Standard Document Classes for \LaTeX{} version 2e*

Copyright (C) 1992 by Leslie Lamport
Copyright (C) 1994-2021 by Frank Mittelbach, Johannes Braams and the \LaTeX{} Project Team

2021/10/04

This file is maintained by the \LaTeX{} Project team. Bug reports can be opened (category \texttt{latex}) at \url{https://latex-project.org/bugs.html}.

Contents

1 The \texttt{docstrip} modules ........................................ 3

2 Initial Code .................................................................. 3

3 Declaration of Options ................................................. 4
  3.1 Setting Paper Sizes .............................................. 4
  3.2 Choosing the type size .......................................... 4
  3.3 Two-side or one-side printing ................................. 5
  3.4 Draft option ....................................................... 5
  3.5 Titlepage option .................................................. 5
  3.6 openright option .................................................. 5
  3.7 Two-column printing ............................................ 5
  3.8 Equation numbering on the left ............................... 5
  3.9 Flush left displays .............................................. 6
  3.10 Open bibliography ............................................... 6

4 Executing Options ..................................................... 6

5 Loading Packages ..................................................... 7

6 Document Layout ..................................................... 7
  6.1 Fonts .......................................................... 7
  6.2 Paragrphing ..................................................... 10
  6.3 Page Layout ..................................................... 11
    6.3.1 Vertical spacing ......................................... 11
    6.3.2 The dimension of text .................................... 12
    6.3.3 Margins ..................................................... 14

*This file has version number v1.4n, last revised 2021/10/04.
6.3.4 Footnotes .................................................. 16
6.3.5 Float placement parameters .............................. 17
6.4 Page Styles .................................................. 19
  6.4.1 Marking conventions .................................. 20
  6.4.2 Defining the page styles .............................. 20

7 Document Markup ............................................ 22
  7.1 The title .................................................. 22
  7.2 Chapters and Sections .................................... 25
    7.2.1 Building blocks ..................................... 25
    7.2.2 Mark commands ..................................... 26
    7.2.3 Define Counters ..................................... 26
    7.2.4 Front Matter, Main Matter, and Back Matter ........ 27
    7.2.5 Parts .................................................. 28
    7.2.6 Chapters ............................................. 31
    7.2.7 Lower level headings ............................... 33
  7.3 Lists ..................................................... 34
    7.3.1 General List Parameters ............................ 34
    7.3.2 Enumerate ........................................... 36
    7.3.3 Itemize ............................................. 36
    7.3.4 Description .......................................... 37
  7.4 Defining new environments ............................... 37
    7.4.1 Abstract ............................................ 37
    7.4.2 Verse ............................................... 38
    7.4.3 Quotation ........................................... 38
    7.4.4 Quote ............................................... 38
    7.4.5 Theorem ............................................. 39
    7.4.6 Titlepage ........................................... 39
    7.4.7 Appendix ............................................ 40
  7.5 Setting parameters for existing environments ........... 40
    7.5.1 Array and tabular .................................. 40
    7.5.2 Tabbing .............................................. 41
    7.5.3 Minipage ............................................ 41
    7.5.4 Framed boxes ........................................ 41
    7.5.5 Equation and eqnarray .............................. 41
  7.6 Floating objects ......................................... 42
    7.6.1 Figure .............................................. 42
    7.6.2 Table ............................................... 43
    7.6.3 Captions ............................................ 43
  7.7 Font changing ............................................ 44

8 Cross Referencing .......................................... 45
  8.1 Table of Contents, etc. .................................. 45
    8.1.1 Table of Contents .................................. 46
    8.1.2 List of figures ...................................... 49
    8.1.3 List of tables ....................................... 49
  8.2 Bibliography ............................................. 50
  8.3 The index ................................................ 51
  8.4 Footnotes ................................................ 52
9 Initialization

1 The docstrip modules

The following modules are used in the implementation to direct docstrip in generating the external files:

- `article` produce the documentclass article
- `report` produce the documentclass report
- `size10` produce the class option for 10pt
- `size11` produce the class option for 11pt
- `size12` produce the class option for 12pt
- `book` produce the documentclass book
- `bk10` produce the book class option for 10pt
- `bk11` produce the book class option for 11pt
- `bk12` produce the book class option for 12pt
- `driver` produce a documentation driver file

2 Initial Code

In this part we define a few commands that are used later on.

\@ptsize
This control sequence is used to store the second digit of the pointsize we are typesetting in. So, normally, it’s value is one of 0, 1 or 2.

1 (`article` | `report` | `book`)
2 `newcommand\@ptsize{}`

\if@restonecol
When the document has to be printed in two columns, we sometimes have to temporarily switch to one column. This switch is used to remember to switch back.

3 `newif\if@restonecol`

\if@titlepage
A switch to indicate if a titlepage has to be produced. For the article document class the default is not to make a separate titlepage.

4 `newif\if@titlepage`
5 `{article}\@titlepagefalse`
6 `{article}\@titlepagetrue`

\if@openright
A switch to indicate if chapters must start on a right-hand page. The default for the report class is no; for the book class it’s yes.

7 `{article}\newif\if@openright`

\if@mainmatter
The switch `\if@mainmatter`, only available in the document class book, indicates whether we are processing the main material in the book.

8 `{book}\newif\if@mainmatter \@mainmattertrue`
3 Declaration of Options

3.1 Setting Paper Sizes

The variables \paperwidth and \paperheight should reflect the physical paper size after trimming. For desk printer output this is usually the real paper size since there is no post-processing. Classes for real book production will probably add other paper sizes and additionally the production of crop marks for trimming. In compatibility mode, these (and some of the subsequent) options are disabled, as they were not present in \LaTeX2.09.

\if@compatibility\else
\DeclareOption{a4paper}{
\setlength\paperheight {297mm}\
\setlength\paperwidth {210mm}}
\DeclareOption{a5paper}{
\setlength\paperheight {210mm}\
\setlength\paperwidth {148mm}}
\DeclareOption{b5paper}{
\setlength\paperheight {250mm}\
\setlength\paperwidth {176mm}}
\DeclareOption{letterpaper}{
\setlength\paperheight {11in}\
\setlength\paperwidth {8.5in}}
\DeclareOption{legalpaper}{
\setlength\paperheight {14in}\
\setlength\paperwidth {8.5in}}
\DeclareOption{executivepaper}{
\setlength\paperheight {10.5in}\
\setlength\paperwidth {7.25in}}
\fi

The option landscape switches the values of \paperheight and \paperwidth, assuming the dimensions were given for portrait paper.

\if@compatibility
\renewcommand\@ptsize{0}
\else
\DeclareOption{10pt}{\renewcommand\@ptsize{0}}
\DeclareOption{11pt}{\renewcommand\@ptsize{1}}
\DeclareOption{12pt}{\renewcommand\@ptsize{2}}
\fi

3.2 Choosing the type size

The type size options are handled by defining \@ptsize to contain the last digit of the size in question and branching on ifcase statements. This is done for historical reasons to stay compatible with other packages that use the \@ptsize variable to select special actions. It makes the declarations of size options less than 10pt difficult, although one can probably use 9 and 8 assuming that a class won’t define both 8pt and 10pt options.

\if@compatibility
\renewcommand\@ptsize{0}
\else
\DeclareOption{10pt}{\renewcommand\@ptsize{0}}
\fi
\DeclareOption{11pt}{\renewcommand\@ptsize{1}}
\DeclareOption{12pt}{\renewcommand\@ptsize{2}}
3.3 Two-side or one-side printing
For two-sided printing we use the switch `\if@twoside`. In addition we have to set the `\if@mparswitch` to get any margin paragraphs into the outside margin.

```
\if@compatibility\else
\DeclareOption{oneside}{\@twosidetrue \@mparswitchfalse}
\fi
\DeclareOption{twoside}{\@twosidetrue \@mparswitchtrue}
```

3.4 Draft option
If the user requests `draft` we show any overfull boxes. We could probably add some more interesting stuff to this option.

```
\DeclareOption{draft}{\setlength\overfullrule{5pt}}
\if@compatibility\else
\DeclareOption{final}{\setlength\overfullrule{0pt}}
\fi
```

3.5 Titlepage option
An article usually has no separate titlepage, but the user can request one.

```
\DeclareOption{titlepage}{\@titlepagetrue}
\if@compatibility\else
\DeclareOption{notitlepage}{\@titlepagefalse}
\fi
```

3.6 Openright option
This option determines whether or not a chapter must start on a right-hand page.

```
⟨\if\@article\if@compatibility\else\fi\fi\if\@book\openrighttrue\else\fi\fi
\DeclareOption{openright}{\@openrighttrue}
\DeclareOption{openany}{\@openrightfalse}
\fi
```

3.7 Two-column printing
Two-column and one-column printing is again realized via a switch.

```
\if@compatibility\else
\DeclareOption{onecolumn}{\@twocolumnfalse}
\fi
\DeclareOption{twocolumn}{\@twocolumntrue}
```

3.8 Equation numbering on the left
The option `leqno` can be used to get the equation numbers on the left side of the equation. It loads code which is generated automatically from the kernel files when the format is built. If the equation number does get a special formatting
then instead of using the kernel file the class would need to provide the code explicitly.

62 \DeclareOption{leqno}{\input{leqno.clo}}}

3.9 Flush left displays
The option fleqn redefines the displayed math environments in such a way that they come out flush left, with an indentation of \texttt{mathindent} from the prevailing left margin. It loads code which is generated automatically from the kernel files when the format is built.

63 \DeclareOption{fleqn}{\input{fleqn.clo}}}

3.10 Open bibliography
The option openbib produces the “open” bibliography style, in which each block starts on a new line, and succeeding lines in a block are indented by \texttt{bibindent}.

64 \DeclareOption{openbib}{%
First some hook into the bibliography environment is filled.
65 \AtEndOfPackage{%
66 \renewcommand\@openbib@code{%
67 \advance\leftmargin\bibindent
68 \itemindent -\bibindent
69 \listparindent \itemindent
70 \parsep \z@
71 }}%
In addition the definition of \texttt{newblock} is overwritten.
72 \renewcommand\newblock{\par}}%
73 }

4 Executing Options
Here we execute the default options to initialize certain variables. Note that the document class ‘book’ always uses two sided printing.

