesw \immediate\write\@auxout
       {\string\bibcite{#1}{\the\value{\@listctr}}}\fi\ignorespaces}
```

`\bibcite`

```
\def\bibcite{\@newl@bel b}
\@onlypreamble\bibcite
```

`\citation`

```
\let\citation\@gobble
```

`\cite`

```
\def\cite{\@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)

```
\def\@citex[#1]#2{%
  \let\@citea\@empty
  \@cite{\@for\@citeb:=#2\do
    {\@citea\def\@citea{,\penalty\@m\ }%
     \edef\@citeb{\expandafter\@firstofone\@citeb}%
     \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
     \@ifundefined{b@\@citeb}{\mbox{\reset@font\bfseries ?}%
       \G@refundefinedtrue\@latex@warning
       {Citation `\@citeb' on page \thepage \space undefined}}%
     {\hbox{\csname b@\@citeb\endcsname}}}}{#1}}
```

\bibdata
\bibstyle

```
\let\bibdata=\@gobble
\let\bibstyle=\@gobble
```

\bibliography

```
\def\bibliography#1{%
  \if@filesw
    \immediate\write\@auxout{\string\bibdata{#1}}%
  \fi
  \@input@{\jobname.bbl}}
```

\bibliographystyle

```
\def\bibliographystyle#1{%
  \ifx\@begindocumenthook\@undefined\else
    \expandafter\AtBeginDocument
  \fi
    {\if@filesw
       \immediate\write\@auxout{\string\bibstyle{#1}}%
     \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.

```
\def\nocite#1{\@bsphack
  \@for\@citeb:=#1\do{%
    \edef\@citeb{\expandafter\@firstofone\@citeb}%
    \if@filesw\immediate\write\@auxout{\string\citation{\@citeb}}\fi
    \@ifundefined{b@\@citeb}{\G@refundefinedtrue
        \@latex@warning{Citation `\@citeb' undefined}}{}}%
  \@esphack}
```

Since `\nocite{*}` should not produce a warning about undefined citation keys (seee 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).

```
\expandafter\let\csname b@*\endcsname\@empty
```

DEFAULT DEFINITIONS

```
\def\@cite#1#2{[{#1\if@tempswa , #2\fi}]}
```

RmS 92/01/14: removed \hfill in definition of \@biblabel

```
\def\@biblabel#1{[#1]}
⟨/2ekernel⟩
```

File J
ltpage.dtx

62 Page styles and related commands

62.1 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.

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.)

62.2 marking conventions

LaTeX extends TeX's `\mark` facility by producing two kinds of marks a 'left' and a 'right' mark, using the following commands:
`\markboth{`*left*`}{`*right*`}` : Adds both marks.
`\markright{`*right*`}` : Adds a 'right' mark.
`\leftmark` : Used in the output routine, gets the current 'left' mark. Works like TeX's `\botmark`.
`\rightmark` : Used in the output routine, gets the current 'right' mark. Works like TeX's `\firstmark`. The marking commands work reasonably well for right marks 'numbered within' left marks—e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if 2 `\markboth`'s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\@mkboth` command, which is `\let` by the pagestyle command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

```
⟨*2ekernel⟩
```

`\pagestyle`

```
\def\pagestyle#1{%
  \@ifundefined{ps@#1}%
    \undefinedpagestyle
    {\@nameuse{ps@#1}}}
```

\thispagestyle

```
\def\thispagestyle#1{%
  \@ifundefined{ps@#1}%
    \undefinedpagestyle
    {\global\@specialpagetrue\gdef\@specialstyle{#1}}}
```

Default Initializations

\ps@empty

```
\def\ps@empty{%
  \let\@mkboth\@gobbletwo\let\@oddhead\@empty\let\@oddfoot\@empty
  \let\@evenhead\@empty\let\@evenfoot\@empty}
```

\ps@plain

```
\def\ps@plain{\let\@mkboth\@gobbletwo
     \let\@oddhead\@empty\def\@oddfoot{\reset@font\hfil\thepage
     \hfil}\let\@evenhead\@empty\let\@evenfoot\@oddfoot}
```

\@leftmark \@rightmark We implement `\@leftmark` and `\@rightmark` in terms of already defined commands to save token space. We can't get rid of them since they are sometimes used in applications.

```
\let\@leftmark\@firstoftwo
\let\@rightmark\@secondoftwo
```

\markboth \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`

```
\def\markboth#1#2{\gdef\@themark{{#1}{#2}}{%
     \let\protect\@unexpandable@protect
     \let\label\relax \let\index\relax \let\glossary\relax
     \mark{\@themark}}\if@nobreak\ifvmode\nobreak\fi\fi}
\def\markright#1{{\let\protect\@unexpandable@protect
     \let\label\relax \let\index\relax \let\glossary\relax
     \expandafter\@markright\@themark
     {#1}\mark{\@themark}}\if@nobreak\ifvmode\nobreak\fi\fi}
```

\@markright \leftmark \rightmark

```
\def\@markright#1#2#3{\gdef\@themark{{#1}{#3}}}
\def\leftmark{\expandafter\@leftmark\botmark{}{}}
\def\rightmark{\expandafter\@rightmark\firstmark{}{}}
```

Initialization

\@themark

```
\def\@themark{{}{}}
```

\mark We initialize the internal TeX `\mark` when we reach `\begin{document}`. Initialization in the format would not survive.

```
%\AtBeginDocument{\mark{{}{}}}
```

OTHER

`\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`.)

`\flushbottom` : Inverse of `\raggedbottom` - makes all pages the same height.

`\sloppy` : Resets TeX's parameters so it accepts worse line and page breaks, and slightly more overfull boxes.

`\fussy` : Resets TeX's parameters to their normal finnicky values.

`\raggedbottom`

```
\def\raggedbottom{%
  \def\@textbottom{\vskip \z@ \@plus.0001fil}\let\@texttop\relax}
```

`\flushbottom`

```
\def\flushbottom{%
  \let\@textbottom\relax \let\@texttop\relax}
```

Default definitions

`\sloppy` `\sloppy` will never (well, hardly ever) produce overfull boxes, but may produce underfull ones. (14 June 85) A sloppypar environment is equivalent to `{\par \sloppy ... \par}`.

```
\def\sloppy{\tolerance 9999
            \emergencystretch 3em
            \hfuzz .5\p@ \vfuzz .5\p@}
```

`sloppypar`

```
\def\sloppypar{\par\sloppy}
\def\endsloppypar{\par}
```

`\fussy`

```
\def\fussy{%
  \emergencystretch\z@
  \tolerance 200
  \hfuzz .1\p@
  \vfuzz .1\p@}
```

`\overfullrule` LaTeX default is no overfull box rule. Changed by document class option.

```
\overfullrule 0pt
⟨/2ekernel⟩
```

File K
ltoutput.dtx

63 Output Routine

63.1 Floats

```
⟨*2ekernel⟩
\message{output,}
```

```
****************************************
*                OUTPUT                *
****************************************
```

PAGE LAYOUT PARAMETERS

```
\topmargin       : Extra space added to top of page.
@twoside         : boolean.  T if two-sided printing
\oddsidemargin   : IF @twoside = T
                      THEN extra space added to left of odd-numbered
                           pages.
                      ELSE extra space added to left of all pages.
\evensidemargin  : IF @twoside = T
                      THEN extra space added to left of
even-numbered
                           pages.
\headheight      : height of head
\headsep         : separation between head and text
\footskip        : distance separation between baseline of last
                   line of text and baseline of foot.
                   Note difference between \footSKIP and \headSEP.
\textheight      : height of text on page, excluding head and foot
\textwidth       : width of printing on page
\columnsep       : IF @twocolumn = T
                      THEN width of space between columns
\columnseprule   : IF @twocolumn = T
                      THEN width of rule between columns (0 if none).
\columnwidth     : IF @twocolumn = T
                      THEN (\textwidth - \columnsep)/2
                      ELSE \textwidth
                   It is set by the \twocolumn and
                   \onecolumn commands.
\@textbottom     : Command executed at bottom of vbox holding text
of
```

page (including figures). The `\raggedbottom` command almost `\let`'s this to `\vfil` (actually sets it to `\vskip \z@` plus.0001fil).
Should have depth 0pt.

`\@texttop` : Command executed at top of vbox holding text of page (including figures). Used by letter style; can also be used to produce centered pages. Let to `\relax` by `\raggedbottom` and `\flushbottom.`

Page layout must initialize `\@colht` and `\@colroom` to `\textheight.`

PAGE STYLE PARAMETERS:

- `\floatsep` : Space left between floats.
- `\textfloatsep` : Space between last top float or first bottom float and the text.
- `\topfigrule` : Command to place rule (or whatever) between floats at top of page and text. Executed in inner vertica mode right before the `\textfloatsep` skip separating the floats from the text. Must occupy zero vertical space. (See `\footnoterule.`)
- `\botfigrule` : Same as `\topfigrule,` but put after the `\textfloatsep` skip separating text from the floats at bottom of page.
- `\intextsep` : Space left on top and bottom of an in-text float.
- `\dblfloatsep` : Space between double-column floats.
- `\dbltextfloatsep` : Space between top double-column floats and text.
- `\dblfigrule` : Similar to `\topfigrule,` but for double-column floats.
- `\@fptop` : Glue to go at top of float column – must be 0pt + stretch
- `\@fpsep` : Glue to go between floats in a float column.
- `\@fpbot` : Glue to go at bottom of float column – must be 0pt + stretch
- `\@dblfptop, \@dblfpsep, \@dblfpbot` : Analogous for double-column float page in two-column format.

FOOTNOTES: As in PLAIN, footnotes use `\insert\footins.`

PAGE LAYOUT SWITCHES AND MACROS

```
@twocolumn         : Boolean.  T if two columns per page globally.

PAGE STYLE MACROS AND SWITCHES

 \@oddhead          : IF @twoside = T
                          THEN macro to generate head of
odd-numbered
                               pages.
                          ELSE macro to generate head of all pages.
 \@evenhead         : IF @twoside = T
                          THEN macro to generate head of
even-numbered
                               pages.
 \@oddfoot          : IF @twoside = T
                          THEN macro to generate foot of
odd-numbered
                               pages.
                          ELSE macro to generate foot of all pages.
 \@evenfoot         : IF @twoside = T
                          THEN macro to generate foot of
even-numbered
                               pages.
 @specialpage      : boolean.  T if current page is to have a special
                               format.
\@specialstyle    : If its value is  foo then
                      IF @specialpage = T
                        THEN the command \ps@foo is executed to
                              temporarily reset the page style parameters
                              before composing the current page.
                              This command should execute only \def's
and
                              \edef's, making only local definitions.

FLOAT PLACEMENT PARAMETERS

The following parameters are set by the macro \@floatplacement.
When \@floatplacement is called,
\@colht is the height of the page or column being built.  I.e.:
         * For single-column page it equals \textheight.
         * For double-column page it equals \textheight - height
           of double-column floats on page.
Note that some are set globally and some locally:
   \@topnum  :=G Maximum number of floats allowed on the top of a
                 column.
   \@toproom :=G Maximum amount of top of column devoted to floats--
                 excluding \textfloatsep separation below the floats
```

```
                  and \floatsep separation between them.  For
                  two-column output, should be computed as a function
                  of \@colht.
   \@botnum, \@botroom
                : Analogous to above.
   \@colnum  :=G Maximum number of floats allowed in a column,
                  including in-text floats.
   \@textmin :=L Minimum amount of text (excluding footnotes) that
                  must appear on a text page.
                     %% 27 Sep 85 : made local to
                     %% \@addtocurcol and \@addtonextcol
                  It is now also used locally in processing double
                  floats.
   \@fpmin   :=L Minimum height of floats in a float column.
```

The macro `\@dblfloatplacement` sets the following parameters.

```
   \@dbltopnum  :=G Maximum number of double-column floats allowed
at
                     the top of a two-column page.
   \@dbltoproom :=G Maximum height of double-column floats allowed at
                     top of two-column page.
   \@fpmin      :=L Minimum height of floats in a float column.
```

It should also perform the following local assignments where necessary – i.e., where the new value differs from the old one:

```
     \@fptop      :=L \@dblfptop
     \@fpsep      :=L \@dblfpsep
     \@fpbot      :=L \@dblfpbot
```

OUTPUT ROUTINE VARIABLES

```
 \@colht : The total height of the current column.  In single column
           style, it equals \textheight.  In two-column style, it is
           \textheight minus the height of the double-column floats
           on the current page.  MUST BE INITIALIZED TO
\textheight.

 \@colroom : The height available in the current column for text and
             footnotes.  It equals \@colht minus the height of all
             floats committed to the top and bottom of the current
             column.

 \@textfloatsheight : The total height of in-text floats on the
                      current page.

 \footins : Footnote insertion number.
```

```
\@maxdepth : Saved value of TeX's \maxdepth.  Must be set
             when any routine sets \maxdepth.
```

CALLING THE OUTPUT ROUTINE

The output routine is called either by TeX's normal page-breaking mechanism, or by a macro putting a penalty < or = -10000 in the output list. In the latter case, the penalty indicates why the output routine was called, using the following code.

penalty	reason
-10000	`\pagebreak`
	`\newpage`
-10001	`\clearpage` (`\penalty` -10000 `\vbox{}` `\penalty` -10001)
-10002	float insertion, called from horizontal mode
-10003	float insertion, called from vertical mode.
-10004	float insertion.

Note: A float or marginpar puts the following sequence in the output list: (i) a penalty of -10004,
(ii) a null `\vbox`
(iii) a penalty of -10002 or -10003.
This solves two special problems:
1. If the float comes right after a `\newpage` or `\clearpage,` then the first penalty is ignored, but the second one invokes the output routine.
2. If there is a split footnote on the page, the second 'page' puts out the rest of the footnote.

THE OUTPUT ROUTINE

FUNCTIONS USED IN THE OUTPUT ROUTINE:

`\@outputpage` : Produces an output page with the contents of box `\@outputbox` as the text part.
Also sets `\@colht` :=G `\textheight.`
The page style is determined as follows.
IF @thispagestyle = true
THEN use `\thispagestyle` style
ELSE use ordinary page style.

`\@tryfcolumn\FLIST` : Tries to form a float column composed of floats from `\FLIST` (if nonempty) with the following parameters:

```
        \@colht : height of box
        \@fpmin : minimum height of floats in the box
        \@fpsep : interfloat 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 float box.
        * @fcolmade   :=G true
        * \FLIST      :=G \FLIST - floats put in box
        * \@freelist  :=G \@freelist + floats 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 float column only if \FLIST is empty.
             Otherwise, it makes a float 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 floats
                  from \@deferlist to go at top and bottom of column,
                  deleting them from \@deferlist.  It does
                  this using \@colht as the total height, the page
                  style parameters \@floatsep and \@textfloatsep, and
                  the float placement parameters \@topnum, \@toproom,
                  \@botnum, \@botroom, \@colnum and \textfraction.
                * Globally sets \@colroom to \@colht minus the height
                  of the added floats.

\@startdblcolumn :
      Calls \@tryfcolumn\@dbldeferlist{8}.  If \@tryfcolumn returns
      with (globally set) @fcolmade = false, then:
               * Globally sets \@dbltoplist to floats from
                 \@dbldeferlist to go at top and bottom of column,
                 deleting them from \@dbldeferlist.
                 It does this using \textheight as the
                 total height, and the parameters \@dblfloatsep, etc.
               * Globally sets \@colht to \textheight minus the height
                 of the added floats.

\@combinefloats : Combines the text from box
             \@outputbox with the floats from \@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 floats 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 float pages,
and determines \@dbltoplist for the next page.

