Contents

1 Introduction 3

2 The \texttt{docstrip} modules 3

3 Initial Code 3

4 Declaration of Options 4
  4.1 Setting Paper Sizes 4
  4.2 Choosing the type size 5
  4.3 Two-side or one-side printing 5
  4.4 Draft option 6
  4.5 Titlepage option 6
  4.6 openright option 6
  4.7 Table of contents formatting 6
  4.8 Formatting of the title 6
  4.9 Twocolumn printing 7
  4.10 Equation numbering on the left 7
  4.11 Flush left displays 7
  4.12 Open bibliography 7

5 Executing Options 7

6 Loading Packages 8

7 Document Layout 8
  7.1 Fonts 8
  7.2 Paragraphing 11
  7.3 Page Layout 13
    7.3.1 Vertical spacing 14
    7.3.2 The dimension of text 14

*This file has version number v2.1e, last revised 2020-11-17.
1 Introduction

This file contains the set of document classes that were made available by Working Group 13 of the NTG (Nederlandstalige \TeX Gebruikersgroep). They are compatible with the standard \LaTeXe document classes, but implement different layouts.

2 The docstrip modules

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

- artikel produce the documentclasses artikel?
- rapport produce the documentclasses rapport?
- 10pt produce the class option for 10pt
- 11pt produce the class option for 11pt
- 12pt produce the class option for 12pt
- boek produce the documentclasses boek?
- type1 produce the ‘1’ variants of the classes
- type2 produce the ‘2’ variants of the classes
- type3 produce the ‘3’ variants of the classes
- driver produce a documentation driver file

3 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.
4 Declaration of Options

4.1 Setting Paper Sizes

The variables \texttt{\paperwidth} and \texttt{\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.

\DeclareOption{a4paper}
  \setlength\paperheight {297mm}\%
\DeclareOption{a5paper}
  \setlength\paperheight {210mm}\%
\DeclareOption{b5paper}
  \setlength\paperheight {250mm}\%
\DeclareOption{letterpaper}
  \setlength\paperheight {11in}\%
\DeclareOption{legalpaper}
  \setlength\paperheight {14in}\%
\DeclareOption{executivepaper}
  \setlength\paperheight {10.5in}\%

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

\DeclareOption{landscape}
  \setlength\@tempdima {\paperheight}\%
  \setlength\paperheight {\paperwidth}\%

4.2 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 18pt options.

\DeclareOption{10pt}{\renewcommand\@ptsize{0}}
\DeclareOption{11pt}{\renewcommand\@ptsize{1}}
\DeclareOption{12pt}{\renewcommand\@ptsize{2}}

4.3 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.

\DeclareOption{oneside}{\@twosidefalse \@mparswitchfalse}
\DeclareOption{twoside}{\@twosidetrue \@mparswitchtrue}
4.4 Draft option
If the user requests draft we show any overfull boxes. We could probably add some more interesting stuff to this option.
43 \DeclareOption{draft}{\setlength\overfullrule{5pt}}
44 \DeclareOption{final}{\setlength\overfullrule{0pt}}

4.5 Titlepage option
An article usually has no separate titlepage, but the user can request one.
45 \DeclareOption{titlepage}{\@titlepagetrue}
46 \DeclareOption{notitlepage}{\@titlepagefalse}

4.6 openright option
This option determines whether or not a chapter must start on a right-hand page request one.
47 \DeclareOption{openright}{\@openrighttrue}
48 \DeclareOption{openany}{\@openrightfalse}

For these document classes there used to be a file voorwerk.sty which was a replacement for titlepag.sty. Therefore we also have the option voorwerk.
49 \DeclareOption{voorwerk}{\@titlepagetrue}
50 \DeclareOption{geenvoorwerk}{\@titlepagefalse}

4.7 Table of contents formatting
This document class uses a new layout for the table of contents, but in order to maintain compatibility with the standard \LaTeX\document classes we supply an extra option: oldtoc. If this option is specified the switch \if@oldtoc will be set true.
51 \DeclareOption{oldtoc}{\@oldtoctrue}

4.8 Formatting of the title
The option titlecentered changes the behaviour of the \maketitle command. It then produces a title like it does for the artikel1 document class.
52 \DeclareOption{titlecentered}{\@titlecenteredtrue}

In the rapport and boek document styles the titlepage uses all capital letters. The option mctitle (for ‘mixed case’) prevents this.
53 (rapport|boek)\DeclareOption{mctitle}{\@allcapsfalse}
54 (rapport|boek)\DeclareOption{uctitle}{\@allcapstrue}
4.9 Two-column printing

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

\DeclareOption{onecolumn}{\@twocolumnfalse}
\DeclareOption{twocolumn}{\@twocolumntrue}

4.10 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.

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

4.11 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.

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

4.12 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}.

\DeclareOption{openbib}{
First some hook into the bibliography environment is filled.
\AtEndOfPackage{%
\renewcommand\@openbib@code{%
\advanceleftmargin\bibindent
\itemindent -\bibindent
\listparindent \itemindent
\parsep \z@}
}%
In addition the definition of \texttt{newblock} is overwritten.
\renewcommand\newblock{\par}}%
}

5 Executing Options

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

\texttt{(*artikel\*)}
6 Loading Packages

These class files do not load additional packages.

7 Document Layout

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

7.1 Fonts

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

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

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

\begin{verbatim}
  \@vpt  5  \@vipt  6  \@viipt  7
  \@viipt  8  \@ixpt  9  \@xpt  10
  \@xipt  10.95  \@xiipt  12  \@xivpt  14.4
...
\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

\begin{verbatim}
\renewcommand{\normalsize}{\@setfontsize{\normalsize}{10pt}{12pt}
\abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
\abovedisplayshortskip \z@ \@plus3\p@
\belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
\belowdisplayskip \abovedisplayskip
\let\@listi\@listI}
\end{verbatim}

The parameters of the first level list are always given by \texttt{\@listI}. Make \texttt{\@normalsize} a synonymm for \texttt{\normalsize}. We initially choose the \texttt{\normalsize} font.

\begin{verbatim}
\texttt{\small} This is similar to \texttt{\normalsize}.
\end{verbatim}

\begin{verbatim}
\DeclareRobustCommand{\small}{\@setfontsize{\small}{10pt}{12pt}
\abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
\abovedisplayshortskip \z@ \@plus3\p@
\belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
\belowdisplayskip \abovedisplayskip
\let\@listi\@listI}
\end{verbatim}
This is similar to \normalsize.

\DeclareRobustCommand\footnotesize{%
\@setfontsize\footnotesize\@viiipt{9.5}%
\abovedisplayskip 6\p@ \@plus2\p@ \@minus4\p@
\abovedisplayshortskip \z@ \@plus\p@
\belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
}\scriptsize\tiny\large\Large\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.
\DeclareRobustCommand\scriptsize{%
\@setfontsize\scriptsize\@viiipt{9.5}%
\abovedisplayskip 6\p@ \@plus2\p@ \@minus4\p@
\abovedisplayshortskip \z@ \@plus\p@
\belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
}\Large\LARGE\Huge

\DeclareRobustCommand\Large{%
\@setfontsize\Large\@xiipt{14}
156 \DeclareRobustCommand\huge{\@setfontsize\huge\@xxpt{25}}
157 \DeclareRobustCommand\Huge{\@setfontsize\Huge\@xxvpt{30}}
158 (10pt)
159 (+11pt)
160 \DeclareRobustCommand\scriptsize{\@setfontsize\scriptsize\@viiipt{9.5}}
161 \DeclareRobustCommand\tiny{\@setfontsize\tiny\@vipt\@viipt}
162 \DeclareRobustCommand\large{\@setfontsize\large\@xipt{14}}
163 \DeclareRobustCommand\Large{\@setfontsize\Large\@xivpt{18}}
164 \DeclareRobustCommand\LARGE{\@setfontsize\LARGE\@xviipt{22}}
165 \DeclareRobustCommand\huge{\@setfontsize\huge\@xxpt{25}}
166 \Huge
167 (10pt)
168 (+12pt)
169 \setlength\lineskip{1\p@}
170 \setlength\normallineskip{1\p@}
171 \baselineskip
172 \parindent
173 \footnotesize
174 \footnoteskip

\lineskip These parameters control \TeX’s behaviour when two lines tend to come too close together.
175 \normallineskip
176 \baselineskip
177 \parindent
178 \footnoteskip

\baselineskip This is used as a multiplier for \baselineskip. 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 \baselineskip is ignored.
179 \renewcommand\baselineskip{}
180 \newdimen\unitindent
181 \parindent
182 \footnoteskip

\unitindent These document classes all use a single dimension for a number of layout parameters:
183 \unitindent
184 \footnoteskip

• the label width in section heading,
185 \parindent
• the \footnoteskip
186 \footnoteskip
• the footnote label indent (= half \unitindent)
187 \footnoteskip
• listindent on the first level
188 \newdimen\unitindent
The default setting accommodates three levels of single digit section numbering.

\othermargin Other indentations are maximal label width plus white space.

\if@needwriteindent If this is not enough, a new width is calculated, set, and the file.aux file contains an instruction that will set \unitindent on the next run.

For this we need a switch

\if@needwriteindent\else\fi\@indentset

And a command that sets the various parameters.

\newcommand*\@indentset{\@ifundefined{unitindent}{\let\@indentset\relax}{}
\global\parindent=\unitindent\global\leftmargini=\unitindent\global\@needwriteindenttrue}

The \end{document} command will call \@writeindent to write the final width of \unitindent on the file.aux file. Also a command is written to set \unitindent. To be compatible with other document classes a check is written to the .aux file for the existence of \unitindent. This prevents nasty errors when another document class is used.

\newcommand*\@writeindent[1]{\immediate\write\@mainaux{\string\@ifundefined{unitindent}{\string\newdimen\string\unitindent\let\string\@indentset\relax}{}\global\string\unitindent=#1\string\relax\string\@indentset \string\relax}}

We need to use the hook into \end{document} to write the final value of \unitindent on the file.aux file for the next run.