74 \begin{verbatim}
75 \ExecuteOptions{letterpaper,10pt,oneside,onecolumn,final}
76 \end{verbatim}
77 \ExecuteOptions{letterpaper,10pt,oneside,onecolumn,final,openany}
78 \ExecuteOptions{letterpaper,10pt,twoside,onecolumn,final,openright}
79 \begin{verbatim}
80 \ProcessOptions
81 \begin{verbatim}
82 \ProcessOptions

The \texttt{ProcessOptions} command causes the execution of the code for every option \texttt{FOO} which is declared and for which the user typed the \texttt{FOO} option in his \texttt{documentclass} command. For every option \texttt{BAR} he typed, which is not declared, the option is assumed to be a global option. All options will be passed as document options to any \texttt{usepackage} command in the document preamble.

83 \ProcessOptions
Now that all the options have been executed we can load the chosen class option file that contains all size dependent code.

\begin{itemize}
\item \texttt{\input{size1@ptsize.clo}}
\item \texttt{\input{bk1@ptsize.clo}}
\end{itemize}

\section{Loading Packages}

The standard class files do not load additional packages.

\section{Document Layout}

In this section we are finally dealing with the nasty typographical details.

\subsection{Fonts}

\LaTeX{} offers the user commands to change the size of the font, relative to the ‘main’ size. Each relative size changing command \texttt{\size} executes the command \texttt{\@setfontsize\size\langle font-size\rangle\langle baselineskip\rangle} where:

\begin{itemize}
\item \langle font-size\rangle The absolute size of the font to use from now on.
\item \langle baselineskip\rangle The normal value of \texttt{\baselineskip} for the size of the font selected. (The actual value will be \texttt{\baselinestretch * \langle baselineskip\rangle}.)
\end{itemize}

A number of commands, defined in the \LaTeX{} kernel, shorten the following definitions and are used throughout. They are:

\begin{itemize}
\item \texttt{\@vpt 5 \@vipt 6 \@viipt 7}
\item \texttt{\@viiipt 8 \@ixpt 9 \@xpt 10}
\item \texttt{\@xiipt 10.95 \@xiipt 12 \@xivpt 14.4}
\end{itemize}

\begin{verbatim}
\renewcommand\normalsize{\@setfontsize\normalsize\@xpt\@xiipt
  \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
  \abovedisplayshortskip \z@ \@plus3\p@
  \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@}
\end{verbatim}

The user level command for the main size is \texttt{\normalsize}. Internally \LaTeX{} uses \texttt{\@normalsize} when it refers to the main size. \texttt{\@normalsize} will be defined to work like \texttt{\normalsize} if the latter is redefined from its default definition (that just issues an error message). Otherwise \texttt{\@normalsize} simply selects a 10pt/12pt size.

The \texttt{\normalsize} macro also sets new values for \texttt{\abovedisplayskip}, \texttt{\abovedisplayshortskip} and \texttt{\belowdisplayshortskip}.

\begin{verbatim}
\renewcommand\normalsize{%
\@setfontsize\normalsize\@xpt\@xiipt
\abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
\abovedisplayshortskip \z@ \@plus3\p@
\belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@}
\end{verbatim}
The \belowdisplayskip is always equal to the \abovedisplayskip. The parameters of the first level list are always given by \@listI.

We initially choose the \normalsize font.

We use \MakeRobust instead of \DeclareRobustCommand above to avoid a log entry for the redefinition. But if we are running in a rollback situation (prior to 2015) we don’t touch it.

This is similar to \normalsize.
This is similar to \normalsize.

\footnotesize
\DeclareRobustCommand\footnotesize{%
\@setfontsize\footnotesize\@viiipt{9.5}\
\belowdisplayskip \abovedisplayskip\}

\scriptsize
\tiny
\large
\Large
\huge
\Huge

These are all much simpler than the previous macros, they just select a new
fontsize, but leave the parameters for displays and lists alone.
6.2 Paragraphing

These parameters control \TeX's behaviour when two lines tend to come too close together.

This is used as a multiplier for \lineskip. The default is to not stretch the baselines. Note that if this command doesn’t resolve to “empty” any plus or minus part in the specification of \lineskip is ignored.

\parskip gives extra vertical space between paragraphs and \parindent is the width of the paragraph indentation. The value of \parindent depends on whether we are in two-column mode.

The values for these three parameters are set in the \LaTeX{} kernel. They should perhaps vary, according to the size option specified. But as they have always had the same value regardless of the size option we do not change them to stay compatible with both \LaTeX{} 2.09 and older releases of \LaTeX{} 2e.
The commands \nopagebreak and \nolinebreak put in penalties to discourage these breaks at the point they are put in. They use \@lowpenalty, \@medpenalty or \@highpenalty, dependent on their argument.

\clubpenalty These penalties are used to discourage club and widow lines. Because we use their default values we only show them here, commented out.
\widowpenalty
\displaywidowpenalty Discourage (but not so much) widows in front of a math display and forbid breaking directly in front of a display. Allow break after a display without a penalty. Again the default values are used, therefore we only show them here.
\predisplaypenalty
\postdisplaypenalty
\interlinepenalty Allow the breaking of a page in the middle of a paragraph.
\brokenpenalty We allow the breaking of a page after a hyphenated line.

6.3 Page Layout
All margin dimensions are measured from a point one inch from the top and lefthand side of the page.

6.3.1 Vertical spacing
The \headheight is the height of the box that will contain the running head. The \headsep is the distance between the bottom of the running head and the top of the text. The \topskip is the \baselineskip for the first line on a page; \LaTeX{}’s output routine will not work properly if it has the value 0pt, so do not do that!

\setlength\headheight{12\p@}
\setlength\headsep {25\p@}
\setlength\headsep {.25in}
\setlength\headsep {.275in}
\setlength\topskip {10\p@}
\setlength\topskip {11\p@}
\setlength\topskip {12\p@}
\setlength\topskip {10\p@}
\setlength\topskip {11\p@}
\setlength\topskip {12\p@}
\footskip \footnotesize The distance from the baseline of the box which contains the running footer to the baseline of last line of text is controlled by the \footskip.

\maxdepth \footnotesize The \TeX{} primitive register \maxdepth has a function that is similar to that of \topskip. The register \@maxdepth should always contain a copy of \maxdepth. This is achieved by setting it internally at \begin{document}. In both plain \TeX{} and L\TeX{} 2.09 \maxdepth had a fixed value of 4pt; in native L\TeX{}2e mode we let the value depend on the typesize. We set it so that \maxdepth + \topskip = typesize \times 1.5. As it happens, in these classes \topskip is equal to the typesize, therefore we set \maxdepth to half the value of \topskip.

6.3.2 The dimension of text

\textwidth \footnotesize When we are in compatibility mode we have to make sure that the dimensions of the printed area are not different from what the user was used to see.

\if@compatibility \setlength\textwidth{4\p@} \else \setlength\textwidth{.5\topskip} \fi

When we are not in compatibility mode we can set some of the dimensions differently, taking into account the paper size for instance.

\else \setlength\@tempdima{\paperwidth} \addtolength\@tempdima{-2in} \setlength\@tempdimb{345\p@} \setlength\textwidth{\@tempdimb} \fi

Now we can set the \textwidth, depending on whether we will be setting one or two columns.

In two-column mode each column shouldn’t be wider than \@tempdimb (which could happen on A3 paper for instance).
\setlength{textwidth}{2\@tempdimb}
\else
\setlength{textwidth}{\@tempdima}
\fi

In one-column mode the text should not be wider than the minimum of the paperwidth (minus 2 inches for the margins) and the maximum length of a line as defined by the number of characters.
\else
\ifdim\@tempdima>\@tempdimb\relax
\setlength{textwidth}{\@tempdimb}
\else
\setlength{textwidth}{\@tempdima}
\fi
\fi

Here we modify the width of the text a little to be a whole number of points.
\if@compatibility\else
\@settopoint{textwidth}
\fi

Now that we have computed the width of the text, we have to take care of the height. The \textheight is the height of text (including footnotes and figures, excluding running head and foot).

First make sure that the compatibility mode gets the same dimensions as we had with \LaTeX\texttt{2.09}. The number of lines was calculated as the floor of the old \textheight minus \topskip, divided by \baselineskip for \normalsize. The old value of \textheight was 528pt.

\if@compatibility
\langle 10\text{pt} &! bk \rangle \setlength{\textheight}{43\baselineskip}
\langle 10\text{pt} & bk \rangle \setlength{\textheight}{41\baselineskip}
\langle 11\text{pt} \rangle \setlength{\textheight}{38\baselineskip}
\langle 12\text{pt} \rangle \setlength{\textheight}{36\baselineskip}
\else
\setlength{\tempdima}{\paperheight}
\addtolength{\tempdima}{-2in}
\addtolength{\tempdima}{-1.5in}
\divide{\tempdima}{\baselineskip}
\@tempcnta=\tempdima
\fi

Again we compute this, depending on the papersize and depending on the baselineskip that is used, in order to have a whole number of lines on the page.

\setlength{\tempdima}{\paperheight}

We leave at least a 1 inch margin on the top and the bottom of the page.

\addtolength{\tempdima}{-2in}

We also have to leave room for the running headers and footers.

\addtolength{\tempdima}{-1.5in}

Then we divide the result by the current \baselineskip and store this in the count register \@tempcnta, which then contains the number of lines that fit on this page.

\divide{\tempdima}{\baselineskip}
\@tempcnta=\tempdima

From this we can calculate the height of the text.

\setlength{\textheight}{\@tempcnta\baselineskip}
\fi
6.3.3 Margins

Most of the values of these parameters are now calculated, based on the papersize in use. In the calculations the \marginparsep needs to be taken into account so we give it its value first.

The horizontal space between the main text and marginal notes is determined by \marginparsep, the minimum vertical separation between two marginal notes is controlled by \marginparpush.

Now we can give the values for the other margin parameters. For native \TeX\ 2\epsilon, these are calculated.

First we give the values for the compatibility mode.