USER COMMANDS THAT CALL OR AFFECT THE
OUTPUT ROUTINE
—————————————————————-

```
\newpage == BEGIN \par\vfil\penalty -10000 END

\clearpage == BEGIN \newpage
                    \write -1{}    % Part of hack to make sure no
                    \vbox{}        % \write's get lost.
                    \penalty -10001
               END

\cleardoublepage == BEGIN \clearpage
                          if @twoside = true and c@page is even
                            then \hbox{} \newpage fi
                    END
```

\twocolumn[BOX] : starts a new page, chnaging to twocolumn setting

```
and puts BOX in a parbox of width \textwidth across the top.
Useful for full-width titles for double-column pages.
SURPRISE: The stretch from \@dbltextfloatsep will be inserted
          between the BOX and the top of the two columns.


              FLOAT-HANDLING MECHANISMS
              -------------------------

The float environment obtains an insertion number B from the
\@freelist (see below for a description of list manipulation), puts
the float into box B and sets \count B to a FLOAT SPECIFIER.  For
a normal (not double-column) float, it then causes a page break
in one of the following two ways:
   - In outer hmode: \vadjust{\penalty -10002}
   - In vmode :        \penalty -10003.
For a double-column float, it puts B onto the \@dbldeferlist.
The float specifier has two components:
     * A PLACEMENT SPECIFICATION, describing where the float may
       be placed.
     * A TYPE, which is a power of two--e.g., figures might be
       type 1 floats, tables type 2 floats, programs type 4 floats, etc.
The float specifier is encoded as follows, where bit 0 is the least
significant bit.

 Bit     Meaning
 ---     -------
  0      1 iff the float may go where it appears in the text.
  1      1 iff the float may go on the top of a page.
  2      1 iff the float may go on the bottom of a page.
  3      1 iff the float may go on a float page.
  4      1 unless the PLACEMENT includes a !
  5      1 iff a type 1 float
  6      1 iff a type 2 float
 etc.

 A negative float specifier is used to indicate a marginal note.

    MACROS AND DATA STRUCTURES FOR PROCESSING FLOATS
    ------------------------------------------------

 A FLOAT LIST consisting of the floats in boxes \boxa ... \boxN has
 the form:
       \@elt \boxa ... \@elt \boxN
 where  \boxI is defined by
       \newinsert\boxI
```

Normally, \@elt is \let to \relax. A test can be performed on the entire float list by locally \def'ing \@elt appropriately and executing the list.
This is a lot more efficient than looping through the list.

The following macros are used for manipulating float lists.

```
  \@next \CS \LIST {NONEMPTY}{EMPTY} ==  %% NOTE: ASSUME
\@elt = \relax
    BEGIN  assume that \LIST == \@elt \B1 ... \@elt \Bn
           if n = 0
             then  EMPTY
             else  \CS    :=L \B1
                   \LIST  :=G \@elt \B2 ... \@elt \Bn
                   NONEMPTY
           fi
    END
```

```
  \@bitor\NUM\LIST : Globally sets switch @test to the disjunction for
         all I of bit  log2 \NUM of the float specifiers of all the
         floats in \LIST.
         I.e., @test is set to true iff there is at least one
         float in \LIST having bit  log2 \NUM  of its float specifier
         equal to 1.

  Note: log2 [(\count I)/32] is the bit number corresponding to the
  type of float I.  To see if there is any float in \LIST having
  the same type as float I, you run \@bitor with
    \NUM = [(\count I)/32] * 32.

 \@bitor\NUM\LIST ==
   BEGIN
      @test :=G false
      { \@elt \CTR ==  if \NUM <> 0 then
                          if \count\CTR / \NUM is odd
                             then  @test := true        fi fi
        \LIST
      }
   END


 \@cons\LIST\NUM : Globally sets \LIST := \LIST * \@elt \NUM

 \@cons\LIST\NUM ==
   BEGIN {  \@elt == \relax
```

```
        \LIST :=G \LIST \@elt \NUM
      }

  BOX LISTS FOR FLOAT-PLACEMENT ALGORITHMS

    \@freelist     : List of empty boxes for placing new floats.
    \@toplist      : List of floats to go at top of current column.
    \@midlist      : List of floats in middle of current column.
    \@botlist      : List of floats to go at bottom of current column.
    \@deferlist    : List of floats to go after current column.
    \@dbltoplist   : List of double-col. floats to go at top of current
                     page.
    \@dbldeferlist : List of double-column floats to go on subsequent
                     pages.

  FLOAT-PLACEMENT ALGORITHMS


  \@addtobot : Tries to put insert \@currbox on \@botlist.
               Called only when:
                  * \ht BOX < \@colroom
                  * type of \@currbox not on \@deferlist
                  * \@colnum > 0
                  * @insert = false
               If it succeeds, then:
                  * sets @insert true
                  * decrements \@botroom by \ht BOX
                  * decrements \@botnum and \@colnum by 1
                  * decrements \@colroom by \ht BOX + either
\floatsep
                    or \textfloatsep, as appropriate.
                  * sets \maxdepth to 0pt

  \@addtotoporbot : Tries to put insert \@currbox on \@toplist or
                    \@botlist.
                    Called only under same conditions as \@addtobot.
                    If it succeeds, then:
                       * sets @insert true
                       * decrements \@toproom or \@botroom by \ht
BOX
                       * decrements \@colnum and either \@topnum or
                         \@botnum by 1
                       * decrements \@colroom by \ht BOX +
\floatsep
                         or \textfloatsep, as appropriate.
```

```
\@addtocurcol : Tries to add \@currbox to current column, setting
                @insert true if it succeeds, false otherwise.
                It will add \@currbox to top only if bit 0 of
                \count \@currbox is 0, and to the bottom only if
                bit 0 = 0 or an earlier float of the same type is
                put on the bottom.
                If the float is put in the text, then
                \penalty\interlinepenalty is put
                right after the float, before the following \vskip,
                and \outputpenalty :=L 0.

\@addtonextcol : Tries to add \@currbox to the next column, setting
                 @insert true if it succeeds, false otherwise.

\@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.

```
\maxdeadcycles = 100
```

```
\let\@elt\relax
```

```
\def\@next#1#2#3#4{\ifx#2\@empty #4\else
   \expandafter\@xnext #2\@@#1#2#3\fi}
```

```
\def\@xnext \@elt #1#2\@@#3#4{\def#3{#1}\gdef#4{#2}}
```

```
\newif \if@test
```

```
\def\@bitor#1#2{\global\@testfalse {\let\@elt\@xbitor
   \@tempcnta #1\relax #2}}
```

RmS 91/11/22: Added test for |\count#1 = 0|.
Suggested by Chris Rowley.

```
\def\@xbitor #1{\@tempcntb \count#1
   \ifnum \@tempcnta =\z@
   \else
     \divide\@tempcntb\@tempcnta
     \ifodd\@tempcntb \global\@testtrue\fi
   \fi}
```

DEFINITION OF FLOAT BOXES:

```
\newinsert\bx@A
\newinsert\bx@B
\newinsert\bx@C
\newinsert\bx@D
\newinsert\bx@E
\newinsert\bx@F
\newinsert\bx@G
\newinsert\bx@H
\newinsert\bx@I
\newinsert\bx@J
\newinsert\bx@K
\newinsert\bx@L
\newinsert\bx@M
\newinsert\bx@N
\newinsert\bx@O
\newinsert\bx@P
\newinsert\bx@Q
\newinsert\bx@R

\gdef\@freelist{\@elt\bx@A\@elt\bx@B\@elt\bx@C\@elt\bx@D\@elt\bx@E
               \@elt\bx@F\@elt\bx@G\@elt\bx@H\@elt\bx@I\@elt\bx@J
                \@elt\bx@K\@elt\bx@L\@elt\bx@M\@elt\bx@N
                \@elt\bx@O\@elt\bx@P\@elt\bx@Q\@elt\bx@R}

\gdef\@toplist{}
\gdef\@botlist{}
\gdef\@midlist{}
\gdef\@currlist{}
\gdef\@deferlist{}
\gdef\@dbltoplist{}
\gdef\@dbldeferlist{}
```

PAGE LAYOUT PARAMETERS

```
\newdimen\topmargin
\newdimen\oddsidemargin
\newdimen\evensidemargin
\let\@themargin=\oddsidemargin
\newdimen\headheight
\newdimen\headsep
\newdimen\footskip
\newdimen\textheight
\newdimen\textwidth
\newdimen\columnwidth
\newdimen\columnsep
\newdimen\columnseprule
\newdimen\marginparwidth
\newdimen\marginparsep
\newdimen\marginparpush
```

\AtBeginDvi
\@begindvibox

We use a box register in which to put stuff that must appear before anything else in the `.dvi` file.

The stuff in the box should not add any typeset material to the page whwn it is unboxed.

```
\newbox\@begindvibox
\def \AtBeginDvi #1{%
  \global \setbox \@begindvibox
    \vbox{\unvbox \@begindvibox #1}%
}
```

\@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?

```
\newdimen\@maxdepth
\@maxdepth = \maxdepth
```

\paperheight
\paperwidth

New `\paper`... registers.

```
\newdimen\paperheight
\newdimen\paperwidth
```

\if@insert
\if@fcolmade
\if@specialpage
\if@firstcolumn
\if@twocolumn
\if@twoside
\if@reversemarginpar
\if@mparswitch
\col@number

Local switches first:

```
\newif \if@insert
```

These should definitely be global:

```
\newif \if@fcolmade
\newif \if@specialpage \@specialpagefalse
```

These should be global but are not always set globally in other files.

```
\newif \if@firstcolumn \@firstcolumntrue
\newif \if@twocolumn   \@twocolumnfalse
```

Not sure about these: two questions. Should things which must apply to a whole doument be local or global (they probably should be 'preamble only' commands)? Are these three such things?

```
\newif \if@twoside     \@twosidefalse
\newif \if@reversemargin \@reversemarginfalse
\newif \if@mparswitch  \@mparswitchfalse
```

This counter has beeen imported from 'multicol'.

```
\newcount \col@number
\col@number \@ne
```

INTERNAL REGISTERS

```
\newcount\@topnum
\newdimen\@toproom
\newcount\@dbltopnum
\newdimen\@dbltoproom
\newcount\@botnum
\newdimen\@botroom
```

```
\newcount\@colnum
\newdimen\@textmin
\newdimen\@fpmin
\newdimen\@colht
\newdimen\@colroom
\newdimen\@pageht
\newdimen\@pagedp
\newdimen\@mparbottom \@mparbottom\z@
\newcount\@currtype
\newbox\@outputbox
\newbox\@leftcolumn
\newbox\@holdpg
```

```
\def\@thehead{\@oddhead} % initialization
\def\@thefoot{\@oddfoot}
```

```
\def\clearpage{\newpage \write\m@ne{}\vbox{}\penalty -\@Mi}
```

```
\def\cleardoublepage{\clearpage\if@twoside \ifodd\c@page\else
    \hbox{}\newpage\if@twocolumn\hbox{}\newpage\fi\fi\fi}
⟨/2ekernel⟩
```

`\onecolumn`

```
⟨*2ekernel | fltrace⟩
\def\onecolumn{%
  \clearpage
  \global\columnwidth\textwidth
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumnfalse
  \col@number \@ne
  \@floatplacement}
```

`\newpage` `\@emptycol` It may be better to use an invisible rule rather than an empty box here.

```
\def \newpage {\par\vfil\penalty -\@M}
\def \@emptycol {\vbox{}\penalty -\@M}
```

`\twocolumn` `\@topnewpage` There are several bug fixes to the two-column stuff here.

```
\def \twocolumn {%
  \clearpage
  \global\columnwidth\textwidth
  \global\advance\columnwidth-\columnsep
  \global\divide\columnwidth\tw@
  \global\hsize\columnwidth
  \global\linewidth\columnwidth
  \global\@twocolumntrue
  \global\@firstcolumntrue
  \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.

```
  \@ifnextchar [\@topnewpage\@floatplacement
}
```

Note that here, getting a box from the freelist can assume success since this comes just after a `\clearpage`.

```
\long\def \@topnewpage [#1]{%
  \@next\@currbox\@freelist{}{}%
  \global \setbox\@currbox
    \color@vbox
      \normalcolor
      \vbox {%
        \hsize\textwidth
        \@parboxrestore
        \col@number \@ne
        #1%
        \vskip -\dbltextfloatsep
             }%
    \color@endbox
```

Added size test and warning message; perhaps we should use an error message.

```
  \ifdim \ht\@currbox>\textheight
    \ht\@currbox \textheight
  \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.

```
  \global \count\@currbox \tw@
  \@tempdima -\ht\@currbox
  \advance \@tempdima -\dbltextfloatsep
  \global \advance \@colht \@tempdima
  \ifx \@dbltoplist \@empty
  \else
    \@latexerr{Float(s) lost}\@ehb
    \let \@dbltoplist \@empty
  \fi
  \@cons \@dbltoplist \@currbox
```

This setting of `\@dbltopnum` is used only to change the typesetting in `\@combinedblfloats`.

```
  \global \@dbltopnum \m@ne
⟨*trace⟩
    \tr@ce{dbltopnum set to -1 (= \the \@dbltopnum) (topnewpage)}%
⟨/trace⟩
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK? This has been extended to cope with having to produce two empty columns.

I now think that the cut-off point should be `3\baselineskip`?

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.

```
  \ifdim \@colht<3\baselineskip
    \@latex@warning@no@line {Optional argument of \noexpand\twocolumn
                  too tall on page \thepage}%
    \@emptycol
    \if@firstcolumn
    \else
      \@emptycol
    \fi
  \else
    \global \vsize \@colht
    \global \@colroom \@colht
    \@floatplacement
  \fi
}
```

\output
\@specialoutput

This needs some small adjustments. We cannot guarantee that the float mechanism will interact correctly with this stuff, but that mecahnism 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.

```
\output {%
  \let \par \@@par
  \ifnum \outputpenalty<-\@M
    \@specialoutput
  \else
    \@makecol
    \@opcol
```

Moved to `\@opcol`: `\@floatplacement`.

```
    \@startcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startcolumn`.

```
    \@whilesw \if@fcolmade \fi
      {%
⟨*trace⟩
        \tr@ce{PAGE: float \if@twocolumn column \else page \fi
                    completed}%
⟨/trace⟩
        \@opcol\@startcolumn}%
  \fi
  \ifnum \outputpenalty>-\@Miv
```

At points such as this we need to check that there is still a minimal amount of room left on the page; this uses an arbitrary small value at present. If there is little room left we just force a page-break, OK?

This bit is essential only if a float has just been processed so maybe it should be moved.

Since this happens only when there is nothing on the page but floats, the empty box should not cause any problem other than some overfull box messages, which is not entirely misleading.

```
    \ifdim \@colroom<2\baselineskip
      \@latex@warning@no@line {Text page \thepage\space
                               contains only floats}%
      \@emptycol
      \if@twocolumn
        \if@firstcolumn
        \else
          \@emptycol
        \fi
      \fi
    \else
      \global \vsize \@colroom
    \fi
  \else
    \global \vsize \maxdimen
  \fi
}
```

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.)

```
\def\@specialoutput{%
   \ifnum \outputpenalty>-\@Mii
     \@doclearpage
   \else
     \ifnum \outputpenalty<-\@Miii
       \ifnum \outputpenalty<-\@MM \deadcycles \z@ \fi
       \global \setbox\@holdpg \vbox {\unvbox\@cclv}%
     \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.

```
     \global \setbox\@holdpg \vbox{%
                    \unvbox\@holdpg
                    \unvbox\@cclv
```

We must now remove the box added by the float mechanism and the `\topskip` glue therefore added above it by TEX.

```
                    \setbox\@tempboxa \lastbox
                    \unskip
                                   }%
```

These two are needed as separate dimensions only by `\@addmarginpar`; for other purposes we put the whole size into `\@pageht` (see below).

```
     \@pagedp \dp\@holdpg
     \@pageht \ht\@holdpg
     \unvbox \@holdpg
     \@next\@currbox\@currlist{%
       \ifnum \count\@currbox>\z@
```

Putting the whole size into `\@pageht` (see above).

```
         \advance \@pageht \@pagedp
         \ifvoid\footins \else
           \advance \@pageht \ht\footins
           \advance \@pageht \skip\footins
           \advance \@pageht \dp\footins
         \fi
         \ifvbox \@kludgeins
```

We want to make the adjustment due to this insert only if the non-star form is used. The *-form will probably not work with floats, but maybe it still could make some adjustment here even so?

```
           \ifdim \wd\@kludgeins=\z@
             \advance \@pageht \ht\@kludgeins
⟨*trace⟩
             \tr@ce {Extra size added: \the \ht\@kludgeins}%
⟨/trace⟩
           \fi
         \fi
```

This version puts the inserts back just before the additional material; it could be moved earlier, before unboxing the page-so-far. Neither is guaranteed not to put things on the wrong page. This version is similar to the original version.

```
         \@reinserts
         \@addtocurcol
       \else
```

```
          \@reinserts
          \@addmarginpar
        \fi
        }\@latexbug
```

A 2e change: use `\addpenalty` instead of `\penalty` here. Some penalty is needed to create a potential break-point immediately after the reinerts (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).

```
      \ifnum \outputpenalty<\z@
        \if@nobreak
        \else
          \addpenalty \interlinepenalty
        \fi
      \fi
    \fi
  \fi
}
⟨/2ekernel | fltrace⟩
```

`\@doclearpage` This is a very much an emergency action, just dumping everything: footnotes first then floats. A more sophisticated version is needed.

