NTG Document Classes for LaTeX version 2e*

2023-01-10

Contents

*This file has version number v2.1f, last revised 2023-01-10.

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 LATEX2e 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.

```
1 ⟨*artikel | rapport | boek⟩
2 \newcommand*\@ptsize{}
3
```

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

```
4 \newif\if@restonecol
```

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

```
5 \newif\if@titlepage
6 ⟨artikel⟩\@titlepagefalse
7 ⟨!artikel⟩\@titlepagetrue
```

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

```
8 ⟨!artikel⟩\newif\if@openright
```

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

```
9 ⟨boek⟩\newif\if@mainmatter \@mainmattertrue
```

`\if@oldtoc` A switch to indicate if 'old' layout of the table of contents should be produced. These document classes normally produce a table of contents that looks quite different from what the standard classes produce.

```
10 \newif\if@oldtoc
11 \@oldtocfalse
```

`\if@allcaps` By default the text on the titlepage is set in capital letters. This can be disabled by the option mctitle, which sets the switch `\if@allcaps` to false.

```
12 \newif\if@allcaps
```

`\if@titlecentered` In the document classes `artikel3` and `rapport3` the default placement of the title that is produced by `\maketitle` is flushleft. This can be changed by the switch `\if@titlecentered`.

```
13 ⟨type3⟩\newif\if@titlecentered
14 ⟨type3⟩\@titlecenteredfalse
```

`\if@revlabel` These document classes need to be able to change the positioning of the label in labeled lists. This switch is used for that purpose.

```
15 \newif\if@revlabel
```

4 Declaration of Options

4.1 Setting Paper Sizes

The variables `\paperwidth` and `\paperheight` should reflect the physical paper size after trimming. For desk printer output this is usually the real paper size

since there is no post-processing. Classes for real book production will probably add other paper sizes and additionally the production of crop marks for trimming.

```
\DeclareOption{a4paper}
   {\setlength\paperheight {297mm}%
    \setlength\paperwidth  {210mm}}
\DeclareOption{a5paper}
   {\setlength\paperheight {210mm}%
    \setlength\paperwidth  {148mm}}
\DeclareOption{b5paper}
   {\setlength\paperheight {250mm}%
    \setlength\paperwidth  {176mm}}
\DeclareOption{letterpaper}
   {\setlength\paperheight {11in}%
    \setlength\paperwidth  {8.5in}}
\DeclareOption{legalpaper}
   {\setlength\paperheight {14in}%
    \setlength\paperwidth  {8.5in}}
\DeclareOption{executivepaper}
   {\setlength\paperheight {10.5in}%
    \setlength\paperwidth  {7.25in}}
```

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

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

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 wont 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.

```
\DeclareOption{draft}{\setlength\overfullrule{5pt}}
\DeclareOption{final}{\setlength\overfullrule{0pt}}
```

4.5 Titlepage option

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

```
\DeclareOption{titlepage}{\@titlepagetrue}
\DeclareOption{notitlepage}{\@titlepagefalse}
```

4.6 openright option

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

```
⟨!artikel⟩\DeclareOption{openright}{\@openrighttrue}
⟨!artikel⟩\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.

```
\DeclareOption{voorwerk}{\@titlepagetrue}
\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 2_ε document classes we supply an extra option: oldtoc. If this option is specified the switch `\if@oldtoc` will be set true.

```
\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.

```
⟨type3⟩\DeclareOption{titlecentered}{\@titlecenteredtrue}
```

In the `rapport` and `boek` document styles the titlepage uses all capital letters. The option mctitle (for 'mixed case') prevents this.

```
⟨rapport | boek⟩\DeclareOption{mctitle}{\@allcapsfalse}
⟨rapport | boek⟩\DeclareOption{uctitle}{\@allcapstrue}
```

4.9 Twocolumn 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 `\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 `\bibindent`.

```
\DeclareOption{openbib}{%
```

First some hook into the bibliography environment is filled.

```
  \AtEndOfPackage{%
    \renewcommand\@openbib@code{%
      \advance\leftmargin\bibindent
      \itemindent -\bibindent
      \listparindent \itemindent
      \parsep \z@
      }%
```

In addition the definition of `\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.

```
⟨*artikel⟩
```

```
\ExecuteOptions{a4paper,10pt,oneside,onecolumn,final,uctitle}
⟨/artikel⟩
⟨*rapport⟩
\ExecuteOptions{a4paper,10pt,oneside,onecolumn,final,uctitle,openany}
⟨/rapport⟩
⟨*boek⟩
\ExecuteOptions{a4paper,10pt,twoside,onecolumn,final,uctitle,openright}
⟨/boek⟩
```

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

```
\ProcessOptions
```

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

```
\input{ntg1\@ptsize.clo}
⟨/artikel | rapport | boek⟩
```

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 `\size` executes the command `\@setfontsize\size`⟨*font-size*⟩⟨*baselineskip*⟩ where:

⟨***font-size***⟩ The absolute size of the font to use from now on.

⟨***baselineskip***⟩ The normal value of `\baselineskip` for the size of the font selected. (The actual value will be `\baselinestretch` * ⟨*baselineskip*⟩.)

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

`\@vpt`	5	`\@vipt`	6	`\@viipt`	7
`\@viiipt`	8	`\@ixpt`	9	`\@xpt`	10
`\@xipt`	10.95	`\@xiipt`	12	`\@xivpt`	14.4
...					

\normalsize
\@normalsize

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

The \normalsize macro also sets new values for \abovedisplayskip, \abovedisplayshortskip and

```
⟨*10pt | 11pt | 12pt⟩
\renewcommand\normalsize{%
⟨*10pt⟩
   \@setfontsize\normalsize\@xpt\@xiipt
   \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
   \abovedisplayshortskip \z@ \@plus3\p@
   \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
⟨/10pt⟩
⟨*11pt⟩
   \@setfontsize\normalsize\@xipt{13.6}%
   \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
   \abovedisplayshortskip \z@ \@plus3\p@
   \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
⟨/11pt⟩
⟨*12pt⟩
   \@setfontsize\normalsize\@xiipt{14.5}%
   \abovedisplayskip 12\p@ \@plus3\p@ \@minus7\p@
   \abovedisplayshortskip \z@ \@plus3\p@
   \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
⟨/12pt⟩
```

The \belowdisplayskip is always equal to the \abovedisplayskip. The parameters of the first level list are always given by \@listI.

```
   \belowdisplayskip \abovedisplayskip
   \let\@listi\@listI}
```

Make \@normalsize a synonymn for \normalsize.

```
\let\@normalsize\normalsize
```

We initially choose the normalsize font.

```
\normalsize
```

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

```
\ifx\MakeRobust\@undefined \else
    \MakeRobust\normalsize
\fi
```

\small

This is similar to \normalsize.

```
\DeclareRobustCommand\small{%
⟨*10pt⟩
   \@setfontsize\small\@ixpt{11}%
```

```
    \abovedisplayskip 8.5\p@ \@plus3\p@ \@minus4\p@
    \abovedisplayshortskip \z@ \@plus2\p@
    \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
⟨/10pt⟩
⟨*11pt⟩
    \@setfontsize\small\@xpt\@xiipt
    \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
⟨/11pt⟩
⟨*12pt⟩
    \@setfontsize\small\@xipt{13.6}%
    \abovedisplayskip 11\p@ \@plus3\p@ \@minus6\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6.5\p@ \@plus3.5\p@ \@minus3\p@
⟨/12pt⟩
    \belowdisplayskip \abovedisplayskip
}
```

`\footnotesize` This is similar to `\normalsize`.

```
\DeclareRobustCommand\footnotesize{%
⟨*10pt⟩
    \@setfontsize\footnotesize\@viiipt{9.5}%
    \abovedisplayskip 6\p@ \@plus2\p@ \@minus4\p@
    \abovedisplayshortskip \z@ \@plus\p@
    \belowdisplayshortskip 3\p@ \@plus\p@ \@minus2\p@
⟨/10pt⟩
⟨*11pt⟩
    \@setfontsize\footnotesize\@ixpt{11}%
    \abovedisplayskip 8\p@ \@plus2\p@ \@minus4\p@
    \abovedisplayshortskip \z@ \@plus\p@
    \belowdisplayshortskip 4\p@ \@plus2\p@ \@minus2\p@
⟨/11pt⟩
⟨*12pt⟩
    \@setfontsize\footnotesize\@xpt\@xiipt
    \abovedisplayskip 10\p@ \@plus2\p@ \@minus5\p@
    \abovedisplayshortskip \z@ \@plus3\p@
    \belowdisplayshortskip 6\p@ \@plus3\p@ \@minus3\p@
⟨/12pt⟩
    \belowdisplayskip \abovedisplayskip
}
```

`\scriptsize` `\tiny` `\large` `\Large` `\LARGE` `\huge` `\Huge` These are all much simpler than the previous macros, they just select a new fontsize, but leave the parameters for displays and lists alone.

```
⟨*10pt⟩
\DeclareRobustCommand\scriptsize{\@setfontsize\scriptsize\@viipt\@viiipt}
\DeclareRobustCommand\tiny{\@setfontsize\tiny\@vpt\@vipt}
\DeclareRobustCommand\large{\@setfontsize\large\@xiipt{14}}
\DeclareRobustCommand\Large{\@setfontsize\Large\@xivpt{18}}
\DeclareRobustCommand\LARGE{\@setfontsize\LARGE\@xviipt{22}}
```

```
\DeclareRobustCommand\huge{\@setfontsize\huge\@xxpt{25}}
\DeclareRobustCommand\Huge{\@setfontsize\Huge\@xxvpt{30}}
⟨/10pt⟩
⟨*11pt⟩
\DeclareRobustCommand\scriptsize{\@setfontsize\scriptsize\@viiipt{9.5}}
\DeclareRobustCommand\tiny{\@setfontsize\tiny\@vipt\@viipt}
\DeclareRobustCommand\large{\@setfontsize\large\@xiipt{14}}
\DeclareRobustCommand\Large{\@setfontsize\Large\@xivpt{18}}
\DeclareRobustCommand\LARGE{\@setfontsize\LARGE\@xviipt{22}}
\DeclareRobustCommand\huge{\@setfontsize\huge\@xxpt{25}}
\DeclareRobustCommand\Huge{\@setfontsize\Huge\@xxvpt{30}}
⟨/11pt⟩
⟨*12pt⟩
\DeclareRobustCommand\scriptsize{\@setfontsize\scriptsize\@viiipt{9.5}}
\DeclareRobustCommand\tiny{\@setfontsize\tiny\@vipt\@viipt}
\DeclareRobustCommand\large{\@setfontsize\large\@xivpt{18}}
\DeclareRobustCommand\Large{\@setfontsize\Large\@xviipt{22}}
\DeclareRobustCommand\LARGE{\@setfontsize\LARGE\@xxpt{25}}
\DeclareRobustCommand\huge{\@setfontsize\huge\@xxvpt{30}}
\let\Huge=\huge
⟨/12pt⟩
⟨/10pt | 11pt | 12pt⟩
```

7.2 Paragraphing

`\lineskip` `\normallineskip` These parameters control TEX's behaviour when two lines tend to come too close together.