Values for two-sided printing:

Values for one-sided printing:

The first line on the page has a height of \topskip.
And values for two-column mode:
\if@twocolumn
\setlength\oddsidemargin {30\p@}
\setlength\evensidemargin {30\p@}
\setlength\marginparwidth {48\p@}
\fi

When we are not in compatibility mode we can take the dimensions of the selected paper into account.

The values for \oddsidemargin and \marginparwidth will be set depending on the status of the \if@twoside.

If \@twoside is true (which is always the case for book) we make the inner margin smaller than the outer one.
\else
\if@twoside
\setlength\@tempdima {\paperwidth}
\addtolength\@tempdima {+\textwidth}
\setlength\oddsidemargin {+.4\@tempdima}
\addtolength\oddsidemargin {+1in}
\fi
\setlength\marginparwidth {+.6\@tempdima}
\addtolength\marginparwidth {\marginparsep}
\addtolength\marginparwidth {-.4in}
\fi

The width of the margin for text is set to the remainder of the width except for a ‘real margin’ of white space of width 0.4in. A check should perhaps be built in to ensure that the (text) margin width does not get too small!
\setlength\marginparwidth {+.6\@tempdima}
\addtolength\marginparwidth {\marginparsep}
\addtolength\marginparwidth {-.4in}

For one-sided printing we center the text on the page, by calculating the difference between \textwidth and \paperwidth. Half of that difference is than used for the margin (thus \oddsidemargin is 1in less).
\else
\setlength\@tempdima {\paperwidth}
\addtolength\@tempdima {+\textwidth}
\setlength\oddsidemargin {+.5\@tempdima}
\addtolength\oddsidemargin {+1in}
\setlength\marginparwidth {+.5\@tempdima}
\addtolength\marginparwidth {\marginparsep}
\addtolength\marginparwidth {-.4in}
\addtolength\marginparwidth {-.4in}
\fi

With the above algorithm the \marginparwidth can come out quite large which we may not want.
Having done these calculations we make them pt values.

The \evensidemargin can now be computed from the values set above.

Setting \evensidemargin to a full point value may produce a small error. However it will lie within the error range a doublesided printer of today’s technology can accurately print.

The \topmargin is the distance between the top of ‘the printable area’—which is 1 inch below the top of the paper—and the top of the box which contains the running head.

It can now be computed from the values set above.

By changing the factor in the next line the complete page can be shifted vertically.

Footnotes

\footnotepsep is the height of the strut placed at the beginning of every footnote. It equals the height of a normal \footnotesize strut in this class, thus no extra space occurs between footnotes.

\footins \skip\footins is the space between the last line of the main text and the top of the first footnote.
6.3.5 Float placement parameters

All float parameters are given default values in the \LaTeX\ kernel. For this reason parameters that are not counters need to be set with \texttt{\renewcommand}.

Limits for the placement of floating objects

\c@topnumber

The \texttt{topnumber} counter holds the maximum number of floats that can appear on the top of a text page.

```
\setcounter{topnumber}{2}
```

\topfraction

This indicates the maximum part of a text page that can be occupied by floats at the top.

```
\renewcommand\topfraction{.7}
```

\c@bottomnumber

The \texttt{bottomnumber} counter holds the maximum number of floats that can appear on the bottom of a text page.

```
\setcounter{bottomnumber}{1}
```

\bottomfraction

This indicates the maximum part of a text page that can be occupied by floats at the bottom.

```
\renewcommand\bottomfraction{.3}
```

\c@totalnumber

This indicates the maximum number of floats that can appear on any text page.

```
\setcounter{totalnumber}{3}
```

\textfraction

This indicates the minimum part of a text page that has to be occupied by text.

```
\renewcommand\textfraction{.2}
```

\floatpagefraction

This indicates the minimum part of a page that has to be occupied by floating objects before a ‘float page’ is produced.

```
\renewcommand\floatpagefraction{.5}
```

\c@dbltopnumber

The \texttt{dbltopnumber} counter holds the maximum number of two-column floats that can appear on the top of a two-column text page.

```
\setcounter{dbltopnumber}{2}
```

\dbltopfraction

This indicates the maximum part of a two-column text page that can be occupied by two-column floats at the top.

```
\renewcommand\dbltopfraction{.7}
```

\dblfloatpagefraction

This indicates the minimum part of a page that has to be occupied by two-column wide floating objects before a ‘float page’ is produced.

```
\renewcommand\dblfloatpagefraction{.5}
```

\(</article | report | book)\)
Floats on a text page

\floatsep When a floating object is placed on a page with text, these parameters control the separation between the float and the other objects on the page. These parameters are used for both one-column mode and single-column floats in two-column mode.

\textfloatsep is the space between adjacent floats that are moved to the top or bottom of the text page.

\intextsep is the space between in-text floats and the text.

\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 floating objects that span the whole \textwidth are placed on a text page when we are in two-column mode the separation between the float and the text is controlled by \dblfloatsep and \dbltextfloatsep.

\dblfloatsep is the space between adjacent floats that are moved to the top or bottom of the text page.

\dbltextfloatsep is the space between the main text and floats at the top or bottom of the page.

\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 separate pages the layout of such pages is controlled by these parameters. At the top of the page \@fptop amount of stretchable whitespace is inserted, at the bottom of the page we get an \@fpbot amount of stretchable whitespace. Between adjacent floats the \@fpsep is inserted.
These parameters are used for the placement of floating objects in one-column mode, or in single-column floats in two-column mode.

Note that at least one of the two parameters \fptop and \fpbot should contain a \plus \fil to allow filling the remaining empty space.

\begin{verbatim}
\setlength\fptop{0\p@ \plus 1fil}
\setlength\fpsep{8\p@ \plus 2fil}
\setlength\fpbot{0\p@ \plus 1fil}
\end{verbatim}

Double-column floats in two-column mode are handled with similar parameters.

\begin{verbatim}
\setlength\dblfptop{0\p@ \plus 1fil}
\setlength\dblfpsep{8\p@ \plus 2fil}
\setlength\dblfpbot{0\p@ \plus 1fil}
\end{verbatim}

6.4 Page Styles

The page style foo is defined by defining the command \ps@foo. This command should make only local definitions. There should be no stray spaces in the definition, since they could lead to mysterious extra spaces in the output (well, that’s something that should be always avoided).

\begin{verbatim}
\oddevenhead
\oddhead
\evenhead
\oddfoot
\evenfoot
\end{verbatim}

The \ps@... command defines the macros \oddhead, \oddfoot, \evenhead, \evenfoot to define the running heads and feet—e.g., \oddhead is the macro to produce the contents of the heading box for odd-numbered pages. It is called inside an \ hbox of width \textwidth.
6.4.1 Marking conventions

To make headings determined by the sectioning commands, the page style defines the commands \chaptermark, \sectionmark, \...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.)

\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 \@oddhead, \@oddfoot, \@evenhead or \@evenfoot macros, it gets the current ‘left’ mark. \leftmark works like \TeX{}’s \botmark command.
\rightmark: Used in the \@oddhead, \@oddfoot, \@evenhead or \@evenfoot macros, it gets the current ‘right’ mark. \rightmark works like \TeX{}’s \firstmark command.

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 two \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.

6.4.2 Defining the page styles

The page styles \empty{} and \plain{} are defined in the \LaTeX{} format.

\ps@headings

The definition of the page style \headings{} has to be different for two sided printing than it is for one sided printing.

\if@twoside
\def\ps@headings{%

The running feet are empty in this page style, the running head contains the page number and one of the marks.

\let\@oddfoot\@empty\let\@evenfoot\@empty
\def\@evenhead{\thepage\hfil\slshape\leftmark}%
\def\@oddhead{\slshape\rightmark\hfil\thepage}%

When using this page style, the contents of the running head is determined by the chapter and section titles. So we \let \@mkboth to \markboth.

\let\@mkboth\markboth

For the article document class we define \sectionmark to clear the right mark and put the number of the section (when it is numbered) and its title in the left mark. The rightmark is set by \subsectionmark to contain the subsection titles.

Note the use of ##1 for the parameter of the \sectionmark command, which will be defined when \ps@headings is executed.

(*article)
The definitions of \texttt{\ps@headings} for one sided printing can be much simpler, because we treat even and odd pages the same. Therefore we don’t need to define \texttt{\@even}... 

\begin{verbatim}
  \else
  \def\ps@headings{
    \let\@oddfoot\@empty
    \def\@oddhead{{\slshape\rightmark}\hfil\thepage}\
    \let\@mkboth\markboth
  }
\end{verbatim}

We use \texttt{\markright} now instead of \texttt{\markboth} as we did for two sided printing.
The definition of the page style \textit{myheadings} is fairly simple because the user determines the contents of the running head himself by using the \texttt{\markboth} and \texttt{\markright} commands.

\begin{verbatim}
def\ps@myheadings{\let\@oddfoot\@empty\let\@evenfoot\@empty \def\@evenhead{\thepage\hfil\slshape\leftmark} \def\@oddhead{\slshape\rightmark\hfil\thepage}}
\end{verbatim}

We have to make sure that the marking commands that are used by the chapter and section headings are disabled. We do this letting them to a macro that gobbles its argument(s).

\begin{verbatim}
\let\@mkboth\@gobbletwo \let\chaptermark\@gobble \let\sectionmark\@gobble \let\subsectionmark\@gobble}
\end{verbatim}

\section{Document Markup}

\subsection{The title}

\texttt{\title}, \texttt{\author} \texttt{\date} These three macros are provided by the \LaTeX{} format to provide information about the title, author(s) and date of the document. The information is stored away in internal control sequences. It is the task of the \texttt{\maketitle} command to use the information provided. The definitions of these macros are shown here for information.

\begin{verbatim}
\% \DeclareRobustCommand*{\title}[1]{\gdef@title#1}}
\% \DeclareRobustCommand*{\author}[1]{\gdef@author#1}}
\% \DeclareRobustCommand*{\date}[1]{\gdef@date#1}}
\end{verbatim}

The \texttt{\date} macro gets today’s date by default.

\begin{verbatim}
\% \date{\today}
\end{verbatim}

\texttt{\maketitle} The definition of \texttt{\maketitle} depends on whether a separate title page is made. This is the default for the report and book document classes, but for the article class it is optional.