Also, it puts any left-over non-boxes (writes, specials, etc.) back after any float pages created: this is a bug.

```
⟨*2ekernel⟩
\def \@doclearpage {%
    \ifvoid\footins
      \setbox\@tempboxa\vsplit\@cclv to\z@ \unvbox\@tempboxa
      \setbox\@tempboxa\box\@cclv
      \xdef\@deferlist{\@toplist\@botlist\@deferlist}%

      \global \let \@toplist \@empty
      \global \let \@botlist \@empty
      \global \@colroom \@colht
      \ifx \@currlist\@empty
      \else
         \@latexerr{Float(s) lost}\@ehb

         \global \let \@currlist \@empty
      \fi
      \@makefcolumn\@deferlist
      \@whilesw\if@fcolmade \fi{\@opcol\@makefcolumn\@deferlist}%
      \if@twocolumn
        \if@firstcolumn
          \xdef\@dbldeferlist{\@dbltoplist\@dbldeferlist}%

          \global \let \@dbltoplist \@empty
          \global \@colht \textheight
          \begingroup
             \@dblfloatplacement
```

```
              \@makefcolumn\@dbldeferlist
              \@whilesw\if@fcolmade \fi{\@outputpage
                                        \@makefcolumn\@dbldeferlist}%
           \endgroup
         \else
           \vbox{}\clearpage
         \fi
       \fi
     \else
       \setbox\@cclv\vbox{\box\@cclv\vfil}%
       \@makecol\@opcol
       \clearpage
     \fi
}
⟨/2ekernel⟩
```

`\@opcol` Several changes in detail here.

```
⟨*2ekernel | fltrace⟩
\def \@opcol {%
  \if@twocolumn
    \@outputdblcol
  \else
    \@outputpage
⟨*trace⟩
    \tr@ce{PAGE: one column (float? see above) page completed}%
⟨/trace⟩
```

Not needed since it comes after `\@outputpage`:

```
%   \global\@colht\textheight
  \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.

```
  \global \@mparbottom \z@ \global \@textfloatsheight \z@
  \@floatplacement
}
⟨/2ekernel | fltrace⟩
```

`\@makecol` We must rewrite this macro to alllow for variations in page-makeup required by changes in page-length.

This uses a different macro if a special-length column is being produced.

```
⟨*2ekernel⟩
\def \@makecol {%
   \ifvoid\footins
     \setbox\@outputbox \box\@cclv
   \else
     \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 chnage is pointless when there are footnotes.

But see also its use when combining floats.

```
      \boxmaxdepth \@maxdepth
      \unvbox \@cclv
```

```
      \vskip \skip\footins
      \color@begingroup
        \normalcolor
        \footnoterule
        \unvbox \footins
      \color@endgroup
      }%
  \fi
```

The h floats have now been finally committed to this page so we can reset their list. The top and bottom floats are then added to the page.

```
  \xdef\@freelist{\@freelist\@midlist}%
```

```
  \global \let \@midlist \@empty
  \@combinefloats
```

The variations start here in case \enlargethispage has been used.

```
  \ifvbox\@kludgeins
    \@makespecialcolbox
  \else
```

This extra reboxing is only needed to add the \@texttop and \@textbotttom but this could be done earlier, when the floats are added.

The \boxmaxdepth resetting here will have no effect unless \@textbottom ends with a box or rule. So is this (or possibly \@maxdepth) the correct value?

The \vskip -\dimen@ ensures that the visible depth of the box does not affect the placement of anything on the page. Thus very deep pages will overprint the footer; but these should have been prevented by suitable settings of the maxdepths at appropriate times.

If \@textbottom ends with a box or rule of non-zero depth then this skip adjustemnt 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 wthin \@texttop and \@textbottom, should it? Is it needed at all?

RmS 91/10/22: Replaced \dimen128 by \dimen@.

```
     \setbox\@outputbox \vbox to\@colht {%
%        \boxmaxdepth \maxdepth                  %??
      \@texttop
      \dimen@ \dp\@outputbox
      \unvbox \@outputbox
```

```
      \vskip -\dimen@
      \@textbottom
      }%
  \fi
  \global \maxdepth \@maxdepth
}
```

`\@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 expidite experimentation.

```
\def \@reinserts{%
  \ifvoid\footins\else\insert\footins{\unvbox\footins}\fi
  \ifvbox\@kludgeins\insert\@kludgeins{\unvbox\@kludgeins}\fi
}
⟨/2ekernel⟩
```

`\@makespecialcolbox` This implements certain variations in page-makeup.

```
⟨*2ekernel | fltrace⟩
\def \@makespecialcolbox {%
⟨*trace⟩
  \tr@ce{Kludgeins ht \the\ht\@kludgeins\space
                        dp \the\dp\@kludgeins\space
                        wd \the\wd\@kludgeins}%
⟨/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'.

```
  \setbox\@outputbox \vbox {%
    \@texttop
    \dimen@ \dp\@outputbox
    \unvbox\@outputbox
    \vskip-\dimen@
    }%
  \@tempdima \@colht
  \ifdim \wd\@kludgeins>\z@
```

Note that in this case (the *-version), the height of the `\@kludgeins` box is not used since its value is somewhat arbitrary: it need only be big enough to ensure that the page-break is not taken prematurely.

Here we calculate how much vertical space needs to be added in order to enable the column to fit into a box of size `\@colht` using the best information we have about the amount of shrink available (another thing which is known internally about a box, but cannot be accessed at the TeX level!).

This needs TeX3 otherwise `\pageshrink` is zero anyway; it may not be exactly the figure we wish as it is the total available from the all the material collected before the page-break decision is made. It will, we think, always be an overestimate of the actual shrink in the box; therefore this should always force the shortest possible column with the possibility of an overfull box.

This should work for bothe flush- and ragged-bottom setting since it makes the contents no smaller than the size (`\@colht`) of the box into which they are put.

Their should perhaps be an upper limit, of 0pt?, on the extra space added to force shrinking.

See above for a discussion of the `\boxmaxdepth` setting here.

```
      \advance \@tempdima -\ht\@outputbox
      \advance \@tempdima \pageshrink
⟨*trace⟩
      \tr@ce {Natural ht of col: \the \ht\@outputbox}%
      \tr@ce {\string \@colht: \the \@colht}%
      \tr@ce {Pageshrink added: \the \pageshrink}%
      \tr@ce {Hence, space added: \the \@tempdima}%
⟨/trace⟩
      \setbox\@outputbox \vbox to \@colht {%
%        \boxmaxdepth \maxdepth
        \unvbox\@outputbox
        \vskip \@tempdima
        \@textbottom
        }%
```

For the unstarred version, the final size of the page is precisely specified. Therefore, at least for the flush-bottom case, we need to ensure that, visually, it has this size exactly.

Thus we calculate this size and set the material in a box of this size, which is then put into a box of size `\@colht` with `\vss` at the bottom.

```
    \else
      \advance \@tempdima -\ht\@kludgeins
⟨*trace⟩
      \tr@ce {Natural ht of col: \the \ht\@outputbox}%
      \tr@ce {\string \@colht: \the \@colht}%
      \tr@ce {Extra size added: -\the \ht \@kludgeins}%
      \tr@ce {Hence, height of inner box: \the \@tempdima}%
      \tr@ce {Max? pageshrink available: \the \pageshrink}%
⟨/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.

```
      \setbox \@outputbox \vbox to \@colht {%
        \vbox to \@tempdima {%
          \unvbox\@outputbox
          \@textbottom}%
        \vss}%
    \fi
```

Finally we need to explicitly make the insert box void.

```
    {\setbox \@tempboxa \box \@kludgeins}%
```

```
}
⟨/2ekernel | fltrace⟩
```

\@texttop
\@textbottom

These do nothing as a default.

```
⟨*2ekernel⟩
\let \@texttop \relax
\let \@textbottom \relax
```

\@resetactivechars
\@activechar@info

RmS 93/09/06: added hook to protect against certain active characters in the output routine. Default checks are for active space and end-of-line.

```
\def\@activechar@info #1{%
      \@latex@info@no@line {Active #1 character found while
                              output routine is active
                              \MessageBreak
                              This may be a bug in a package file
                              you are using}%
}
```

Do not put any spaces in this next bit!

```
\begingroup
\obeylines\obeyspaces%
\gdef\@resetactivechars{%
\def^^M{\@activechar@info{EOL}\space}%
\def {\@activechar@info{space}\space}}%
\endgroup
```

\@outputpage
\@shipoutsetup

The `\color@hbox` hooks here are used to avoid putting just a colour special into an otherwise empty box (in a header or footer). These boxes are often set to be completely empty and so adding a special produces a very undeffull 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 `\write`s 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 `\write`s because the aftergroup token list is only constructed when popping the save-stack, it is not expanded until after the shipout is completed.

```
\def\@outputpage{%
  \let \protect \noexpand
  \shipout \vbox{%
    \set@typeset@protect
    \aftergroup\set@typeset@protect
    \@shipoutsetup
```

```
    \@begindvi
    \vskip \topmargin
    \moveright\@themargin \vbox {%
      \setbox\@tempboxa \vbox to\headheight{%
        \vfil
        \color@hbox
          \normalcolor
          \hb@xt@\textwidth {%
            \let \label \@gobble
            \let \index \@gobble
            \let \glossary \@gobble %% 21 Jun 91
            \@thehead
            }%
        \color@endbox
        }%                        %% 22 Feb 87
      \dp\@tempboxa \z@
      \box\@tempboxa
      \vskip \headsep
      \box\@outputbox
      \baselineskip \footskip
      \color@hbox
        \normalcolor
        \hb@xt@\textwidth{%
          \let \label \@gobble
          \let \index \@gobble      %% 22 Feb 87
          \let \glossary \@gobble   %% 21 Jun 91
          \@thefoot
          }%
      \color@endbox
      }%
    }%
  \global \@colht \textheight
  \stepcounter{page}%
```

It is now clear that this does something useful, thanks to Piet van Oostrum. It is needed because a float page is made without using TeX's page-builder; thus the output routine is never called so the marks are not updated.

```
  \let\firstmark\botmark
}
```

Now here is all the set-up stuff.

```
\def \@shipoutsetup {%
```

This first bit has been moved inside the shipped out box.

RmS 93/08/19: Redefined accents to allow changes in font encoding.

The `\catcode‘\ = 10` was removed as it was considered useless.

```
     \@resetactivechars
     \let\-\@dischyph
```

With the new encoding setup the following line can use `\let`.

```
     \let\'\@acci\let\‘\@accii\let\=\@acciii
```

```
    \let\\\@normalcr
    \if@specialpage
      \global\@specialpagefalse\@nameuse{ps@\@specialstyle}%
    \fi
    \if@twoside
      \ifodd\count\z@ \let\@thehead\@oddhead \let\@thefoot\@oddfoot
           \let\@themargin\oddsidemargin
      \else \let\@thehead\@evenhead
         \let\@thefoot\@evenfoot \let\@themargin\evensidemargin
      \fi
    \fi
```

The rest was always inside the box.

RmS 91/08/15: aded this line:

```
    \reset@font
```

RmS 93/08/06 Added `\lineskiplimit=0pt` to guard against it being nonzero: e.g. by `\offinterlineskip` being in effect.

```
    \normalsize
    \baselineskip\z@skip \lineskip\z@skip \lineskiplimit\z@
    \let\par\@@par          %% 15 Sep 87
}
```

`\@begindvi` This unboxes stuff that must appear before anything else in the `.dvi` file, then returns that box register to the free list and cancels itself.

The stuff in the box should not add any typeset material to the page.

```
\def \@begindvi{%
  \unvbox \@begindvibox
  \global\let \@begindvi \@empty
}
```

`\@combinefloats` `\@cflt` `\@cflb` 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.

```
\def \@combinefloats {%
%    \boxmaxdepth \maxdepth
    \ifx \@toplist\@empty \else \@cflt \fi
    \ifx \@botlist\@empty \else \@cflb \fi
}
```

```
\def \@cflt{%
    \let \@elt \@comflelt
    \setbox\@tempboxa \vbox{}%
    \@toplist
    \setbox\@outputbox \vbox{%
                             \boxmaxdepth \maxdepth
                             \unvbox\@tempboxa
                             \vskip -\floatsep
                             \topfigrule
                             \vskip \textfloatsep
```

```
                                \unvbox\@outputbox
                                }%
    \let\@elt\relax
    \xdef\@freelist{\@freelist\@toplist}%
    \global\let\@toplist\@empty
}
```

```
\def \@cflb {%
    \let\@elt\@comflelt
    \setbox\@tempboxa \vbox{}%
    \@botlist
    \setbox\@outputbox \vbox{%
                                \unvbox\@outputbox
                                \vskip \textfloatsep
                                \botfigrule
                                \unvbox\@tempboxa
                                \vskip -\floatsep
                                }%
    \let\@elt\relax
    \xdef\@freelist{\@freelist\@botlist}%
    \global \let \@botlist\@empty
}
```

\@comflelt
\@comdblflelt
\@combinedblfloats

```
\def\@comflelt#1{\setbox\@tempboxa
      \vbox{\unvbox\@tempboxa\box #1\vskip\floatsep}}
```

```
\def\@comdblflelt#1{\setbox\@tempboxa
      \vbox{\unvbox\@tempboxa\box #1\vskip\dblfloatsep}}
```

```
\def \@combinedblfloats{%
  \ifx \@dbltoplist \@empty
  \else
    \setbox\@tempboxa \vbox{}%
    \let \@elt \@comdblflelt
    \@dbltoplist
    \let \@elt \relax
    \xdef \@freelist {\@freelist\@dbltoplist}%
    \global\let \@dbltoplist \@empty
    \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.

```
      {%\boxmaxdepth\maxdepth   %% probably not needed, CAR
       \unvbox\@tempboxa\vskip-\dblfloatsep
```

Here we need different typesetting if the top float comes from `\@topnewpage`.

```
       \ifnum \@dbltopnum>\m@ne
         \dblfigrule
       \fi
       \vskip \dbltextfloatsep
```

```
      \box\@outputbox
      }%
  \fi
}
⟨/2ekernel⟩
```

\@startcolumn \@startdblcolumn

We could combine (most of) these two into `\@startcol <list>`. Note that `\@xstartcol` was only used once (i.e. in `\@startcolumn`); it has therefore been removed. This is not quite as efficient but it now has the same structure as `\@startdblcolumn`.

The empty-list test has been moved to `\@tryfcolumn`.

```
⟨*2ekernel | fltrace⟩
\def \@startcolumn {%
  \global \@colroom \@colht
  \@tryfcolumn \@deferlist
  \if@fcolmade
⟨*trace⟩
    \tr@ce{PAGE: float \if@twocolumn column \else page \fi
                completed}%
⟨/trace⟩
  \else

    \begingroup
      \let \reserved@b \@deferlist
      \global \let \@deferlist \@empty
      \let \@elt \@scolelt
      \reserved@b
    \endgroup
  \fi
}
```

This one does not need to set `\@colht`.

```
\def \@startdblcolumn {%
```

Not needed since this always comes after `\@outputpage`:

```
% \global \@colht \textheight
  \@tryfcolumn \@dbldeferlist
  \if@fcolmade
⟨*trace⟩
    \tr@ce{PAGE: double float page completed}%
⟨/trace⟩
  \else

    \begingroup
      \let \reserved@b \@dbldeferlist
      \global \let \@dbldeferlist \@empty
      \let \@elt \@sdblcolelt
      \reserved@b
    \endgroup
  \fi
}
```

`\@tryfcolumn` Now tests if its list is empty before any further exertion.