```
⟨*artikel | rapport | boek⟩
\setlength\lineskip{1\p@}
\setlength\normallineskip{1\p@}
```

`\baselinestretch` 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.

```
\renewcommand\baselinestretch{}
```

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

- the label width in section heading,
- the `\parindent`
- the footnote label indent (= half `\unitindent`)
- listindent on the first level

```
\newdimen\unitindent
```

The default setting accomodates three levels of single digit section numbering.

```
⟨*type1 | type3⟩
{\setbox0\hbox{\normalsize\rmfamily 2.2.2\hskip.5em}
 \global\unitindent=\wd0}
⟨/type1 | type3⟩
```

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

```
\newdimen\othermargin
{\setbox0\hbox{\normalsize (m)\hskip.6em}\global\othermargin=\wd0}
```

`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

```
⟨*type1 | type3⟩
\newif\if@needwriteindent
```

`\@indentset` And a command that sets the various parameters.

```
\newcommand*\@indentset{%
⟨!type3⟩                \global\parindent=\unitindent
                \global\leftmargini=\unitindent
                \global\@needwriteindenttrue}
```

`\@writeindent` The `\end{document}` command will call `\@writeindent` to write the final width of `\unitindent` on the `.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}{}}
    \immediate\write\@mainaux{\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` om the file.aux file for the next run.

```
\AtEndDocument{%
    \if@filesw
      \if@needwriteindent
        \@writeindent{\the\unitindent}
      \fi
    \fi}
⟨/type1 | type3⟩
```

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

```
⟨type2⟩\unitindent=2\othermargin
```

\parskip
\parindent

\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.

```
⟨*type1⟩
\setlength\parskip{0\p@}
\setlength\parindent{\unitindent}
⟨/type1⟩
⟨*type3⟩
\setlength\parskip{.5\baselineskip \@plus .1\baselineskip
                                   \@minus .1\baselineskip}
\setlength\parindent{\z@}
⟨/type3⟩
```

\@lowpenalty
\@medpenalty
\@highpenalty

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   51
\@medpenalty  151
\@highpenalty 301
```

\clubpenalty
\widowpenalty

These penalties are use to discourrage club and widow lines. Because we use their default values we only show them here, commented out.

```
% \clubpenalty  150
% \widowpenalty 150
```

\displaywidowpenalty
\predisplaypenalty
\postdisplaypenalty

Discourrage (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.

```
% \displaywidowpenalty 50
% \predisplaypenalty   10000
% \postdisplaypenalty  0
```

\interlinepenalty

Allow the breaking of a page in the middle of a paragraph.

```
% \interlinepenalty 0
```

\brokenpenalty

We allow the breaking of a page after a hyphenated line.

```
% \brokenpenalty 0
⟨/artikel | rapport | boek⟩
```

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 \headsep \topskip

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.

```
⟨*10pt | 11pt | 12pt⟩
\setlength\headheight{12\p@}
\setlength\headsep   {25\p@}
⟨10pt⟩\setlength\topskip   {10\p@}
⟨11pt⟩\setlength\topskip   {11\p@}
⟨12pt⟩\setlength\topskip   {12\p@}
```

\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{30\p@}   %
```

\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 LATEX2e mode we let the value depend on the typesize. We set it so that \maxdepth + \topskip = typesize ×1.5. As it happens, in these classes \topskip is equal to the typesize, therefor we set \maxdepth to half the value of \topskip.

```
\if@compatibility
  \setlength\maxdepth{4\p@}
\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{410\p@}
  \else
⟨10pt⟩    \setlength\textwidth{345\p@}
⟨11pt⟩    \setlength\textwidth{360\p@}
⟨12pt⟩    \setlength\textwidth{390\p@}
  \fi
```

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

```
\else
```

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}
⟨10pt⟩  \setlength\@tempdimb{361\p@}
⟨11pt⟩  \setlength\@tempdimb{376\p@}
⟨12pt⟩  \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
```

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
\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 LaTeX2.09. The number of lines was calculated as the floor of the old `\textheight` minus `\topskip`, divided by `\baselineskip` for `\normalsize`. The old value of `\textheight` was 528pt.

```
\if@compatibility
⟨10pt⟩  \setlength\textheight{43\baselineskip}
⟨11pt⟩  \setlength\textheight{38\baselineskip}
⟨12pt⟩  \setlength\textheight{36\baselineskip}
```

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.

```
\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`.

```
\advance\textheight by \topskip
```

7.3.3 Margins

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

`\marginparsep` `\marginparpush`

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`.

```
\if@twocolumn
  \setlength\marginparsep {10\p@}
\else
⟨10pt⟩   \setlength\marginparsep{11\p@}
⟨11pt⟩   \setlength\marginparsep{10\p@}
⟨12pt⟩   \setlength\marginparsep{10\p@}
\fi
⟨10pt | 11pt⟩\setlength\marginparpush{5\p@}
⟨12pt⟩\setlength\marginparpush{7\p@}
```

Now we can give the values for the other margin parameters. For native LaTeX 2_ε, these are calculated.

`\oddsidemargin` `\evensidemargin` `\marginparwidth`

First we give the values for the compatibility mode.

Values for two-sided printing:

```
\if@compatibility
  \if@twoside
⟨10pt⟩      \setlength\oddsidemargin   {44\p@}
```

```
⟨11pt⟩      \setlength\oddsidemargin   {36\p@}
⟨12pt⟩      \setlength\oddsidemargin   {21\p@}
⟨10pt⟩      \setlength\evensidemargin  {82\p@}
⟨11pt⟩      \setlength\evensidemargin  {74\p@}
⟨12pt⟩      \setlength\evensidemargin  {59\p@}
⟨10pt⟩      \setlength\marginparwidth {107\p@}
⟨11pt⟩      \setlength\marginparwidth {100\p@}
⟨12pt⟩      \setlength\marginparwidth {85\p@}
```

Values for one-sided printing:

```
  \else
⟨10pt⟩      \setlength\oddsidemargin   {63\p@}
⟨11pt⟩      \setlength\oddsidemargin   {54\p@}
⟨12pt⟩      \setlength\oddsidemargin   {39.5\p@}
⟨10pt⟩      \setlength\evensidemargin  {63\p@}
⟨11pt⟩      \setlength\evensidemargin  {54\p@}
⟨12pt⟩      \setlength\evensidemargin  {39.5\p@}
⟨10pt⟩      \setlength\marginparwidth  {90\p@}
⟨11pt⟩      \setlength\marginparwidth  {83\p@}
⟨12pt⟩      \setlength\marginparwidth  {68\p@}
  \fi
```

And values for two column mode:

```
  \if@twocolumn
     \setlength\oddsidemargin  {30\p@}
     \setlength\evensidemargin {30\p@}
     \setlength\marginparwidth {48\p@}
  \fi
```

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

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

If `@twoside` is true (which is always the case for boek) we make the inner margin smaller than the outer one.

```
\else
  \if@twoside
    \setlength\@tempdima        {\paperwidth}
    \addtolength\@tempdima      {-\textwidth}
    \setlength\oddsidemargin    {.4\@tempdima}
    \addtolength\oddsidemargin  {-1in}
```

The width of the margin for text is set to the remainder of the width except for a 'real margin' of white space of width 0.4in. A check should perhaps be built in to ensure that the (text) margin width does not get too small!

```
    \setlength\marginparwidth   {.6\@tempdima}
    \addtolength\marginparwidth {-\marginparsep}
    \addtolength\marginparwidth {-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 than used for the

margin (thus \oddsidemargin is 1in less).

```
\else
  \setlength\@tempdima        {\paperwidth}
  \addtolength\@tempdima      {-\textwidth}
  \setlength\oddsidemargin    {.5\@tempdima}
  \addtolength\oddsidemargin  {-1in}
  \setlength\marginparwidth   {.5\@tempdima}
  \addtolength\marginparwidth {-\marginparsep}
  \addtolength\marginparwidth {-.4in}
\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 todays technology can accuratly 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}
  \addtolength\topmargin{-\footskip}     % this might be wrong!
```

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 `\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.

```
⟨10pt⟩\setlength\footnotesep{6.65\p@}
⟨11pt⟩\setlength\footnotesep{7.7\p@}
⟨12pt⟩\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.

```
⟨10pt⟩\setlength{\skip\footins}{9\p@ \@plus 4\p@ \@minus 2\p@}
⟨11pt⟩\setlength{\skip\footins}{10\p@ \@plus 4\p@ \@minus 2\p@}
⟨12pt⟩\setlength{\skip\footins}{10.8\p@ \@plus 4\p@ \@minus 2\p@}
⟨/10pt | 11pt | 12pt⟩
```

7.3.5 Float placement parameters

All float parameters are given default values in the LaTeX 2_ε 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.

```
⟨*artikel | rapport | boek⟩
\setcounter{topnumber}{2}
```

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

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

\c@bottomnumber The *bottomnumber* counter holds the maximum number of floats that can appear on the bottom of a text page.

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

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

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

\c@totalnumber This indicates the maximum number of floats that can appear on any text page.

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

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

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

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

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

\c@dbltopnumber The *dbltopnumber* counter holds the maximum number of two column floats that can appear on the top of a two column text page.

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

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

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

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

```
\renewcommand\dblfloatpagefraction{.5}
⟨/artikel | rapport | boek⟩
```

Floats on a text page

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

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

\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.

```
⟨*10pt⟩
\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@}
⟨/10pt⟩
⟨*11pt⟩
\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@}
⟨/11pt⟩
⟨*12pt⟩
\setlength\floatsep    {12\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\textfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\intextsep   {14\p@ \@plus 4\p@ \@minus 4\p@}
⟨/12pt⟩
```

\dblfloatsep \dbltextfloatsep 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.