\AtEndDocument{%
  \if@filesw
    \if@needwriteindent
      \@writeindent{\the\unitindent}
    \fi
  \fi
}

In the document class artikel2 the width of \unitindent is fixed and related to \othermargin.

\newcommand*\artikel2\unitindent=2\othermargin

\parskip \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.

\setlength\parskip{0\p@}
\setlength\parindent{\unitindent}
\setlength\parskip{.5\baselineskip \@plus .1\baselineskip \@minus .1\baselineskip}
\setlength\parindent{\z@}

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.

\@lowpenalty 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.
\@medpenalty
\@highpenalty

\clubpenalty These penalties are use 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.

7.3 Page Layout

All margin dimensions are measured from a point one inch from the top and lefthand side of the page.
7.3.1 Vertical spacing

\headheight 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. \topskip is the \baselineskip for the first line on a page.

\headheight \headsep \topskip
\setlength{\headheight}{12pt}
\setlength{\headsep}{25pt}
\setlength{\topskip}{10pt}
\setlength{\topskip}{11pt}
\setlength{\topskip}{12pt}

\footskip 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. Bottom of page:

\setlength{\footskip}{30pt}

\maxdepth 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. In both plain \TeX and \LaTeX 2.09 \maxdepth had a fixed value of 4pt; in native \LaTeXe 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.

\if@compatibility
\setlength{\maxdepth}{4pt}
\else
\setlength{\maxdepth}{.5\topskip}
\fi

7.3.2 The dimension of text

\textwidth 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
\if@twocolumn
\setlength{\textwidth}{410pt}
\else
\setlength{\textwidth}{345pt}
\fi
\else
\setlength{\textwidth}{410pt}
\fi

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

\if@compatibility
\setlength{\textwidth}{345pt}
\setlength{\textwidth}{360pt}
\setlength{\textwidth}{390pt}
\else
\setlength{\textwidth}{410pt}
\fi

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

\if@compatibility
\else
First, we calculate the maximum textwidth, which will we will allow on the selected paper and store it in \@tempdima. Then we store the length of a line with approximately 60 – 70 characters in \@tempdimb. The values given are taken from
the file `a4.sty` by Johannes Braams and Nico Poppelier and are more or less suitable when Computer Modern fonts are used.

\setlength{\@tempdima}{\paperwidth}
\addtolength{\@tempdima}{-2in}
\setlength{\@tempdimb}{361\p@}
\setlength{\@tempdimb}{376\p@}
\setlength{\@tempdimb}{412\p@}

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).

\if@twocolumn
\ifdim\@tempdima>2\@tempdimb\relax
\setlength{\textwidth}{2\@tempdimb}
\else
\setlength{\textwidth}{\@tempdima}
\fi
\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

\textheight

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\,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 528\,pt.

\if@compatibility
\else
\setlength{\textheight}{43\baselineskip}
\setlength{\textheight}{38\baselineskip}
\setlength{\textheight}{36\baselineskip}
\fi

15
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.
\begin{verbatim}
\else
\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
The first line on the page has a height of \topskip.
\end{verbatim}

\subsection{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.
\begin{verbatim}
\marginparsep
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.
\end{verbatim}
\begin{verbatim}
\if@twocolumn
\setlength{\marginparsep}{10\p@}
\else
(10pt)\setlength{\marginparsep}{11\p@}
(11pt)\setlength{\marginparsep}{10\p@}
(12pt)\setlength{\marginparsep}{9\p@}
\fi
\end{verbatim}
\begin{verbatim}
\setlength{\marginparpush}{5\p@}
\end{verbatim}
\begin{verbatim}
\setlength{\marginparpush}{7\p@}
\end{verbatim}
\begin{verbatim}
\end{verbatim}
Now we can give the values for the other margin parameters. For native
\LaTeX, these are calculated.
\begin{verbatim}
\oddsidemargin
\evensidemargin
\marginparwidth
\end{verbatim}
\begin{verbatim}
\if@compatibility
\if@twoside
\setlength{\oddsidemargin}{44\p@}
\fi
\fi
\end{verbatim}
\begin{verbatim}
\setlength{\oddsidemargin}{44\p@}
\end{verbatim}

16
Values for one-sided printing:
\else
\if@twoside
\setlength\@tempdima \paperwidth
\addtolength\@tempdima \textwidth
\setlength\oddsidemargin \@tempdima
\addtolength\oddsidemargin \-1in
\setlength\evensidemargin \@tempdima
\addtolength\evensidemargin \-0.4in
\fi
\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 boek) we make the inner margin smaller than the outer one.
\else
\if@twoside
\setlength\tempdima \paperwidth
\addtolength\tempdima \textwidth
\setlength\oddsidemargin \@tempdima
\addtolength\oddsidemargin \-1in
\addtolength\evensidemargin \@tempdima
\addtolength\evensidemargin \-0.4in
\fi
\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 \-0.4in

For one-sided printing we center the text on the page, by calculating the difference between textwidth and \paperwidth. Half of that difference is then 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}
\fi

With the above algorithm the \marginparwidth can come out quite large which
we may not want.
\ifdim \marginparwidth >2in
\setlength\marginparwidth{2in}
\fi

Having done these calculations we make them pt values.
@settopoint\oddsidemargin
@settopoint\marginparwidth

The \evensidemargin can now be computed from the values set above.
\setlength\evensidemargin {\paperwidth}
\addtolength\evensidemargin{-2in}
\addtolength\evensidemargin{-\textwidth}
\addtolength\evensidemargin{-\oddsidemargin}

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.
@settopoint\evensidemargin
\fi
\topmargin
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.
@if@compatibility
\setlength\topmargin{27pt}
\else
\setlength\topmargin{\paperheight}
\addtolength\topmargin{-2in}
\addtolength\topmargin{-\headheight}
\addtolength\topmargin{-\headsep}
\addtolength\topmargin{-\textheight} % this might be wrong!
\addtolength\topmargin{-\footskip}
\fi

By changing the factor in the next line the complete page can be shifted vertically.
\addtolength\topmargin{-.5\topmargin}
@settopoint\topmargin
\fi
7.3.4 Footnotes

\footnotesep 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.

362 \setlength{\footnotesep}{6.65\p@}
363 \setlength{\footnotesep}{7.7\p@}
364 \setlength{\footnotesep}{8.4\p@}

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

365 \setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}
366 \setlength{\skip\footins}{10\p@ \@plus 4\p@ \@minus 2\p@}
367 \setlength{\skip\footins}{10.8\p@ \@plus 4\p@ \@minus 2\p@}
368 \setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}

7.3.5 Float placement parameters

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

Limits for the placement of floating objects

\c@topnumber The topnumber counter holds the maximum number of floats that can appear on the top of a text page.
369 \setcounter{topnumber}{2}
370 \renewcommand{\topfraction}{.7}

\c@bottomnumber The bottomnumber counter holds the maximum number of floats that can appear on the bottom of a text page.
372 \setcounter{bottomnumber}{1}
373 \renewcommand{\bottomfraction}{.3}

\c@totalnumber This indicates the maximum number of floats that can appear on any text page.
374 \setcounter{totalnumber}{3}
375 \renewcommand{\textfraction}{.2}
376 \renewcommand{\floatpagefraction}{.5}
The `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}

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}

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}

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 the main text and floats at the top or bottom of the 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@}

When floating objects that span the whole `\textwidth` are placed on a text page when we are in twocolumn 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 4\p@}
\setlength{\dbltextfloatsep}{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength{\intextsep}{14\p@ \@plus 4\p@ \@minus 4\p@}
Floats on their own page or column

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 \ldots \fil to allow filling the remaining empty space.

Double column floats in two column mode are handled with similar parameters.
7.4 Page Styles

The page style \texttt{foo} is defined by defining the command \texttt{\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).

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

\texttt{\thispagestyle} Several commands (\texttt{\index}, \texttt{\maketitle}) give a \texttt{\thispagestyle{plain}} command, which will overrule a \texttt{\pagestyle{empty}} command. This situation is almost always unwanted. Therefore we provide a more careful definition.

First save the original definition.

\let\Thispagestyle\thispagestyle

Then we provide the new definition, for which we must also adapt \texttt{\pagestyle} a little.

\newcommand*{\emptypagestyle}{empty}
\renewcommand*{\pagestyle}[1]{\@nameuse{ps@#1}\def\@currentpagestyle{#1}}
\renewcommand*{\thispagestyle}[1]{\ifx\@currentpagestyle\emptypagestyle\else\global\@specialpagetrue\gdef\@specialstyle{#1}\fi}

7.4.1 Marking conventions

To make headings determined by the sectioning commands, the page style defines the commands \texttt{\chaptermark}, \texttt{\sectionmark}, \ldots, where \texttt{\chaptermark{(TEXT)}} is called by \texttt{\chapter} to set a mark, and so on.

The \ldots\texttt{mark} commands and the \ldots\texttt{head} macros are defined with the help of the following macros. (All the \ldots\texttt{mark} commands should be initialized to no-ops.)

\texttt{\TeX} extends \texttt{\TeX}'s \texttt{\mark} facility by producing two kinds of marks, a ‘left’ and a ‘right’ mark, using the following commands:

\texttt{\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.