```
\def \@tryfcolumn #1{%
  \global \@fcolmadefalse
  \ifx #1\@empty
  \else
⟨*trace⟩
     \tr@ce{PAGE: try float \if@twocolumn column/page\else page\fi
                  ---\string #1}%
     \tr@ce{----- \string #1: #1}%
⟨/trace⟩
    \xdef\@trylist{#1}%
    \global \let \@failedlist \@empty
    \begingroup
      \let \@elt \@xtryfc \@trylist
    \endgroup
    \if@fcolmade
      \@vtryfc #1%
    \fi
  \fi
}
⟨/2ekernel | fltrace⟩
⟨*2ekernel⟩
\def\@scolelt#1{\def\@currbox{#1}\@addtonextcol}
\def\@sdblcolelt#1{\def\@currbox{#1}\@addtodblcol}
\def\@vtryfc #1{\global\setbox\@outputbox\vbox{}\let\@elt\@wtryfc
      \@flsucceed \global\setbox\@outputbox
      \vbox to\@colht{\vskip \@fptop
      \vskip -\@fpsep \unvbox \@outputbox
      \vskip \@fpbot}\let\@elt\relax
      \xdef #1{\@failedlist\@flfail}%
      \xdef\@freelist{\@freelist\@flsucceed}}

\def\@wtryfc #1{\global\setbox\@outputbox\vbox{\unvbox\@outputbox
    \vskip\@fpsep\box #1}}
\def\@xtryfc #1{\@next\reserved@a\@trylist{}{}\@currtype
  \count #1\divide\@currtype\@xxxii \multiply\@currtype\@xxxii
  \@bitor \@currtype \@failedlist \@testfp #1\ifdim
    \ht #1>\@colht \global\@testtrue\fi
    \if@test \@cons\@failedlist #1\else \@ytryfc #1\fi}
\def\@ytryfc #1{\begingroup \gdef\@flsucceed{\@elt #1}%
  \global\let\@flfail\@empty
  \@tempdima\ht #1\let\@elt\@ztryfc \@trylist
  \ifdim \@tempdima >\@fpmin
     \global\@fcolmadetrue \else \@cons\@failedlist #1\fi
  \endgroup \if@fcolmade \let\@elt\@gobble \fi}
```

```
\def\@ztryfc #1{\@tempcnta\count #1\divide\@tempcnta\@xxxii
    \multiply\@tempcnta\@xxxii \@bitor \@tempcnta {\@failedlist
    \@flfail}\@testfp #1\@tempdimb\@tempdima
    \advance\@tempdimb\ht #1\advance
    \@tempdimb\@fpsep \ifdim \@tempdimb >\@colht \global\@testtrue\fi
    \if@test \@cons\@flfail #1\else
      \@cons\@flsucceed #1\@tempdima\@tempdimb
    \fi}
⟨/2ekernel⟩
```

The major changes for float suppression and the changes to the float mechanism to make it conform to the documentation are in these next macros.

`\@addtobot` Lots of changes.

```
⟨*2ekernel | fltrace⟩
\def \@addtobot {%
⟨*trace⟩
   \tr@ce{***Start addtobot}%
⟨/trace⟩
   \@getfpsbit 4\relax
⟨*trace⟩
   \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi bot:
                                                      \the \@fpstype}%
⟨/trace⟩
   \ifodd \@tempcnta
     \@flsetnum \@botnum
     \ifnum \@botnum>\z@
       \@tempswafalse
       \@flcheckspace \@botroom \@botlist
       \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?

```
         \global \maxdepth \z@
         \@flupdates \@botnum \@botroom \@botlist
⟨*trace⟩
         \tr@ce{colroom (after-bot) = \the \@colroom}%
         \tr@ce{colnum (after-bot) = \the \@colnum}%
         \tr@ce{botnum (after-bot) = \the \@botnum}%
         \tr@ce{***Success: bot}%
⟨/trace⟩
         \@inserttrue
       \fi
⟨*trace⟩
     \else
       \tr@ce{Fail: botnum = \the \@botnum:
                             fpstype \the \@fpstype=ORD?}%
       \ifnum \@fpstype<\sixt@@n
         \tr@ce{ERROR: !b float not successful (addtobot)}%
       \fi
```

```
⟨/trace⟩
    \fi
  \fi
}
```

\@addtotoporbot Lots of changes.

```
\def \@addtotoporbot {%
⟨*trace⟩
  \tr@ce{***Start addtotoporbot}%
⟨/trace⟩
  \@getfpsbit \tw@
⟨*trace⟩
  \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi top:
                                                  \the \@fpstype}%
⟨/trace⟩
  \ifodd \@tempcnta
    \@flsetnum \@topnum
    \ifnum \@topnum>\z@
      \@tempswafalse
      \@flcheckspace \@toproom \@toplist
      \if@tempswa
        \@bitor\@currtype{\@midlist\@botlist}%
⟨*trace⟩
          \tr@ce{(mid+bot)list: \@midlist, \@botlist:
                             (addtotoporbot-before)}%
⟨/trace⟩
        \if@test
⟨*trace⟩
          \tr@ce{type already on list: mid or bot---sent to addtobot}%
⟨/trace⟩
        \else
         \@flupdates \@topnum \@toproom \@toplist
⟨*trace⟩
         \tr@ce{colroom (after-top) = \the \@colroom}%
         \tr@ce{colnum (after-top) = \the \@colnum}%
         \tr@ce{topnum (after-top) = \the \@topnum}%
         \tr@ce{***Success: top}%
⟨/trace⟩
         \@inserttrue
        \fi
      \fi
⟨*trace⟩
    \else
      \tr@ce{Fail: topnum = \the \@topnum: fpstype
                                         \the \@fpstype=ORD?}%
      \ifnum \@fpstype<\sixt@@n
        \tr@ce{ERROR: !t float not successful (addtotoporbot)}%
      \fi
⟨/trace⟩
    \fi
```

```
  \fi
  \if@insert
  \else
⟨*trace⟩
    \tr@ce{sent to addtobot (addtotoporbot)}%
⟨/trace⟩
    \@addtobot
  \fi
}
⟨/2ekernel | fltrace⟩
```

\@addtocurcol Lots of changes.

```
⟨*2ekernel | fltrace | flafter⟩
\def \@addtocurcol {%
⟨*trace⟩
 \tr@ce{***Start addtocurcol}%
⟨/trace⟩
  \@insertfalse
  \@setfloattypecounts
  \ifnum \@fpstype=8
⟨*trace⟩
    \tr@ce{fpstype !p only (addtocurcol): \the \@fpstype = 8?}%
⟨/trace⟩
  \else
    \ifnum \@fpstype=24
⟨*trace⟩
      \tr@ce{fpstype p only (addtocurcol): \the \@fpstype = 24?}%
⟨/trace⟩
    \else
      \@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 floats, so before comparing it with `\@textmin`, we add this to `\@textmin` also.

```
⟨*trace⟩
      \tr@ce{textfloatsheight (before) = \the \@textfloatsheight}%
⟨/trace⟩
      \advance \@textmin \@textfloatsheight
      \@reqcolroom \@pageht
```

This line must be removed since `\@specialoutput` changed.

```
%       \advance \@reqcolroom \@pagedp
⟨*trace⟩
      \tr@ce{textmin + textfloatsheight: \the \@textmin}%
      \tr@ce{page-so-far: \the \@reqcolroom}%
⟨/trace⟩
      \ifdim \@textmin>\@reqcolroom
        \@reqcolroom \@textmin
⟨*trace⟩
```

```
          \tr@ce{ORD? textmin being used}%
⟨/trace⟩
        \fi
        \advance \@reqcolroom \ht\@currbox
⟨*trace⟩
        \tr@ce{float size = \the \ht \@currbox (addtocurcol)}%
        \tr@ce{colroom = \the \@colroom (addtocurcol)}%
        \tr@ce{reqcolroom = \the \@reqcolroom (addtocurcol)}%
⟨/trace⟩
        \ifdim \@colroom>\@reqcolroom
          \@flsetnum \@colnum
          \ifnum \@colnum>\z@
            \@bitor\@currtype\@deferlist
⟨*trace⟩
            \tr@ce{deferlist: \@deferlist: (addtocurcol-before)}%
⟨/trace⟩
            \if@test
⟨*trace⟩
              \tr@ce{type already on list: defer (addtocurcol)}%
⟨/trace⟩
            \else
              \@bitor\@currtype\@botlist
⟨*trace⟩
            \tr@ce{botlist: \@botlist: (addtocurcol-before)}%
⟨/trace⟩
              \if@test
⟨*trace⟩
                \tr@ce{type already on list: bot---sent to addtobot}%
⟨/trace⟩
                \@addtobot
              \else
⟨*trace⟩
                \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi
                       here: \the \@fpstype}%
⟨/trace⟩
                \ifodd \count\@currbox
                  \advance \@reqcolroom \intextsep
                  \ifdim \@colroom>\@reqcolroom
                    \global \advance \@colnum \m@ne
                    \global \advance \@textfloatsheight \ht\@currbox
```

This may sometimes give an overestimate.

```
                    \global \advance \@textfloatsheight 2\intextsep
                    \@cons \@midlist \@currbox
⟨*trace⟩
                    \tr@ce{***Success: here}%
                    \tr@ce{textfloatsheight (after-here) =
                         \the \@textfloatsheight}%
                    \tr@ce{colnum (after-here) = \the \@colnum}%
⟨/trace⟩
```

CHANGE TO `\@addtocurcol`:

`\penalty\z@` changed to `\penalty\interlinepenalty` so `\samepage` works properly with figure and table environments. (Changed 23 Oct 86)

There is also an `\addpenalty\interlinepenalty` above.

Since in 2e `\samepage` is no longer supported, these could be removed.

Although it is best to use `\addvspace` in case two h floats come together, this makes other spacing more difficult to adjust; whereas if a user specifies two h floats together then they can more easily get the spacing correct by ad hoc commands.

It is necessary to adjust for the addition of `\parskip` here in case the float is added betweeen paragraphs (i.e. when in vertical mode).

Typesetting starts here.

```
                \if@nobreak
                \else
                  \addpenalty \interlinepenalty
                \fi
                \vskip \intextsep
                \box\@currbox
                \penalty\interlinepenalty
                \vskip\intextsep
                \ifnum\outputpenalty <-\@Mii \vskip -\parskip\fi
```

Typesetting ends here.

```
                \outputpenalty \z@
                \@inserttrue
⟨*trace⟩
              \else
                \tr@ce{Fail---no room at 2nd test of colroom
                              (addtocorcol \string\intextsep)}%
⟨/trace⟩
              \fi
            \fi
            \if@insert
            \else
⟨*2ekernel | fltrace⟩
⟨*trace⟩
              \tr@ce{not here: sent to addtotoporbot}%
⟨/trace⟩
              \@addtotoporbot
⟨/2ekernel | fltrace⟩
⟨*!2ekernel | fltrace⟩
⟨*trace⟩
              \tr@ce{not here: sent to addtobot}%
⟨/trace⟩
              \@addtobot
⟨/!2ekernel | fltrace⟩
            \fi
          \fi
        \fi
⟨*trace⟩
```

```
     \else
      \tr@ce{Fail: colnum = \the \@colnum:
            fpstype \the \@fpstype=ORD?}%
      \ifnum \@fpstype<\sixt@@n
       \tr@ce{ERROR: BANG float not successful (addtocurcol)}%
      \fi
⟨/trace⟩
     \fi
⟨*trace⟩
    \else
     \tr@ce{Fail---no room: fl box ht: \the \ht \@currbox
                           (addtocurcol)}%
⟨/trace⟩
    \fi
   \fi
  \fi
  \if@insert
  \else
   \@resethfps
⟨*trace⟩
   \tr@ce{put on deferlist (addtocurcol)}%
⟨/trace⟩
   \@cons\@deferlist\@currbox
⟨*trace⟩
   \tr@ce{deferlist: \@deferlist: (addtocurcol-after)}%
⟨/trace⟩
  \fi
}
⟨/2ekernel | fltrace | flafter⟩
```

`\@addtonextcol` Lots of changes.

```
⟨*2ekernel | fltrace⟩
\def\@addtonextcol{%
 \begingroup
⟨*trace⟩
  \tr@ce{***Start addtonextcol}%
⟨/trace⟩
  \@insertfalse
  \@setfloattypecounts
  \ifnum \@fpstype=8
⟨*trace⟩
   \tr@ce{fpstype not curcol: \the \@fpstype = 8?}%
⟨/trace⟩
  \else
   \ifnum \@fpstype=24
⟨*trace⟩
    \tr@ce{fpstype not curcol: \the \@fpstype = 24?}%
⟨/trace⟩
   \else
    \@flsettextmin
```

```
⟨*trace⟩
     \tr@ce{text-so-far: 0pt (top of col)}%
⟨/trace⟩
      \@reqcolroom \ht\@currbox
⟨*trace⟩
     \tr@ce{float size: \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
      \advance \@reqcolroom \@textmin
⟨*trace⟩
      \tr@ce{colroom = \the \@colroom (addtonextcol)}%
      \tr@ce{reqcolroom = \the \@reqcolroom (addtonextcol)}%
⟨/trace⟩
      \ifdim \@colroom>\@reqcolroom
        \@flsetnum \@colnum
        \ifnum\@colnum>\z@
           \@bitor\@currtype\@deferlist
⟨*trace⟩
           \tr@ce{deferlist: \@deferlist: (addtonextcol-before)}%
⟨/trace⟩
           \if@test
⟨*trace⟩
             \tr@ce{type already on list: defer (addtonextcol)}%
⟨/trace⟩
           \else
⟨*trace⟩
             \tr@ce{sent to addtotoporbot (addtonextcol)}%
⟨/trace⟩
             \@addtotoporbot
           \fi
        \fi
⟨*trace⟩
      \else
        \tr@ce{Fail---no room: fl box ht: \the \ht \@currbox
                                                 (addtonextcol)}%
⟨/trace⟩
      \fi
    \fi
  \fi
  \if@insert
  \else
⟨*trace⟩
    \tr@ce{put back on deferlist (addtonextcol)}%
⟨/trace⟩
    \@cons\@deferlist\@currbox
⟨*trace⟩
    \tr@ce{deferlist: \@deferlist: (addtonextcol-after)}%
⟨/trace⟩
  \fi
⟨*trace⟩
  \tr@ce{End of addtonextcol -- locally counts:}%
```

```
  \tr@ce{ col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
 \endgroup
⟨*trace⟩
 \tr@ce{End of addtonextcol -- globally counts:}%
 \tr@ce{col: \the \@colnum. top: \the \@topnum. bot: \the \@botnum.}%
⟨/trace⟩
}
```

\@addtodblcol Lots of changes.

```
\def\@addtodblcol{%
 \begingroup
⟨*trace⟩
 \tr@ce{***Start addtodblcol}%
⟨/trace⟩
  \@insertfalse
  \@setfloattypecounts
  \@getfpsbit \tw@
⟨*trace⟩
  \tr@ce{fpstype \ifodd \@tempcnta OK \else not \fi dbltop:
                                                   \the \@fpstype}%
⟨/trace⟩
  \ifodd\@tempcnta
    \@flsetnum \@dbltopnum
    \ifnum \@dbltopnum>\z@
      \@tempswafalse
      \ifdim \@dbltoproom>\ht\@currbox
        \@tempswatrue
⟨*trace⟩
        \tr@ce{Space OK: \@dbltoproom =
               \the \@dbltoproom > \the \ht \@currbox
                                      (dbltoproom)}%
⟨/trace⟩
      \else
⟨*trace⟩
        \tr@ce{fpstype: \the \@fpstype (addtodblcol)}%
⟨/trace⟩
        \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
          \tr@ce{BANG float ignoring \@dbltoproom}%
          \tr@ce{\@spaces \@dbltoproom = \the \@dbltoproom.
                          Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
```

Need to check that there is room on the page, using the local value of `\@textmin` to make the necessary adjustment to `\@dbltoproom`.