```
⟨*10pt⟩
```

```
\setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
⟨/10pt⟩
⟨*11pt⟩
\setlength\dblfloatsep    {12\p@ \@plus 2\p@ \@minus 2\p@}
\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
⟨/11pt⟩
⟨*12pt⟩
\setlength\dblfloatsep    {14\p@ \@plus 2\p@ \@minus 4\p@}
\setlength\dbltextfloatsep{20\p@ \@plus 2\p@ \@minus 4\p@}
⟨/12pt⟩
```

Floats on their own page or column

`\@fptop` `\@fpsep` `\@fpbot`

When floating objects are placed on seperate 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 paramaters are used for the placement of floating objects in one column mode, or in single column floats in two column mode.

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

```
⟨*10pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/10pt⟩
⟨*11pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{8\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/11pt⟩
⟨*12pt⟩
\setlength\@fptop{0\p@ \@plus 1fil}
\setlength\@fpsep{10\p@ \@plus 2fil}
\setlength\@fpbot{0\p@ \@plus 1fil}
⟨/12pt⟩
```

`\@dblfptop` `\@dblfpsep` `\@dblfpbot`

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

```
⟨*10pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/10pt⟩
⟨*11pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{8\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/11pt⟩
```

```
⟨*12pt⟩
\setlength\@dblfptop{0\p@ \@plus 1fil}
\setlength\@dblfpsep{10\p@ \@plus 2fil}
\setlength\@dblfpbot{0\p@ \@plus 1fil}
⟨/12pt⟩
⟨*artikel | rapport | boek⟩
```

7.4 Page Styles

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

`\@evenhead` `\@oddhead` `\@evenfoot` `\@oddfoot`

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

`\thispagestyle`

Several commands (`\index`, `\maketitle`) give a `\thispagestyle{plain}` command, which will overrule a `\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 `\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 `\chaptermark`, `\sectionmark`, ...,
where `\chaptermark{`⟨*TEXT*⟩`}` is called by `\chapter` to set a mark, and so on.

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

LaTeX extends TeX's `\mark` facility by producing two kinds of marks, a 'left' and a 'right' mark, using the following commands:

`\markboth{`⟨*LEFT*⟩`}{`⟨*RIGHT*⟩`}`: Adds both marks.

`\markright{`⟨*RIGHT*⟩`}`: Adds a 'right' mark.
`\leftmark`: Used in the `\@oddhead`, `\@oddfoot`, `\@evenhead` or `\@evenfoot` macros, it gets the current 'left' mark. `\leftmark` works like TEX's `\botmark` command.
`\rightmark`: Used in the `\@oddhead`, `\@oddfoot`, `\@evenhead` or `\@evenfoot` macros, it gets the current 'right' mark. `\rightmark` works like TEX's `\firstmark` command.

The marking commands work reasonably well for right marks 'numbered within' left marks–e.g., the left mark is changed by a `\chapter` command and the right mark is changed by a `\section` command. However, it does produce somewhat anomalous results if two `\markboth`'s occur on the same page.

Commands like `\tableofcontents` that should set the marks in some page styles use a `\@mkboth` command, which is `\let` by the pagestyle command (`\ps@...`) to `\markboth` for setting the heading or to `\@gobbletwo` to do nothing.

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.

`\ps@plain`

```
448 \renewcommand*\ps@plain{%
```

The running heads are empty in this pagestyle, the page number appears in the running foot.

```
449     \let\@oddhead\@empty\let\@evenhead\@empty
450     \def\@oddfoot{\hfil\PageFont\thepage}%
451     \if@twoside
452       \def\@evenfoot{\PageFont\thepage\hfil}%
453     \else
454       \let\@evenfoot\@oddfoot
455     \fi
```

Because the running heads should be empty we `let \@mkboth` to `\@gobbletwo`, thus disabling the mark commands.

```
456     \let\@mkboth\@gobbletwo}
```

`\ps@headings` The definition of the page style *headings* has to be different for two sided printing than it is for one sided printing.

```
457 \if@twoside
458     \def\ps@headings{%
```

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

```
459       \let\@oddfoot\@empty\let\@evenfoot\@empty
460       \def\@evenhead{{\PageFont\thepage}\hfil\MarkFont\leftmark}%
461       \def\@oddhead{{\MarkFont\rightmark}\hfil\PageFont\thepage}%
```

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

```
462       \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.

```
463 ⟨*artikel⟩
464       \def\sectionmark##1{%
465         \markboth {\MakeUppercase{%
466             \ifnum \c@secnumdepth >\z@
467               \thesection\quad
468             \fi
469             ##1}}{}}%
470       \def\subsectionmark##1{%
471         \markright {%
472           \ifnum \c@secnumdepth >\@ne
473             \thesubsection\quad
474           \fi
475           ##1}}}
476 ⟨/artikel⟩
```

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.

```
477 ⟨*rapport | boek⟩
478       \def\chaptermark##1{%
479         \markboth {\MakeUppercase{\ifnum \c@secnumdepth >\m@ne
480 ⟨boek⟩            \if@mainmatter
481                 \@chapapp\ \thechapter. \ %
482 ⟨boek⟩            \fi
483             \fi
484             ##1}}{}}%
485       \def\sectionmark##1{%
486         \markright {\MakeUppercase{\ifnum \c@secnumdepth >\z@
487             \thesection. \ \fi
488             ##1}}}}
489 ⟨/rapport | boek⟩
```

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...`.

```
490 \else
491   \def\ps@headings{%
492     \let\@oddfoot\@empty
```

```
    \def\@oddhead{{\MarkFont\rightmark}\hfil\PageFont\thepage}%
    \let\@mkboth\markboth
```

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

```
⟨*artikel⟩
    \def\sectionmark##1{%
      \markright {\MakeUppercase{%
          \ifnum \c@secnumdepth >\m@ne
            \thesection\quad
          \fi
          ##1}}}}
⟨/artikel⟩
⟨*rapport | boek⟩
    \def\chaptermark##1{%
      \markright {\MakeUppercase{%
          \ifnum \c@secnumdepth >\m@ne
⟨boek⟩              \if@mainmatter
                \@chapapp\ \thechapter. \ %
⟨boek⟩              \fi
          \fi
          ##1}}}}
⟨/rapport | boek⟩
\fi
```

`\ps@myheadings` 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.

```
\def\ps@myheadings{%
    \let\@oddfoot\@empty\let\@evenfoot\@empty
    \def\@evenhead{{\PageFont\thepage}\hfil\MarkFont\leftmark}%
    \def\@oddhead{{\MarkFont\rightmark}\hfil\PageFont\thepage}%
```

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

```
    \let\@mkboth\@gobbletwo
⟨!artikel⟩    \let\chaptermark\@gobble
    \let\sectionmark\@gobble
⟨artikel⟩     \let\subsectionmark\@gobble
    }
```

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

```
\newcommand*\PageFont{\rmfamily}
\newcommand*\MarkFont{\slshape}
```

`\RunningFonts` Use this macro to change the fonts that are used in the running heads.

```
\newcommand*\RunningFonts[2]{%
  \renewcommand*\PageFont{#1}\renewcommand*\MarkFont{#2}}
```

8 Document Markup

8.1 The title

\title \author \date

These three macros are provided by `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 `\maketitle` command to use the information provided. The definitions of these macros are shown here for information.

```
% \newcommand*\title[1]{\gdef\@title{#1}}
% \newcommand*\author[1]{\gdef\@author{#1}}
% \newcommand*\date[1]{\gdef\@date{#1}}
```

The `\date` macro gets today's date by default.

```
% \gdef\@date{\today}
```

\TitleFont

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

```
\newcommand*\TitleFont{\bfseries}
```

\maketitle

The definition of `\maketitle` depends on whether a seperate 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 mctitle.

When we are making a title page, we locally redefine `\footnotesize` and `\footnoterule` to change the appearance of the footnotes that are produced by the `\thanks` command.

```
⟨!boek⟩\if@titlepage
\renewcommand*\TitleFont{\rmfamily}
\newcommand*\maketitle{%
  \begin{titlepage}%
    \let\footnotesize\small
    \let\footnoterule\relax
    \let \footnote \thanks
```

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

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

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

```
    \if@compatibility\setcounter{page}{0}\fi
    \null\vfil
    \vskip 60\p@
```

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}\par}%
      \vskip 1.5em%
      {\large
        \if@allcaps
          \uppercase\expandafter{\@date}%
        \else
          \@date
        \fi\par}%
      \end{center}\par
```

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
```

```
  \global\let\author\relax
  \global\let\date\relax
  \global\let\and\relax
}
```

We want to have the title, author and date information in uppercase, but we have to be very carefull not to put too much text in uppercase. The macros that perform the filtering of texts that shouldn't be in uppercase were developped with th 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.

```
\def\uc@nothanks#1\thanks{\uppercase{#1}\futurelet\@tempa\uc@tx}
```

`\uc@authornothanks` A document can have more than one author. Usually they are seperated 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 than passsed 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`.

```
\def\uc@authornothanks#1\and{\uc@nothanks#1\thanks\relax
```

Then we have to check whether the `\and` we ound earlier was put in by the user, in which case information for another user will follow, or by the call from another macro, in which case the `\and` will be followed by a `\relax` token. The `\futurelet` contstruct stores the first token *after* the `\and` in `\@tempa` to be inspected by `\u@ax`.

```
    \futurelet\@tempa\uc@ax}
```

`\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.

```
\def\uc@ax{%
  \ifx\@tempa\relax
  \else
    \\ \andname \\ \expandafter\uc@authornothanks
  \fi}
```

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

```
\def\uc@tx{\ifx\@tempa\relax
    \else \expandafter\uc@ty \fi}
```

`\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.

```
⟨*!boek⟩
\else
\newcommand*\maketitle{\par
  \begingroup
    \renewcommand*\thefootnote{\@fnsymbol\c@footnote}%
⟨!type2⟩    \def\@makefnmark{\rlap{%
⟨!type2⟩      \@textsuperscript{\normalfont\@thefnmark}}}%
⟨!type2⟩    \long\def\@makefntext{\@xmakefntext{%
⟨!type2⟩      \@textsuperscript{\normalfont\@thefnmark}}}%
⟨*type2⟩
    \long\def\@makefntext##1{\parindent\z@
      \def\labelitemi{\textendash}%
      \leavevmode\hb@xt@.5\unitindent{%
        \@textsuperscript{\normalfont\@thefnmark}\hfil}##1}
⟨/type2⟩
```

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`.

```
    \if@twocolumn
      \ifnum \col@number=\@ne
        \@maketitle
      \else
        \twocolumn[\@maketitle]%
      \fi
    \else
```

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.

```
      \newpage
      \global\@topnum\z@
      \@maketitle
    \fi
```

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

```
    \thispagestyle{plain}\@thanks
```

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\@title\@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 seperate 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%
⟨type3⟩\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}%
⟨*type3⟩
 \else
     {\LARGE \TitleFont \head@style \@title \par} \vskip 1.5em
     {\large \TitleFont \lineskip .5em \tabcolsep\z@
             \def\and{%%% \begin{tabular} has already started
                      \end{tabular}\hskip 1em plus .17fil
                      \begin{tabular}[t]{l}}%% \end{tabular} will come
             \begin{tabular}[t]{l}\@author\end{tabular}\par}
     \vskip 1em {\large \TitleFont \@date}
 \fi
```

```
666 ⟨/type3⟩
667   \par
668   \vskip 1.5em}
669 \fi
670 ⟨/!boek⟩
```

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`⟨*name*⟩⟨*level*⟩⟨*indent*⟩⟨*beforeskip*⟩⟨*afterskip*⟩⟨*style*⟩ optional * `[`⟨*altheading*⟩`]`⟨*heading*⟩

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

⟨***name***⟩ The name of the user level command, e.g., 'section'.

⟨***level***⟩ 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 ⟨*level*⟩ <= the value of the *secnumdepth* counter.

⟨***indent***⟩ The indentation of the heading from the left margin

⟨***beforeskip***⟩ 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.

⟨***afterskip***⟩ 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.

⟨***style***⟩ Commands to set the style of the heading. Since the June 1996 release of LaTeX 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.

⟨***altheading***⟩ 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.

⟨***heading***⟩ 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`⟨*unstarcmds*⟩⟨*starcmds*⟩

⟨***unstarcmds***⟩ Used for the normal form of the sectioning command.

⟨***starcmds***⟩ Used for the ∗-form of the sectioning command.

You can use `\secdef` as follows:

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

`\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.

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

`\@sect` The definition of this macro from `latex.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`.

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

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 rememeber to store the new value in the `.aux`-file.