When we are making a title page, we locally redefine \texttt{\footnotesize} and \texttt{\footnoterule} to change the appearance of the footnotes that are produced by the \texttt{\thanks} command; these changes affect all footnotes.

\begin{verbatim}
\if@titlepage\newcommand\maketitle{\begin{titlepage}\begin{center} \end{center}\end{titlepage}}\else
\maketitle\fi
\end{verbatim}
We center the entire title vertically; the centering is set off a little by adding a \vskip. (In compatibility mode the page number is set to 0 by the titlepage environment to keep the behaviour of \LaTeX\ 2.09 style files.)

Then we set the title, in a \LARGE font; leave a little space and set the author(s) in a \large font. We do this inside a tabular environment to get them in a single column. Before the date we leave a little whitespace again.

Then we call \@thanks to print the information that goes into the footnote and finish the page.

We reset the footnote counter, disable \thanks and \maketitle and save some storage space by emptying the internal information macros.

After the title is set the declaration commands \title, etc. can vanish. The definition of \and makes only sense within the argument of \author so this can go as well.

When the title is not on a page of its own, the layout of the title is a little different. We use symbols to mark the footnotes and we have to deal with two-column documents.

Therefore we first start a new group to keep changes local. Then we redefine \thefootnote to use \fnsymbol: and change \@makefnmark so that footnotemarks have zero width (to make the centering of the author names look better).
If this is a two-column document we start a new page in two-column mode, with the title set to the full width of the text. The actual printing of the title information is left to \maketitle.

When this is not a two-column document we just start a new page, prevent floating objects from appearing on the top of this page and print the title information.

This page gets a plain layout. We call \thanks to produce the footnotes.

Now we can close the group, reset the footnote counter, disable \thanks, \maketitle and \maketitle and save some storage space by emptying the internal information macros.

\maketitle This macro takes care of formatting the title information when we have no separate title page.

We always start a new page, leave some white space and center the information. The title is set in a \LARGE font, the author names and the date in a \large font.
7.2 Chapters and Sections

7.2.1 Building blocks

The definitions in this part of the class file make use of two internal macros, \@startsection and \secdef. To understand what is going on here, we describe their syntax.

The macro \@startsection has 6 required arguments, optionally followed by a *, an optional argument and a required argument:

\@startsection\{name\}{level\}{indent\}{beforeskip\}{afterskip\}{style\} optional *

It is a generic command to start a section, the arguments have the following meaning:

\langle name\rangle The name of the user level command, e.g., ‘section’.

\langle level\rangle A number, denoting the depth of the section – e.g., chapter=1, section = 2, etc. A section number will be printed if and only if \langle level\rangle \leq \text{the value of the secnumdepth counter}.

\langle indent\rangle The indentation of the heading from the left margin

\langle beforeskip\rangle The absolute value of this argument gives the skip to leave above the heading. If it is negative, then the paragraph indent of the text following the heading is suppressed.

\langle afterskip\rangle If positive, this gives the skip to leave below the heading, else it gives the skip to leave to the right of a run-in heading.

\langle style\rangle Commands to set the style of the heading.

* When this is missing the heading is numbered and the corresponding counter is incremented.

\langle altheading\rangle Gives an alternative heading to use in the table of contents and in the running heads. This should not be present when the * form is used.

\langle heading\rangle The heading of the new section.
A sectioning command is normally defined to \@startsection and its first six arguments.

The macro \seccdef can be used when a sectioning command is defined without using \@startsection. It has two arguments:

\seccdef( unstarcmds )( starcmd )

\langle unstarcmds \rangle Used for the normal form of the sectioning command.

\langle starcmd \rangle Used for the *-form of the sectioning command.

You can use \seccdef as follows:

\def\chapter{ ... \seccdef \CMDA \CMDB }
\def\CMDA[#1]{ ... } % Command to define \chapter[#1]{...}
\def\CMDB{ ... } % Command to define \chapter*{...}

7.2.2 Mark commands

\chaptermark
\sectionmark
\subsectionmark
\subsubsectionmark
\paragraphmark
\subparagraphmark

Default initializations of \...mark commands. These commands are used in the definition of the page styles (see section 6.4.2) Most of them are already defined by the \LaTeX{} format, so they are only shown here.

\chaptermark \sectionmark \subsectionmark \subsubsectionmark \paragraphmark \subparagraphmark

7.2.3 Define Counters

\c@secnumdepth

The value of the counter secnumdepth gives the depth of the highest-level sectioning command that is to produce section numbers.

\setcounter{secnumdepth}{3}
\setcounter{secnumdepth}{2}

\c@part
\c@chapter
\c@section
\c@subsection
\c@paragraph
\c@subparagraph

These counters are used for the section numbers. The macro \newcounter{\langle newctr \rangle}{\langle oldctr \rangle} defines \langle newctr \rangle to be a counter, which is reset to zero when counter \langle oldctr \rangle is stepped. Counter \langle oldctr \rangle must already be defined.
For any counter \texttt{CTR}, \footnote{\texttt{\textbackslash theCTR} is a macro that defines the printed version of counter \texttt{CTR}. It is defined in terms of the following macros:}{\texttt{\textbackslash theCTR}} is a macro that defines the printed version of counter \texttt{CTR}. It is defined in terms of the following macros:

\begin{itemize}
\item \texttt{\textbackslash arabic\{COUNTER\}} prints the value of \texttt{COUNTER} as an arabic numeral.
\item \texttt{\textbackslash roman\{COUNTER\}} prints the value of \texttt{COUNTER} as a lowercase roman numeral.
\item \texttt{\textbackslash Roman\{COUNTER\}} prints the value of \texttt{COUNTER} as an uppercase roman numeral.
\item \texttt{\textbackslash alph\{COUNTER\}} prints the value of \texttt{COUNTER} as a lowercase letter: 1 = a, 2 = b, etc.
\item \texttt{\textbackslash Alph\{COUNTER\}} prints the value of \texttt{COUNTER} as an uppercase letter: 1 = A, 2 = B, etc.
\end{itemize}

Actually to save space the internal counter representations and the commands operating on those are used.

\begin{verbatim}
\renewcommand \thepart {\@Roman\c@part}
\renewcommand \thesection {\@arabic\c@section}
\renewcommand \thechapter {\@arabic\c@chapter}
\renewcommand \thesection {\thechapter.\@arabic\c@section}
\renewcommand \thesubsection {\thesection.\@arabic\c@subsection}
\renewcommand \thesubparagraph {\theparagraph.\@arabic\c@subparagraph}
\end{verbatim}

\@chapapp \@chapapp is initially defined to be \texttt{\textbackslash chaptername}'. The \texttt{\textbackslash appendix} command redefines it to be \texttt{\textbackslash appendixname}'.

\begin{verbatim}
\newcommand\@chapapp{
\chaptername}
\end{verbatim}

\subsection{Front Matter, Main Matter, and Back Matter}

A book contains these three (logical) sections. The switch \texttt{\@mainmatter} is true iff we are processing Main Matter. When this switch is false, the \texttt{\chapter} command does not print chapter numbers.

Here we define the commands that start these sections.

\texttt{\textbackslash frontmatter} This command starts Roman page numbering and turns off chapter numbering. Since this restarts the page numbering from 1, it should also ensure that a recto page is used.

\begin{verbatim}
\newcommand\frontmatter{%
% \if@openright
% \cleardoublepage
% \else
% \clearpage
% \fi
\% \@mainmatterfalse
\pagenumbering{roman}
\end{verbatim}

\texttt{\textbackslash mainmatter} This command clears the page, starts arabic page numbering and turns on chapter numbering. Since this restarts the page numbering from 1, it should also ensure that a recto page is used.

\begin{verbatim}
\newcommand\mainmatter{%
\end{verbatim}

27
7.2.5 Parts

The command to start a new part of our document.

In the article class the definition of \part is rather simple; we start a new paragraph, add a little white space, suppress the indentation of the first paragraph and make use of \secdef. As in other sectioning commands (cf. \@startsection in the \LaTeX \textsc{ε} kernel), we need to check the @noskipsec switch and force horizontal mode if it is set.

\newcommand\part{%
  \if@noskipsec\leavevmode\fi
  \par
  \addvspace{4ex}%
  \@afterindentfalse
  \secdef\@part\@spart}

For the report and book classes we things a bit different.

We start a new (righthand) page and use the \textit{plain} pagemode.

\newcommand\part{%
  \if@openright
    \cleardoublepage
  \else
    \clearpage
  \fi
  \thispagestyle{plain}%
}

When we are making a two-column document, this will be a one column page. We use \@tempswa to remember to switch back to two columns.

\if@twocolumn
  \onecolumn
\else
  \@tempswafalse
\fi
We need an empty box to prevent the filament glue from disappearing.

Here we use \texttt{\textbackslash secdet} to indicate which commands to use to make the actual heading.

This macro does the actual formatting of the title of the part. Again the macro is differently defined for the article document class than for the document classes report and book.

When \texttt{secnumdepth} is larger than \texttt{-1} for the document class article, we have a numbered part, otherwise it is unnumbered.

We print the title flush left in the article class. Also we prevent breaking between lines and reset the font.

We print the title flush left in the article class. Also we prevent breaking between lines and reset the font.

When this is a numbered part we have to print the number and the title. The \texttt{\textbackslash nobreak} should prevent a page break here.

Now we empty the mark registers, leave some white space and let \texttt{\textbackslash afterheading} take care of suppressing the indentation.

When \texttt{secnumdepth} is larger than \texttt{-2} for the document class report and book, we have a numbered part, otherwise it is unnumbered.
We empty the mark registers and center the title on the page in the report and book document classes. Also we prevent breaking between lines and reset the font.

\markboth{}{}
\centering
\interlinepenalty \@M
\normalfont

When this is a numbered part we have to print the number.

\ifnum \c@secnumdepth >-2\relax
\huge \bfseries \partname \thepart 
\par
\fi
\Huge \bfseries #2 \par
\@endpart

\endpart
This macro finishes the part page, for both \part and \@spart.