```
          \advance \@dbltoproom \@textmin
⟨*trace⟩
          \tr@ce{Local value of texmin: \the\@textmin}%
          \tr@ce{\@spaces space on page = \the \@dbltoproom.
```

```
                        Ht float: \the \ht \@currbox-BANG}%
⟨/trace⟩
        \ifdim \@dbltoproom>\ht\@currbox
          \@tempswatrue
⟨*trace⟩
          \tr@ce{Space OK BANG: space on page = \the \@dbltoproom >
                                           \the \ht \@currbox}%
        \else
          \tr@ce{fpstype: \the \@fpstype}%
          \tr@ce{Fail---no room dbltoproom-BANG?:}%
          \tr@ce{\@spaces space on page = \the \@dbltoproom.
                        Ht float: \the \ht \@currbox}%
⟨/trace⟩
        \fi
        \advance \@dbltoproom -\@textmin
⟨*trace⟩
      \else
        \tr@ce{fpstype: \the \@fpstype}%
        \tr@ce{Fail---no room dbltoproom-ORD?:}%
        \tr@ce{\@spaces \@dbltoproom = \the \@dbltoproom.
                        Ht float: \the \ht \@currbox}%
⟨/trace⟩
      \fi
    \fi
    \if@tempswa
        \@bitor \@currtype \@dbldeferlist
⟨*trace⟩
        \tr@ce{dbldeferlist: \@dbldeferlist: (before)}%
⟨/trace⟩
        \if@test
⟨*trace⟩
           \tr@ce{type already on list: dbldefer}%
⟨/trace⟩
        \else
           \@tempdima -\ht\@currbox
           \advance\@tempdima
             -\ifx \@dbltoplist\@empty \dbltextfloatsep \else
                                       \dblfloatsep \fi
           \global \advance \@dbltoproom \@tempdima
           \global \advance \@colht \@tempdima
           \global \advance \@dbltopnum \m@ne
           \@cons \@dbltoplist \@currbox
⟨*trace⟩
           \tr@ce{dbltopnum (after) = \the \@dbltopnum}%
           \tr@ce{***Success: dbltop}%
⟨/trace⟩
           \@inserttrue
        \fi
    \fi
⟨*trace⟩
```

```
      \else
        \tr@ce{Fail: dbltopnum = \the \@dbltopnum: fpstype
                                                \the \@fpstype=ORD?}%
        \ifnum \@fpstype<\sixt@@n
          \tr@ce{ERROR: !t float not successful (addtodblcol)}%
        \fi
⟨/trace⟩
      \fi
    \fi
    \if@insert
    \else
⟨*trace⟩
      \tr@ce{put on dbldeferlist}%
⟨/trace⟩
      \@cons\@dbldeferlist\@currbox
⟨*trace⟩
      \tr@ce{dbldeferlist: \@dbldeferlist: (after)}%
⟨/trace⟩
    \fi
⟨*trace⟩
    \tr@ce{End of addtodblcol -- locally count:}%
    \tr@ce{ dbltop: \the \@dbltopnum.}%
⟨/trace⟩
  \endgroup
⟨*trace⟩
  \tr@ce{End of addtodblcol -- globally count:}%
  \tr@ce{dbltop: \the \@dbltopnum.}%
⟨/trace⟩
}
⟨/2ekernel | fltrace⟩
```

```
⟨*2ekernel⟩
\def\@addmarginpar{\@next\@marbox\@currlist{\@cons\@freelist\@marbox
    \@cons\@freelist\@currbox}\@latexbug\@tempcnta\@ne
    \if@twocolumn
        \if@firstcolumn \@tempcnta\m@ne \fi
    \else
      \if@mparswitch
         \ifodd\c@page \else\@tempcnta\m@ne \fi
      \fi
      \if@reversemargin \@tempcnta -\@tempcnta \fi
    \fi
    \ifnum\@tempcnta <\z@  \global\setbox\@marbox\box\@currbox \fi
    \@tempdima\@mparbottom
    \advance\@tempdima -\@pageht
    \advance\@tempdima\ht\@marbox
    \ifdim\@tempdima >\z@
      \@latex@warning@no@line {Marginpar on page \thepage\space moved}%
    \else
      \@tempdima\z@
```

```
    \fi
    \global\@mparbottom\@pageht
    \global\advance\@mparbottom\@tempdima
    \global\advance\@mparbottom\dp\@marbox
    \global\advance\@mparbottom\marginparpush
    \advance\@tempdima -\ht\@marbox
```

Putting box movement inside the 'marbox': CAR 92/12/12

Corrected typo: FMi 92/12/30

```
    \global\setbox \@marbox
                   \vbox {\vskip \@tempdima
                          \box \@marbox}%
    \global \ht\@marbox \z@
    \global \dp\@marbox \z@
```

Sticking (rather than gluing:-}) the 'marbox' to the line above, changed vskip to kern:
CAR92/12/12

```
    \kern -\@pagedp
    \nointerlineskip
    \hb@xt@\columnwidth
      {\ifnum \@tempcnta >\z@
          \hskip\columnwidth \hskip\marginparsep
       \else
          \hskip -\marginparsep \hskip -\marginparwidth
       \fi
       \box\@marbox \hss}%
```

```
\nobreak              %% No longer needed.  CAR92/12
\vskip -\@tempdima    %% No longer needed.  CAR92/12
```

```
    \nointerlineskip
    \hbox{\vrule \@height\z@ \@width\z@ \@depth\@pagedp}}
```

63.1.1 Kludgeins

This part of the file is part of the implementation of the following two new commands for LaTeX2e.

```
\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 execpt 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).

`\@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 chnaged.

```
\newinsert \@kludgeins
\dimen\@kludgeins \maxdimen
\count\@kludgeins 1000
```

`\enlargethispage` The user command.

`\enlargethispage*`

```
\def \enlargethispage {%
  \@ifstar
    {%
⟨*trace⟩
      \tr@ce{Enlarging page height * }%
⟨/trace⟩
      \@enlargepage{\hbox{\kern\p@}}}%
    {%
⟨*trace⟩
      \tr@ce{Enlarging page height exactly---}%
⟨/trace⟩
      \@enlargepage\@empty}%
}
```

`\@enlargepage` This actually inserts the insert, after checking for extreme values of the change.

```
\def\@enlargepage#1#2{%
⟨*trace⟩
  \tr@ce{\@spaces\@spaces by #2}%
⟨/trace⟩
  \@tempskipa#2\relax
  \ifdim \@tempskipa>.5\maxdimen
    \@latexerr{Suggested extra height (\the\@tempskipa)
                 dangerously large}\@eha
  \else
    \ifdim \vsize<.5\maxdimen
⟨*trace⟩
     \tr@ce {Kludgeins added--pagegoal before: \the\pagegoal}%
⟨/trace⟩
     \@bsphack
       \insert\@kludgeins{#1\vskip-\@tempskipa}%
     \@esphack
```

This next bit is for tracing only:

```
⟨*trace⟩
```

```
        \ifvmode \par
          \tr@ce {Kludgeins added--pagegoal after: \the \pagegoal}%
        \fi
⟨/trace⟩
      \else
        \@latexerr{Page height already too large}\@eha
      \fi
    \fi
}
⟨/2ekernel⟩
```

63.1.2 Float control

This part implements controllable floats and other changes to the float mechanism.

It provides, at the doument level, the following command for inclusion in LaTeX2e.

```
\suppressfloats
```

This suppresses all further floats on the current page.

With an optional argument it suppresses only floats only in certain positions on the current page.

> `[t]` suppresses only floats at the top of the page `[b]` suppresses only floats at the bottom of the page

It also enables the use of an extra specifier, `!`, in the location optional argument of a float. If this is present then, just for this particular float, whenever it is processed by the float mechanism the followinhg 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 suplies a reasonably exhaustive, and somewhat baroque, means of tracing some aspects of the float mechanism.

More tracing.

`\tr@ce` `\notrace` `\tracefloats` `\@traceval` `\tracefloatvals` `\@tracemessage`

Set-up tracing for floats independent of other tracing as it produces mega-output. Default is no tracing.

```
⟨*trace⟩
\def \@tracemessage #1{\typeout{LaTeX2e: #1}}
\def \tracefloats{\let \tr@ce \@tracemessage}
\def \notrace {\let \tr@ce \@gobble}
```

```
\notrace
\def \@traceval #1{\tr@ce{\string #1 = \the #1}}
\def \tracefloatvals{%
  \@dblfloatplacement
  \@floatplacement
  \@traceval\@colnum
  \@traceval\@colroom
  \@traceval\@topnum
  \@traceval\@toproom
  \@traceval\@botnum
  \@traceval\@botroom
  \@traceval\@fpmin
  \tr@ce{\string\textfraction = \textfraction}%
  \@traceval\@dbltopnum
  \@traceval\@dbltoproom
}
⟨/trace⟩
⟨*flafter⟩
\providecommand\tr@ce[1]{}
⟨/flafter⟩
```

`\suppressfloats` `\@flstop` Float suppression commands: these set the relevant counter globally to zero. Thus they are overridden for a particular float by an ! specifier.

```
⟨*2ekernel⟩
\def \suppressfloats {%
   \@ifnextchar [%
     \@flstop
    {\global \@colnum \z@}%
}
```

Maybe this should be a loop over `#1`?

```
\def \@flstop [#1]{%
   \if t#1%
     \global \@topnum \z@
   \fi
   \if b#1%
     \global \@botnum \z@
   \fi
}
```

Manipulation of float placement and type; both their strings and the corresponding count registers.

`\@fpstype` `\@reqcolroom` `\@textfloatsheight` First a new count register to go with `\@currtype`.

Then a new skip register, for information needed to remove the `\@maxsep` conservatism: it is possible that this could use a temporary register.

Finally a dimension register to hold the total height of in-text floats on the current page. This is needed to implement a major change in the functionality of `\@addtocurcol` which is, nevertheless, a bug fix. It is not local and therefore cannot be a temporary register.

```
\newcount \@fpstype
\newdimen \@reqcolroom
\newdimen \@textfloatsheight
⟨/2ekernel⟩
```

`\@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@\@captype \endcsname}%
   \edef \reserved@a {\the\@temptokena}%
   \@onelevel@sanitize \reserved@a
   \edef \@fps {\@fps\reserved@a}%
}
```

```
⟨*2ekernel | fltrace⟩
\def \@fpsadddefault {%
⟨*trace⟩
   \tr@ce{fps changed from: \@fps}%
⟨/trace⟩
   \edef \@fps {\@fps\csname fps@\@captype \endcsname}%
   \@latex@warning {%
     No positions in optional float specifier.\MessageBreak
     Default added (so using `\@fps')}%
}
```

`\@setfloattypecounts` Sets counters `\@fpstype` and `\@currtype`.

BANG == bit4 of `\count\@currbox` = 0.

```
\def \@setfloattypecounts {%
  \@currtype \count\@currbox
  \@fpstype \count\@currbox
  \divide\@currtype\@xxxii \multiply\@currtype\@xxxii
  \advance \@fpstype -\@currtype
⟨*trace⟩
  \tr@ce{(mod 32) fpstype: \the \@fpstype}%
  \tr@ce{(mult of 32) currtype: \the \@currtype}%
% Tracing only: but some should be changed into real errors/warnings?
  \ifnum \@fpstype<\sixt@@n
    \ifnum \@fpstype=\z@
      \tr@ce{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 0?}%
    \fi
    \ifnum \@fpstype=\@ne
      \tr@ce{WARNING: only h, fpstype = \the \@fpstype = 1?}%
    \fi
    \tr@ce{BANG float}%
  \else
```

```
    \ifnum \@fpstype=\sixt@@n
      \tr@ce{ERROR: no PLACEMENT, fpstype = \the \@fpstype = 16?}%
    \fi
    \ifnum \@fpstype=17
      \tr@ce{WARNING: only h, fpstype = \the \@fpstype = 17?}%
    \fi
    \tr@ce{ORD float}%
  \fi
⟨/trace⟩
}
⟨/2ekernel | fltrace⟩
```

Macros for getting, testing and setting bits of the fps.

`\@getfpsbit` Sets `\@tempcnta` to required bit of `\count\@currbox`.

```
⟨*2ekernel⟩
\def \@getfpsbit {%
   \@boxfpsbit \@currbox
}
```

`\@boxfpsbit` Used above.

```
\def \@boxfpsbit #1#2{%
   \@tempcnta \count#1%
   \divide \@tempcnta #2\relax
}
```

`\@testfp` New definition of the float page test.

```
\def \@testfp #1{%
   \@boxfpsbit #18\relax % Really `#1 8' for human readers!
   \ifodd \@tempcnta
   \else
     \global \@testtrue
   \fi
}
```

`\@setfpsbit` Sets required bit of `\@tempcnta` (to 1).

```
\def \@setfpsbit #1{%
   \@tempcntb \@tempcnta
   \divide \@tempcntb #1\relax
   \ifodd \@tempcntb
   \else
     \advance \@tempcnta #1\relax
   \fi
}
⟨/2ekernel⟩
```

`\@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`.

```
⟨*2ekernel | fltrace⟩
\def \@resethfps {%
   \let\reserved@a\@empty
   \ifnum \@fpstype=\@ne
      \def \reserved@a {!}%
      \@fpstype 17
   \fi
   \ifnum \@fpstype=17
     \global \advance \count\@currbox \tw@
     \@latex@warning@no@line {%
       `\reserved@a h' float specifier changed to `\reserved@a ht'}%
⟨*trace⟩
     \tr@ce{%
        `t' added to `\reserved@a h'- new Count: \the \count\@currbox}%
⟨/trace⟩
   \fi
}
```

Special stuff for BANG floats.

`\@flsetnum` Ignores any zero float counter value in case BANG.

It uses a local assignment to the normally global counter: a bit naughty, perhaps?

These assgnments are safe so long as the counter involved is only consulted once (i.e. only for the 'bang float') with the changed value. This is the case within `\@addtocurcol` because it is used only once within a call of the output routine (which forms a group).

For `\@addtonextcol` this is achieved by putting a group around its code; this is needed because it is called (by `\@startcolumn`) for each float which was on the deferlist. Almost identical considerations pertain to `\@addtodblcol`. There may be more efficient ways to handle this, but the group seems to be the simplest.

```
\def \@flsetnum #1{%
⟨*trace⟩
   \tr@ce{fpstype: \the \@fpstype (flsetnum \string#1)}%
⟨/trace⟩
   \ifnum \@fpstype<\sixt@@n
     \ifnum #1=\z@
⟨*trace⟩
      \tr@ce{BANG float resetting \string#1 to 1}%
⟨/trace⟩
       #1\@ne
     \fi
   \fi
⟨*trace⟩
   \tr@ce{#1 (before) = \the #1}%
⟨/trace⟩
}
```

`\@flsettextmin` This ignores `\textfraction` space restriction in case BANG.

```
\def \@flsettextmin {%
⟨*trace⟩
   \tr@ce{fpstype: \the \@fpstype (flsettextmin)}%
⟨/trace⟩
   \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
     \tr@ce{BANG ignoring textmin}%
⟨/trace⟩
     \@textmin \z@
   \else
     \@textmin \textfraction\@colht
⟨*trace⟩
     \tr@ce{ORD textmin = \the \@textmin}%
⟨/trace⟩
   \fi
}
```

`\@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 `@tempswa` true if there is room for `\@currbox`.

```
\def \@flcheckspace #1#2{%
   \advance \@reqcolroom
     \ifx #2\@empty \textfloatsep \else \floatsep \fi
⟨*trace⟩
   \tr@ce{colroom = \the \@colroom (flcheckspace \string#1 \string#2)}%
   \tr@ce{reqcolroom = \the \@reqcolroom
                                    (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
   \ifdim \@colroom>\@reqcolroom
     \ifdim #1>\ht\@currbox
       \@tempswatrue
⟨*trace⟩
       \tr@ce{Space OK: #1 = \the #1 > \the \ht \@currbox
                                    (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
     \else
⟨*trace⟩
       \tr@ce{fpstype: \the \@fpstype
                                    (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
       \ifnum \@fpstype<\sixt@@n
⟨*trace⟩
         \tr@ce{BANG float ignoring #1
                                    (flcheckspace \string#1 \string#2):}%
         \tr@ce{\@spaces #1 = \the #1.  Ht float: \the \ht \@currbox
                                                              BANG}%
⟨/trace⟩
         \@tempswatrue
```

```
⟨*trace⟩
      \else
        \tr@ce{Fail---no room (flcheckspace \string#1 \string#2)
                      (fpstype \the \@fpstype=ORD?):}%
        \tr@ce{\@spaces #1 = \the #1.  Ht float: \the \ht \@currbox
                                                        ORD?}%
⟨/trace⟩
      \fi
    \fi
⟨*trace⟩
  \else
    \tr@ce{Fail---no room at 2nd test of colroom
                     (flcheckspace \string#1 \string#2)}%
⟨/trace⟩
  \fi
}
⟨/2ekernel | fltrace⟩
```

`\@flupdates` This updates everything when a float is placed.