7.4.2 Defining the page styles

The pagestyle empty is defined in latex.dtx, but the pagestyle plain is slightly altered here. The difference is that the page numbers are set flush right in onesided and flush left and right in the twosided style.

\begin{verbatim}
\ps@plain
\renewcommand*{\ps@plain}{%  
  The running heads are empty in this pagestyle, the page number appears in the running foot.
  \let\@oddhead\@empty\let\@evenhead\@empty
  \def\@oddfoot{\hfil\PageFont\thepage}%
  \if@twoside
    \def\@evenfoot{\PageFont\thepage\hfil}%
  \else
    \let\@evenfoot\@oddfoot
  \fi
  Because the running heads should be empty we let \@mkboth to \@gobbletwo, thus disabling the mark commands.
  \let\@mkboth\@gobbletwo}
\end{verbatim}

\begin{verbatim}
\ps@headings
\if@twoside
  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{{\PageFont\thepage}\hfill\MarkFont\leftmark}%
  \def\@oddhead{{\MarkFont\rightmark}\hfill\PageFont\thepage}%
\else
\fi
\end{verbatim}
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 artikel document classes 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.


def\sectionmark##1{%
  \markboth {\MakeUppercase{\ifnum \c@secnumdepth >\z@ \thesection\quad \fi \##1}}%}
def\subsectionmark##1{%
  \markright {\MakeUppercase{\ifnum \c@secnumdepth >\@ne \thesubsection\quad \fi \##1}}}

In the rapport and boek document classes we use the `\chaptermark` and `\sectionmark` macros to fill the running heads.

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

\def\chaptermark##1{%
  \markboth {\MakeUppercase{\ifnum \c@secnumdepth >\m@ne \if@mainmatter \@chapapp\ \thechapter. \ % \fi \fi \##1}}%}
\def\sectionmark##1{%
  \markright {\MakeUppercase{\ifnum \c@secnumdepth >\z@ \thechapter. \ \fi \##1}}}%

The definition of `\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 `\@even...`.

\else
\def\ps@headings{%
  \let\@oddfoot\@empty
  \let\@evenfoot\@empty
  \let\@oddfoot\@empty
  \let\@evenfoot\@empty

24
We use `\markright` now instead of `\markboth` as we did for two sided printing.

The definition of the page style `myheadings` is fairly simple because the user determines the contents of the running head himself by using the `\markboth` and `\markright` commands.

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

These macros are to store the fonts that are used to typeset the pagenumber (`\PageFont`) and the marks (`\MarkFont`) in the running head and feet.

Use this macro to change the fonts that are used in the running heads.
8 Document Markup

8.1 The title

\title These three macros are provided by \texttt{latex.dtx} 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}
\newcommand*{\title}[1]{\gdef\@title{#1}}
\newcommand*{\author}[1]{\gdef\@author{#1}}
\newcommand*{\date}[1]{\gdef\@date{#1}}
\end{verbatim}

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

\begin{verbatim}
\gdef\@date{\today}
\end{verbatim}

\TitleFont This selects the font to use in the title of the document.

\begin{verbatim}
\newcommand*{\TitleFont}{\bfseries}
\end{verbatim}

\maketitle The definition of \texttt{\maketitle} depends on whether a separate title page is made. This is the default for the rapport and boek document classes, but for the artikel classes it is optional. Note that the title, author and date information is printed in capital letters by default. This can be changed by the option \texttt{mctitle}.

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.

\begin{verbatim}
\if@titlepage
\renewcommand*{\TitleFont}{\rmfamily}
\newcommand*{\maketitle}{% 
\begin{titlepage}{% 
\let\footnotesize\small
\let\footnoterule\relax
\let \footnote \thanks
\long\def\makefntext##1{\parindent\z@
\def\labelitemi{\textendash}\@revlabeltrue
\leavevmode\@textsuperscript{\@thefnmark}\kern1em\relax ##1}
\renewcommand*{\thefootnote}{\@fnsymbol{\c@footnote}}%
\end{titlepage}}%
\let\@makefntext\makefntext
\let\footnote \thanks
\end{verbatim}

Footnotes on the titlepage, generated by the use of \texttt{\thanks}, use symbols in these document classes.

\begin{verbatim}
\long\def\makefntext#1{\parindent\z%
\def\labelitemi{\textendash}\@revlabeltrue
\leavevmode\@textsuperscript{\@thefnmark}\kern1em\relax #1}
\renewcommand*{\thefootnote}{\@fnsymbol{\c@footnote}}%
\end{verbatim}

We center the entire title vertically; the centering is set off a little by adding a \texttt{vskip}. In compatibility mode the pagename is set to 0 to keep the behaviour of \LaTeX\ 2.09 style files

\begin{verbatim}
\if@compatibility\setcounter{page}{0}\fi
\null\vfil
\vskip 60\p@
\end{verbatim}
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.

\begin{center}
  \TitleFont
  {\LARGE \def\{"{\penalty-\@M}
  \if@allcaps
    \expandafter\uc@nothanks\@title\thanks\relax
  \else
    \@title
    \fi\par}\
  \vskip 3em
  {\large
    \lineskip .75em \parindent\z@
    \begin{tabular}[t]{c}
    \if@allcaps
      \expandafter\uc@authornothanks\@author\and\relax
    \else
      \@author
    \fi
    \end{tabular}
  \vskip 1.5em
  {\large
    \if@allcaps
      \uppercase\expandafter{\@date}\
    \else
      \@date
    \fi\par}
  \end{center}

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

\@thanks
\vfil\null
\end{titlepage}

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

\setcounter{footnote}{0}\
\global\let\thanks\relax
\global\let\maketitle\relax
\global\let\@thanks\@empty
\global\let\@author\@empty
\global\let\@title\@empty
\global\let\@date\@empty

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.

\global\let\title\relax
We want to have the title, author and date information in uppercase, but we have to be very careful not to put too much text in uppercase. The macros that perform the filtering of texts that shouldn’t be in uppercase were developed with the help of Howard Trickey.

\uc@nothanks

This macro takes all the text up to the first use of \thanks and passes it to \uppercase. The use of \futurelet will store the token after the \thanks in @\tempa. The macro \u@tx uses that information to determine what to do next.

\uc@authornothanks

A document can have more than one author. Usually they are separated with \and. For each author a footnote –using \thanks can be present. Therefore this macro takes all the text up to the first use of \and, thus picking up all the information for one author. This is then passed to \uc@nothanks, which checks for the presence of \thanks. For this to work the argument of \uc@nothanks has to be delimited by \thanks\relax.

\uc@ax

When @\tempa contains a \relax token nothing needs to be done, when it doesn’t we put in a linebreak \ \ the word ‘and’ (stored in \andname so that this control sequence can be redeined for other languages), another linebreak and we call \uc@authornothanks to continue processing. The \expandafter lets \TeX see the \fi first.

\uc@tx

This macro simply checks whether @\tempa contains a \relax token. When it doesn’t further processing is performed by \u@ty.

\uc@ty

The macro \uc@ty gets executed when the \thanks that delimited text earlier on in the processing had a real argument. In that case it was a \thanks put in by
the user, not by these macros. Therefore the argument is now passed to \thanks and processing continues by calling \uc@nothanks.

\def\uc@ty#1{\thanks{#1}\uc@nothanks}

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). We also want raised footnotemarkers in the footnotes here.

\maketitle

If this is a twocolumn document we start a new page in twocolumn 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 twocolumn 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.

\endgroup
\setcounter{footnote}{0}\%\global\let\thanks\relax\global\let\maketitle\relax\global\let\@maketitle\relax\global\let\@thanks\@empty\global\let\@author\@empty\global\let\@date\@empty\global\let\title\relax\global\let\author\relax\global\let\date\relax\global\let\and\relax
}\@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 in a \large font.
\def\@maketitle{\newpage\null\vskip 2em\langle\type3\rangle\if@titlecentered\begin{center}\let\footnote\thanks{\LARGE \TitleFont \@title \par}\vskip 1.5em{\large \TitleFont \lineskip .5em\begin{tabular}\[t\]{c}\@author\end{tabular}\par}\vskip 1em{\large \TitleFont \@date}\end{center}\langle\ast\type3\rangle\else{\LARGE \TitleFont \head@style \@title \par}\vskip 1.5em{\large \TitleFont \lineskip .5em\begin{tabular}\[t\]{l}\@author\end{tabular}\par}\vskip 1em{\large \TitleFont \@date}\fi\relax\{\LARGE \TitleFont \head@style \@title \par\vskip 1.5em{\large \TitleFont \lineskip .5em \tabcolsep\z@\def\and{\begin{tabular}\[t\]{l}}\end{tabular}\hskip 1em plus .17fil\begin{tabular}\[t\]{l}\@author\end{tabular}\par}\vskip 1em{\large \TitleFont \@date}\relax\{\LARGE \TitleFont \head@style \@title \par\vskip 1.5em{\large \TitleFont \lineskip .5em \tabcolsep\z@\begin{tabular}\[t\]{l}has already started\end{tabular}\hskip 1em plus .17fil\begin{tabular}\[t\]{l}will come\end{tabular}\@author\end{tabular}\par}\vskip 1em{\large \TitleFont \@date}\fi
8.2 Chapters and Sections

8.2.1 Building blocks

The definitions in this part of the class file make use of two macros, \@startsection and \secdef, which are defined by latex.dtx. 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\langle name\rangle\langle level\rangle\langle indent\rangle\langle beforeskip\rangle\langle afterskip\rangle\langle style\rangle optional *

\[\langle altheading\rangle\langle heading\rangle\]

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 <\= 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. Since the June 1996 release of \TeX the last command in this argument may be a command such as \MakeUppercase or \fbox that takes an argument. The section heading will be supplied as the argument to this command. So setting \#6 to, say, \bfseries\MakeUppercase would produce bold, uppercase headings.

* 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 be not 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 `\secdef` can be used when a sectioning command is defined without using `\@startsection`. It has two arguments:

`\secdef\textit{(unstarcmds)(starcmds)}`

\textit{(unstarcmds)} Used for the normal form of the sectioning command.

\textit{(starcmds)} Used for the \*\-form of the sectioning command.

You can use `\secdef` as follows:

```latex
\def\chapter { ... \secdef \CMDA \CMDB }
\def\CMDA [#1]#2{ ... } % Command to define \chapter[...]{}
\def\CMDB #1{ ... } % Command to define \chapter*{...}
```

\texttt{\head@style} In the definition of chapter and section commands a number of settings frequently occur. Therefore we store them in a control sequence.