First we fill the current page.

\@endpart
This macro does the actual formatting of the title of the part when the star form of the user command was used. In this case we never print a number. Otherwise the formatting is the same.

The differences between the definition of this macro in the article document class and in the report and book document classes are similar as they were for \@part.

\def\@spart#1{\parindent \z@ \raggedright
\interlinepenalty \@M
\normalfont
\huge \bfseries #1 \par
\nobreak
\vskip 3ex
\@afterheading}
\@afterheading}

\if@twoside
\if@openright
\vfil
\newpage
Then, when we are in twosided mode and chapters are supposed to be on right hand sides, we produce a completely blank page.

\if@twoside
\if@openright
7.2.6 Chapters

A chapter should always start on a new page therefore we start by calling \clearpage and setting the pagestyle for this page to plain.

Then we prevent floats from appearing at the top of this page because it looks weird to see a floating object above a chapter title.

Then we suppress the indentation of the first paragraph by setting the switch \@afterindent to false. We use \secdef to specify the macros to use for actually setting the chapter title.

This macro is called when we have a numbered chapter. When \seclnumdepth is larger than −1 and, in the book class, \@mainmatter is true, we display the chapter number. We also inform the user that a new chapter is about to be typeset by writing a message to the terminal.

After having written an entry to the table of contents we store the (alternative) title of this chapter with \chaptermark and add some white space to the lists of figures and tables.
Then we call upon \@makechapterhead to format the actual chapter title. We have to do this in a special way when we are in two-column mode in order to have the chapter title use the entire \textwidth. In one-column mode we call \@afterheading which takes care of suppressing the indentation.

\@makechapterhead

The macro above uses \@makechapterhead\langle text\rangle to format the heading of the chapter.

We begin by leaving some white space. The we open a group in which we have a paragraph indent of 0pt, and in which we have the text set ragged right. We also reset the font.

\def\@makechapterhead#1{\vspace*{50\p@}%
\parindent \z@ \raggedright \normalfont

Then we check whether the number of the chapter has to be printed. If so we leave some whitespace between the chapternumber and its title.

\ifnum \c@secnumdepth >\m@ne \huge\bfseries \@chapapp\space \thechapter
\par
\vskip 20\p@
\fi

Now we set the title in a large bold font. We prevent a pagebreak from occurring in the middle of or after the title. Finally we leave some whitespace before the text begins.

\interlinepenalty\@M
\Huge \bfseries #1
\vskip 40\p@}

This macro is called when we have an unnumbered chapter. It is much simpler than \@chapter because it only needs to typeset the chapter title.

\def\@schapter#1{\if@twocolumn
\@topnewpage{\@makeschapterhead{#1}}%
\else
\@makeschapterhead{#1}%
\@afterheading
\fi}

The macro above uses \@makeschapterhead\langle text\rangle to format the heading of the chapter. It is similar to \@makechapterhead except that it never has to print a chapter number.
7.2.7 Lower level headings

These commands all make use of \@startsection.

\section This gives a normal heading with white space above and below the heading, the title set in \Large\bfseries, and no indentation on the first paragraph.
\subsection This gives a normal heading with white space above and below the heading, the title set in \large\bfseries, and no indentation on the first paragraph.
\subsubsection This gives a normal heading with white space above and below the heading, the title set in \normalsize\bfseries, and no indentation on the first paragraph.
\paragraph This gives a run-in heading with white space above and to the right of the heading, the title set in \normalsize\bfseries.
\subparagraph This gives an indented run-in heading with white space above and to the right of the heading, the title set in \normalsize\bfseries.
7.3 Lists

7.3.1 General List Parameters

The following commands are used to set the default values for the list environment’s parameters. See the \LaTeX{} manual for an explanation of the meanings of the parameters. Defaults for the list environment are set as follows. First, \texttt{\rightmargin}, \texttt{\listparindent} and \texttt{\itemindent} are set to 0pt. Then, for a Kth level list, the command \texttt{\@listK} is called, where ‘K’ denotes ‘i’, ‘ii’, ..., ‘vi’. (I.e., \texttt{\@listiii} is called for a third-level list.) By convention, \texttt{\@listK} should set \texttt{\leftmargin} to \texttt{\leftmarginK}.

\begin{verbatim}
\leftmargin \leftmargini
\leftmarginii \leftmarginiii \leftmarginiv \leftmarginv \leftmarginvi
\end{verbatim}

When we are in two-column mode some of the margins are set somewhat smaller.

\begin{verbatim}
\if@twocolumn
\setlength\leftmargini {2em}
\else
\setlength\leftmargini {2.5em}
\fi
\end{verbatim}

Until the whole of the parameter setting in these files is rationalised, we need to set the value of \texttt{\leftmargin} at this outer level.

\begin{verbatim}
\leftmargin \leftmargini
\end{verbatim}

The following three are calculated so that they are larger than the sum of \texttt{\labelsep} and the width of the default labels (which are ‘(m)’, ‘vii.’ and ‘M.’).

\begin{verbatim}
\if@twocolumn
\setlength\leftmarginii {2.2em}
\setlength\leftmarginiii {1.87em}
\setlength\leftmarginiv {1.7em}
\else\fi
\end{verbatim}

\begin{verbatim}
\if@twocolumn
\setlength\leftmarginv {.5em}
\setlength\leftmarginvi {.5em}
\else
\setlength\leftmarginv {1em}
\setlength\leftmarginvi {1em}
\fi
\end{verbatim}

\begin{verbatim}
\setlength \labelsep {.5em}
\setlength \labelwidth{\leftmargini}
\addtolength\labelwidth{-\labelsep}
\end{verbatim}

\begin{verbatim}
\partopsep
\end{verbatim}

\begin{verbatim}
\partopsep is the distance between the label and the text of an item; \texttt{\labelwidth} is the width of the label.
\end{verbatim}

\begin{verbatim}
\setlength \labelsep {.5em}
\setlength \labelwidth{\leftmargini}
\addtolength\labelwidth{-\labelsep}
\end{verbatim}

\begin{verbatim}
\parindent
\end{verbatim}

\begin{verbatim}
When the user leaves a blank line before the environment an extra vertical space of \texttt{\parindent} is inserted, in addition to \texttt{\parskip} and \texttt{\topsep}.
\end{verbatim}

\begin{verbatim}
\if@twocolumn
\setlength\parindent{2\p@ \@plus 1\p@ \@minus 1\p@}
\else
\setlength\parindent{3\p@ \@plus 1\p@ \@minus 1\p@}
\fi
\end{verbatim}

\begin{verbatim}
\@beginparpenalty \@endparpenalty
\end{verbatim}

These penalties are inserted before and after a list or paragraph environment.

\begin{verbatim}
\@itempenalty
\end{verbatim}

This penalty is inserted between list items.

\begin{verbatim}
\@article | report | book
\end{verbatim}

34
\@listi \@listi defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\@listii \@listi defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\@listiii \@listii defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\@listiv \@listiii defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\@listv \@listiv defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\@listvi \@listv defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

We initialise the parameters although strictly speaking that is not necessary.

Here are the same macros for the higher level lists. Note that they don’t have saved versions and are not modified by the font size commands. In other words this class assumes that nested lists only appear in \normalsize, i.e. the main document size.

\def\@listii \@listvi defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\def\@listiii \@listi defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\def\@listiv \@listii defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\def\@listv \@listiii defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.

\def\@listvi \@listiv defines the values of \leftmargin, \parsep, \topsep, \itemsep, etc. for the lists that appear on top-level. Its definition is modified by the font-size commands (eg within \small the list parameters get “smaller” values).

For this reason listI is defined to hold a saved copy of listi so that \normalsize can switch all parameters back.
7.3.2 Enumerate

The enumerate environment uses four counters: enumi, enumii, enumiii and enumiv, where enumN controls the numbering of the Nth level enumeration.

\theenumi \theenumii \theenumiii \theenumiv

The counters are already defined in the LaTeX format, but their representation is changed here.

\renewcommand\theenumi{\@arabic{\c@enumi}}
\renewcommand\theenumii{\@alph{\c@enumii}}
\renewcommand\theenumiii{\@roman{\c@enumiii}}
\renewcommand\theenumiv{\@Alph{\c@enumiv}}

\newcommand\labelenumi{\theenumi.}
\newcommand\labelenumii{\textbullet}
\newcommand\labelenumiii{\textbullet}
\newcommand\labelenumiv{\textbullet}

7.3.3 Itemize

Itemization is controlled by four commands: \labelitemi, \labelitemii, \labelitemiii, \labelitemiv, which define the labels of the various itemization levels: the symbols used are bullet, bold endash, centered asterisk and centred dot.

\newcommand\labelitemi{\textbullet}
The default definition for \labelitemfont is to reset the font to \normalfont so that always the same symbol is produced regardless of surrounding conditions. Possible alternatives would be, for example,
\renewcommand\labelitemfont
{\normalfont\fontfamily{lmss}\selectfont}
\renewcommand\labelitemfont
{\rmfamily\normalshape}
the first would use symbols from Latin Modern Sans, the second would only allow changes in the font series so that an itemize in a bold context would produce bolder symbols.

7.3.4 Description

The description environment is defined here – while the itemize and enumerate environments are defined in the \LaTeX{} format.

\newenvironment{description}
{\list{}{\labelwidth\z@ \itemindent-\leftmargin\let\makelabel\descriptionlabel}}
{\endlist}
\descriptionlabel
To change the formatting of the label, you must redefine \descriptionlabel.

7.4 Defining new environments

7.4.1 Abstract

When we are producing a separate titlepage we also put the abstract on a page of its own. It will be centred vertically on the page.

Note that this environment is not defined for books.