```
⟨*2ekernel⟩
\def \@flupdates #1#2#3{%
  \global \advance #1\m@ne
  \global \advance \@colnum \m@ne
  \@tempdima -\ht\@currbox
  \advance \@tempdima
    -\ifx #3\@empty \textfloatsep \else \floatsep \fi
  \global \advance #2\@tempdima
  \global \advance \@colroom \@tempdima
  \@cons #3\@currbox
}
⟨/2ekernel⟩
```

Interesting facts about float mechanisms past and present, together with a summary of various features, some unresolved:

1. The value `\textfraction` does not affect the processing of doublecol floats: this seems sensible, but should be documented.
2. `\twocolumn` floatplacement was wrong: dbl not needed, ord needed.
3. `\@floatplacement` was not called after `\@startdblcol` or `\@topnewpage`. This has been changed; it is clearly a bug fix.
4. The use `\@topnewpage` when `\dblfigrule` is non-trivial produced a rule in the wrong place. This has been fixed by not using `\dblfigrule` when processing the 'float' from `\@topnewpage`.
5. If the specifier was just h and the float could not be put here, it went on the deferlist and stayed there until a clearpage. It now gets changed to a 'th': this is only an error-recovery action, putting just h or !h should be deprecated.

6. `\@dblmaxsep` was 'the maximum of `\dblfloatsep` and `\dbltexfloatsep`'. But it was never used! Now gone completely, like `\@maxsep`.

7. After an h float is put on a page, it was counted as text when applying the `\textfraction` test; this is possibly too big a change although it is a bug fix?

8. Two consecutive h floats are separated by twice `\intextsep`: this could be changed to one by use of `\addvspace`, OK? Note that it would also mean that less space is put in if an h float immediaiely follows other spaces. This is also possibly too big a change, at least for compatibility mode? Or it may be simply wrong! It has not been changed.

9. Now `\@addtocurcol` checks first for just p fps. I think that this is an increase in efficiency, but maybe the coding should be made even more efficient.

10. `\@tryfcolumn` now tests if the list is empty first, otherwise lots of wasted time! Thus this test has been removed from `\@startcolumn`. As Frank pointed out, this makes `\@startcolumn` less efficient. But it is now the same as `\@startdblcolumn`: I can see no reason why they should be different, but which is best?

11. Why is `\@colroom` set in `\@doclearpage`?

12. Footnotes. Check what `\clearpage` does when footnotes are left over. Footnotes are not put on float pages and, also, `\@addtonextcol` ignores the existence of held-over footnotes in deciding what floats can go on the page. Not changed.

13. `\clearpage` can still lose non-boxes, at least when floats are involved. It also moves some to the 'wrong page', but this may be a coding problem.

14. The ! option makes it necessary to check in `\output` that there is enough room left on the page after adding a float. (This would have been necessary anyway if anyone set `\@textmin` too close to zero! A similar danger existed also if the text in a `\twocolumn[text]` entity gets too large.) The current implementation of this also makes the normal case a little less efficient, OK? Not enough room means, at present, less than `\baselineskip`, with a warning: is this OK? Should it be made generic (another parameter)?

15. There are four possibilities for supporting this:

    `\twocolumn[\maketitle more text]`

    One is to change `\maketitle` slightly to allow this. Another is to change `\@topnewpage` so that more than one `\twocolumn[]` command is allowed; in this case `\maketitle\twoclumn[more text]` will work. The former is more robust from the user's viewpoint, but makes the code for `\maketitle` rather ad hoc (maybe it is already?). Another is to misuse the global twocolumn flag locally within `\@topnewpage`. Yet another is to move the column count register from the multicol package into the kernel. This has beeen done.

16. Where should the reinserts be put to maximise the probability that footmotes come out on the correct page? Or should we go for as much comaptibility as psossible (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 adjustement 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.

```
⟨*2ekernel | fltrace⟩
\def\@makefcolumn #1{%
  \begingroup
    \@fpmin \z@
    \let \@testfp \@gobble
    \@tryfcolumn #1%
  \endgroup
⟨*trace⟩
  \if@fcolmade
    \tr@ce{PAGE: in \string\clearpage \if@twocolumn ---twocolumn\fi---}%
    \tr@ce{----- float column/page completed from \string#1}%
  \fi
⟨/trace⟩
}
```

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.

```
\def\@outputdblcol{%
  \if@firstcolumn
    \global \@firstcolumnfalse
    \global \setbox\@leftcolumn \box\@outputbox
⟨*trace⟩
    \tr@ce{PAGE: first column boxed}%
⟨/trace⟩
  \else
    \global \@firstcolumntrue
    \setbox\@outputbox \vbox {%
                         \hb@xt@\textwidth {%
                           \hb@xt@\columnwidth {%
                             \box\@leftcolumn \hss}%
                           \hfil
                           \vrule \@width\columnseprule
                           \hfil
                           \hb@xt@\columnwidth {%
                             \box\@outputbox \hss}%
                                             }%
                            }%
⟨*trace⟩
    \tr@ce{PAGE: second column also boxed}%
⟨/trace⟩
    \@combinedblfloats
    \@outputpage
⟨*trace⟩
    \tr@ce{PAGE: two column page completed}%
⟨/trace⟩
    \begingroup
```

```
      \@dblfloatplacement
      \@startdblcolumn
```

This loop could be replaced by an `\expandafter` tail recursion in `\@startdblcolumn`.

```
      \@whilesw\if@fcolmade \fi
        {\@outputpage
⟨*trace⟩
      \tr@ce{PAGE: double float page completed}%
⟨/trace⟩
         \@startdblcolumn}%
    \endgroup
  \fi
}
⟨/2ekernel | fltrace⟩
```

63.1.3 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 douments 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.

```
⟨*2ekernel⟩
\newcount\c@topnumber
\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.

```
\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.

```
\newcount\c@bottomnumber
\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 floats at the bottom.

```
\newcommand\bottomfraction{.3}
```

`\c@totalnumber` This counter holds the maximum number of floats that can appear on any text page or column.

```
\newcount\c@totalnumber
\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.

```
\newcommand\textfraction{.2}
```

`\floatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page or column that must be occupied by floating objects before a 'float page' is produced.

```
\newcommand\floatpagefraction{.5}
```

`\c@dbltopnumber` This counter holds the maximum number of double-column floats that can appear on the top of a two-column text page.

```
\newcount\c@dbltopnumber
\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 floats at the top.

```
\newcommand\dbltopfraction{.7}
```

`\dblfloatpagefraction` This macro holds the minimum proportion (as a decimal number) of a page that must be occupied by double-column floating objects before a 'double-column float page' is produced.

```
\newcommand\dblfloatpagefraction{.5}
```

Floats on a text page

`\floatsep` `\textfloatsep` `\intextsep` When a floating object is placed on a page with text, these parameters control the seperation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode. They are all rubber lengths.

`\floatsep` is the space between adjacent floats that are placed at the top or bottom of the text page or column.

`\textfloatsep` is the space between the main text and floats at the top or bottom of the page or column.

`\intextsep` is the space between in-text floats and the text.

```
\newskip\floatsep
\newskip\textfloatsep
\newskip\intextsep
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {12\p@ \@plus 2\p@ \@minus 2\p@}
```

`\dblfloatsep` `\dbltextfloatsep` When double-column floats (floating objects that span the whole `\textwidth`) are placed at the top of a text page in two-column mode, the separation between the float and the text is controlled by `\dblfloatsep` and `\dbltextfloatsep`. They are rubber lengths.

`\dblfloatsep` is the space between adjacent double-column floats placed at the top of the text page.

\dbltextfloatsep is the space between the main text and double-column floats at the top of the page.

```
\newskip\dblfloatsep
\newskip\dbltextfloatsep
\setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
```

Floats on their own page or column

\@fptop \@fpsep \@fpbot When floating objects are placed on a seperate 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 ais inserted. Between adjacent floats \@fpsep is inserted.

These paramaters 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.

```
\newskip\@fptop
\newskip\@fpsep
\newskip\@fpbot
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
```

\@dblfptop \@dblfpsep \@dblfpbot Double-column 'float pages' in two-column mode use similar parameters.

```
\newskip\@dblfptop
\newskip\@dblfpsep
\newskip\@dblfpbot
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
```

\topfigrule \botfigrule \dblfigrule 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.

```
\let\topfigrule=\relax
\let\botfigrule=\relax
\let\dblfigrule=\relax
⟨/2ekernel⟩
```

File L

ltclass.dtx

64 Introduction

This file implements the following declarations, which replace `\documentstyle` in LaTeX 2ε 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 2ε and LaTeX2.09 is that LaTeX 2ε 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.

65 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 LaTeX2.09 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 LaTeX2.09 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 ⟨*package-option-list*⟩, each package processes the ⟨*main-option-list*⟩. This means that options that affect all of the packages can be given globally, rather than repeated for every package.

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

`filecontents` 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.

The environment is allowed only before `\documentclass` to ensure that all packages or options necessary for this particular run are present when needed.

65.1 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.

66 Class and Package interface

66.1 Class name and version

`\ProvidesClass` A class can identify itself with the `\ProvidesClass{`⟨*name*⟩`}[`⟨*version*⟩`]` command. The ⟨*version*⟩ should begin with a date in the format `YYYY/MM/DD`.

66.2 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`.

66.3 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` `\PassOptionsToClass` Packages can pass options to other packages using:
`\PassOptionsToPackage{`⟨*options*⟩`}{`⟨*package*⟩`}`.
This adds the ⟨*options*⟩ to the options list of any future `\RequirePackage` or `\usepackage` command. For example:

```
\PassOptionsToPackage{foo,bar}{fred}
\RequirePackage[baz]{fred}
```

is the same as:

```
\RequirePackage[foo,bar,baz]{fred}
```

`\@ifpackageloaded` `\@ifclassloaded` To find out if a package has already been loaded, use
`\@ifpackageloaded{`⟨*package*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`.

`\@ifpackagelater` `\@ifclasslater` To find out if a package has already been loaded with a version more recent than ⟨*version*⟩, use `\@ifpackagelater{`⟨*package*⟩`}{`⟨*version*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`.

`\@ifpackagewith` `\@ifclasswith` To find out if a package has already been loaded with at least the options ⟨*options*⟩, use `\@ifpackagewith{`⟨*package*⟩`}{`⟨*options*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`.

66.4 Declaring new options

Options for classes and packages are built using the same macros.

`\DeclareOption` To define a builtin option, use `\DeclareOption{`⟨*name*⟩`}{`⟨*code*⟩`}`.

`\DeclareOption*` To define the default action to perform for local options which have not been declared, use `\DeclareOption*{`⟨*code*⟩`}`.

Note: there should be no use of
`\RequirePackage`, `\DeclareOption`, `\DeclareOption*` or `\ProcessOptions` inside `\DeclareOption` or `\DeclareOption*`.

Possible uses for `\DeclareOption*` include:

`\DeclareOption*{}`

Do nothing. Silently accept unknown options. (This suppresses the usual warnings.)

`\DeclareOption*{\@unkownoptionerror}`

Complain about unknown local options. (The initial setting for package files.)

`\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{`⟨*pkg-name*⟩`}`

Handle the 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.

66.5 Safe Input Macros

`\InputIfFileExists` `\InputIfFileExists{`⟨*file*⟩`}{`⟨*then*⟩`}{`⟨*else*⟩`}`
Inputs ⟨*file*⟩ if it exists. Immediately before the input, ⟨*then*⟩ is executed. Otherwise ⟨*else*⟩ is executed.

`\IfFileExists` As above, but does not input the file.

One thing you might like to put in the ⟨*else*⟩ clause is

`\@missingfileerror` This starts an interactive request for a filename, supplying default extensions. Just hitting return causes the whole input to be skipped. (Perhaps we should let `x` denote quit the current run, as long as no one wants to input a file of that name.

`\input` This has been redefined from the LaTeX2.09 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.

67 Implementation

```
⟨*2ekernel⟩
```

`\if@compatibility` The flag for compatibilty mode.

```
\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`.

```
\def\@documentclasshook{%
   \ifx\@normalsize\@undefined
      \let\@normalsize\normalsize
   \fi
}
```

`\@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`.

```
\let\@declaredoptions\@empty
```

`\@classoptionslist` List of options of the main class.

```
\let\@classoptionslist\relax
```

`\@unusedoptionlist` List of options of the main class that haven't been declared or loaded as class option files.

```
\let\@unusedoptionlist\@empty
```

`\CurrentOption` Name of current package or option.

```
\let\CurrentOption\@empty
```

`\@currname` Name of current package or option.

```
\let\@currname\@empty
```

`\@currext` The current file extension.

```
\global\let\@currext=\@empty
```

`\@clsextension` `\@pkgextension` The two possible values of `\@currext`.

```
\def\@clsextension{cls}
\def\@pkgextension{sty}
\@onlypreamble\@clsextension
\@onlypreamble\@pkgextension
```

`\@pushfilename` `\@popfilename` `\@currnamestack` Commands to push and pop the file name and extension.
`#1` current name.
`#2` current extension.
`#3` current catcode of `@`.
`#4` Rest of the stack.

```
\def\@pushfilename{%
  \xdef\@currnamestack{%
    {\@currname}%
    {\@currext}%
    {\the\catcode`\@}%
    \@currnamestack}}
\@onlypreamble\@pushfilename
```

```
\def\@popfilename{\expandafter\@p@pfilename\@currnamestack\@nil}
\@onlypreamble\@popfilename
```

```
\def\@p@pfilename#1#2#3#4\@nil{%
  \gdef\@currname{#1}%
  \gdef\@currext{#2}%
  \catcode`\@#3\relax
  \gdef\@currnamestack{#4}}
\@onlypreamble\@p@pfilename
```

```
\gdef\@currnamestack{}
\@onlypreamble\@currnamestack
```

`\@ptionlist`

Returns the option list of the file.

```
\def\@ptionlist#1{%
  \@ifundefined{opt@#1}\@empty{\csname opt@#1\endcsname}}
\@onlypreamble\@ptionlist
```

`\@ifpackageloaded`
`\@ifclassloaded`

`\@ifpackageloaded{`⟨*name*⟩`}` Checks to see whether a file has been loaded.

```
\def\@ifpackageloaded{\@ifl@aded\@pkgextension}
\def\@ifclassloaded{\@ifl@aded\@clsextension}
\@onlypreamble\@ifpackageloaded
\@onlypreamble\@ifclassloaded
```

```
\def\@ifl@aded#1#2{%
  \expandafter\ifx\csname ver@#2.#1\endcsname\relax
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
\@onlypreamble\@ifl@aded
```

`\@ifpackagelater`
`\@ifclasslater`

`\@ifpackagelater{`⟨*name*⟩`}{YYYY/MM/DD}` Checks that the package loaded is more recent than the given date.

```
\def\@ifpackagelater{\@ifl@ter\@pkgextension}
\def\@ifclasslater{\@ifl@ter\@clsextension}
\@onlypreamble\@ifpackagelater
\@onlypreamble\@ifclasslater
```

```
\def\@ifl@ter#1#2{%
  \expandafter\@ifl@t@r
    \csname ver@#2.#1\endcsname}
\@onlypreamble\@ifl@ter
```

This internal macro is also used in `\NeedsTeXFormat`.

```
\def\@ifl@t@r#1#2{%
  \ifnum\expandafter\@parse@version#1//00\@nil<%
        \expandafter\@parse@version#2//00\@nil
    \expandafter\@secondoftwo
  \else
    \expandafter\@firstoftwo
  \fi}
\@onlypreamble\@ifl@t@r
```

```
\def\@parse@version#1/#2/#3#4#5\@nil{#1#2#3#4 }
\@onlypreamble\@parse@version
```

\@ifpackagewith
\@ifclasswith

`\@ifpackagewith{`⟨*name*⟩`}{`⟨*option-list*⟩`}` Checks that ⟨*option-list*⟩ is a subset of the options **with** which ⟨*name*⟩ was loaded.

```
\def\@ifpackagewith{\@if@ptions\@pkgextension}
\def\@ifclasswith{\@if@ptions\@clsextension}
\@onlypreamble\@ifpackagewith
\@onlypreamble\@ifclasswith
```

```
\def\@if@ptions#1#2{%
  \@expandtwoargs\@if@pti@ns{\@ptionlist{#2.#1}}}
\@onlypreamble\@if@ptions
```

Probably shouldnt use `\CurrentOption` here. . . (changed to `\reserved@b`.)

```
\def\@if@pti@ns#1#2{%
  \let\reserved@a\@firstoftwo
  \@for\reserved@b:=#2\do{%
    \expandafter\in@\expandafter{\expandafter,\reserved@b,}{,#1,}%
    \ifin@\else\let\reserved@a\@secondoftwo\fi}%
  \reserved@a}
\@onlypreamble\@if@pti@ns
```

\ProvidesPackage

Checks that the current filename is correct, and defines `\ver@filename`.