Section headings are to be set extremely raggedright, with no hyphenations, not even at explicit hyphens.

```latex
\newcommand*{\head@style}{%\interlinepenalty \@M \hyphenpenalty=\@M \exhyphenpenalty=\@M \rightskip=0cm plus .7\hsize\relax}
```

\texttt{\@sect} The definition of this macro from \latex\texttt{.dtx} needs to be repeated here because we want to modify its behaviour with respect to:

1. the width of the number, which is fixed;
2. checking the value of `\unitindent`;
3. formatting the section title ragged right;
4. changing the argument of `\contentsline`.

```latex
\def\@sect#1#2#3#4#5#6[#7]#8{%\ifnum #2>\c@secnumdepth \let\@svsec\@empty \else \refstepcounter{#1} % \chapter[...]{}
```

The following code (within the group) checks the value of `\unitindent`. If the sectionnumber is wider than `\unitindent` its value is adapted and a flag is set to remember to store the new value in the `.aux`-file.
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.

This \{ used to be after the argument to \@hangfrom but was moved here to allow commands such as \MakeUppercase to be used at the end of \#6.

This macro was introduced in \LaTeX\ 2\epsilon, its definition is changed here to get the fixed with of the section number.
Similar changes need to be made to the definition of \@ssect, which is used in 'starred' sections.

\@tempskipa #3\relax
\ifdim \@tempskipa>\z@
\begingroup
This \ifdim used to be after the argument to \hangfrom but was moved here to allow commands such as \MakeUppercase to be used at the end of \#6.
\endgraf
\endgroup
\else
\def\@svsechd{#4\hskip #1\relax #5}\
\fi
\@xsect{#3}

8.2.2 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 7.4.2) Most of them are already defined by \latex\text, so they are only shown here.

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

These counters are used for the section numbers. The macro \newcounter defines \newctr to be a counter, which is reset to zero when counter \oldctr is stepped. Counter \oldctr must already be defined.

\newcounter {part}
\newcounter {chapter}
\newcounter {section}
\newcounter {subsection}
\newcounter {subsubsection}
\newcounter {paragraph}
\newcounter {subparagraph}

8.2.3 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}

These counters are used for the section numbers. The macro \newcounter defines \newctr to be a counter, which is reset to zero when counter \oldctr is stepped. Counter \oldctr must already be defined.

\newcounter {part}
\newcounter {chapter}
\newcounter {section}
\newcounter {subsection}
\newcounter {subsubsection}
\newcounter {paragraph}
\newcounter {subparagraph}
For any counter $CTR$, \texttt{\theCTR} is a macro that defines the printed version of counter $CTR$. It is defined in terms of the following macros:

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

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

\renewcommand*{\thepart}{\@Roman\c@part}
\renewcommand{\thesection}{\@arabic\c@section}
\renewcommand*{\thechapter}{\@arabic\c@chapter}
\renewcommand*{\thesubsection}{\thesection.\@arabic\c@subsection}
\renewcommand*{\thesubsubsection}{\thesubsection.\@arabic\c@subsubsection}
\renewcommand*{\theparagraph}{\thesubsubsection.\@arabic\c@paragraph}
\renewcommand*{\thesubparagraph}{\theparagraph.\@arabic\c@subparagraph}

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

8.2.4 Front Matter, Main Matter, and Back Matter

A boek contains these three sections. First, we define the switch \texttt{\@mainmatter} that is true if 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.

\frontmatter This command starts Roman page numbering and turns off chapter numbering.
This command clears the page, starts arabic page numbering and turns on chapter numbering.
\newcommand*{\mainmatter}{\cleardoublepage\@mainmattertrue \pagenumbering{arabic}}

This clears the page, turns off chapter numbering and leaves page numbering unchanged.
\newcommand*{\backmatter}{\if@openright\cleardoublepage\else\clearpage\fi \@mainmatterfalse}

8.2.5 Parts
\part The command to start a new part of our document.

In the artikel classes the definition of \part is rather simple; we start a new paragraph, add a little white space, suppress the indentation of the first paragraph (not for the artikel2 document class) and make use of \@secdef.
\newcommand*{\part}{\if@noskipsec \leavevmode \fi \par \addvspace{4ex} \afterindentfalse \@afterindenttrue \secdef\@part\@spart}

For the rapport and boek classes we things a bit different.
We start a new (righthand) page and use the empty pagestyle.
\newcommand*{\part}{\cleardoublepage \thispagestyle{empty}}

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 \@tempswatrue \else \@tempswafalse \fi

We need an empty box to prevent the fil glue from disappearing.
\null\vfil
Here we use \texttt{\textbackslash secd\textbackslash def} to indicate which commands to use to make the actual heading.

\begin{verbatim}
\secdef\@part\@spart}
\end{verbatim}

\texttt{\textbackslash @part} This macro does the actual formatting of the title of the part. Again the macro is differently defined for the artikel document classes than for the document classes rapport and boek.

\texttt{\textbackslash PartFont} The font used to typeset the part is stored in this macro.

\begin{verbatim}
\newcommand*{\PartFont}{\bfseries}
\end{verbatim}

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

\begin{verbatim}
\def\@part[#1]{%\ifnum \c@secnumdepth >\m@ne
\refstepcounter{part}\
\addcontentsline{toc}{part}{\protect\numberline{\thepart}#1}%
\else
\addcontentsline{toc}{part}{#1}%
\fi}{\head@style
\parindent\unitindent
\normalfont
\texttt{\textbackslash \Large\PartFont\noindent \partname\nobreakspace\thepart
\Large\PartFont\indent \partname\nobreakspace\thepart
\par}\nobreak}
\end{verbatim}

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

\begin{verbatim}
\texttt{\textbackslash \Large \PartFont \noindent #2%}
\end{verbatim}

Then we empty the mark registers, leave some white space and call \texttt{\textbackslash @afterheading} to take care of suppressing the indentation.

\begin{verbatim}
\texttt{\textbackslash markboth{}{}\par}%
\texttt{\nobreak}
\texttt{\vskip 3ex}
\texttt{\textbackslash @afterheading}
\end{verbatim}

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

When this is a numbered part we have to print the number. We have to expand \partname before \uppercase is called, therefore we use a temporary control sequence that, when called will execute \MakeUppercase on the contents of \partname.

We leave some space before we print the title and leave the finishing up to \@endpart.

\@spart 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 artikel document classes and in the rapport and boek document classes are similar as they were for \@part.
858 \( \text{\textcopyright{rapport | boek}} \)
859 \( \text{\def\@spart#1{\protect\centering\interlinepenalty\@M\normalfont\Large\partFont #1\par}} \)
860 \( \text{\@endpart} \)
861 \( \text{This macro finishes the part page, for both \part and \@spart.} \)
862 \( \text{First we fill the current page.} \)
863 \( \text{\def\@endpart{\vfil\newpage} \)\)
864 \( \text{Then, when we are in twosided mode and chapters are supposed to be on right} \)
865 \( \text{hand sides, we produce a completely blank page.} \)
866 \( \text{\if@twoside \)\)
867 \( \text{\if@openright \)\null\)\)
868 \( \text{\thispagestyle{empty}\)\newpage\)\fi \)\fi \)
869 \( \text{When this was a two column document we have to switch back to two column} \)
870 \( \text{mode.} \)
871 \( \text{\if@tempswa \)\twocolumn\)\fi} \)
872 \( \text{\chapter} \)
873 \( \text{A chapter should always start on a new page therefore we start by calling} \)
874 \( \text{\clearpage and setting the pagestyle for this page to plain.} \)
875 \( \text{\def\@chapter{\if@openright\cleardoublepage\else\clearpage\fi} \)\)
876 \( \text{\if@tempswa \)\twocolumn\)\fi} \)
877 \( \text{A chapter should always start on a new page therefore we start by calling} \)
878 \( \text{\clearpage and setting the pagestyle for this page to plain.} \)
879 \( \text{\def\@chapter{\if@openright\cleardoublepage\else\clearpage\fi} \)\)
880 \( \text{\if@tempswa \)\twocolumn\)\fi} \)
881 \( \text{Then we prevent floats from appearing at the top of this page because it looks} \)
882 \( \text{weird to see a floating object above a chapter title.} \)
883 \( \text{\global\@topnum\z@} \)
884 \( \text{Then we suppress the indentation of the first paragraph by setting the switch} \)
885 \( \text{\@afterindent to false. We use \seccdef to specify the macros to use for actually} \)
886 \( \text{setting the chapter title.} \)
887 \( \text{\def\@afterindentfalse} \)
888 \( \text{\seccdef{\chapter}{@chapter}} \)
889 \( \text{This macro is called when we have a numbered chapter. When secnumdepth is} \)
890 \( \text{larger than \textendash{}1 and, in the boek class, \@mainmatter is true, we display the chapter} \)
number. We also inform the user that a new chapter is about to be typeset by writing a message to the terminal.

883 \def\@chapter[#1]{%  
884 \ifnum \c@secnumdepth >\m@ne  
885 \refstepcounter{chapter}  
886 \typeout{\@chapapp\space\thechapter.}  
887 \addcontentsline{toc}{chapter}{\protect\numberline{\thechapter}#1}  
888 \ifboek  
889 \else  
890 \addcontentsline{toc}{chapter}{#1}  
891 \fi  
892 \else  
893 \addcontentsline{toc}{chapter}{#1}  
894 \fi}

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.

898 \chaptermark{#1}  
899 \addtocontents{lof}{\protect\addvspace{10\p@}}  
900 \addtocontents{lot}{\protect\addvspace{10\p@}}

Then we call upon \@makechapterhead to format the actual chapter title. We have to do this in a special way when we are in twocolumn 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.

901 \if@twocolumn  
902 \@topnewpage[\@makechapterhead{#2}]  
903 \else  
904 \@makechapterhead{#2}  
905 \@afterheading  
906 \fi}

\ChapFont The font used to typeset the chapters is stored in this macro.

907 \newcommand*\ChapFont{\textbf}

\@makechapterhead The macro above uses \@makechapterhead{\text} 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.