% \changes{v1.3m}{1995/10/23}{Added setting of \cs{beginparpenalty} to discourage page break before abstract heading.}
\maketitle
\if@titlepage
\newenvironment{abstract}{%
When we are not making a separate titlepage—the default for the article document class—we have to check if we are in two-column mode. In that case the abstract is as a \section*, otherwise the quotation environment is used to typeset the abstract.

\begin{Verbatim}
\hspace{0pt}
\end{Verbatim}

\subsection{Verse}

The verse environment is defined by making clever use of the list environment’s parameters. The user types `\` to end a line. This is implemented by `\let`’ing `\` equal `\@centercr`.

\begin{Verbatim}
\hspace{0pt}
\end{Verbatim}

\subsection{Quotation}

The quotation environment is also defined by making clever use of the list environment’s parameters. The lines in the environment are set smaller than `\textwidth`. The first line of a paragraph inside this environment is indented.

\begin{Verbatim}
\hspace{0pt}
\end{Verbatim}

\subsection{Quote}

The quote environment is like the quotation environment except that paragraphs are not indented.

\begin{Verbatim}
\hspace{0pt}
\end{Verbatim}
7.4.5 Theorem
This document class does not define its own theorem environments, the defaults, supplied by the \LaTeX format are available.

7.4.6 Titlepage

In the normal environments, the titlepage environment does nothing but start and end a page, and inhibit page numbers. When \LaTeX is in two-column mode, the environment temporarily switches to one-column mode. In the report class, it also resets the page number to one, and then, in two-column mode, sets it back to one at the end. For the book class the environment makes sure that the title page is on a recto page by issuing a \cleardoublepage command. In compatibility mode, it sets the page number to zero. This is incorrect since it results in using the page parameters for a right-hand page but it is the way it was.

First we do give the definition for compatibility mode.

\[
\begin{align*}
\text{if@compatibility} \\
\text{newenvironment{titlepage} } \\
\text{%} \\
\text{(book)} \quad \text{\cleardoublepage} \\
\text{\if@twocolumn} \\
\text{\if@twocolumn} \\
\text{\if@restonecoltrue\onecolumn} \\
\text{\else} \\
\text{\if@restonecolfalse\newpage} \\
\text{\fi} \\
\text{\thispagestyle{empty}%} \\
\text{\setcounter{page}\z@}%\text{%} \\
\text{\if@restonecol\twocolumn \else \newpage \fi} \\
\text{\}} \\
\text{\And here is the one for native \LaTeX 2ε.} \\
\text{\else} \\
\text{\newenvironment{titlepage} } \\
\text{%} \\
\text{(book)} \quad \text{\cleardoublepage} \\
\text{\if@twocolumn} \\
\text{\if@twocolumn} \\
\text{\if@restonecoltrue\onecolumn} \\
\text{\else} \\
\text{\if@restonecolfalse\newpage} \\
\text{\fi} \\
\text{\thispagestyle{empty}%} \\
\text{\setcounter{page}\@ne}%\text{%} \\
\text{\if@restonecol\twocolumn \else \newpage \fi} \\
\text{\}}
\end{align*}
\]

If we are not in two-side mode the first page after the title page should also get page number 1.

\[
\begin{align*}
\text{\if@twoside} \text{\else} \\
\text{\setcounter{page}\@ne} \\
\text{\}}
\end{align*}
\]
7.4.7 Appendix

The `\appendix` command is not really an environment, it is a macro that makes some changes in the way things are done.

In the article document class the `\appendix` command must do the following:

- reset the section and subsection counters to zero,
- redefine `\thesection` to produce alphabetic appendix numbers. This redefinition is done globally to ensure that it survives even if `\appendix` is issued within an environment such as `multicols`.

```
\newcommand\appendix{\par
\setcounter{section}{0}\
\setcounter{subsection}{0}\
\gdef\thesection{\@Alph\c@section}}
```

In the report and book document classes the `\appendix` command must do the following:

- reset the chapter and section counters to zero,
- set `\@chapapp` to `\appendixname` (for messages),
- redefine the chapter counter to produce appendix numbers,
- possibly redefine the `\chapter` command if appendix titles and headings are to look different from chapter titles and headings. This redefinition is done globally to ensure that it survives even if `\appendix` is issued within an environment such as `multicols`.

```
\newcommand\appendix{\par
\setcounter{chapter}{0}\
\setcounter{section}{0}\
\gdef\@chapapp{\appendixname}\
\gdef\thechapter{\@Alph\c@chapter}}
```

7.5 Setting parameters for existing environments

7.5.1 Array and tabular

`\arraycolsep` The columns in an array environment are separated by 2`\arraycolsep`.

```
\setlength{\arraycolsep}{5\p@}
```

`\tabcolsep` The columns in an tabular environment are separated by 2`\tabcolsep`.

```
\setlength{\tabcolsep}{6\p@}
```
The width of rules in the array and tabular environments is given by `arrayrulewidth`.

1061 \setlength{arrayrulewidth}{.4\p@}

The space between adjacent rules in the array and tabular environments is given by `doublerulesep`.

1062 \setlength{doublerulesep}{2\p@}

### 7.5.2 Tabbing

`tabbingsep` This controls the space that the `\` command puts in. (See `LaTeX` manual for an explanation.)

1063 \setlength{tabbingsep}{\labelsep}

### 7.5.3 Minipage

`minipagerestore` The macro `\minipagerestore` is called upon entry to a minipage environment to set up things that are to be handled differently inside a minipage environment. In the current classes, it does nothing.

`mpfootins` Minipages have their own footnotes; `\skip\mpfootins` plays same rôle for footnotes in a minipage as `\skip\footins` does for ordinary footnotes.

1064 `\skip\mpfootins = \skip\footins`

### 7.5.4 Framed boxes

`fboxsep` The space left by `\fbox` and `\framebox` between the box and the text in it.

`fboxrule` The width of the rules in the box made by `\fbox` and `\framebox`.

1065 \setlength{fboxsep}{2\p@}
1066 \setlength{fboxrule}{.4\p@}

### 7.5.5 Equation and eqnarray

`theequation` When within chapters, the equation counter will be reset at the beginning of a new chapter and the equation number will be prefixed by the chapter number.

This code must follow the `\chapter` definition or, more exactly, the definition of the chapter counter.

1067 `{article}\renewcommand\theequation{\@arabic\c@equation}
1068 \{*report|book\}
1069 \@addtoreset{equation}{chapter}
1070 \renewcommand\theequation
1071 {\ifnum \c@chapter>0 \thechapter.\fi \@arabic\c@equation}
1072 \{/report|book\}

`\jot` \jot is the extra space added between lines of an eqnarray environment. The default value is used.

1073 % \setlength{\jot}{3pt}

`\seqnum` The macro `\seqnum` defines how equation numbers are to appear in equations. Again the default is used.

1074 % \def\seqnum{\{\theequation\}}
7.6 Floating objects

The \LaTeX{} format only defines a number of tools with which floating objects can be defined. This is done in the document class. It needs to define the following macros for each floating object of type \texttt{TYPE} (e.g., \texttt{TYPE} = \texttt{figure}).

\texttt{\fps@TYPE} The default placement specifier for floats of type \texttt{TYPE}.

\texttt{\ftype@TYPE} The type number for floats of type \texttt{TYPE}. Each \texttt{TYPE} has associated a unique positive \texttt{TYPE} number, which is a power of two. E.g., figures might have type number 1, tables type number 2, programs type number 4, etc.

\texttt{\ext@TYPE} The file extension indicating the file on which the contents list for float type \texttt{TYPE} is stored. For example, \texttt{\ext@figure} = ‘lof’.

\texttt{\fnum@TYPE} A macro to generate the figure number for a caption. For example, \texttt{\fnum@TYPE} == ‘Figure \thefigure’.

\texttt{\@makecaption⟨num⟩⟨text⟩} A macro to make a caption, with \texttt{⟨num⟩} the value produced by \texttt{\fnum@…} and \texttt{⟨text⟩} the text of the caption. It can assume it’s in a \texttt{parbox} of the appropriate width. This will be used for all floating objects.

The actual environment that implements a floating object such as a figure is defined using the macros \texttt{\@float} and \texttt{\end@float}, which are defined in the \LaTeX{} format.

An environment that implements a single-column floating object is started with \texttt{\@float\{TYPE\}[⟨placement⟩]} of type \texttt{TYPE} with \texttt{⟨placement⟩} as the placement specifier. The default value of \texttt{⟨PLACEMENT⟩} is defined by \texttt{\fps@TYPE}.

The environment is ended by \texttt{\end@float}. E.g., \texttt{\figure} == \texttt{\@floatfigure}, \texttt{\endfigure} == \texttt{\end@float}.

7.6.1 Figure

Here is the implementation of the figure environment.

\texttt{\c@figure} First we have to allocate a counter to number the figures.

In the report and book document classes figures within chapters are numbered per chapter.

\begin{verbatim}
\def\fps@figure{tbp}
\def\ftype@figure{1}
\def\ext@figure{lof}
\def\fnum@figure{\figurename\nobreakspace \thefigure}
\end{verbatim}

\texttt{\fps@figure} Here are the parameters for the floating objects of type ‘figure’.

\begin{verbatim}
\def\fps@figure{tbp}
\def\ftype@figure{1}
\def\ext@figure{lof}
\def\fnum@figure{\figurename\nobreakspace\lof}\fi \@arabic\c@figure}
\end{verbatim}

42
And the definition of the actual environment. The form with the \* is used for double-column figures.

\begin{figure}
\end{figure}

\begin{figure*}
\end{figure*}

\newenvironment{figure}
{\@float{figure}}
{\end@float}
\newenvironment{figure*}
{\@dblfloat{figure}}
{\end@dblfloat}

7.6.2 Table

Here is the implementation of the table environment. It is very much the same as the figure environment.

\newcounter{table}
\renewcommand\thetable{\@arabic\c@table}
\newcounter{table}[chapter]
\renewcommand\thetable{\ifnum\c@chapter>\z@ \thechapter.\fi \@arabic\c@table}
\def\fps@table{tbp}
\def\ftype@table{2}
\def\ext@table{lot}
\def\fnum@table{\tablename\ \thetable}

And the definition of the actual environment. The form with the \* is used for double-column tables.

\begin{table}
\end{table}

\begin{table*}
\end{table*}

7.6.3 Captions

\makecaption
The \caption command calls \makecaption to format the caption of floating objects. It gets two arguments, \textit{(number)}, the number of the floating object and \textit{text}, the text of the caption. Usually \texttt{(number)} contains a string such as ‘Figure 3.2’. The macro can assume it is called inside a \texttt{parbox} of right width, with \texttt{normalsize}.