```
\def\ProvidesPackage#1{%
  \xdef\@gtempa{#1}%
  \ifx\@gtempa\@currname\else
    \@latex@warning@no@line{You have requested
      \@cls@pkg\space`\@currname',\MessageBreak
       but the \@cls@pkg\space provides `#1'}%
  \fi
  \@ifnextchar[\@pr@videpackage{\@pr@videpackage[]}}%]
\@onlypreamble\ProvidesPackage
```

```
\def\@pr@videpackage[#1]{%
  \expandafter\xdef\csname ver@\@currname.\@currext\endcsname{#1}%
  \ifx\@currext\@clsextension
    \typeout{Document Class: \@gtempa\space#1}%
  \else
    \wlog{Package: \@gtempa\space#1}%
  \fi}
\@onlypreamble\@pr@videpackage
```

\ProvidesClass

Like `\ProvidesPackage`, but for classes.

```
\let\ProvidesClass\ProvidesPackage
\@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.

```
\def\ProvidesFile#1{%
```

```
  \begingroup
    \catcode`\ 10 %
    \@makeother\/%
    \@ifnextchar[{\@providesfile{#1}}{\@providesfile{#1}[]}}
\def\@providesfile#1[#2]{%
    \wlog{File: #1 #2}%
    \expandafter\xdef\csname ver@#1\endcsname{#2}%
  \endgroup}
```

\PassOptionsToPackage
\PassOptionsToClass

If the package has been loaded, we check that it was first loaded with the options. Otherwise we add the option list to that of the package.

```
\def\@pass@ptions#1#2#3{%
  \expandafter\xdef\csname opt@#3.#1\endcsname{%
    \@ifundefined{opt@#3.#1}\@empty
      {\csname opt@#3.#1\endcsname,}%
    \zap@space#2 \@empty}}
\@onlypreamble\@pass@ptions
\def\PassOptionsToPackage{\@pass@ptions\@pkgextension}
\def\PassOptionsToClass{\@pass@ptions\@clsextension}
\@onlypreamble\PassOptionsToPackage
\@onlypreamble\PassOptionsToClass
```

\DeclareOption
\DeclareOption*

Adds an option as a `\ds@` command, or the default `\default@ds` command.

```
\def\DeclareOption{%
  \let\@fileswith@pti@ns\@badrequireerror
  \@ifstar\@defdefault@ds\@declareoption}
\long\def\@declareoption#1#2{%
   \xdef\@declaredoptions{\@declaredoptions,#1}%
   \toks@{#2}%
   \expandafter\edef\csname ds@#1\endcsname{\the\toks@}}
\long\def\@defdefault@ds#1{%
  \toks@{#1}%
  \edef\default@ds{\the\toks@}}
\@onlypreamble\DeclareOption
\@onlypreamble\@declareoption
\@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.

```
\def\OptionNotUsed{%
  \ifx\@currext\@clsextension
    \xdef\@unusedoptionlist{%
      \ifx\@unusedoptionlist\@empty\else\@unusedoptionlist,\fi
      \CurrentOption}%
  \fi}
\@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*`.

```
% \let\default@ds\OptionNotUsed
```

\ProcessOptions
\ProcessOptions*

`\ProcessOptions` calls `\ds@option` for each known package option, then calls `\default@ds` for each option on the local options list. Finally resets all the declared options to `\relax`. The empty option does nothing, this has to be reset on the off chance it's set to `\relax` if an empty element gets into the `\@declaredoptions` list.

The star form is similar but executes options given in the order specified in the document, not the order they are declared in the file. In the case of packages, global options are executed before local ones.

```
\def\ProcessOptions{%
  \let\ds@\@empty
  \edef\@curroptions{\@ptionlist{\@currname.\@currext}}%
  \@ifstar\@xprocess@ptions\@process@ptions}
\@onlypreamble\ProcessOptions
```

```
\def\@process@ptions{%
  \@for\CurrentOption:=\@declaredoptions\do{%
    \ifx\CurrentOption\@empty\else
      \@expandtwoargs\in@{,\CurrentOption,}{%
         ,\ifx\@currext\@clsextension\else\@classoptionslist,\fi
         \@curroptions,}%
      \ifin@
        \@use@ption
        \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
      \fi
    \fi}%
  \@process@pti@ns}
\@onlypreamble\@process@ptions
```

```
\def\@xprocess@ptions{%
  \ifx\@currext\@clsextension\else
    \@for\CurrentOption:=\@classoptionslist\do{%
      \ifx\CurrentOption\@empty\else
        \@expandtwoargs\in@{,\CurrentOption,}{,\@declaredoptions,}%
        \ifin@
          \@use@ption
          \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
        \fi
      \fi}%
  \fi
  \@process@pti@ns}
\@onlypreamble\@xprocess@ptions
```

The common part of `\ProcessOptions` and `\ProcessOptions*`.

```
\def\@process@pti@ns{%
```

```
  \@for\CurrentOption:=\@curroptions\do{%
    \@ifundefined{ds@\CurrentOption}%
      {\@use@ption
       \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`.

```
       \@use@ption}%
```

Clear all the definitions for option code. First set all the declared options to `\relax`, then reset the 'default' and 'empty' options. and the lst of declared options.

```
  \@for\CurrentOption:=\@declaredoptions\do{%
    \expandafter\let\csname ds@\CurrentOption\endcsname\relax}%
  \let\@fileswith@pti@ns\@@fileswith@pti@ns
  \AtEndOfPackage{\let\@unprocessedoptions\relax}}
\@onlypreamble\@process@pti@ns
```

`\@options` `\@options` is a synonym for `\ProcessOptions*` for upward compatibility with LaTeX2.09 style files.

```
\def\@options{\ProcessOptions*}
\@onlypreamble\@options
```

`\@use@ption` Execute the code for the current option.

```
\def\@use@ption{%
  \@expandtwoargs\@removeelement\CurrentOption
  \@unusedoptionlist\@unusedoptionlist
  \csname ds@\CurrentOption\endcsname}
\@onlypreamble\@use@ption
```

`\ExecuteOptions` `\ExecuteOptions{`⟨*option-list*⟩`}` executes the code declared for each option.

```
\def\ExecuteOptions#1{%
  \def\reserved@a##1\@nil{%
    \@for\CurrentOption:=#1\do{\csname ds@\CurrentOption\endcsname}%
    \edef\CurrentOption{##1}}%
  \expandafter\reserved@a\CurrentOption\@nil}
\@onlypreamble\ExecuteOptions
```

The top-level commands, which just set some parameters then call the internal command, `\@fileswithoptions`.

`\documentclass` The main new-style class declaration.

```
\def\documentclass{%
  \let\documentclass\@twoclasseserror
  \if@compatibility\else\let\usepackage\RequirePackage\fi
  \@fileswithoptions\@clsextension}
\@onlypreamble\documentclass
```

\documentstyle 2.09 style class ‘style’ declaration.

```
\def\documentstyle{%
  \makeatletter\input{latex209.def}\makeatother
  \documentclass}
\@onlypreamble\documentstyle
```

\RequirePackage Load package if not already loaded.

```
\def\RequirePackage{%
  \@fileswithoptions\@pkgextension}
\@onlypreamble\RequirePackage
```

\LoadClass Load class.

```
\def\LoadClass{%
  \ifx\@currext\@pkgextension
     \@latex@error
      {\noexpand\LoadClass in package file}%
      {You may only use \noexpand\LoadClass in a class file.}%
  \fi
  \@fileswithoptions\@clsextension}
\@onlypreamble\LoadClass
```

\usepackage To begin with, `\usepackage` produces an error. This is reset by `\documentclass`.

```
\def\usepackage#1#{%
  \@latex@error
    {\noexpand \usepackage before \string\documentclass}%
    {\noexpand \usepackage may only appear in the document
      preamble, i.e.,\MessageBreak
      between \noexpand\documentclass and
      \string\begin{document}.}%
  \@gobble}
\@onlypreamble\usepackage
```

\NeedsTeXFormat Check that the document is running on the correct system.

```
\def\NeedsTeXFormat#1{%
  \def\reserved@a{#1}%
  \ifx\reserved@a\fmtname
    \expandafter\@needsformat
  \else
     \@latex@error{This file needs format `\reserved@a'%
       \MessageBreak but this is `\fmtname'}{%
       The current input file will not be processed
       further,\MessageBreak
       because it was written for some other flavor of
       TeX.\MessageBreak\@ehd}%
```

If the file is not meant to be processed by LaTeX 2_ε we stop inputting it, but we do not end the run. We just end inputting the current file.

```
     \endinput \fi}
\@onlypreamble\NeedsTeXFormat
```

```
\def\@needsformat{%
  \@ifnextchar[%]
    \@needsf@rmat
    {}}
\@onlypreamble\@needsformat
```

```
\def\@needsf@rmat[#1]{%
    \@ifl@t@r\fmtversion{#1}{}%
    {\@latex@warning@no@line
        {You have requested release `#1' of LaTeX,\MessageBreak
         but only release `\fmtversion' is available}}}
\@onlypreamble\@needsf@rmat
```

`\zap@space`

`\zap@space foo`⟨*space*⟩`\@empty` removes all spaces from `foo` that are not protected by `{ }` groups.

```
\def\zap@space#1 #2{%
  #1%
  \ifx#2\@empty\else\expandafter\zap@space\fi
  #2}
```

`\@fileswithoptions`

The common part of `\documentclass` and `\usepackage`.

```
\def\@fileswithoptions#1{%
  \@ifnextchar[%]
    {\@fileswith@ptions#1}%
    {\@fileswith@ptions#1[]}}
\@onlypreamble\@fileswithoptions
```

```
\def\@fileswith@ptions#1[#2]#3{%
  \@ifnextchar[%]
  {\@fileswith@pti@ns#1[#2]#3}%
  {\@fileswith@pti@ns#1[#2]#3[]}}
\@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.

```
\def\@fileswith@pti@ns#1[#2]#3[#4]{%
  \ifx#1\@clsextension
    \ifx\@classoptionslist\relax
      \xdef\@classoptionslist{\zap@space#2 \@empty}%
      \def\reserved@a{%
        \@onefilewithoptions#3[#2][#4]#1%
```

```
        \@documentclasshook}%
    \else
      \def\reserved@a{%
        \@onefilewithoptions#3[#2][#4]#1}%
    \fi
  \else
```

build up a list of calls to `\@onefilewithoptions` (one for each package) without thrashing the parameter stack.

```
    \def\reserved@b##1,{%
      \ifx\@nil##1\relax\else
        \ifx\relax##1\relax\else
         \noexpand\@onefilewithoptions##1[#2][#4]\noexpand\@pkgextension
        \fi
        \expandafter\reserved@b
      \fi}%
      \edef\reserved@a{\zap@space#3 \@empty}%
      \edef\reserved@a{\expandafter\reserved@b\reserved@a,\@nil,}%
  \fi
  \reserved@a}
\@onlypreamble\@fileswith@pti@ns
```

Have the main argument as `#1`, so we only need one `\expandafter` above.

```
\def\@onefilewithoptions#1[#2][#3]#4{%
  \@pushfilename
  \xdef\@currname{#1}%
  \global\let\@currext#4%
  \expandafter\let\csname\@currname.\@currext-h@@k\endcsname\@empty
  \let\CurrentOption\@empty
  \@reset@ptions
  \makeatletter
```

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

```
  \def\reserved@a{%
    \@ifl@aded\@currext{#1}%
      {\@if@ptions\@currext{#1}{#2}{}%
        {\@latex@error
            {Option clash for \@cls@pkg\space #1}%
            {The package #1 has already been loaded
             with options:\MessageBreak
             \space\space[\@ptionlist{#1.\@currext}]\MessageBreak
             There has now been an attempt to load it
              with options\MessageBreak
             \space\space[#2]\MessageBreak
             Adding the line:\MessageBreak
             \space\space\string\usepackage
                  [\@ptionlist{#1.\@currext},#2]{#1}\MessageBreak
             to your document may fix this.\MessageBreak
             Try typing \space <return> \space to proceed.}}}%
      {\@pass@ptions\@currext{#2}{#1}%
```

```
      \global\expandafter
      \let\csname ver@\@currname.\@currext\endcsname\@empty
      \InputIfFileExists
        {\@currname.\@currext}%
        {}%
        {\@missingfileerror\@currname\@currext}%
```

`\@unprocessedoptions` will generate an error for each specified option in a package unless a `\ProcessOptions` has appeared in the package file.

```
    \let\@unprocessedoptions\@@unprocessedoptions
    \csname\@currname.\@currext-h@@k\endcsname
    \expandafter\let\csname\@currname.\@currext-h@@k\endcsname
             \@undefined
    \@unprocessedoptions}
    \@ifl@ter\@currext{#1}{#3}{}%
      {\@latex@warning@no@line
         {You have requested,\on@line,
          version\MessageBreak
            `#3' of \@cls@pkg\space #1,\MessageBreak
          but only version\MessageBreak
           `\csname ver@#1.\@currext\endcsname'\MessageBreak
          is available}}%
    \ifx\@currext\@clsextension\let\LoadClass\@twoloadclasserror\fi
    \@popfilename
    \@reset@ptions}%
  \reserved@a}
\@onlypreamble\@onefilewithoptions
```

`\@@fileswith@pti@ns` Save the definition (for error checking).

```
\let\@@fileswith@pti@ns\@fileswith@pti@ns
\@onlypreamble\@@fileswith@pti@ns
```

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

```
\def\@reset@ptions{%
  \global\ifx\@currext\@clsextension
    \let\default@ds\OptionNotUsed
   \else
    \let\default@ds\@unknownoptionerror
  \fi
  \global\let\ds@\@empty
  \global\let\@declaredoptions\@empty}
\@onlypreamble\@reset@ptions
```

67.1 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 \@enddocumenthook

Stuff to appear at the begining or end of the document.

```
\let\@begindocumenthook\@empty
\let\@enddocumenthook\@empty
```

\g@addto@macro

Globally add to the end of a macro.

```
\long\def\g@addto@macro#1#2{{%
  \toks@\expandafter{#1#2}%
  \xdef#1{\the\toks@}}}
```

\AtEndOfPackage \AtEndOfClass \AtBeginDocument \AtEndDocument

The access functions.