908 \def\@makechapterhead#1{  
909 \ifboek  
910 \vspace{(50)\p@ \dplus 5\p@}  
911 \else  
912 \vspace{(50)\p@ \dplus 20\p@}  
913 \setlength\parindent{=0}%  
914 \setlength\parskip {=0}%  
915 \head@style \normalfont  
916 \head@style \normalfont
Then we check whether the number of the chapter has to be printed. If so we leave some whitespace between the chapter number and its title.

```
\ifnum \c@secnumdepth >\m@ne
  \if@mainmatter
    \Large\ChapFont \@chapapp{} \thechapter
    \par\nobreak
    \vskip 20\p@
  \fi
\fi
```

Now we set the title in a large bold font. We prevent a pagebreak at this point and leave some whitespace before the text begins.

```
\Large \ChapFont #1\par
\nobreak
\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 to format the heading of the chapter. It is similar to \@makechapterhead except that it never has to print a chapter number.

```
\def\@makeschapterhead#1{%
  \vspace*{50\p@\@plus 5\p@}%
  \vspace*{50\p@\@plus 20\p@}%
  {\setlength\parindent{\z@}%
    \setlength\parskip{\z@}%
    \head@style
    \normalfont
    \Large \ChapFont #1\par
    \nobreak
    \vskip 40\p@
  }}
```

### 8.2.7 Lower level headings

These commands all make use of \@startsection.  

```
\section
```

This gives a normal heading with white space above the heading (the whitespace below the heading will be generated by the \texttt{\parskip} that is inserted at the start
of the first paragraph), the title set in \large\bfseries, and no indentation on
the first paragraph.
\newcommand*{\section}{
\@startsection {section}{1}{\z@}{-2\baselineskip\@plus -1\baselineskip \@minus -.5\baselineskip}{\@type 1}|\@type 3}{
\@startsection {section}{1}{\unitindent}{2\baselineskip\@plus .5\baselineskip \@minus .25\baselineskip}{\@type 2}{\@type 1}{.25\baselineskip}{\@type 2}|\@type 3}{.01\baselineskip}{\normalfont\large\SectFont}}
\SectFont The font used to typeset the sections is stored in this mako.
\newcommand*{\subsection}{
\@startsection{subsection}{2}{\z@}{-1\baselineskip plus -.5\baselineskip minus -.25\baselineskip}{\@type 1}|\@type 3}{\@startsection{subsection}{2}{\unitindent}{1\baselineskip plus .5\baselineskip minus .25\baselineskip}{\@type 2}{\@type 1}{.25\baselineskip}{\@type 2}|\@type 3}{.01\baselineskip}{\normalfont\normalsize\SSectFont}}
\SSectFont The font used to typeset the subsections is stored in this mako.
\newcommand*{\subsubsection}{
\@startsection{subsubsection}{3}{\z@}{-1\baselineskip plus -.5\baselineskip minus -.25\baselineskip}{\@type 1}|\@type 3}{\@startsection{subsubsection}{3}{\unitindent}{1\baselineskip plus .5\baselineskip minus .25\baselineskip}{\@type 2}{\@type 1}{.25\baselineskip}{\@type 2}|\@type 3}{.01\baselineskip}{\normalfont\normalsize\SSSectFont}}
\SSSectFont The font used to typeset the sub-subsections is stored in this mako.
\SSSectFont  The font used to typeset the subsubsections is stored in this maro.

\paragraph  This gives a run-in heading with white space above and to the right of the heading, the title set in \normalsize\slshape.

\ParaFont  The font used to typeset the paragraphs is stored in this maro.

\subparagraph  This gives an indented run-in heading with white space above and to the right of the heading, the title set in \normalsize\slshape.

\Headingfonts  To change the fonts that are used to typeset the title, part, chapter and section headings this macro can be used.

\HeadingFonts  \newcommand*{\HeadingFonts}{\{7\}}
\newcommand*{\TitleFont}{#1}
\newcommand*{\PartFont}{#2}
\newcommand*{\SectFont}{#3}
\newcommand*{\SSectFont}{#4}
\newcommand*{\SSSectFont}{#5}
\newcommand*{\ParaFont}{#6}
\newcommand*{\SParaFont}{#7}

\HeadingFonts  \newcommand*{\HeadingFonts}{\{8\}}
\newcommand*{\TitleFont}{#1}
\newcommand*{\PartFont}{#2}
\newcommand*{\SectFont}{#3}
\newcommand*{\SSectFont}{#4}
\newcommand*{\SSSectFont}{#5}
8.3 Lists

8.3.1 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, \rightmargin, \listparindent and \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}
\texttt{\leftmargini \leftmarginii \leftmarginiii \leftmarginiv \leftmarginv \leftmarginvi}
\end{verbatim}

Here we set the top level leftmargin.

\texttt{\setlength{\labelsep} {5\p@}}

\texttt{\setlength{\labelwidth}{\leftmargini}}

\texttt{\addtolength{\labelwidth}{-\labelsep}}

\texttt{\setlength{\partopsep}{\z@}}

\texttt{\setlength{\topsep}{\z@}}

\texttt{\@beginparpenalty}

\texttt{\@endparpenalty}

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

\texttt{\@beginparpenalty}

\texttt{\@endparpenalty}

They are set to a bonus value to encourage page breaking at these points.
\itempenalty This penalty is inserted between list items.

1033 \beginparpenalty \- \endparpenalty
1034 \- \lowpenalty
1035 \- \lowpenalty

\listi \listi defines values of \leftmargin, \parsp, \topsep, and \itemsep, etc.
\listI 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.

1036 \def\listi{\%
1037 \type2 \leftmargin \unitindent
1038 \type2 \leftmargini \leftmargini
1039 \type2 \labsep .5em\%
1040 \type2 \labsep .45em\%
1041 \labelwidth \leftmargini
1042 \advance \labelwidth \- \labsep
1043 \parsp \z@ \topsep 0\p@ \@plus\p@
1044 \topsep -.5\parskip \@plus\p@
1045 \parsp \z@ \itemsep \z@ \@plus\p@}
1046 \let \listI \listi
1047 \%

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

1048 \listi

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

\listv 1049 \def\listii {\leftmargini \leftmarginii
1050 \type2 \labsep .5em\%
1051 \type2 \labsep .3em\%
1052 \labelwidth \leftmarginii
1053 \advance \labelwidth \- \labsep
1054 \topsep 0\p@ \@plus\p@ \topsep -.5\parskip \@plus\p@
1055 \parsp \z@ \itemsep \z@ \@plus\p@}
1056 \def\listiii {\leftmargini \leftmarginiii
1057 \type2 \labsep .5em\%
1058 \type2 \labsep .3em\%
1059 \labelwidth \leftmarginiii
1060 \advance \labelwidth \- \labsep
1061 \topsep 0\p@ \@plus\p@ \topsep -.5\parskip \@plus\p@
1062 \parsp \z@ \itemsep \z@ \@plus\p@}
1063 \def\listiv {\leftmargini \leftmarginiv
1064 \type2 \labsep .5em\%
1065 \type2 \labsep .3em\%
1066 \labelwidth \leftmarginiv
1067 \advance \labelwidth \- \labsep
1068 \topsep 0\p@ \@plus\p@ \topsep -.5\parskip \@plus\p@
1069 \parsp \z@ \itemsep \z@ \@plus\p@}
1070 \itemsep \z@ \@plus\p@}
1071 \itemsep \z@ \@plus\p@
8.3.2 Enumerate

The enumerate environment uses four counters: \texttt{enumi}, \texttt{enumii}, \texttt{enumiii} and \texttt{enumiv}, where \texttt{enumN} controls the numbering of the \texttt{N}th level enumeration.

\begin{verbatim}
\theenumi \theenumii \theenumiii \theenumiv
\end{verbatim}

The countes are already defined in \texttt{latex.dtx}, but their representation is changed here.

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

The label for each item is generated by the commands

\begin{verbatim}
\labelenumi \labelenumii \labelenumiii \labelenumiv
\end{verbatim}

The expansion of \texttt{\p@enum\theenumN} defines the output of a \texttt{\ref} command when referencing an item of the \texttt{N}th level of an enumerated list.

\begin{verbatim}
\p@enumi 
\p@enumii 
\p@enumiii 
\p@enumiv
\end{verbatim}
enumerate We want to have different label positioning on different levels of list. To achieve this we have to redefine the enumerate environment.

\begin{verbatim}
\renewenvironment{enumerate}{%  
  \ifnum\@enumdepth>\thr@@  \@toodeep  \else  \advance\@enumdepth\@ne  \edef\@enumctr{enum\romannumeral\the\@enumdepth}%  \expandafter\list\csname label\@enumctr\endcsname  \usecounter{\@enumctr}⟨\textbullet⟩  \ifnum\@listdepth=1  ⟨∗\textbullet⟩  \if@revlabel  \def\makelabel##1{hskip.5\unitindent{##1\hfil}}%  \else  ⟨!\textbullet⟩  \def\makelabel##1{\hfil##1}⟨\textbullet⟩  \def\makelabel##1{##1\hfil}  \fi  ⟨/\textbullet⟩  \else  ⟨\textbullet\textendash⟩  \else  ⟨\textbullet\textperiodcentered⟩  \fi\endlist}
\end{verbatim}

We try to suppress spaces after these list constructs.

\begin{verbatim}
{\global\@ignoretrue \endlist}
\end{verbatim}

8.3.3 Itemize

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

\begin{verbatim}
\newcommand\labelitemi{\labelitemfont\textbullet}
\newcommand\labelitemii{\labelitemfont\bfseries\textendash}
\newcommand\labelitemiii{\labelitemfont\textasteriskcentered}
\newcommand\labelitemiv{\labelitemfont\textperiodcentered}
\end{verbatim}

\labelitemfont The default definition for \labelitemfont is to reset the font to \normalfont so that always the same symbol is produced regardless of surrounding conditions.

A possible alternative would be
which resets series and shape doesn’t touch the family.

\texttt{\newcommand\labelitemfont{\normalfont}}

\textbf{itemize} We want to have different label positioning on different levels of list. To achieve this we have to redefine the \texttt{itemize} environment.

\texttt{\renewenvironment{itemize}{...}}

We try to suppress spaces after these list constructs.

\texttt{\descriptionlabel} To change the formatting of the label, you must redefine \texttt{\descriptionlabel}.

\texttt{\newcommand*\descriptionlabel[1]{\hspace{\labelsep}\normalfont\bfseries #1}}
8.4 Adapting existing environments

Because we globally set \topsep to zero, we need to modify the definitions of a number of environments slightly to get a little whitespace around them in the document classes artikel1 and rapport1.

**center**  Add a little surrounding whitespace.