```
⟨∗type1 | type3⟩
    \begingroup
      \setbox\@tempboxa=\hbox{#6\relax
```

```
                                \csname the#1\endcsname
                                \hskip.5em}
      \ifdim\wd\@tempboxa>\unitindent
        \global\unitindent=\wd\@tempboxa
        \@indentset
      \fi
    \endgroup
⟨/type1 | type3⟩
```

Since `\@seccntformat` might end with an improper `\hskip` which is scanning forward for `plus` or `minus` we end the definition of `\@svsec` with `\relax` as a precaution.

```
    \protected@edef\@svsec{\@seccntformat{#1}\relax}%
  \fi
  \@tempskipa #5\relax
  \ifdim \@tempskipa>\z@
    \begingroup
```

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`.

```
    #6{%
⟨*type1 | type3⟩
      \@hangfrom{\hskip #3\relax\@svsec}\head@style #8\endgraf}%
⟨/type1 | type3⟩
⟨*type2⟩
       \@hangfrom{\hskip #3}
                  \head@style\@svsec \hskip.3em\relax #8\endgraf}
⟨/type2⟩
    \endgroup
    \csname #1mark\endcsname{#7}%
    \addcontentsline{toc}{#1}{%
      \ifnum #2>\c@secnumdepth
      \else
        \protect\numberline{\csname the#1\endcsname}%
      \fi
      #7}%
    \else
      \def\@svsechd{#6\hskip #3\relax
        \@svsec #8\csname #1mark\endcsname{#7}%
        \addcontentsline{toc}{#1}{%
          \ifnum #2>\c@secnumdepth
          \else
            \protect\numberline{\csname the#1\endcsname}%
          \fi
          #7}}%
    \fi
    \@xsect{#5}}
```

This macro was introduced in LaTeX 2_ε, its definition is changed here to get the fixed with of the section number.

```
\def\@seccntformat#1{%
```

```
⟨!type2⟩  \hb@xt@\unitindent{\csname the#1\endcsname \hfil}%
⟨type2⟩  \csname the#1\endcsname\hskip.3em\relax
  }
```

\@ssect Similar changes need to be made to the definition of \@ssect, which is used in 'starred' sections.

```
\def\@ssect#1#2#3#4#5{\@tempskipa #3\relax
  \ifdim \@tempskipa>\z@
    \begingroup
```

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

```
      #4{%
        \@hangfrom{\hskip #1}\head@style #5\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.tex, so they are only shown here.

```
⟨!artikel⟩\newcommand*\chaptermark[1]{}
% \newcommand*\sectionmark[1]{}
% \newcommand*\subsectionmark[1]{}
% \newcommand*\subsubsectionmark[1]{}
% \newcommand*\paragraphmark[1]{}
% \newcommand*\subparagraphmark[1]{}
```

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.

```
⟨artikel⟩\setcounter{secnumdepth}{3}
⟨!artikel⟩\setcounter{secnumdepth}{2}
```

\c@part \c@chapter \c@section \c@subsection \c@subsubsection \c@paragraph \c@subparagraph These counters are used for the section numbers. The macro
\newcounter{⟨*newctr*⟩}[⟨*oldctr*⟩]
defines ⟨*newctr*⟩ to be a counter, which is reset to zero when counter ⟨*oldctr*⟩ is stepped. Counter ⟨*oldctr*⟩ must already be defined.

```
\newcounter {part}
⟨artikel⟩\newcounter {section}
⟨*rapport | boek⟩
\newcounter {chapter}
\newcounter {section}[chapter]
```

```
⟨/rapport | boek⟩
\newcounter {subsection}[section]
\newcounter {subsubsection}[subsection]
\newcounter {paragraph}[subsubsection]
\newcounter {subparagraph}[paragraph]
```

\thepart
\thechapter
\thesection
\thesubsection
\thesubsubsection
\theparagraph
\thesubparagraph

For any counter *CTR*, `\theCTR` is a macro that defines the printed version of counter *CTR*. It is defined in terms of the following macros:

`\arabic{`*COUNTER*`}` prints the value of *COUNTER* as an arabic numeral.

`\roman{`*COUNTER*`}` prints the value of *COUNTER* as a lowercase roman numberal.

`\Roman{`*COUNTER*`}` prints the value of *COUNTER* as an uppercase roman numberal.

`\alph{`*COUNTER*`}` prints the value of *COUNTER* as a lowercase letter: 1 = a, 2 = b, etc.

`\Alph{`*COUNTER*`}` prints the value of *COUNTER* as an uppercase letter: 1 = A, 2 = B, etc.

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

```
\renewcommand*\thepart{\@Roman\c@part}
⟨artikel⟩\renewcommand\thesection{\@arabic\c@section}
⟨*rapport | boek⟩
\renewcommand*\thechapter{\@arabic\c@chapter}
\renewcommand*\thesection{\thechapter.\@arabic\c@section}
⟨/rapport | boek⟩
\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

`\@chapapp` is initially defined to be '`\chaptername`'. The `\appendix` command redefines it to be '`\appendixname`'.

```
⟨rapport | boek⟩\newcommand*\@chapapp{\chaptername}
```

8.2.4 Front Matter, Main Matter, and Back Matter

A boek contains these three sections. First, we define the switch `\@mainmatter` that is true iff we are processing Main Matter. When this switch is false, the `\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.

```
⟨*boek⟩
\newcommand*\frontmatter{%
  \cleardoublepage
  \@mainmatterfalse
  \pagenumbering{roman}}
```

`\mainmatter` This command clears the page, starts arabic page numbering and turns on chapter numbering.

```
\newcommand*\mainmatter{%
  \cleardoublepage
  \@mainmattertrue
  \pagenumbering{arabic}}
```

`\backmatter` This clears the page, turns off chapter numbering and leaves page numbering unchanged.

```
\newcommand*\backmatter{%
  \if@openright\cleardoublepage\else\clearpage\fi
  \@mainmatterfalse}
⟨/boek⟩
```

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`.

```
⟨*artikel⟩
\newcommand*\part{%
  \if@noskipsec \leavevmode \fi
  \par
  \addvspace{4ex}%
⟨!type2⟩    \@afterindentfalse
⟨type2⟩    \@afterindenttrue
  \secdef\@part\@spart}
⟨/artikel⟩
```

For the rapport and boek classes we things a bit different.

We start a new (righthand) page and use the *empty* pagestyle.

```
⟨*rapport | boek⟩
\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 `\secdef` to indicate which commands to use to make the actual heading.

```
  \secdef\@part\@spart}
⟨/rapport | boek⟩
```

`\@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.

`\PartFont` The font used to typeset the part is stored in this maro.

```
\newcommand*\PartFont{\bfseries}
```

When *secnumdepth* is larger than -1 for the artikel document classes, we have a numbered part, otherwise it is unnumbered.

```
⟨*artikel⟩
\def\@part[#1]#2{%
    \ifnum \c@secnumdepth >\m@ne
      \refstepcounter{part}%
      \addcontentsline{toc}{part}{\protect\numberline{\thepart}#1}%
    \else
      \addcontentsline{toc}{part}{#1}%
    \fi
```

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

```
    {\head@style
     \parindent\unitindent
     \normalfont
```

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

```
      \ifnum \c@secnumdepth >\m@ne
⟨!type2⟩        \Large\PartFont\noindent \partname\nobreakspace\thepart
⟨type2⟩        \Large\PartFont\indent \partname\nobreakspace\thepart
        \par\nobreak
      \fi
⟨!type2⟩      \Large \PartFont \noindent #2%
⟨type2⟩      \Large \PartFont #2%
```

Then we empty the mark registers, leave some white space and call `\@afterheading` to takes care of suppressing the indentation.

```
      \markboth{}{}\par}%
    \nobreak
    \vskip 3ex
    \@afterheading}
⟨/artikel⟩
```

When *secnumdepth* is larger than -2 for the document class rapport and boek, we have a numbered part, otherwise it is unnumbered.

```
⟨*rapport | boek⟩
```

```
\def\@part[#1]#2{%
    \ifnum \c@secnumdepth >-2\relax
      \refstepcounter{part}%
      \addcontentsline{toc}{part}{\protect\numberline{\thepart}{#1}}%
    \else
      \addcontentsline{toc}{part}{\toc@case{#1}}%
    \fi
```

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.

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

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`.

```
     \ifnum \c@secnumdepth >-2\relax
       \Large\PartFont
       \edef\@tempa{\noexpand\MakeUppercase{\partname}}\@tempa
       \nobreakspace\thepart
       \par
```

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

```
       \vskip 20\p@
     \fi
     \Large \PartFont \MakeUppercase{#2}\par}%
    \@endpart}
⟨/rapport | boek⟩
```

`\@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`.

```
⟨*artikel⟩
\def\@spart#1{%
    {\parindent \z@
     \head@style
     \normalfont
⟨!type2⟩      \Large \PartFont \noindent #1\par}%
⟨type2⟩      \Large \PartFont \indent #1\par}%
     \nobreak
     \vskip 3ex
     \@afterheading}
⟨/artikel⟩
```

```
⟨*rapport | boek⟩
\def\@spart#1{%
    {\centering
     \interlinepenalty \@M
     \normalfont
     \Large \PartFont #1\par}%
    \@endpart}
```

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

First we fill the current page.

```
\def\@endpart{\vfil\newpage
```

Then, when we are in twosided mode and chapters are supposed to be on right hand sides, we produce a completely blank page.

```
⟨!boek⟩             \if@twoside
                  \if@openright
                    \null
                    \thispagestyle{empty}%
                    \newpage
                  \fi
⟨!boek⟩             \fi
```

When this was a two column document we have to switch back to two column mode.

```
               \if@tempswa
                 \twocolumn
               \fi}
⟨/rapport | boek⟩
```

8.2.6 Chapters

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

```
⟨*rapport | boek⟩
\newcommand*\chapter{\if@openright\cleardoublepage\else\clearpage\fi
                     \thispagestyle{plain}%
```

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

```
                     \global\@topnum\z@
```

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

```
                     \@afterindentfalse
                     \secdef\@chapter\@schapter}
```

`\@chapter` This macro is called when we have a numbered chapter. When *secnumdepth* is larger than -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.

```
\def\@chapter[#1]#2{%
    \ifnum \c@secnumdepth >\m@ne
⟨boek⟩      \if@mainmatter
        \refstepcounter{chapter}%
        \typeout{\@chapapp\space\thechapter.}%
        \addcontentsline{toc}{chapter}%
                    {\protect\numberline{\thechapter}#1}%
⟨∗boek⟩
      \else
        \addcontentsline{toc}{chapter}{#1}%
      \fi
⟨/boek⟩
    \else
      \addcontentsline{toc}{chapter}{#1}%
    \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.