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

```
\def\AtBeginDocument{\g@addto@macro\@begindocumenthook}
\def\AtEndDocument{\g@addto@macro\@enddocumenthook}
\@onlypreamble\AtBeginDocument
```

\@cls@pkg

The current file type.

```
\def\@cls@pkg{%
  \ifx\@currext\@clsextension
    document class%
  \else
    package%
  \fi}
\@onlypreamble\@cls@pkg
```

\@unknownoptionerror

Bad option.

```
\def\@unknownoptionerror{%
  \@latex@error
    {Unknown option `\CurrentOption' for \@cls@pkg\space`\@currname'}%
    {The option `\CurrentOption' was not declared in
     \@cls@pkg\space`\@currname', perhaps you\MessageBreak
      misspelled its name.
     Try typing \space <return>
     \space to proceed.}}
\@onlypreamble\@unknownoptionerror
```

\@@unprocessedoptions

Declare an error for each option, unless a `\ProcessOptions` occured.

```
\def\@@unprocessedoptions{%
  \ifx\@currext\@pkgextension
    \edef\@curroptions{\@ptionlist{\@currname.\@currext}}%
    \@for\CurrentOption:=\@curroptions\do\@unknownoptionerror
  \fi}
\@onlypreamble\@unprocessedoptions
\@onlypreamble\@@unprocessedoptions
```

`\@badrequireerror` `\RequirePackage` or `\LoadClass` occurs in the options section.

```
\def\@badrequireerror#1[#2]#3[#4]{%
  \@latex@error
    {\noexpand\RequirePackage or \noexpand\LoadClass
         in Options Section}%
    {The \@cls@pkg\space `\@currname' is defective.\MessageBreak
     It attempts to load `#3' in the options section, i.e.,\MessageBreak
     between \noexpand\DeclareOption and \string\ProcessOptions.}}
\@onlypreamble\@badrequireerror
```

`\@twoloadclasserror` Two `\LoadClass` in a class.

```
\def\@twoloadclasserror{%
  \@latex@error
    {Two \noexpand\LoadClass commands}%
    {You may only use one \noexpand\LoadClass in a class file}}
\@onlypreamble\@twoloadclasserror
```

`\@twoclasseserror` Two `\documentclass` or `\documentstyle`.

```
\def\@twoclasseserror#1#{%
  \@latex@error
    {Two \noexpand\documentclass or \noexpand\documentstyle commands}%
    {The document may only declare one class.}\@gobble}
\@onlypreamble\@twoclasseserror
```

67.2 Providing shipment

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

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

`\filecontents`
`\endfilecontents` This environment implements inline files. The star-form does not write extra comments into the file.

```
\begingroup%
\gdef\filecontents{\@tempswatrue\filec@ntents}%
\catcode`\*=11 %
\gdef\filecontents*{\@tempswafalse\filec@ntents}%
\catcode`\^^M\active%
\catcode`\^^L\active\let^^L\relax%
\catcode`\^^I\active%
\gdef\filec@ntents#1{%
  \openin\@inputcheck#1 %
  \ifeof\@inputcheck%
    \@latex@warning@no@line%
        {Writing file `\@currdir#1'}%
    \chardef\verbatim@out15 %
    \ch@ck7\verbatim@out\write%
  \else%
    \closein\@inputcheck%
    \@latex@warning@no@line%
```

```
            {File `#1' already exists on the system.\MessageBreak%
             Not generating it from this source}%
    \def\immediate##1\relax{}%
  \fi%
  \immediate\openout\verbatim@out=#1\relax%
  \if@tempswa%
    \immediate\write\verbatim@out{%
      \@percentchar\@percentchar\space LaTeX2e file `#1'^^J%
      \@percentchar\@percentchar\space  generated by the %
         `filecontents' environment^^J%
      \@percentchar\@percentchar\space from source `\jobname' on %
         \number\year/\two@digits\month/\two@digits\day.^^J%
      \@percentchar\@percentchar}\relax%
  \fi%
  \let\do\@makeother\dospecials%
  \@vwrend%
\catcode`\^^L\active%
\let\L\@undefined%
\def^^L{\@ifundefined L^^J^^J}%
\catcode`\^^I\active%
\let\I\@undefined%
\def^^I{\@ifundefined I\space\space}%
  \catcode`\^^M\active%
  \def^^M##1^^M%
    {\def\reserved@b{##1}%
     \ifx\reserved@b\reserved@a%
       \expandafter\end\expandafter{\@currenvir}%
     \else%
       \immediate\write\verbatim@out{##1}\relax\expandafter^^M%
     \fi}}%
\endgroup%
\begingroup
\catcode`/=0
\catcode`<=\catcode`{
\catcode`>=\catcode`}
\catcode`|=\catcode`\%
/catcode`/\=12
/catcode`/{=12
/catcode`/}=12
/catcode`/%=12
/catcode`/*=11
/gdef/@percentchar<%>
/gdef/@vwrend</edef/reserved@a<\end{/@currenvir}>>
/gdef/endfilecontents<|
  /immediate/closeout/verbatim@out/relax
  /ifx/L/@undefined/else
    /@latex@warning@no@line<Form Feed has been converted to Blank Line>|
  /fi
  /ifx/I/@undefined/else
    /@latex@warning@no@line<Tab has been converted to Space>|
```

```
  /fi
  /immediate/write/@unused<>/relax>
/global/let/endfilecontents*/endfilecontents|
/@onlypreamble/filecontents
/@onlypreamble/endfilecontents
/@onlypreamble/filecontents*
/@onlypreamble/endfilecontents*
/endgroup
\@onlypreamble\filec@ntents
\@onlypreamble\@vwrend
```

```
⟨/2ekernel⟩
```

68 After Preamble

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

```
⟨*afterpreamble⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{pkgindoc}
         [1994/10/20 v1.1 Package Interface in Document (DPC)]
\def\reserved@a#1\do\@clsextension#2\do\@vwrend#3\relax{%
  \gdef\@preamblecmds{#1#3}}
\expandafter\reserved@a\@preamblecmds\relax
⟨/afterpreamble⟩
```

File M
lthyphen.dtx

This file contains the code for loading hyphenation patterns into LaTeX. Most of this will end up in a file called `hyphen.ltx`. If you wish to customize your LaTeX 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 LaTeX 2_{ε}.

```
⟨*driver⟩
\documentclass{ltxdoc}
\begin{document}
\DocInput{lthyphen.dtx}
\end{document}
⟨/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 TeX'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.

```
⟨*default⟩
\InputIfFileExists{hyphen.tex}%
   {\message{Loading hyphenation patterns for US english.}%
    \language=0
    \lefthyphenmin=2 \righthyphenmin=3 }%
```

Otherwise, since we cannot do anything without any hyphenation patterns, an error message is printed and the IniTeX run is terminated by invoking `\@@end` (which is the LaTeX 2_{ε} name for TeX's `\end` primitive).

```
   {\errhelp{The configuration for hyphenation is incorrectly
             installed.^^J%
             If you don't understand this error message you need
             to seek^^Jexpert advice.}%
    \errmessage{OOPS! I can't find any hyphenation patterns for
                US english.^^J \space Think of getting some or the
                latex2e setup will never succeed}\@@end}
⟨/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-us
\language=1
\input ghyphen3
\language=0
\lefthyphenmin=2
\righthyphenmin=3
\endinput
```

Another possibility is to use the package `babel`, by Johannes Braams. In that case, the file `babel.hyphen` can be used as `hyphen.cfg`; by renaming it or (if your operating system supports such an operation) by creating a (symbolic) link.

File N
ltfinal.dtx

69 Final settings

This section contains the final settings for LaTeX. It initialises some debugging and typesetting parameters, sets the default `\catcode`s and uc/lc codes, and inputs the hyphenation file.

69.1 Debugging

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

```
⟨*2ekernel⟩
\def\showoverfull{\tracingonline\@ne}
\def\showoutput{\tracingonline\@ne\tracingoutput\@ne
    \showboxbreadth\maxdimen\showboxdepth\maxdimen\errorstopmode}
```

By default, LaTeX shows statistics:

```
\tracingstats1
```

Empty the error help message, which may have some rubbish:

```
\errhelp{}
```

69.2 Typesetting parameters

`\@lowpenalty` `\@medpenalty` `\@highpenalty` These are penalties used internally.

```
\newcount\@lowpenalty
\newcount\@medpenalty
\newcount\@highpenalty
```

The default values of the picture and `\fbox` parameters:

```
\unitlength = 1pt
\fboxsep = 3pt
\fboxrule = .4pt
```

The saved value of TeX's `\maxdepth`:

```
\@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}`

```
\vsize = 1000pt
\@colroom = \vsize
\@colht = \vsize
```

Initialise `\textheight` `\textwidth` and page style, to avoid internal errors if they are not set by the class.

```
\textheight=.5\maxdimen
\textwidth=\textheight
\ps@empty
```

69.3 Hyphenation

The following code will be compiled into the format file. It checks for the existance of `hyphen.cfg` in inputs that file if found. Otherwise it inputs `hyphen.ltx`. Note that these are loaded in *before* the `\catcode`s are set, so local hyphenation files can use 8-bit input.

We try to load the customized hyphenation description file.

```
\InputIfFileExists{hyphen.cfg}
            {\typeout{*******************************************^^J%
                      * Local configuration file hyphen.cfg used^^J%
                      *******************************************}}
            {\input{hyphen.ltx}}
```

69.4 Font loading

Fonts loaded during the formatting process might already have changed the `\font@submax` from `0pt` to something higher. If so be put out a bold warning.

```
\ifdim \font@submax >\z@
   \@font@warning{Size substitutions with differences\MessageBreak
                  up to \font@submax\space have occured.\MessageBreak
                 \MessageBreak
                 Please check the transcript file
                 carefully\MessageBreak
                 and redo the format generation if necessary!
                 \@gobbletwo}%
   \errhelp{Only stopped, to give you time to
            read the above message.}
   \errmessage{}
```

We reset the macro. Otherwise every user will get a warning on every job.

```
\def\font@submax{0pt}
\fi
```

69.5 Input encoding

We temporarily define `\reserved@a` to apply `\reserved@c` to all the numbers in the range of its arguments.

```
\def\reserved@a#1#2{%
   \@tempcnta#1\relax
   \@tempcntb#2\relax
   \reserved@b
}
```

```
\def\reserved@b{%
   \ifnum\@tempcnta>\@tempcntb\else
      \reserved@c\@tempcnta
      \advance\@tempcnta\@ne
      \expandafter\reserved@b
   \fi
}
```

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.

```
\catcode`\ =10
\catcode`\#=6
\catcode`\$=3
\catcode`\%=14
\catcode`\&=4
\catcode`\\=0
\catcode`\^=7
\catcode`\_=8
\catcode`\{=1
\catcode`\}=2
\catcode`\~=13
\catcode`\@=11
\catcode`\^^I=10
\catcode`\^^J=12
\catcode`\^^L=13
\catcode`\^^M=5
```

Set the ‘other’ catcodes.

```
\def\reserved@c#1{\catcode#1=12\relax}
\reserved@c{`\!}
\reserved@c{`\"}
\reserved@a{`\'}{`\?}
\reserved@c{`\[}
\reserved@c{`\]}
\reserved@c{`\`}
\reserved@c{`\|}
```

Set the ‘letter’ catcodes.

```
\def\reserved@c#1{\catcode#1=11\relax}
\reserved@a{`\A}{`\Z}
\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.

```
\def\reserved@c#1{\catcode#1=15\relax}
\reserved@a{0}{`\^^H}
\reserved@c{`\^^K}
\reserved@a{`\^^N}{31}
```

```
%\ifnum\inputlineno=\m@ne
  \catcode"7F=15
%\else
%  \reserved@a{"7F}{"FF}
%\fi
```

The lower case characters need their `\uccode` and `\lccode` values set. Depending on the TeX version, we might not be allowed to do this for non-ASCII characters.

```
% \changes{v1.0n}{1994/06/09}{For \TeX2, do not set codes for higher
%                    half of character table.}
\def\reserved@c#1{%
   \count@=#1\advance\count@ by -"20
   \uccode#1=\count@
   \lccode#1=#1
}
\reserved@a{`\a}{`\z}
\ifnum\inputlineno=\m@ne\else
  \reserved@a{"A0}{"BC}
  \reserved@a{"E0}{"FF}
\fi
```

The upper case characters need their `\uccode` and `\lccode` values set, and their `\sfcode` set to 999.

```
\def\reserved@c#1{%
   \count@=#1\advance\count@ by "20
   \uccode#1=#1
   \lccode#1=\count@
   \sfcode#1=999
}
\reserved@a{`\A}{`\Z}
\ifnum\inputlineno=\m@ne\else
  \reserved@a{"80}{"9C}
  \reserved@a{"C0}{"DF}
\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.

```
\uccode`\^^Y=`\I     % dotless i
\lccode`\^^Y=`\^^Y   % dotless i
\uccode`\^^Z=`\J     % dotless j, ae in OT1
\lccode`\^^Z=`\^^Z   % dotless j, ae in OT1
\ifnum\inputlineno=\m@ne\else
  \lccode`\^^9d=`\i    % dotted I
  \uccode`\^^9d=`\^^9d % dotted I
  \lccode`\^^9e=`\^^9e % d-bar
  \uccode`\^^9e=`\^^d0 % d-bar
\fi
```

And just to make sure nobody relies on those definitions of `\reserved@a` and friends:

```
\let\reserved@a=\@undefined
```

```
\let\reserved@b=\@undefined
\let\reserved@c=\@undefined
```

\MakeUppercase
\MakeUppercase
\@uclclist

And whilst we're doing things with uc/lc tables, here are two commands to upper- and lower-case a string.

Note that this implementation is subject to change! At the moment we're not providing any way to extend the list of uc/lc commands, since finding a good interface is difficult. These commands have some nasty features, such as uppercasing mathematics, environment names, labels, etc. A much better long-term solution is to use all-caps fonts, but these aren't generally available.

```
\DeclareRobustCommand{\MakeUppercase}[1]{{%
      \def\i{I}\def\j{J}%
      \def\reserved@a##1##2{\let##1##2\reserved@a}%
      \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%
      \protected@edef\reserved@a{\uppercase{#1}}%
      \reserved@a
   }}
\DeclareRobustCommand{\MakeLowercase}[1]{{%
      \def\reserved@a##1##2{\let##2##1\reserved@a}%
      \expandafter\reserved@a\@uclclist\reserved@b{\reserved@b\@gobble}%
      \protected@edef\reserved@a{\lowercase{#1}}%
      \reserved@a
   }}
\def\@uclclist{\oe\OE\o\O\ae\AE\aa\AA
      \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 uppercasing is only done to the first letter of the contents name, since the mark expands out to:

```
\mark{\protect\MakeUppercase Table of Contents}
     {\protect\MakeUppercase Table of Contents}
```

In order to get round this, we redefine `\MakeUppercase` and `\MakeLowercase` to grab their argument and brace it. This is a very low-level hack, and is *not* recommended practice! This is an instance of a general problem that makes it unsafe to grab arguments unbraced, and probably needs a more general solution. For the moment though, this hack will do:

```
\protected@edef\MakeUppercase#1{\MakeUppercase{#1}}
\protected@edef\MakeLowercase#1{\MakeLowercase{#1}}
```

69.6 Applying Patch files

Between major releases, small patches will be distributed in files `ltpatch.ltx` which must be added at this point.

```
\IfFileExists{ltpatch.ltx}
  {\typeout{***********************************^^J%
            * Applying patch file ltpatch.ltx *^^J%
            ***********************************}
   \def\fmtversion@topatch{unknown}
   \input{ltpatch.ltx}
   \ifx\fmtversion\fmtversion@topatch
      \ifx\patch@level\@undefined
        \typeout{^^J^^J^^J%
         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
         !! Patch file `ltpatch.ltx' not suitable for this^^J%
         !! version of LaTeX.^^J^^J%
         !! Please check if initex found an old patch file:^^J%
         !! --- if so, rename it or delete it, and redo the^^J%
         !! initex run.^^J%
         !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
        \batchmode \@@end
      \else
```

The code below adds the 'patch level' string to the first `\typeout` in the startup banner.

```
        \def\fmtversion@topatch{0}%
        \ifx\fmtversion@topatch\patch@level\else
          \def\reserved@a\typeout##1##2\reserved@a{%
                 \typeout{##1 patch level \patch@level}##2}
          \everyjob\expandafter\expandafter\expandafter{%
             \expandafter\reserved@a\the\everyjob\reserved@a}
          \let\reserved@a\relax
          \the\everyjob
        \fi
      \fi
   \else
      \typeout{^^J^^J^^J%
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J%
     !! Patch file `ltpatch.ltx' (for version <\fmtversion@topatch>)^^J%
     !! is not suitable for version <\fmtversion> of LaTeX.^^J^^J%
     !! Please check if initex found an old patch file:^^J%
     !! --- if so, rename it or delete it, and redo the^^J%
     !!     initex run.^^J%
     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!^^J}%
       \batchmode \@@end
   \fi
   \let\fmtversion@topatch\relax
  }{}
```

69.7 Initialise file list

Reset **`\@filelist`** so files input while making the format are not listed.

```
\let\@filelist\@gobble
```

69.8 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.

```
\makeatother
\errorstopmode
\dump
⟨/2ekernel⟩
```

File O
ltpatch

Things we did wrong...

```
% \iffalse meta-comment
%
% Copyright 1993-1995 the LaTeX3 project and any individual authors
% listed elsewhere in this file.  All rights reserved.
%
% For further copyright information see the file legal.txt, and any
% other copyright notices in this file.
%
% This file is part of the LaTeX2e system.
% ----------------------------------------
%   This system is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
%
%   For error reports concerning UNCHANGED versions of this file no more
%   than one year old, see bugs.txt.
%
%   Please do not request updates from us directly.  Primary
%   distribution is through the CTAN archives.
%
%
% IMPORTANT COPYRIGHT NOTICE:
%
% You are NOT ALLOWED to distribute this file alone.
%
% You are allowed to distribute this file under the condition that it is
% distributed together with all the files listed in manifest.txt.
%
% If you receive only some of these files from someone, complain!
%
% Permission is granted to copy this file to another file with a clearly
% different name and to customize the declarations in that copy to serve
% the needs of your installation, provided that you comply with
% the conditions in the file legal.txt.
%
% However, NO PERMISSION is granted to produce or to distribute a
% modified version of this file under its original name.
%
% You are NOT ALLOWED to change this file.
%
%
% \fi

%%%%
```

```
%%%% Patch file for the LaTeX2e kernel dated 1995/06/01
%%%% (1995/06/14)

\def\fmtversion@topatch{1995/06/01} % This patch will not work with
                                    % any other release.

\def\patch@level{0}


%%%%%%%%%%%%%%%

\endinput
```