```
1166 +(type1)
1167 \def\center
1168 {\topsep=.25\baselineskip ++ .1\baselineskip
1169 \@minus .1\baselineskip
1170 \trivlist \centering\item[]}
1171 \let\endcenter\endtrivlist
```

**flushleft**  Add a little surrounding whitespace.

```
1172 \def\flushleft
1173 {\topsep=.25\baselineskip ++ .1\baselineskip
1174 \@minus .1\baselineskip
1175 \trivlist \raggedright\item[]}
1176 \let\endflushleft=\endtrivlist
```

**flushright**  Add a little surrounding whitespace.

```
1177 \def\flushright
1178 {\topsep=.25\baselineskip ++ .1\baselineskip
1179 \@minus .1\baselineskip
1180 \trivlist \raggedleft\item[]}
1181 \let\endflushright=\endtrivlist
1182 /+(type1)
```

**verbatim**  In verbatim we add a little surrounding whitespace, \textemdash which for artikel3 and rapport3 is negative to compensate for the positive \parskip \textemdash but also an indent for the artikel1 and rapport1 document classes.

```
1183 \def\verbatim%+ 
1184 +(type1 | type2)
1185 \topsep=.25\baselineskip ++ .1\baselineskip
1186 \@minus .1\baselineskip
1187 \@verbatim
1188 /+(type1 | type2)
1189 (type1) \leftskip\unitindent
1190 (type2) \leftskip\z@ 
1191 +(type3)
1192 \topsep=−.5\parskip
1193 \@verbatim
1194 /+(type3)
1195 \frenchspacing\@obeyspaces \@verbatim
1196 (type1) \def\endverbatim\if@newlist \leavevmode \fi\endtrivlist
```

49
8.5 Defining new environments

8.5.1 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.

When we are not making a separate titlepage—the default for the artikel document classes—we have to check if we are in twocolumn mode. In that case the abstract is set as a \section*, otherwise the abstract is typeset flushleft, an amount \unitindent smaller as the normal text.

As always, the artikel2 document class has a different implementation.

Which implies that the definition of \end{abstract} is also different.
8.5.2 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{verse}
\let\@centercr
\list{}\{\itemsep\z@\itemindent-1.5em\%\listparindent\itemindent\rightmargin\leftmargin\advance\leftmargin1.5em\%\item\relax\endlist
\end{verse}

8.5.3 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{quotation}
\list{}{\langle\!\!\type2\!\!\rangle\listparindent\z@\langle\!\!\type2\!\!\rangle\listparindent\unitindent\langle\!\!\boek\!\!\rangle\listparindent1.5em\%\itemindent\listparindent\rightmargin\leftmargin\parsep\z@ \@plus\p@\%\item\relax\endlist
\end{quotation}

8.5.4 Quote

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

\begin{quote}
\list{}{\rightmargin\leftmargin}\item\relax\endlist
\end{quote}

8.5.5 Theorem

These document classes have a slightly modified theorem environment style. Surrounding whitespace is added and an initialisation of `\labelsep`. Finally a slanted font instead of an italic font is used.

\begin{verbatim}
\def\@begintheorem#1#2{%
\vskip\baselineskip \labelsep=.5em\%
\trivlist\begin{itemize}
\item\relax\end{itemize}\\%\end{verbatim}

51
8.5.6 Titlepage

titlepage In the normal environments, the titlepage environment does nothing but start and end a page, and inhibit page numbers. It also resets 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. In two-column style, it still makes a one-column page.

8.5.7 Appendix

\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 artikel document classes the \appendix command must do the following:

- reset the section and subsection counters to zero,
- redefine \thesection to produce alphabetic appendix numbers.
In the rapport and boek 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.

8.6 Setting parameters for existing environments

8.6.1 Array and tabular

\arraycolsep The columns in an array environment are separated by 2\arraycolsep.
1296 \setlength\arraycolsep{5\p@}
\tabcolsep The columns in a tabular environment are separated by 2\tabcolsep.
1297 \setlength\tabcolsep{6\p@}
\arrayrulewidth The width of rules in the array and tabular environments is given by \arrayrulewidth.
1298 \setlength\arrayrulewidth{.4\p@}
\doublerulesep The space between adjacent rules in the array and tabular environments is given by \doublerulesep.
1299 \setlength\doublerulesep{2\p@}

8.6.2 Tabbing

\tabbingsep This controls the space that the \' command puts in. (See \LaTeX\ manual for an explanation.)
1300 \setlength\tabbingsep{\labelsep}

8.6.3 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.
1301 \def\minipagerestore{\parindent\unitindent}
1302 \def\minipagerestore{\parindent\unitindent}
1303 \def\@minipagerestore{\%
1304 \parskip=.5\baselineskip \@plus .1\baselineskip
1305 \@minus .1\baselineskip\}
1306 (/type3)
\@mpfootins Minipages have their own footnotes; \skip\@mpfootins plays same rôle for foot-
notes in a minipage as \skip\footins does for ordinary footnotes.
1307 \skip\@mpfootins = \skip\footins

8.6.4 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.
1308 \setlength\fboxsep{3\p@}
1309 \setlength\fboxrule{.4\p@}

8.6.5 Equation and eqnarray
\theequation When within chapters, the equation counter will be reset at 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.
1310 (artikel)\renewcommand*{\theequation{\@arabic\c@equation}}
1311 (+rapport + boek)
1312 \@addtoreset{equation}{chapter}
1313 \renewcommand*{\theequation{%
1314 \ifnum \c@chapter >\z@ \thechapter. \fi\@arabic\c@equation}
1315 (/rapport + boek)
\jot \jot is the extra space added between lines of an eqnarray environment. The
default value is used.
1316 \% \setlength{\jot}{3pt}
\@eqnnum The macro \@eqnnum defines how equation numbers are to appear in equations.
Again the default is used.
1317 \% \def\@eqnnum{\{\theequation\}}

8.7 Floating objects
The file latex.dtx 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 TYPE (e.g., TYPE = figure).
\fps@TYPE The default placement specifier for floats of type TYPE.
\ftype@TYPE The type number for floats of type TYPE. Each TYPE has associated
a unique positive 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.
The file extension indicating the file on which the contents list for float type TYPE is stored. For example, \texttt{\textbackslash 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 \texttt{latex.dtx}.

An environment that implements a single column floating object is started with \texttt{\@float{TYPE}}\texttt{[⟨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 == \@floatfigure, \endfigure == \end@float}.

8.7.1 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 rapport and boek document classes the figures are numbered per chapter.

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

\texttt{\type@figure} And the definition of the actual environment. The form with the * is used for double column figures.
8.7.2 Table

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

\c@table First we have to allocate a counter to number the tables. In the rapport and boek document classes the tables are numbered per chapter.

\newcounter{table}
\renewcommand*{\thetable}{{@arabic}{c@table}}
\fps@table
\ftype@table
\ext@table
\num@table

Here are the parameters for the floating objects of type ‘table’.
\def\fps@table{tbp}
\def\ftype@table{2}
\def\ext@table{lot}
\def\fnum@table{\tablename\nobreakspace\thetable}

And the definition of the actual environment. The form with the * is used for double column tables.
\newenvironment{table}
{\@float{table}}
{\end@float}
\newenvironment{table*}
{\@dblfloat{table}}
{\end@dblfloat}

8.7.3 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 \textit{number} contains a string such as ‘Figure 3.2’. The macro can assume it is called inside a \parbox of right width, with \normalsize.

These lengths contain the amount of white space to leave above and below the caption.
\newlength{abovecaptionskip}
\newlength{belowcaptionskip}
\setlength{abovecaptionskip}{10\p@}
\setlength{belowcaptionskip}{0\p@}

56
The definition of this macro is \texttt{\long} in order to allow more than one paragraph in a caption.

\begin{verbatim}
def \@makecaption#1#2{
    \vskip \abovecaptionskip
    \sbox \@tempboxa{{\CaptionLabelFont#1:} \CaptionTextFont#2}\
    \ifdim \wd \@tempboxa > \hsize
        \CaptionLabelFont#1: \CaptionTextFont#2\par
    \else
        \global \@minipagefalse
        \hb@xt@\hsize{\hfil \box \@tempboxa \hfil}\
    \fi
    \vskip \belowcaptionskip}
\end{verbatim}

These macros can contain the fonts used for typesetting captions. By default they do nothing.

\begin{verbatim}
captionlabelfont \captiontextfont
\end{verbatim}

These macros may be used to change the fonts that are used to typeset captions.

\begin{verbatim}
\captionlabelfont\captiontextfont
\end{verbatim}

\section{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{\DeclareOldFontCommand}, 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 \LaTeX2.09 behaviour.
\end{verbatim}

\begin{verbatim}
\sf \DeclareOldFontCommand{\rm}{\normalfont \rmfamily}{\mathrm}
\end{verbatim}

\begin{verbatim}
\tt \DeclareOldFontCommand{\sf}{\normalfont \sffamily}{\mathsf}
\end{verbatim}

\begin{verbatim}
\tt \DeclareOldFontCommand{\tt}{\normalfont \ttfamily}{\mathsf}
\end{verbatim}

\begin{verbatim}
\end{verbatim}
The command to change to the bold series. One should use `\mdseries` to explicitly switch back to medium series.

```
\bf
```

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. One should use `\upshape` to explicitly change back to the upright shape.

```
\sl
\it
\sc
```

The commands `\cal` and `\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’.