```
      \chaptermark{#1}%
      \addtocontents{lof}{\protect\addvspace{10\p@}}%
      \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.

```
      \if@twocolumn
        \@topnewpage[\@makechapterhead{#2}]%
      \else
        \@makechapterhead{#2}%
        \@afterheading
      \fi}
```

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

```
\newcommand*\ChapFont{\bfseries}
```

`\@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.

```
\def\@makechapterhead#1{%
⟨!boek⟩  \vspace*{50\p@ \@plus 5\p@}%
⟨boek⟩  \vspace*{50\p@ \@plus 20\p@}%
  {\setlength\parindent{\z@}%
   \setlength\parskip  {\z@}%
   \head@style \normalfont
```

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

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

`\@schapter` 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}
```

`\@makeschapterhead` The macro above uses `\@makeschapterhead`⟨*text*⟩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{%
⟨!boek⟩  \vspace*{50\p@\@plus 5\p@}%
⟨boek⟩  \vspace*{50\p@\@plus 20\p@}%
  {\setlength\parindent{\z@}%
    \setlength\parskip{\z@}%
    \head@style
    \normalfont
    \Large \ChapFont  #1\par
    \nobreak
    \vskip 40\p@
  }}
⟨/rapport | boek⟩
```

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 `\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{%
⟨*type1 | type3⟩
  \@startsection {section}{1}{\z@}%
    {-2\baselineskip\@plus -1\baselineskip \@minus -.5\baselineskip}%
⟨/type1 | type3⟩
⟨*type2⟩
  \@startsection {section}{1}{\unitindent}%
    {2\baselineskip\@plus \baselineskip \@minus .5\baselineskip}%
⟨/type2⟩
⟨type1⟩    {.5\baselineskip}%
⟨type2 | type3⟩    {.01\baselineskip}%
    {\normalfont\large\SectFont}}
```

`\SectFont` The font used to typeset the sections is stored in this maro.

```
\newcommand*\SectFont{\bfseries}
```

`\subsection` This gives a normal heading with white space above the heading, the title set in `\normalsize\bfseries`, and no indentation on the first paragraph.

```
\newcommand*\subsection{%
⟨*type1 | type3⟩
  \@startsection{subsection}{2}{\z@}%
    {-1\baselineskip\@plus -.5\baselineskip \@minus -.25\baselineskip}%
⟨/type1 | type3⟩
⟨*type2⟩
  \@startsection{subsection}{2}{\unitindent}%
    {1\baselineskip\@plus .5\baselineskip \@minus .25\baselineskip}%
⟨/type2⟩
⟨type1⟩    {.25\baselineskip}%
⟨type2 | type3⟩    {.01\baselineskip}%
    {\normalfont\normalsize\SSectFont}}
```

`\SSectFont` The font used to typeset the subsections is stored in this maro.

```
\newcommand*\SSectFont{\bfseries}
```

`\subsubsection` This gives a normal heading with white space above the heading, the title set in `\normalsize\tm`, and no indentation on the first paragraph.

```
\newcommand*\subsubsection{%
⟨*type1 | type3⟩
  \@startsection{subsubsection}{3}{\z@}%
    {-1\baselineskip plus -.5\baselineskip minus -.25\baselineskip}%
⟨/type1 | type3⟩
⟨*type2⟩
  \@startsection{subsubsection}{3}{\unitindent}%
    {1\baselineskip plus .5\baselineskip minus .25\baselineskip}%
⟨/type2⟩
⟨type1⟩    {.25\baselineskip}%
⟨type2 | type3⟩    {.01\baselineskip}%
    {\normalfont\normalsize\SSSectFont}}
```

`\SSSectFont` The font used to typeset the subsubsections is stored in this maro.

```
⟨artikel & (type1 | type3)⟩\newcommand*\SSSectFont{\rmfamily}
⟨type2⟩\newcommand*\SSSectFont{\scshape}
⟨rapport | boek⟩\newcommand*\SSSectFont{\slshape}
```

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

```
\newcommand*\paragraph{%
⟨!type2⟩   \@startsection{paragraph}{4}{\z@}%
⟨type2⟩   \@startsection{paragraph}{4}{\unitindent}%
    {3.25ex \@plus1ex \@minus.2ex}%
    {-1em}%
    {\normalfont\normalsize\ParaFont}}
```

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

```
⟨!type2⟩\newcommand*\ParaFont{\slshape}
⟨type2⟩\newcommand*\ParaFont{\scshape}
```

`\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`.

```
\newcommand*\subparagraph{%
⟨!type2⟩   \@startsection{subparagraph}{5}{\parindent}%
⟨type2⟩   \@startsection{subparagraph}{5}{\unitindent}%
    {3.25ex \@plus1ex \@minus .2ex}%
    {-1em}%
    {\normalfont\normalsize\SParaFont}}
```

`\SParaFont` The font used to typeset the subparagraphs is stored in this maro.

```
\newcommand*\SParaFont{\slshape}
```

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

```
⟨*artikel⟩
\newcommand*\HeadingFonts[7]{%
  \renewcommand*\TitleFont{#1}%
  \renewcommand*\PartFont{#2}%
  \renewcommand*\SectFont{#3}%
  \renewcommand*\SSectFont{#4}%
  \renewcommand*\SSSectFont{#5}%
  \renewcommand*\ParaFont{#6}%
  \renewcommand*\SParaFont{#7}}
⟨/artikel⟩
⟨*rapport | boek⟩
\newcommand*\HeadingFonts[8]{%
  \renewcommand*\TitleFont{#1}%
  \renewcommand*\PartFont{#2}%
  \renewcommand*\ChapFont{#3}%
  \renewcommand*\SectFont{#4}%
  \renewcommand*\SSectFont{#5}%
```

```
    \renewcommand*\SSSectFont{#6}%
    \renewcommand*\ParaFont{#7}%
    \renewcommand*\SParaFont{#8}}
⟨/rapport | boek⟩
```

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 `\@listK` is called, where 'K' denotes 'i', ''i', ... , 'vi'. (I.e., `\@listiii` is called for a third-level list.) By convention, `\@listK` should set `\leftmargin` to `\leftmarginK`.

`\leftmargin` `\leftmargini` For efficiency, level-one list's values are defined at top level, and `\@listi` is defined to set only `\leftmargin`.

`\leftmarginii` `\leftmarginiii` `\leftmarginiv` `\leftmarginv` `\leftmarginvi`

```
⟨!type2⟩\setlength\leftmargini  {\unitindent}
⟨type2⟩\setlength\leftmargini  {\othermargin}
\setlength\leftmarginii {\othermargin}
\setlength\leftmarginiii{\othermargin}
\setlength\leftmarginiv {\othermargin}
\setlength\leftmarginv  {\othermargin}
\setlength\leftmarginvi {1em}
```

Here we set the top level leftmargin.

```
\setlength\leftmargin    {\leftmargini}
```

`\labelsep` `\labelwidth` `\labelsep` is the distance between the label and the text of an item; `\labelwidth` is the width of the label.

```
\setlength  \labelsep  {5\p@}
\setlength  \labelwidth{\leftmargini}
\addtolength\labelwidth{-\labelsep}
```

`\partopsep` When the user leaves a blank line before the environment an extra vertical space of `\partopsep` is inserted, in addition to `\parskip` and `\topsep`.

```
\setlength\partopsep{\z@}
```

`\topsep` Extra vertical space, in addition to `\parskip`, added above and below list and paragraphing environments.

```
\setlength\topsep{\z@}
```

`\@beginparpenalty` `\@endparpenalty` These penalties are inserted before and after a list or paragraph environment. They are set to a bonus value to encourage page breaking at these points.

\@itempenalty This penalty is inserted between list items.

```
\@beginparpenalty -\@lowpenalty
\@endparpenalty   -\@lowpenalty
\@itempenalty     -\@lowpenalty
```

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

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

```
\def\@listi{%
⟨!type2⟩             \leftmargin\unitindent
⟨type2⟩             \leftmargin\leftmargini
⟨!type2⟩             \labelsep.5em%
⟨type2⟩             \labelsep.45em%
            \labelwidth\leftmargin
            \advance\labelwidth-\labelsep
            \parsep \z@
⟨!type3⟩             \topsep 0\p@ \@plus\p@
⟨type3⟩             \topsep -.5\parskip \@plus\p@
            \itemsep 0\p@ \@plus1\p@}
\let\@listI\@listi
```

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

```
\@listi
```

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

```
\def\@listii {\leftmargin\leftmarginii
⟨!type2⟩              \labelsep  .5em%
⟨type2⟩              \labelsep  .3em%
              \labelwidth\leftmarginii
              \advance\labelwidth-\labelsep
⟨!type3⟩              \topsep    0\p@ \@plus\p@
⟨type3⟩              \topsep    -.5\parskip\@plus\p@
              \parsep    \z@
              \itemsep   \z@ \@plus\p@}
\def\@listiii{\leftmargin\leftmarginiii
⟨!type2⟩              \labelsep  .5em%
⟨type2⟩              \labelsep  .3em%
              \labelwidth\leftmarginiii
              \advance\labelwidth-\labelsep
⟨!type3⟩              \topsep    0\p@ \@plus\p@
⟨type3⟩              \topsep    -.5\parskip\@plus\p@
              \parsep    \z@
              \partopsep \z@ \@plus\p@
              \itemsep   \z@ \@plus\p@}
```

```
\def\@listiv {\leftmargin\leftmarginiv
⟨!type2⟩               \labelsep  .5em%
⟨type2⟩               \labelsep  .3em%
               \labelwidth\leftmarginiv%
               \advance\labelwidth-\labelsep
⟨!type3⟩               \topsep    0\p@ \@plus\p@
⟨type3⟩               \topsep    -.5\parskip\@plus\p@
               \parsep    \z@
               \itemsep   \z@ \@plus\p@}
\def\@listv  {\leftmargin\leftmarginv
⟨!type2⟩               \labelsep  .5em%
⟨type2⟩               \labelsep  .3em%
               \labelwidth\leftmarginv
               \advance\labelwidth-\labelsep%
⟨!type3⟩               \topsep    0\p@ \@plus\p@
⟨type3⟩               \topsep    -.5\parskip\@plus\p@
               \parsep    \z@
               \itemsep   \z@ \@plus\p@}
\def\@listvi {\leftmargin\leftmarginvi
⟨!type2⟩               \labelsep  .5em
⟨type2⟩               \labelsep  .3em
               \labelwidth\leftmarginvi
               \advance\labelwidth{-\labelsep}%
⟨!type3⟩               \topsep    0\p@ \@plus\p@
⟨type3⟩               \topsep    -.5\parskip\@plus\p@
               \parsep    \z@
               \itemsep   \z@ \@plus\p@}
```

8.3.2 Enumerate

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

`\theenumi` `\theenumii` `\theenumiii` `\theenumiv`

The counters are already defined in `latex.dtx`, but their representation is changed here.

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

`\labelenumi` `\labelenumii` `\labelenumiii` `\labelenumiv`

The label for each item is generated by the commands `\labelenumi` ... `\labelenumiv`.