\abovecaptionskip
\belowcaptionskip
These lengths contain the amount of white space to leave above and below the caption.
The definition of this macro is \texttt{\long} in order to allow more than one paragraph in a caption.

\begin{verbatim}
\longdef\makecaption#1#2{%
\vskip\abovecaptionskip
We want to see if the caption fits on one line on the page, therefore we first typeset it in a temporary box.
\setbox\@tempboxa{#1: #2}\
We can then measure its width. It that is larger than the current \texttt{\hsize} we typeset the caption as an ordinary paragraph.
\ifdim\wd\@tempboxa >\hsize
#1: #2\par
If the caption fits, we center it. Because this uses an \texttt{\hbox} directly in vertical mode, it does not execute the \texttt{\everypar} tokens; the only thing that could be needed here is resetting the ‘minipage flag’ so we do this explicitly.
\else
\global\@minipagetrue
\hb@xt@\hsize{\hfil\box\@tempboxa\hfil}\
\fi
\vskip\belowcaptionskip}
\end{verbatim}

7.7 Font changing

Here we supply the declarative font changing commands that were common in \LaTeX version 2.09 and earlier. These commands work in text mode \textit{and} in math mode. They are provided for compatibility, but one should start using the \texttt{\text...} and \texttt{\math...} commands instead. These commands are defined using \texttt{\DeclareTextFontCommand}, a command with three arguments: the user command to be defined; \LaTeX\ commands to execute in text mode and \LaTeX\ commands to execute in math mode.

\begin{verbatim}
\rm The commands to change the family. When in compatibility mode we select the
\tt 'default' font first, to get \LaTeX\2.09 behaviour.
\sf
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
\bf
The command to change to the bold series. One should use \texttt{\mdseries} to explicitly
switch back to medium series.
\DeclareOldFontCommand{\bf}{\normalfont\bfseries}{\mathbf}
\sl
And the commands to change the shape of the font. The slanted and small caps
shapes are not available by default as math alphabets, so those changes do nothing
in math mode. However, we do warn the user that the selection will not have any
effect. One should use \texttt{\upshape} to explicitly change back to the upright shape.
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
\end{verbatim}
The commands \texttt{\cal} and \texttt{\mit} should only be used in math mode, outside math mode they have no effect. Currently the New Font Selection Scheme defines these commands to generate warning messages. Therefore we have to define them ‘by hand’.

\newcommand*{\cal}{\@fontswitch\relax\mathcal}
\newcommand*{\mit}{\@fontswitch\relax\mathnormal}

\section*{8 Cross Referencing}

\subsection*{8.1 Table of Contents, etc.}

A \texttt{\section} command writes a \texttt{\contentsline}{\texttt{section}}{\langle\texttt{title}\rangle}{\langle\texttt{page}\rangle} command on the .toc file, where \langle\texttt{title}\rangle contains the contents of the entry and \langle\texttt{page}\rangle is the page number. If sections are being numbered, then \langle\texttt{title}\rangle will be of the form \texttt{\numberline}\langle\texttt{num}\rangle{\langle\texttt{heading}\rangle} where \langle\texttt{num}\rangle is the number produced by \texttt{\thesection}. Other sectioning commands work similarly.

A \texttt{\caption} command in a ‘figure’ environment writes
\texttt{\contentsline}{\texttt{figure}}{\langle\texttt{num}\rangle}{\langle\texttt{caption}\rangle}{\langle\texttt{page}\rangle}
on the .lof file, where \langle\texttt{num}\rangle is the number produced by \texttt{\thefigure} and \langle\texttt{caption}\rangle is the figure caption. It works similarly for a ‘table’ environment.

The command \texttt{\contentsline}{\texttt{name}} expands to \texttt{\l@\langle\texttt{name}\rangle}. So, to specify the table of contents, we must define \texttt{\l@chapter}, \texttt{\l@section}, \texttt{\l@subsection}, ... ; to specify the list of figures, we must define \texttt{\l@figure}; and so on. Most of these can be defined with the \texttt{\dottedtocline} command, which works as follows.

\texttt{\dottedtocline}{\langle\texttt{level}\rangle}{\langle\texttt{indent}\rangle}{\langle\texttt{numwidth}\rangle}{\langle\texttt{title}\rangle}{\langle\texttt{page}\rangle}

\langle\texttt{level}\rangle An entry is produced only if \langle\texttt{level}\rangle \leq \textnormal{value of the tocdepth counter}.

\langle\texttt{indent}\rangle The indentation from the outer left margin of the start of the contents line.

\langle\texttt{numwidth}\rangle The width of a box in which the section number is to go, if \langle\texttt{title}\rangle includes a \texttt{\numberline} command.

This command uses the following three parameters, which are set with a \texttt{\newcommand} (so em’s can be used to make them depend upon the font).

\texttt{\pnumwidth} The width of a box in which the page number is put.

\texttt{\tocrmarg} The right margin for multiple line entries. One wants \texttt{\tocrmarg} \geq \texttt{\pnumwidth}

\texttt{\dotsep} Separation between dots, in mu units. Should be defined as a number like 2 or 1.7

\newcommand{\pnumwidth}{1.55em}
\newcommand{\tocrmarg}{2.55em}
\newcommand{\dotsep}{4.5}
\setcounter{tocdepth}{3}
\setcounter{tocdepth}{2}
8.1.1 Table of Contents

\tableofcontents

This macro is used to request that \LaTeX{} produces a table of contents. In the report and book document classes the tables of contents, figures etc. are always set in single-column mode.

\newcommand\tableofcontents{%
\if\twocolumn
  \if@restonecoltrue\onecolumn
  \else
  \@restonecolfalse
  \fi
\fi

The title is set using the \chapter* command, making sure that the running head – if one is required – contains the right information.

\chapter*{\contentsname}
\chapter*{(report | book)
\section*{The code for \@mkboth is placed inside the heading to avoid any influence on vertical spacing after the heading (in some cases). For other commands, such as \listoffigures below this has been changed from the \LaTeX{}2.09 version as it will produce a serious bug if used in two-column mode (see, pr/3285). However \tableofcontents is always typeset in one-column mode in these classes, therefore the somewhat inconsistent setting has been retained for compatibility reasons.

\@mkboth{%
\MakeUppercase\contentsname}{\MakeUppercase\contentsname}}%

The actual table of contents is made by calling \@starttoc{toc}. After that we restore two-column mode if necessary.

\@starttoc{toc}%
\if@restonecol\twocolumn\fi

Each sectioning command needs an additional macro to format its entry in the table of contents, as described above. The macro for the entry for parts is defined in a special way.

First we make sure that if a pagebreak should occur, it occurs before this entry. Also a little whitespace is added and a group begun to keep changes local.

\newcommand*{\l@part}[2]{%
\ifnum \c@tocdepth >-2\relax
  \addpenalty\@secpenalty
  \addvspace{2.25em \plus\p@}\
\setlength{\@tempdima}{3em}%
\begingroup

The macro \numberline requires that the width of the box that holds the part number is stored in \LaTeX{}’s scratch register \@tempdima. Therefore we initialize it there even though we do not use \numberline internally—the value used is quite large so that something like \numberline{VIII} would still work.

\setlength{\@tempdima}{3em}%
\begingroup
We set \parindent to 0pt and use \rightskip to leave enough room for the page numbers.\footnote{We should really set \rightskip to \@tocmarg instead of \@pnumwidth (no version of \LaTeX{} ever did this), otherwise the \rightskip is too small. Unfortunately this can’t be changed in \LaTeX{} as we don’t want to create different versions of \LaTeX{} which produce different typeset output unless this is absolutely necessary; instead we suspend it for \LaTeX{}\protect\textsuperscript{3}.} To prevent overfull box messages the \parfillskip is set to a negative value.

\parindent \z@ \rightskip \@pnumwidth
\parfillskip -\@pnumwidth

Now we can set the entry, in a large bold font. We make sure to leave vertical mode, set the part title and add the page number, set flush right.

\leavevmode
\large \bfseries #1\hfil
\hb@xt\@pnumwidth\{\hss #2\%
\kern-\p@ \kern\p@\}

Prevent a pagebreak immediately after this entry, but use \everypar to reset the \if@nobreak switch. Finally we close the group.

\nobreak
\langle\textit{article}\rangle
\if@compatibility
\global\@nobreaktrue
\everypar\{\global\@nobreakfalse\everypar\%
\fi
\endgroup

\l@chapter This macro formats the entries in the table of contents for chapters. It is very similar to \l@part

First we make sure that if a pagebreak should occur, it occurs before this entry. Also a little whitespace is added and a group begun to keep changes local.

\begin{verbatim}
(*report | book*)
\newcommand\l@chapter[2]{%
\ifnum \c@tocdepth >\m@ne
\addpenalty{-\@highpenalty}%
\vskip 1.0em \@plus\p@
\end{verbatim}

The macro \numberline requires that the width of the box that holds the part number is stored in \LaTeX{}’s scratch register \@tempdima. Therefore we initialize it there even though we do not use \numberline internally (the position as well as the values seems questionable but can’t be changed without producing compatibility problems). We begin a group, and change some of the paragraph parameters (see also the remark at \l@part regarding \rightskip).

\setlength\@tempdima{1.5em}%
\begingroup
\parindent \z@ \rightskip \@pnumwidth
\parfillskip -\@pnumwidth

Then we leave vertical mode and switch to a bold font.

\leavevmode \bfseries

Because we do not use \numberline here, we have do some fine tuning ‘by hand’, before we can set the entry. We discourage but not disallow a pagebreak immediately after a chapter entry.
In the article document class the entry in the table of contents for sections looks much like the chapter entries for the report and book document classes.

First we make sure that if a pagebreak should occur, it occurs before this entry. Also a little whitespace is added and a group begun to keep changes local.

\l@section

The macro `\numberline` requires that the width of the box that holds the part number is stored in \TeX's scratch register `\@tempdima`. Therefore we put it there. We begin a group, and change some of the paragraph parameters (see also the remark at `\l@part` regarding `\rightskip`).