```
\cal
\mit
```

The definition of `\em` is changed here to have slanted instead of italic fonts.

```
\em
```

Cross Referencing

9.1 Table of Contents, etc.

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

A `\caption` command in a ‘figure’ environment writes

```
\contentsline{figure}{\numberline{⟨num⟩}{⟨caption⟩}}{⟨page⟩}
```

on the `.lof` file, where ⟨num⟩ is the number produced by `\thefigure` and ⟨caption⟩ is the figure caption. It works similarly for a ‘table’ environment.

The command `\contentsline{⟨name⟩}` expands to \l@⟨name⟩. So, to specify the table of contents, we must define \l@chapter, \l@section, \l@subsection, ... ; to specify the list of figures, we must define \l@figure; and so on. Most of these can be defined with either the `\dottedtocline` or the `\regtocline` command, which work as follows.
An entry is produced only if \( \text{level} \leq \text{value of the tocdepth counter.} \)
Note, \texttt{\chapter} is level 0, \texttt{\section} is level 1, etc.

The indentation from the outer left margin of the start of the contents line.

The width of a box in which the section number is to go, if \texttt{\title} 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).

The width of a box in which the page number is put.

The right margin for multiple line entries. One wants \@tocrmarg \( \geq \) \@pnumwidth

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}

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

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

The \texttt{\tableofcontents} This macro is used to request that \LaTeX{} produces a table of contents. In the rapport and boek document classes the tables of contents, figures etc. are always set in single-column style.

The title is set using the \texttt{\chapter*} command, making sure that the running head –if one is required– contains the right information.
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	extsubscript{2.09} version as it will produce a serious bug if used in two-column mode (see, \LaTeX	extsubscript{2.09} 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.

\begin{verbatim}
\@mkboth{\MakeUppercase{\contentsname}}% \MakeUppercase{\contentsname}%
\end{verbatim}

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

\begin{verbatim}
\@starttoc{toc}\
\@starttoc{toc}
\langle !\texttt{artikel} \rangle \if@restonecol\twocolumn\fi
\}
\end{verbatim}

\@starttoc The internal \LaTeX	extsubscript{2} macro \@starttoc needs to be adapted for the \texttt{artikel3} and \texttt{rapport3} document classes, in order to deal with a the fact that for these document classes the \texttt{parskip} is normally non-zero. We don’t want that in the table of contents.

\begin{verbatim}
\@starttoc{toc}(	exttt{begin})\end{verbatim}

\@regtocline These document classes use a different format for the table of contents than the standard classes from which they were developed. In order to achieve this different format we defined the macro \@regtocline.

\begin{verbatim}
\@regtocline{3}{\texttt{begingroup}}\makeatletter\parskip\z@\@input{\jobname.#1}\
\if@filesw\expandafter\newwrite\csname tf@#1\endcsname\immediate\openout\csname tf@#1\endcsname\jobname.#1\relax\fi\nobreakfalse\endgroup}
\end{verbatim}

\@regtocline This macro defines a different format for the table of contents. In order to achieve this different format we defined the macro \@regtocline.

\begin{verbatim}
\@regtocline[3]{\texttt{begingroup}}\makeatletter\parskip\z@\@input{\jobname.#1}\
\if@filesw\expandafter\newwrite\csname tf@#1\endcsname\immediate\openout\csname tf@#1\endcsname\jobname.#1\relax\fi\nobreakfalse\endgroup}
\end{verbatim}

\@regtocline These document classes use a different format for the table of contents than the standard classes from which they were developed. In order to achieve this different format we defined the macro \@regtocline.

\begin{verbatim}
\@regtocline[3]{\texttt{begingroup}}\makeatletter\parskip\z@\@input{\jobname.#1}\
\if@filesw\expandafter\newwrite\csname tf@#1\endcsname\immediate\openout\csname tf@#1\endcsname\jobname.#1\relax\fi\nobreakfalse\endgroup}
\end{verbatim}

\@regtocline This macro defines a different format for the table of contents. In order to achieve this different format we defined the macro \@regtocline.
This internal macro is redefined for the \texttt{article2} document class.

The changed definition of \texttt{@sect} that we use, selects a different font for the table of contents for the various header levels. It does this using \texttt{toc@font}.

A line of the table of contents contains \texttt{numberline} and the section number as its first two elements. We don’t want to set the section number using \texttt{toc@font}, therefore we give it two additional arguments and pass them on first, before changing the font. Note that we need to re-insert the braces around the second argument.

When the user wants to produce a hyper-document using \texttt{hyperref} we need to take special precautions to make it work for the table of contents. We check for the existence of \texttt{hyper@linkstart} to detect this situation at \texttt{\begin{document}}.

\texttt{Hyperref} injects extra tokens (\texttt{\hyper@linkstart\{link\}\texttt{Hy@tocdestname}}) into the stream in front of the real contentsline. The command \texttt{\hyper@linkstart} and its arguments need to be protected from expanding too early or being “uppercased” themselves.

In the contentslines for chapters, sections etc., the command selection of the appropriate font needs to come after the code that \texttt{hyperref} injects. We do this with some argument shuffling.
\let\ORG@hyper@linkstart\hyper@linkstart
\protected\def\hyper@linkstart#1#2{%\lowercase{\ORG@hyper@linkstart{#1}{#2}}}\fi
\fi
\toc@case
In the rapport and boek document classes, the entries for parts are typeset in capital letters in the new style of the table of contents. In the old style this isn’t done. The macro \toc@case is used to switch this.

\if@oldtoc
\newcommand*{\toc@case}{\relax}
\else
\newcommand*{\toc@case}{\MakeUppercase}
\fi
\l@part
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.

First we have the definition from the standard classes.

\if@oldtoc
\newcommand*{\l@part}[2]{%\ifnum \c@tocdepth >-2\relax ⟨artikel⟩ \addpenalty\@secpenalty
⟨artikel⟩ \addpenalty{-\@highpenalty} \addvspace{2.25em @plus\p@} \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 put it there.
\setlength{\@tempdima}{3em}%
The we set \parindent to 0pt and use \rightskip to leave enough room for the pagenumbers. 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 pagonenumber, set flush right.
\leavevmode{\large \bfseries #1\hfil \hb@xt@0\@pnumwidth{\hss #2} \kern-\p@\kern\p@}}\par
Prevent a pagebreak immediately after this entry, but use \everypar to reset the \if@nobreak switch. Finally we close the group.
\nobreak
62
Then we can introduce our new definition.

\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.

Again we first present the ‘standard’ definition

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 put it there. We begin a group, and change some of the paragraph parameters.

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

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.
Then we present our new definition.

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:

\newcommand*\l@section[2]{\@secpenalty 500 \addvspace{1.0em} #1
\begingroup \parindent 0pt \rightskip \@pnumwidth \parfillskip -\@pnumwidth #1
\nobreak\hfil \nobreak \par}

The new definition:
All lower level entries are defined using the macro `\dottedtocline` or `\regtocline` (see above).

```
\newcommand*{\l@subsection}{\dottedtocline{2}{1.5em}{2.3em}}
\newcommand*{\l@subsubsection}{\dottedtocline{3}{3.8em}{3.2em}}
\newcommand*{\l@paragraph}{\dottedtocline{4}{7.0em}{4.1em}}
\newcommand*{\l@subparagraph}{\dottedtocline{5}{10em}{5em}}
```

9.1.2 List of figures

This macro is used to request that \LaTeX produces a list of figures. It is very similar to `\tableofcontents`.

```
\newcommand*{\listoffigures}{% 
\if@twocolumn
@restonecoltrue \onecolumn
\else
@restonecolfalse
\fi
\chapter*{\listfigurename}%
\@mkboth{\MakeUppercase{\listfigurename}}{\MakeUppercase{\listfigurename}}
\@starttoc{lof}
\fi
```

```
\newcommand*{\l@figure}{\dottedtocline{1}{1.5em}{2.3em}}
\else
\newcommand*{\l@figure}{\regtocline{1}}
\fi
```

65
9.1.3 List of tables

\listoftables\ This macro is used to request that \LaTeX\ produces a list of tables. It is very similar to \tableofcontents.

\newcommand\listoftables{\%\if@twocolumn\@restonecoltrue\onecolumn\else\@restonecolfalse\fi\chapter*{\listtablename}\%\@mkboth{\MakeUppercase\listtablename}{\MakeUppercase\listtablename}\%\@starttoc{lot}\if@restonecol\twocolumn\fi}This macro produces an entry in the list of tables.

\let\l@table\l@figure

9.2 Bibliography

\bibindent\ The “open” bibliography format uses an indentation of \bibindent.
\newdimen\bibindent\setlength\bibindent{1.5em}\newblock This is a dummy definition for this macro which is used in the \bibliography environment.

\newcommand\newblock{}\thebibliography\ The ‘thebibliography’ environment executes the following commands: \renewcommand\newblock{\hspace{.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\textperiodcentered=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 \enumerate counter internally to generate the labels of the list. When an empty ‘thebibliography’ environment is found, a warning is issued.
The default definition for `\newblock` is to produce a small space.

```
% \changes{v2.0t}{1996/04/01}{use \cs{renewcommand} instead of \cs{newcommand}}
\renewcommand\newblock{\hskip.11em\@plus.33em\@minus.07em}
```

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

```
\let\@openbib@code@empty
```

The label for a `\bibitem[...]` command is produced by this macro. The default from `latex.dtx` is used.

```
% \renewcommand*{\@biblabel[1]{[#1]\hfill}}
```

The output of the `\cite` command is produced by this macro. The default from `latex.dtx` is used.

```
% \renewcommand*{\cite[1]{[#1]}}
```

### 9.3 The index

`theindex` 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 `\item`, `\subitem` and `\subsubitem` are used to produce index entries of various levels. When a new letter of the alphabet is encountered an amount of `\indexspace` white space can be added.