```
\newcommand*\labelenumi{\theenumi.}
\newcommand*\labelenumii{(\theenumii)}
\newcommand*\labelenumiii{\theenumiii.}
\newcommand*\labelenumiv{\theenumiv.}
```

`\p@enumii` `\p@enumiii` `\p@enumiv`

The expansion of `\p@enumN\theenumN` defines the output of a `\ref` command when referencing an item of the Nth level of an enumerated list.

```
\renewcommand*\p@enumii{\theenumi}
\renewcommand*\p@enumiii{\theenumi(\theenumii)}
\renewcommand*\p@enumiv{\p@enumiii\theenumiii}
```

enumerate We want to have different label positioning on different levels of list. To acheive this we have to redefine the enumerate environment.

```
\renewenvironment{enumerate}{%
  \ifnum \@enumdepth >\thr@@
    \@toodeep
  \else
    \advance\@enumdepth \@ne
    \edef\@enumctr{enum\romannumeral\the\@enumdepth}%
    \expandafter
    \list
      \csname label\@enumctr\endcsname
         {\usecounter{\@enumctr}%
⟨type1⟩            \ifnum \@listdepth=1
⟨*type1 | type3⟩
              \if@revlabel
                \def\makelabel##1{\hskip .5\unitindent{##1\hfil}}%
              \else
⟨!type3⟩                \def\makelabel##1{\hfil##1}
⟨type3⟩                \def\makelabel##1{##1\hfil}
              \fi
⟨/type1 | type3⟩
⟨type1⟩            \else
⟨type1 | type2⟩              \def\makelabel##1{##1\hfil}%
⟨type1⟩            \fi
         }%
  \fi}
```

We try to suppress spaces after these list constructs.

```
{\global\@ignoretrue \endlist}
```

8.3.3 Itemize

\labelitemi
\labelitemii
\labelitemiii
\labelitemiv

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

```
\newcommand\labelitemi  {\labelitemfont \textbullet}
\newcommand\labelitemii {\labelitemfont \bfseries \textendash}
\newcommand\labelitemiii{\labelitemfont \textasteriskcentered}
\newcommand\labelitemiv {\labelitemfont \textperiodcentered}
```

\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

```
\renewcommand\labelitemfont{%
   \fontseries\seriesdefault
   \fontshape\shapedefault\selectfont}
```

which resets series and shape doesn't touch the family.

```
\newcommand\labelitemfont{\normalfont}
```

`itemize` We want to have differen label positioning on different levels of list. To acheive this we have to redefine the itemize environment.

```
\renewenvironment{itemize}{%
  \ifnum \@itemdepth >\thr@@
    \@toodeep
  \else
    \advance\@itemdepth \@ne
    \edef\@itemitem{labelitem\romannumeral\the\@itemdepth}%
    \expandafter
    \list
      \csname\@itemitem\endcsname
         {%
⟨type1⟩            \ifnum \@listdepth=1\relax
⟨*type1 | type3⟩
             \if@revlabel
               \def\makelabel##1{\hskip .5\unitindent{##1\hfil}}\else
⟨type1⟩              \def\makelabel##1{\hfil##1}
⟨type3⟩              \def\makelabel##1{##1\hfil}
             \fi
⟨/type1 | type3⟩
⟨type1⟩          \else
⟨type1 | type2⟩            \def\makelabel##1{##1\hfil}
⟨type1⟩          \fi
           }%
  \fi}
```

We try to suppress spaces after these list constructs.

```
{\global\@ignoretrue \endlist}
```

8.3.4 Description

`description` The description environment is defined here – while the itemize and enumerate environments are defined in `latex.dtx`.

```
\newenvironment{description}
               {\list{}{\labelwidth\z@ \itemindent-\leftmargin
                        \let\makelabel\descriptionlabel}}
               {\endlist}
```

`\descriptionlabel` To change the formatting of the label, you must redefine `\descriptionlabel`.

```
\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 litle whitespace around them in the document classes `artikel1` and `rapport1`.

`center` Add a litle surrounding whitespace.

```
⟨*type1⟩
\def\center
   {\topsep=.25\baselineskip \@plus .1\baselineskip
                             \@minus .1\baselineskip
    \trivlist \centering\item[]}
\let\endcenter\endtrivlist
```

`flushleft` Add a litle surrounding whitespace.

```
\def\flushleft
   {\topsep=.25\baselineskip \@plus .1\baselineskip
                             \@minus .1\baselineskip
    \trivlist \raggedright\item[]}
\let\endflushleft=\endtrivlist
```

`flushright` Add a litle surrounding whitespace.

```
\def\flushright
   {\topsep=.25\baselineskip \@plus .1\baselineskip
                             \@minus .1\baselineskip
    \trivlist \raggedleft\item[]}
\let\endflushright=\endtrivlist
⟨/type1⟩
```

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

```
\def\verbatim{%
⟨*type1 | type2⟩
  \topsep=.25\baselineskip \@plus .1\baselineskip
                           \@minus .1\baselineskip
   \@verbatim
⟨/type1 | type2⟩
⟨type1⟩   \leftskip\unitindent
⟨type2⟩   \leftskip\z@
⟨*type3⟩
  \topsep=-.5\parskip
  \@verbatim
⟨/type3⟩
  \frenchspacing\@vobeyspaces \@xverbatim}
⟨type1⟩\def\endverbatim{\if@newlist \leavevmode\fi\endtrivlist}
```

8.5 Defining new environments

8.5.1 Abstract

`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 boeks.

```
⟨!boek⟩\if@titlepage
  \newenvironment{abstract}{%
      \titlepage
      \null\vfil
      \@beginparpenalty\@lowpenalty
      \hbox{\SectFont \abstractname}%
      \@endparpenalty\@M
      \noindent\ignorespaces}
     {\par\vfil\null\endtitlepage}
```

When we are not making a seperate 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.

```
⟨*artikel | rapport⟩
\else
  \newenvironment{abstract}{%
      \if@twocolumn
        \section*{\abstractname}%
      \else
        \small
⟨*type1 | type3⟩
        \bgroup\rightskip=\unitindent
        \hbox{\SectFont \abstractname}%
        \noindent\ignorespaces
⟨/type1 | type3⟩
```

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

```
⟨*type2⟩
        \hbox{\hskip\unitindent\SectFont \abstractname}%
        \list{}{\setlength\listparindent{\unitindent}%
                \setlength\parindent    {\z@}%
                \setlength\leftmargin   {\unitindent}%
                \setlength\rightmargin  {\unitindent}%
                \setlength\parsep       {\z@}}%
        \item[]%
⟨/type2⟩
      \fi}
```

Which implies that the definition of `\end{abstract}` is also different.

```
⟨!type2⟩       {\if@twocolumn\else\par\egroup\fi}
⟨type2⟩       {\if@twocolumn\else\par\endlist\fi}
\fi
⟨/artikel | rapport⟩
```

8.5.2 Verse

`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`.

```
\newenvironment{verse}
               {\let\\\@centercr
                \list{}{\itemsep\z@
                        \itemindent-1.5em%
                        \listparindent\itemindent
                        \rightmargin\leftmargin
                        \advance\leftmargin1.5em}%
                \item\relax}
               {\endlist}
```

8.5.3 Quotation

`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.

```
\newenvironment{quotation}
               {\list{}{%
⟨!type2⟩                         \listparindent\z@
⟨type2⟩                         \listparindent\unitindent
⟨boek⟩                         \listparindent1.5em%
                        \itemindent\listparindent
                        \rightmargin\leftmargin
                        \parsep\z@ \@plus\p@}%
                \item\relax}
               {\endlist}
```

8.5.4 Quote

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

```
\newenvironment{quote}
               {\list{}{\rightmargin\leftmargin}%
                \item\relax}
               {\endlist}
```

8.5.5 Theorem

`\@begintheorem` `\@opargbegintheorem` `\@endtheorem` 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.

```
\def\@begintheorem#1#2{%
    \vskip\baselineskip \labelsep=.5em%
    \trivlist
```

```
    \item[\hskip \labelsep{\bfseries #1\ #2}]\slshape}
\def\@opargbegintheorem#1#2#3{%
    \vskip\baselineskip \labelsep=.5em%
    \trivlist
    \item[\hskip \labelsep{\bfseries #1\ #2\ (#3)}]\slshape}
\def\@endtheorem{\endtrivlist \vskip\baselineskip}
```

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.

```
\newenvironment{titlepage}
    {
⟨boek⟩      \cleardoublepage
      \if@twocolumn
        \@restonecoltrue\onecolumn
      \else
        \@restonecolfalse\newpage
      \fi
      \thispagestyle{empty}%
      \if@compatibility
        \setcounter{page}\z@
⟨*artikel | rapport⟩
      \else
        \setcounter{page}\@ne
⟨/artikel | rapport⟩
      \fi}
     {\if@restonecol\twocolumn \else \newpage \fi
⟨artikel | rapport⟩      \setcounter{page}\@ne
      }
```

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.

```
⟨*artikel⟩
\newcommand*\appendix{\par
  \setcounter{section}{0}%
  \setcounter{subsection}{0}%
  \gdef\thesection{\@Alph\c@section}}
⟨/artikel⟩
```

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.

```
⟨*rapport | boek⟩
\newcommand*\appendix{\par
  \setcounter{chapter}{0}%
  \setcounter{section}{0}%
  \gdef\@chapapp{\appendixname}%
  \gdef\thechapter{\@Alph\c@chapter}}
⟨/rapport | boek⟩
```

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`.