Then we leave vertical mode and switch to a bold font.

Because we do not use `\numberline` here, we have done some fine tuning ‘by hand’, before we can set the entry. We discourage but not disallow a pagebreak immediately after a chapter entry.

In the report and book document classes the definition for `\l@section` is much simpler.

All lower level entries are defined using the macro `\@dottedtocline` (see above).
8.1.2 List of figures
\listoffigures This macro is used to request that \LaTeX produces a list of figures. It is very similar to \tableofcontents.

8.1.3 List of tables
\listoftables This macro is used to request that \LaTeX produces a list of tables. It is very similar to \tableofcontents.
\l@table  This macro produces an entry in the list of tables.
\let\l@table\l@figure

8.2 Bibliography

\bibindent  The “open” bibliography format uses an indentation of \bibindent.
\newdimen\bibindent  \setlength\bibindent{1.5em}

\thebibliography  The ‘\thebibliography’ environment executes the following commands:
\renewcommand{\newblock}{\hskip.11em @plus.33em @minus.07em}  — Defines the “closed” format, where the blocks (major units of information) of an entry run together.
\sloppy  — Used because it’s rather hard to do line breaks in bibliographies,
\sfcode\.'=.1000\relax  — Causes a ‘.’ (period) not to produce an end-of-sentence space.

The implementation of this environment is based on the generic list environment. It uses the \emph{enumiv} counter internally to generate the labels of the list.

When an empty ‘\thebibliography’ environment is found, a warning is issued.
\newenvironment{thebibliography}[1]  \section*{\refname}  \@mkboth{\MakeUppercase\refname}{\MakeUppercase\refname}  \chapter*{\bibname}  \@mkboth{\MakeUppercase\bibname}{\MakeUppercase\bibname}  \list{\@biblabel{\@arabic\c@enumiv}}  \settowidth\labelwidth{\@biblabel{#1}}  \leftmargin\labelwidth  \advance\leftmargin\labelsep  \@openbib@code  \usecounter{enumiv}  \let\p@enumiv\@empty  \renewcommand{\theenumiv}{\@arabic\c@enumiv}  \sloppy

This is setting the normal (non-infinite) value of \texttt{clubpenalty} for the whole of this environment, so we must reset its stored value also. (Why is there a \% after the second 4000 below?)
\clubpenalty4000  \clubpenalty4000  \clubpenalty \clubpenalty  \widowpenalty4000  \widowpenalty4000  \sfcode'\.'\%\m
\def\noitemerr  \latex@warning{Empty ‘thebibliography’ environment}\%  \endlist
The default definition for \texttt{\newblock} is to produce a small space.

\texttt{\@openbib@code} The default definition for \texttt{\@openbib@code} is to do nothing. It will be changed by the \texttt{openbib} option.

\texttt{\@biblabel} The label for a \texttt{\bibitem[...]} command is produced by this macro. The default from the \LaTeX{} format is used.

\texttt{\@cite} The output of the \texttt{\cite} command is produced by this macro. The default from the \LaTeX{} format is used.

8.3 The index

The environment ‘theindex’ can be used for indices. It makes an index with two columns, with each entry a separate paragraph. At the user level the commands \texttt{\item}, \texttt{\subitem} and \texttt{\subsubitem} are used to produce index entries of various levels. When a new letter of the alphabet is encountered an amount of \texttt{\indexspace} white space can be added.

```
1292 \newenvironment{theindex}
1293 {
1294 \if@twocolumn
1295 \@restonecolfalse
1296 \else
1297 \@restonecoltrue
1298 \fi
1299 ⟨article⟩
1300 \twocolumn[\section*{\indexname}]
1301 ⟨article⟩
1302 \@makeschapterhead{\indexname}
1303 \@mkboth{\MakeUppercase\indexname}{\MakeUppercase\indexname}
1304 \thispagestyle{plain}\parindent\z@
```

Parameter changes to \texttt{\columnseprule} and \texttt{\columnsep} have to be done after \texttt{\twocolumn} has acted. Otherwise they can affect the last page before the index.

```
1305 \parskip\z@ \@plus .3\p@\relax
1306 \columnseprule \z@
1307 \columnsep 35\p@
1308 \let\item\@idxitem
```

When the document continues after the index and it was a one-column document we have to switch back to one column after the index.

```
1309 \if@restonecol\onecolumn\else\clearpage\fi
```

\texttt{\@idxitem} These macros are used to format the entries in the index.

```
1310 \newcommand\@idxitem{\par\hangindent 40\p@}
```

\texttt{\indexspace} The amount of white space that is inserted between ‘letter blocks’ in the index.

```
1311 \newcommand\indexspace{\par \vskip 10\p@ \@plus5\p@ \@minus3\p@\relax}
```
8.4 Footnotes

\footnoterule

Usually, footnotes are separated from the main body of the text by a small rule. This rule is drawn by the macro \footnoterule. We have to make sure that the rule takes no vertical space (see plain.tex) so we compensate for the natural height of the rule of 0.4pt by adding the right amount of vertical skip.

To prevent the rule from colliding with the footnote we first add a little negative vertical skip, then we put the rule and make sure we end up at the same point where we began this operation.

1312 \renewcommand\footnoterule{%
1313 \kern-3\p@
1314 \hrule\@width.4\columnwidth
1315 \kern2.6\p@}

\c@footnote

Footnotes are numbered within chapters in the report and book document classes.

\@makefntext

The footnote mechanism of \TeX calls the macro \@makefntext to produce the actual footnote. The macro gets the text of the footnote as its argument and should use \@thefnmark as the mark of the footnote. The macro \@makefntext is called when effectively inside a \parbox of width \columnwidth (i.e., with \hsize = \columnwidth).

An example of what can be achieved is given by the following piece of \TeX code.

\newcommand\@makefntext[1]{%\@setpar{%\@@par
1316 \@tempdima = \hsize
1317 \advance\@tempdima-10pt
1318 \parshape \@ne 10pt \@tempdima)%
1319 \par
1320 \parindent 1em
1321 \noindent
1322 \hb@xt@1.8em{\hss\@makefnmark}#1}

The effect of this definition is that all lines of the footnote are indented by 10pt, while the first line of a new paragraph is indented by 1em. To change these dimensions, just substitute the desired value for ‘10pt’ (in both places) or ‘1em’. The mark is flushright against the footnote.

In these document classes we use a simpler macro, in which the footnote text is set like an ordinary text paragraph, with no indentation except on the first line of a paragraph, and the first line of the footnote. Thus, all the macro must do is set \parindent to the appropriate value for succeeding paragraphs and put the proper indentation before the mark.

1317 \newcommand\@makefntext[1]{%
1318 \parindent 1em%
1319 \noindent
1320 \hb@xt@1.8em{\hss\@makefnmark}#1}

\@makefnmark

The footnote markers that are printed in the text to point to the footnotes should be produced by the macro \@makefnmark. We use the default definition for it.

1321 \%\renewcommand\@makefnmark{\hbox{\@textsuperscript
1322 % {\normalfont\@thefnmark}}}}
9 Initialization

9.1 Words

This document class is for documents prepared in the English language. To prepare a version for another language, various English words must be replaced. All the English words that require replacement are defined below in command names. These commands may be redefined in any class or package that is customising \LaTeX{} for use with non-English languages.

\newcommand\contentsname{Contents}
\newcommand\listfigurename{List of Figures}
\newcommand\listtablename{List of Tables}

\newcommand\refname{References}
\newcommand\bibname{Bibliography}
\newcommand\indexname{Index}

\newcommand\figurename{Figure}
\newcommand\tablename{Table}

\newcommand\partname{Part}
\newcommand\chaptername{Chapter}
\newcommand\appendixname{Appendix}
\newcommand\abstractname{Abstract}

9.2 Date

\today This macro uses the \TeX{} primitives \month, \day and \year to provide the date of the \LaTeX{}-run.

At \begin{document} this definition will be optimised so that the names of all the ‘wrong’ months are not stored. This optimisation is not done here as that would ‘freeze’ \today in any special purpose format made by loading the class file into the format file.

\def\today{\ifcase\month\or January\or February\or March\or April\or May\or June\or July\or August\or September\or October\or November\or December\fi \space\number\day, \number\year}

9.3 Two-column mode

\columnsep This gives the distance between two columns in two-column mode.
\setlength{\columnsep}{10\p@}

\columnseprule This gives the width of the rule between two columns in two-column mode. We have no visible rule.
\setlength{\columnseprule}{0\p@}
9.4 The page style

We have plain pages in the document classes article and report unless the user specified otherwise. In the ‘book’ document class we use the page style headings by default. We use arabic page numbers.

\begin{verbatim}
\pagestyle{plain}
\pagestyle{headings}
\pagenumbering{arabic}
\end{verbatim}

9.5 Single or double sided printing

When the twoside option wasn’t specified, we don’t try to make each page as long as all the others.

\begin{verbatim}
\if@twoside
  \raggedbottom
\else
  \fi
\end{verbatim}

When the twocolumn option was specified we call \texttt{\twocolumn} to activate this mode. We try to make each column as long as the others, but call \texttt{\sloppy} to make our life easier.

\begin{verbatim}
\if@twocolumn
  \twocolumn
  \sloppy
  \flushbottom
\else
  \onecolumn
\fi
\end{verbatim}

Normally we call \texttt{\onecolumn} to initiate typesetting in one column.

\begin{verbatim}
\else
  \onecolumn
\fi
\end{verbatim}

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

\begin{verbatim}
Symbols  \chapter  \cite  \endparpenalty  \chapter  \endpart  \eqnnum  \evenfoot  \evenhead  \fnsymbol  \fontswitch  \fpbot  \fptop  \highpenalty  \idxitem  \chapter  \cite  \endparpenalty
\Roman  \afterheading  \clubpenalty  \date  \afterindentfalse  \@afterheading  \@afterindentfalse  \@author  \@beginparpenalty  \@dblfpbot  \@dblpsep  \@dbltop  \@beginparpenalty  \@dblblabel  \@chapter  \@chapapp
\end{verbatim}

54