```
\newenvironment{theindex}{%\relax
% \if@twocolumn
\renewcommand{\@restonecolfalse}
% \else
\renewcommand{\@restonecoltrue}
% \fi
\end{list}}{\endlist}
```

The default definition for `\@biblabel` is used.
Parameter changes to \texttt{\textbackslash columnseprule} and \texttt{\textbackslash columns} have to be done after \texttt{\textbackslash twocolumn} has acted. Otherwise they can affect the last page before the index.

\begin{verbatim}
\columnseprule \z@
\columns 35\p@
\parskip \z@ \@plus .3\p@ \relax
\let\item\@idxitem
\newcommand\@idxitem {
\par \hangindent 40\p@}
\newcommand\subitem {
@idxitem \hspace*{20\p@}}
\newcommand\subsubitem{
@idxitem \hspace*{30\p@}}
\indexspace
\end{verbatim}

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

\begin{verbatim}
\renewcommand\footnoterule{\kern-3\p@\langle \texttt{\textbackslash type1} \textbar \texttt{\textbackslash type3}\rangle\hrule \@width \unitindent\kern .4\baselineskip\langle \texttt{\textbackslash type2}\rangle\hrule \@width 3\unitindent\kern 2.6\p@}
\end{verbatim}

\footnoterule

\footnoterule

9.4 Footnotes

This rule is drawn by the macro \texttt{\textbackslash footnoterule}. The standard \LaTeX{} document classes make sure that the rule takes no vertical space (see \texttt{plain.tex}) and compensate for the natural height of the rule of 0.4pt by adding the right amount of vertical skip. For the \texttt{artikel2} document class this is still true, but for the others the amount of whitespace between the last line of the text and the start of the footnotes is increased by giving \texttt{\textbackslash footnoterule} a positive height\footnote{This should perhaps have been done by increasing the value of \texttt{\textbackslash skip\textbackslash footins}, but changing that now would mean changing the formatting of existing documents. (JLB, 08/09/1997)}.

To prevent the rule from colliding with the footnote we first add a little negative vertical skip, then we put the rule and add some positive vertical skip.

\begin{verbatim}
\renewcommand\footnoterule{
\kern-3\p@
\langle \texttt{\textbackslash type1} \textbar \texttt{\textbackslash type3}\rangle
\hrule\@width \unitindent
\kern 2.6\p@}
\end{verbatim}
Footnotes are numbered within chapters in the rapport and boek document styles.

% \newcounter{footnote}
\@addtoreset{footnote}{chapter}

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

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

\long\def\@xmakefntext#1#2{%
\parindent=.5\unitindent
%<!type3> \parindent=.5\unitindent
%<type3> \parindent=.5\unitindent
\def\labelitemi{--}\@revlabeltrue
\setbox0=\hbox {#1\hskip.5em plus 1fil}\
\dimen0=2\wd0
\ifdim\dimen0>\unitindent
\global\unitindent=\dimen0
\indentset
\fi
\@setpar{\@@par
\@tempdima \hsize
\advance\@tempdima-.5\unitindent
\parshape \@ne .5\unitindent \@tempdima)}%
\par
\noindent\llap{\hb@xt@.5\unitindent{#1\hfil}}#2}

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 the footnote. Thus, all the macro must do is set \texttt{\parindent} to the appropriate value for succeeding paragraphs and put the proper indentation before the mark. We change the label of itemized lists inside footnotes and need to check that the \texttt{\unitindent} is large enough for our purposes.

For most of the document classes produced from this file we need a slightly modified \texttt{\@makefntext} on the title page, so we introduce an extra macro, \texttt{\@xmakefntext}.

\newcommand*{\xmakefntext}[1]{\@xmakefntext{\normalfont\@thefnmark}{#1}}
\newcommand*{\xmakefntext}[1]{%
For the \texttt{article2} document class we have a simpler definition of \texttt{@makefntext}.

\begin{verbatim}
\@makefnmark
\end{verbatim}

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

\begin{verbatim}
\%\renewcommand\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
\end{verbatim}

\section{Initialization}

\subsection{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.

\begin{verbatim}
\newcommand\contentsname\texttt{Contents}
\newcommand\listfigurename\texttt{List of Figures}
\newcommand\listtablename\texttt{List of Tables}
\newcommand\refname\texttt{References}
\newcommand\bibname\texttt{Bibliography}
\newcommand\indexname\texttt{Index}
\newcommand\figurename\texttt{Figure}
\newcommand\tablename\texttt{Table}
\newcommand\partname\texttt{Part}
\newcommand\chaptername\texttt{Chapter}
\newcommand\appendixname\texttt{Appendix}
\newcommand\abstractname\texttt{Abstract}
\newcommand\seename\texttt{see}
\newcommand\andname\texttt{and}
\end{verbatim}
10.2 Date
\today  This macro uses the \TeX primitives \month, \day and \year to provide the date of the \LaTeX-run.
1702 \newcommand*{\today{}}
To save space we define \today in a way that it is expanded when the class file is read in. This means that low-level changes to the internal \TeX registers that are happening later on (e.g. if some packages goes \month=5) are not reflected in \today.
1703 \def\today{\ifcase\month\or
1704 January\or February\or March\or April\or May\or June\or
1705 July\or August\or September\or October\or November\or December\fi
1706 \space\number\day, \number\year}

10.3 Two column mode
\columnsep  This gives the distance between two columns in two column mode.
1707 \setlength{\columnsep}{10\p@}
\columnseprule  This gives the width of the rule between two columns in two column mode. We have no visible rule.
1708 \setlength{\columnseprule}{0\p@}

10.4 The page style
We have plain pages in the document classes artikel and rapport unless the user specified otherwise. In the boek document class we use the page style headings by default. We use arabic pagenumbers.
1709 ⟨\boek⟩ \pagestyle{plain}
1710 ⟨\boek⟩ \pagestyle{headings}
1711 \pagenumbering{arabic} % Arabic page numbers

10.5 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.
1712 ⟨∗artikel⟩
1713 \if@twoside
1714 \else
1715 \raggedbottom
1716 \fi
1717 ⟨/artikel⟩
When the twocolumn option was specified we call \twocolumn to activate this mode. We try to make each column as long as the others, but call sloppy to make our life easier.
1718 \if@twocolumn
\twocolumn
\sloppy
\flushbottom

Normally we call \onecolumn to initiate typesetting in one column.
\else
\onecolumn
\fi

\frenchspacing Controls the amount of space after a punctuation mark.
\frenchspacing

\langle /article \mid rapport \mid boek \rangle

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.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>@dblfptop</th>
<th>. . . . . . . . . . 423</th>
<th>@idxitem</th>
<th>. . 1648, 1651</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Roman</td>
<td>755</td>
<td>@dotsep</td>
<td>1092</td>
<td>@ifundefined</td>
</tr>
<tr>
<td>@afterheading</td>
<td>823, 856, 905, 929</td>
<td>@dottedtocline</td>
<td>1544, 1551–1554, 1557–1560, 1584</td>
<td>@indentset</td>
</tr>
<tr>
<td>@afterindentfalse</td>
<td>784, 881</td>
<td>@emptypagestyle</td>
<td>. . 440, 443</td>
<td>@itemdepth</td>
</tr>
<tr>
<td>@afterindenttrue</td>
<td>785, 1426</td>
<td>@endparpenalty</td>
<td>. . 1107, 1110, 1111</td>
<td>@listI</td>
</tr>
<tr>
<td>@allcapsfalse</td>
<td>53</td>
<td>@listdepth</td>
<td>. . 1116, 1146</td>
<td>@listI</td>
</tr>
<tr>
<td>@allcapstrue</td>
<td>54</td>
<td>@listdepth</td>
<td>. . 102, 1036</td>
<td>@listI</td>
</tr>
<tr>
<td>@author</td>
<td>528, 559, 561, 579, 631, 651, 663</td>
<td>@numctr</td>
<td>. . 111, 1114, 1115</td>
<td>@listII</td>
</tr>
<tr>
<td>@beginparpenalty</td>
<td>. . 1033, 1201</td>
<td>@numdepth</td>
<td>. . 1116, 1146</td>
<td>@listII</td>
</tr>
<tr>
<td>@begintheorem</td>
<td>. . 1255</td>
<td>@lowpenalty</td>
<td>. . 217</td>
<td>@listII</td>
</tr>
<tr>
<td>@biblabel</td>
<td>1616, 1617, 1633</td>
<td>@makecaption</td>
<td>@mainaux</td>
<td>. . 195, 198</td>
</tr>
<tr>
<td>@chapapp</td>
<td>481, 508, 765, 887, 916, 1293</td>
<td>@makechapterhead</td>
<td>@mainmatterfalse</td>
<td>. . 769, 777</td>
</tr>
<tr>
<td>@chapter</td>
<td>. . 882, 883</td>
<td>@msc</td>
<td>@makesymbol</td>
<td>. . 542, 602</td>
</tr>
<tr>
<td>@cite</td>
<td>1634</td>
<td>@fnsymbol</td>
<td>@math {mainmatterfalse}</td>
<td>. . 769, 777</td>
</tr>
<tr>
<td>@currentpagestyle</td>
<td>@fontswitch</td>
<td>@fontswitch</td>
<td>@fontswitch</td>
<td>. . 769, 777</td>
</tr>
<tr>
<td>@date</td>
<td>. . 529, 530, 567, 569, 581, 633, 654, 664</td>
<td>@fptop</td>
<td>. . 408</td>
<td>@makecaption</td>
</tr>
<tr>
<td>@dblfloat</td>
<td>1335, 1354</td>
<td>@hangfrom</td>
<td>@maketable</td>
<td>. . 902, 904, 908</td>
</tr>
<tr>
<td>@dblftop</td>
<td>. . 423</td>
<td>@highpenalty</td>
<td>@makefnmark</td>
<td>. . 603, 1687</td>
</tr>
<tr>
<td>@dblfpsep</td>
<td>. . 423</td>
<td>@latexwarning</td>
<td>@maketitle</td>
<td>. . 1217</td>
</tr>
<tr>
<td>/artikel</td>
<td>rapport</td>
<td>boek</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>