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

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

```
\setlength\tabcolsep{6\p@}
```

`\arrayrulewidth` The width of rules in the array and tabular environments is given by `\arrayrulewidth`.

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

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

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

8.6.2 Tabbing

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

```
\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.

```
⟨type1⟩\def\@minipagerestore{\parindent\unitindent}
⟨*type3⟩
```

```
\def\@minipagerestore{%
              \parskip=.5\baselineskip \@plus .1\baselineskip
                                       \@minus .1\baselineskip}
⟨/type3⟩
```

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

```
\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`.

```
\setlength\fboxsep{3\p@}
\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.

```
⟨artikel⟩\renewcommand*\theequation{\@arabic\c@equation}
⟨*rapport | boek⟩
\@addtoreset{equation}{chapter}
\renewcommand*\theequation{%
  \ifnum \c@chapter>\z@ \thechapter.\fi\@arabic\c@equation}
⟨/rapport | boek⟩
```

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

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

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

```
% \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.

`\ext@TYPE` The file extension indicating the file on which the contents list for float type `TYPE` is stored. For example, `\ext@figure` = 'lof'.

`\fnum@TYPE` A macro to generate the figure number for a caption. For example, `\fnum@TYPE` == 'Figure `\thefigure`'.

`\@makecaption`⟨*num*⟩⟨*text*⟩ A macro to make a caption, with ⟨*num*⟩ the value produced by `\fnum@...` and ⟨*text*⟩ the text of the caption. It can assume it's in a `\parbox` of the appropriate width. 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 `\@float` and `\end@float`, which are defined in `latex.dtx`.

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

The environment is ended by `\end@float`. E.g., `\figure` == `\@float`figure, `\endfigure` == `\end@float`.

8.7.1 Figure

Here is the implementation of the figure environment.

`\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.

```
⟨*artikel⟩
\newcounter{figure}
\renewcommand*\thefigure{\@arabic\c@figure}
⟨/artikel⟩
⟨*rapport | boek⟩
\newcounter{figure}[chapter]
\renewcommand*\thefigure{%
  \ifnum\c@chapter>\z@\thechapter.\fi\@arabic\c@figure}
⟨/rapport | boek⟩
```

`\fps@figure` Here are the parameters for the floating objects of type 'figure'.

`\ftype@figure`
`\ext@figure`
`\num@figure`

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

`figure` And the definition of the actual environment. The form with the `*` is used for
`figure*` double column figures.

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

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.

```
⟨*artikel⟩
\newcounter{table}
\renewcommand*\thetable{\@arabic\c@table}
⟨/artikel⟩
⟨*rapport | boek⟩
\newcounter{table}[chapter]
\renewcommand*\thetable{%
  \ifnum\c@chapter>\z@\thechapter.\fi\@arabic\c@table}
⟨/rapport | boek⟩
```

`\fps@table` Here are the parameters for the floating objects of type 'table'.

`\ftype@table` `\ext@table` `\num@table`

```
\def\fps@table{tbp}
\def\ftype@table{2}
\def\ext@table{lot}
\def\fnum@table{\tablename\nobreakspace\thetable}
```

`table` `table*` 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, ⟨*number*⟩, the number of the floating object and ⟨*text*⟩, the text of the caption. Usually ⟨*number*⟩ contains a string such as 'Figure 3.2'. The macro can assume it is called inside a `\parbox` of right width, with `\normalsize`.

`\abovecaptionskip` `\belowcaptionskip` 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@}
```

The definition of this macro is `\long` in order to allow more then one paragraph in a caption.

```
\long\def\@makecaption#1#2{%
  \vskip\abovecaptionskip
```

We want to see if the caption fits on one line on the page, therefore we first typeset it in a temporary box.

```
  \sbox\@tempboxa{{\CaptionLabelFont#1:} \CaptionTextFont#2}%
```

We can the measure its width. It that is larger than the current `\hsize` we typeset the caption as an ordinary paragraph.

```
  \ifdim \wd\@tempboxa >\hsize
    {\CaptionLabelFont#1:} \CaptionTextFont#2\par
```

If the caption fits, we center it. Because this uses an `\hbox` directly in vertical mode, it does not execute the `\everypar` tokens; the only thing that could be needed here is resetting the 'minipage flag' so we do this explicitly.

```
  \else
    \global \@minipagefalse
    \hb@xt@\hsize{\hfil\box\@tempboxa\hfil}%
  \fi
  \vskip\belowcaptionskip}
```

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

```
\newcommand*\CaptionLabelFont{\relax}
\newcommand*\CaptionTextFont{\relax}
```

`\CaptionFonts` To change the fonts that are used to typeset captions this macro can be used.

```
\newcommand*\CaptionFonts[2]{%
  \renewcommand*\CaptionLabelFont{#1}%
  \renewcommand*\CaptionTextFont{#2}%
  }
```

8.8 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 *and* in math mode. They are provided for compatibility, but one should start using the `\text...` and `\math...` commands instead. These commands are defined using `\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.

`\rm` `\tt` `\sf` The commands to change the family. When in compatibility mode we select the 'default' font first, to get LaTeX2.09 behviour.

```
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\sf}{\normalfont\sffamily}{\mathsf}
\DeclareOldFontCommand{\tt}{\normalfont\ttfamily}{\mathtt}
```

\bf The command to change to the bold series. One should use \mdseries to explicitly switch back to medium series.

```
\DeclareOldFontCommand{\bf}{\normalfont\bfseries}{\mathbf}
```

\sl \it \sc 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.

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
\DeclareOldFontCommand{\sl}{\normalfont\slshape}{\relax}
\DeclareOldFontCommand{\sc}{\normalfont\scshape}{\relax}
```

\cal \mit 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'.

```
\DeclareRobustCommand*\cal{\@fontswitch\relax\mathcal}
\DeclareRobustCommand*\mit{\@fontswitch\relax\mathnormal}
```

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

```
\DeclareRobustCommand*\em{%
  \@nomath\em
  \ifdim\fontdimen\@ne\font>\z@
    \upshape
  \else
    \slshape
  \fi}
```

9 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 \thesection. 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.

`\@dottedtocline{`⟨*level*⟩`}{`⟨*indent*⟩`}{`⟨*numwidth*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}`
`\@regtocline{`⟨*level*⟩`}{`⟨*title*⟩`}{`⟨*page*⟩`}`

⟨***level***⟩ An entry is produced only if⟨ *level*⟩ <= value of the *tocdepth* counter. Note, `\chapter` is level 0, `\section` is level 1, etc.

⟨***indent***⟩ The indentation from the outer left margin of the start of the contents line.

⟨***numwidth***⟩ The width of a box in which the section number is to go, if ⟨*title*⟩ includes a `\numberline` command.

`\@pnumwidth` `\@tocrmarg` `\@dotsep`

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

`\@pnumwidth` The width of a box in which the page number is put.

`\@tocrmarg` The right margin for multiple line entries. One wants `\@tocrmarg` ≥ `\@pnumwidth`

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

```
\newcommand*\@pnumwidth{1.55em}
\newcommand*\@tocrmarg {2.55em}
\newcommand*\@dotsep{4.5}
⟨artikel⟩\setcounter{tocdepth}{3}
⟨!artikel⟩\setcounter{tocdepth}{2}
```

9.1.1 Table of Contents

`\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.

```
\newcommand*\tableofcontents{%
⟨*rapport | boek⟩
    \if@twocolumn
      \@restonecoltrue\onecolumn
    \else
      \@restonecolfalse
    \fi
```

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

```
    \chapter*{\contentsname}%
⟨/rapport | boek⟩
⟨artikel⟩    \section*{\contentsname}%
```

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 LaTeX2.09 version as it will produce a serious bug if used in two-column mode (see, LaTeX pr/3285). However `\tableofcontents` is always typeset in one-column mode in these classes, therefore the somewhat inconsistent setting has been retained for compatibility reasons.

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

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

```
    \@starttoc{toc}%
⟨!artikel⟩    \if@restonecol\twocolumn\fi
    }
```

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

```
⟨*type3⟩
\def\@starttoc#1{\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}
⟨/type3⟩
```

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

```
\newcommand*\@regtocline[3]{%
  \ifnum #1>\c@tocdepth
  \else
    \vskip\z@\@plus.2\p@
    {\hangindent\z@ \@afterindenttrue \interlinepenalty\@M
     \leftskip\unitindent
     \rightskip\unitindent\@plus 1fil
     \parfillskip\z@
     \@tempdima\unitindent
⟨type2⟩      \advance\@tempdima by \othermargin
     \parindent\z@
     \leavevmode
     \hbox{}\hskip -\leftskip\relax
     \ifnum#1<0\toc@case{#2}\else
```

```
      \toc@font#1 #2\fi\nobreak
      \hskip 1em \nobreak{\slshape #3}\par
      }%
  \fi}
```

`\numberline` This internal macro is redefined for the `artikel2` document class.

```
⟨type2⟩\def\numberline#1{\hb@xt@\@tempdima{\hfil#1\hskip.3em}}
```

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

```
\if@oldtoc
  \newcommand*\toc@font[1]{\relax}
\else
```

A line of the table of contents contains `\numberline` and the section number as its first two elements. We don't want to set the section number using `\toc@font`, therefor 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.

```
  \newcommand*\toc@font[4]{%
    \ifx\Hy@toclinkstart#2%
      \def\@next{#2\toc@font#1#3#4}%
    \else
      \def\@next{#2{#3}\toc@fontsel#1#4}%
    \expandafter\fi\@next}
  \newcommand*\toc@fontsel[1]{%
⟨*artikel⟩
    \ifcase#1\relax
⟨type2⟩    \Large\bfseries
    \or\bfseries
    \or\slshape
    \or\rmfamily
⟨/artikel⟩
⟨*rapport | boek⟩
    \ifcase#1\relax
    \bfseries
    \or\slshape
    \or\rmfamily
⟨/rapport | boek⟩
    \fi}
```

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

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

```
\AtBeginDocument{%
  \ifx\hyper@linkstart\undefined
  \else
```

In the contentslines for chapters, sections etc., the command selection of the appropriate font needs to come after the code that 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 pagenumber, set flush right.

```
      {\leavevmode
        \large \bfseries #1\hfil \hb@xt@\@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
⟨artikel⟩       \if@compatibility
        \global\@nobreaktrue
        \everypar{\global\@nobreakfalse\everypar{}}%
⟨artikel⟩       \fi
    \endgroup
  \fi}
```

Then we can introduce our new definition.

```
\else
   \newcommand*\l@part{%
     \ifnum \c@tocdepth >-2\let\l@@part\relax
⟨artikel⟩      \addpenalty\@secpenalty
⟨!artikel⟩      \addpenalty{-\@highpenalty}%
     \addvspace{2.25em \@plus \p@}%
     \def\l@@part{\@regtocline{-1}}%
   \fi\l@@part}
\fi
```

`\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

```
⟨∗rapport | boek⟩
\if@oldtoc
\newcommand*\l@chapter[2]{%
    \addpenalty{-\@highpenalty}%
    \vskip 1.0em \@plus\p@
```

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.

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

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

```
    \leavevmode \bfseries
```

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

```
    \advance\leftskip\@tempdima
    \hskip -\leftskip
    #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss #2%
```

```
                                    \kern-\p@\kern\p@}\par
    \penalty\@highpenalty
  \endgroup}
```

Then we present our new definition.

```
\else
  \newcommand*\l@chapter{\@regtocline{0}}
\fi
⟨/rapport | boek⟩
```

`\l@section` In the artikel document classes the entry in the table of contents for sections looks much like the chapter entries for the rapport and boek document classes.

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

```
⟨*artikel⟩
\if@oldtoc
\newcommand*\l@section[2]{%
    \addpenalty\@secpenalty
    \addvspace{1.0em \@plus\p@}%
```

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 paramters.

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

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

```
    \leavevmode \bfseries
```

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

```
    \advance\leftskip\@tempdima
    \hskip -\leftskip
    #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss #2%
                                    \kern-\p@\kern\p@}\par
  \endgroup}
```

The new definition:

```
\else
  \newcommand*\l@section{\@regtocline{1}}
\fi
⟨/artikel⟩
```

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

```
⟨*rapport | boek⟩
\if@oldtoc
```

```
  \newcommand*\l@section      {\@dottedtocline{1}{1.5em}{2.3em}}
\else
  \newcommand*\l@section      {\@regtocline{1}}
\fi
⟨/rapport | boek⟩
```

\l@subsection
\l@subsubsection
\l@paragraph
\l@subparagraph

All lower level entries are defined using the macro \@dottedtocline or \@regtocline (see above).

```
\if@oldtoc
⟨*artikel⟩
  \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}}
⟨/artikel⟩
⟨*rapport | boek⟩
  \newcommand*\l@subsection   {\@dottedtocline{2}{3.8em}{3.2em}}
  \newcommand*\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
  \newcommand*\l@paragraph    {\@dottedtocline{4}{10em}{5em}}
  \newcommand*\l@subparagraph {\@dottedtocline{5}{12em}{6em}}
⟨/rapport | boek⟩
\else
  \newcommand*\l@subsection   {\@regtocline{2}}
  \newcommand*\l@subsubsection{\@regtocline{3}}
  \newcommand*\l@paragraph    {\@regtocline{4}}
  \newcommand*\l@subparagraph {\@regtocline{5}}
\fi
```

9.1.2 List of figures

\listoffigures

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

```
\newcommand*\listoffigures{%
⟨*rapport | boek⟩
    \if@twocolumn
      \@restonecoltrue\onecolumn
    \else
      \@restonecolfalse
    \fi
    \chapter*{\listfigurename}%
⟨/rapport | boek⟩
⟨artikel⟩    \section*{\listfigurename}%
    \@mkboth{\MakeUppercase{\listfigurename}}%
            {\MakeUppercase{\listfigurename}}%
    \@starttoc{lof}%
⟨rapport | boek⟩    \if@restonecol\twocolumn\fi
    }
```

\l@figure

This macro produces an entry in the list of figures.

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

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{%
⟨*rapport | boek⟩
    \if@twocolumn
      \@restonecoltrue\onecolumn
    \else
      \@restonecolfalse
    \fi
    \chapter*{\listtablename}%
⟨/rapport | boek⟩
⟨artikel⟩    \section*{\listtablename}%
    \@mkboth{\MakeUppercase{\listtablename}}%
            {\MakeUppercase{\listtablename}}%
    \@starttoc{lot}%
⟨rapport | boek⟩    \if@restonecol\twocolumn\fi
    }
```

`\l@table` 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 thebibliography environment.

```
\newcommand*\newblock{}
```

`thebibliography` The 'thebibliography' environment executes the following commands:

`\renewcommand\newblock{\hskip .11em \@plus .33em \@minus .07em}` – Defines the "closed" format, where the blocks (major units of information) of an entry run together.

`\sloppy` – Used because it's rather hard to do line breaks in bibliographies,

`\sfcode‘\.=1000\relax` – Causes a '.' (period) not to produce an end-of-sentence space.

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

When an empty 'thebibliography' environment is found, a warning is issued.

```
\newenvironment{thebibliography}[1]
⟨*artikel⟩
     {\section*{\refname}%
       \@mkboth{\MakeUppercase\refname}{\MakeUppercase\refname}%
⟨/artikel⟩
⟨*!artikel⟩
     {\chapter*{\bibname}%
       \@mkboth{\MakeUppercase\bibname}{\MakeUppercase\bibname}%
⟨/!artikel⟩
      \list{\@biblabel{\@arabic\c@enumiv}}%
           {\settowidth\labelwidth{\@biblabel{#1}}%
            \leftmargin\labelwidth
            \advance\leftmargin\labelsep
            \@openbib@code
            \usecounter{enumiv}%
            \let\p@enumiv\@empty
            \renewcommand*\theenumiv{\@arabic\c@enumiv}}%
      \sloppy\clubpenalty4000\widowpenalty4000%
      \sfcode`\.\@m}
     {\def\@noitemerr
       {\@latex@warning{Empty `thebibliography' environment}}%
      \endlist}
```

`\newblock` 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}
```

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

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

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

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

`\@cite` 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 seperate 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}{%
  \if@twocolumn
    \@restonecolfalse
  \else
    \@restonecoltrue
  \fi
⟨artikel⟩  \twocolumn[\section*{\indexname}]%
⟨!artikel⟩  \twocolumn[\@makeschapterhead{\indexname}]%
  \@mkboth{\MakeUppercase{\indexname}}{\MakeUppercase{\indexname}}%
  \thispagestyle{plain}\parindent\z@
```

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

```
  \columnseprule \z@
  \columnsep 35\p@
  \parskip\z@ \@plus .3\p@\relax
  \let\item\@idxitem
  }{%
```

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

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

`\@idxitem` `\subitem` `\subsubitem` Thsee macros are used to format the entries in the index.

```
\newcommand*\@idxitem  {\par\hangindent 40\p@}
\newcommand*\subitem   {\@idxitem\hspace*{20\p@}}
\newcommand*\subsubitem{\@idxitem\hspace*{30\p@}}
```

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

```
\newcommand*\indexspace{\par\vskip10\p@\@plus5\p@\@minus3\p@\relax}
```

9.4 Footnotes

`\footnoterule` Usually, footnotes are separated from the main body of the text by a small rule. This rule is drawn by the macro `\footnoterule`. The standard LaTeX document classes make sure that the rule takes no vertical space (see `plain.tex`) and compensate for the natural heigth of the rule of 0.4pt by adding the right amount of vertical skip. For the `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 `\footnoterule` a positive height[1].

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.

```
\renewcommand*\footnoterule{%
  \kern-3\p@
⟨*type1 | type3⟩
  \kern.5\baselineskip
```

[1] This should perhaps have been done by increasing the value of `\skip\footins`, but changing that now would mean changing the formatting of existing documents. (`JLB, 08/09/1997`)

```
    \hrule\@width\unitindent
    \kern.4\baselineskip
⟨/type1 | type3⟩
⟨*type2⟩
    \hrule\@width 3\unitindent
    \kern 2.6\p@
⟨/type2⟩
}
```

`\c@footnote` Footnotes are numbered within chapters in the rapport and boek document styles.

```
% \newcounter{footnote}
⟨!artikel⟩\@addtoreset{footnote}{chapter}
```

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

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

```
\long\def\@xmakefntext#1#2{%
%<!type3>  \parindent=.5\unitindent
%<type3>  \parindent=\z@\parskip=.5\baselineskip
  \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 `\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 `\unitindent` is large enough for our purposes.

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

```
⟨*type1 | type3⟩
\newcommand*\@makefntext{\@xmakefntext{\normalfont\@thefnmark.}}
\newcommand*\@xmakefntext[1]{%
    \parindent\z@
    \def\labelitemi{\textendash}\@revlabeltrue
    {\setbox0\hbox {#1\hskip.5em plus 1fil}
      \dimen0=2\wd0\relax
      \ifdim\dimen0>\unitindent
        \global\unitindent\dimen0\relax
        \@indentset
      \fi}
    \leavevmode\hb@xt@.5\unitindent{#1\hfil}}
⟨/type1 | type3⟩
```

For the `artikel2` document class we have a simpler definition of `\@makefntext`.

```
⟨*type2⟩
\newcommand\@makefntext[1]{%
    \parindent\othermargin
    \noindent\hb@xt@\othermargin{\normalfont\@thefnmark\hfil\relax}#1}
⟨/type2⟩
```

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

```
%\renewcommand\@makefnmark{\hbox{\@textsuperscript{\normalfont\@thefnmark}}}
```

10 Initialization

10.1 Words

`\contentsname` `\listfigurename` `\listtablename` 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.

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

`\refname` `\bibname` `\indexname`

```
⟨artikel⟩\newcommand*\refname{References}
⟨rapport | boek⟩\newcommand*\bibname{Bibliography}
\newcommand*\indexname{Index}
```

`\figurename` `\tablename`

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

```
    \partname
 \chaptername 1700 \newcommand*\partname{Part}
\appendixname 1701 ⟨rapport | boek⟩\newcommand*\chaptername{Chapter}
\abstractname 1702 \newcommand*\appendixname{Appendix}
     \seename 1703 ⟨!boek⟩\newcommand*\abstractname{Abstract}
     \andname 1704 \newcommand*\seename{see}
              1705 \newcommand*\andname{and}
```

10.2 Date

`\today` This macro uses the TEX primitives `\month`, `\day` and `\year` to provide the date of the LATEX-run.

```
1706 \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`.

```
1707 \def\today{\ifcase\month\or
1708   January\or February\or March\or April\or May\or June\or
1709   July\or August\or September\or October\or November\or December\fi
1710   \space\number\day, \number\year}
```

10.3 Two column mode

`\columnsep` This gives the distance between two columns in two column mode.

```
1711 \setlength\columnsep{10\p@}
```

`\columnseprule` This gives the width of the rule between two columns in two column mode. We have no visible rule.

```
1712 \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.

```
1713 ⟨!boek⟩\pagestyle{plain}
1714 ⟨boek⟩\pagestyle{headings}
1715 \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.

```
1716 ⟨*artikel⟩
1717 \if@twoside
```

```
\else
  \raggedbottom
\fi
⟨/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.

```
\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
⟨/artikel | rapport | boek⟩
```