The KOMA–Script Package

Frank Neukam Markus Kohm

Jun 10th, 2000

This system is distributed in the hope that it will be useful, but *without any warranty*, without even the implied warranty of *merchantability* or *fitness for a particular purpose*.

Contents

1 General Remarks

1.1 Introduction

Originally the Script package was written by Frank Neukam for LaTeX 2.09. He wanted to implement European rules of typography and paper formats as documented in [JTsch87]. The first version appeared in June 1992 with great resonance.

Markus Kohm reimplemented version 2.0 from scratch for LaTeX 2ε (together with its LaTeX 2.09 compatibility mode) as a docstrip module. Most of this documentation is taken directly from Frank Neukam's Guide to Script 2.0 (in German). Please refer to the German version of this documentation for compatibility questions concerning the original Script 2.0 package for LaTeX.

You can reach the author at the following address or in the newsgroup de.comp.text.tex:

MausNet: Markus Kohm @ HD

snail mail: Markus Kohm, Fichtenstraße 63, 68535 Edingen-Neckarhausen, Germany

1.2 Legal Stuff

This system is distributed in the hope that it will be useful, but **without any warranty**, without even the implied warranty of **merchantability** or **fitness for a particular purpose**.

Permission is granted to distribute verbatim copies of all files together with all other files of the KOMA–Script bundle. You are allowed to modify one or all of the dtx-files if and only if you change the name of the modified file. You are allowed to distribute the modfied files but only **together** with the unmodified versions. You have to document all changes and the name of the author of the changes.

No other permissions to copy or distribute this file in any form are granted.

You are **not allowed** to take money for the distribution or use of either this file or a changed version, except for a nominal charge for copying etc.

Standard class	KOMA–Script class
article	scrartcl
report	scrreprt
book	scrbook
letter	scrlettr

Table 1.1: Relation between standard classes and KOMA–Script classes.

1.3 Compatibility Remarks

All commands which are available in the standard classes are implemented also in the KOMA–Script package. Additionally a few macros has been added, and some of the original commands behave differently. Nevertheless it should be possible to process any document with KOMA–Script which has been written for the standard classes.

Table 1.1 shows the relation between standard classes and KOMA–Script classes.

2 The New Layout

2.1 The New Fonts

`\sfb`

All occurrences of Extended Boldface (i.e. captions, index, and `description` environment) has been replaced with Sansserif Bold Extended (`\sffamily\bfseries`). This is a font without serifs having semibold weight which can be selected with the old font selection command `\sfb` or with the new font commands `\sffamily\bfseries`. Note that the smallest available size is 10pt, thus `\tiny\sfb` will select normal bold extended automatically.

2.2 The Type Area

2.2.1 Fundamental Remarks

The type area defines the size of the printed area of a page[1] and its exact position, the text width depends on the selected font and font size (60–70 characters per line give optimal readability).

Jan Tschichold suggests in [JTsch87] a simple algorithm to get good–looking results even for 'non–classical' paper formats like DIN A4 (which is the default paper format for all KOMA–Script classes). Type area and paper format have the same width/length ratios, even the margins have simple proportions (inner margin/outer margin and upper margin/lower margin respectively have the proportion 1:2, without the `twoside` option inner and outer margin are equal).

2.2.2 The Construction

The construction is based on a devision of the width and height of a page in n equal parts which will be called HLU and VLU (horizontal/vertical length unit).

[1] not counting headers, footers, and footnotes

Example: If we divide a DIN A4 page (210 mm × 297 mm) in ten parts ($n = 10$) we get a horizontal length unit HLU = 210 mm ÷ 10 = 21 mm and a vertical length unit VLU = 297 mm ÷ 10 = 29.7 mm.

Now we simply set in case of double sided printing the inner margin[2] to 1 HLU and the outer margin to 2 HLU. In case of single page printing inner and outer margin will be both 1.5 HLU. Analogously we set the upper margin to 1 VLU and the lower margin to 2 VLU. Thus the text area will have $(n - 3)$ HLU width and $(n - 3)$ VLU height. Marginal notes will be output in the outer margin and limited to a width of 1.5 HLU for twosided and 1 HLU for onesided printing.

Example: In case of dividing the page into 6 units ($n = 6$) left and right margin together have 3 HLU, upper and lower margin have 3 VLU. Thus the text area is exactly half as wide and high as the entire page.

TEX's demand that `\textwidth` should be a multiple of the `typewriter` character width will not be accomplished with the above described algorithm. But even the original styles don't follow this rule, and only text written entirely with `typewriter` type will be affected. See section 2.2.4 for a method how to adjust the type area for `\tt` fonts.

The second rule of the type area construction, namely that `\textheight` should equal `\topskip` plus an integer multiple of `\baselineskip`, will be fulfilled. This ensures that for pages completely filled with ordinary text (i.e. pages without captions, tables, displayed material etc.) paragraphs don't need to be stretched to assure that the lowest text line will align with the lower margin of the text area. Thus the type area height will be at most one line larger than the exact construction's value.

2.2.3 Selection of the Type Area

It's easy to see that for small n the text area will be small too compared to the page size, and that for large n the text area fills the page almost completely (it's hardly possible to output marginal notes correctly for n values greater than 12). Modifying n enables the user to change the size of the type area within a great range having always the guarantee of an optimal construction.

Optionally you can provide a binding correction *BCOR* which will be taken into account while constructing the type area. The algorithm simply reduces the page width by *BCOR*, computes the type area again and finally increases the inner margin by *BCOR*.

Example: Assuming that a glue binding reduces the visible page width of 210 mm (ISO/DIN A4) by an amount of 8 mm (see below how to specify *BCOR*) the new

[2] here we mean the *visible* margin, not the margin values taken from the `odd-` and `evensidemargin` registers, we will evaluate these parameters to get the wanted visible margins.

text font size	*DIV*
10pt	8
11pt	10
12pt	12

Table 2.1: *DIV* defaults

text proportion will be 202:297 instead of the original 210:297. Now for example the type area will be centered again on the visible part of the page for onesided printing.

`\typearea[BCOR]{DIV}`

The command `\typearea` will compute the type area anew. Because internal values like `\oddsidemargin` etc. will be changed, this command is only allowed in the preamble (i.e. before `\begin{document}`). The optional argument *BCOR* will adjust the binding correction, whereas `{`*DIV*`}` sets n as described above. *BCOR* needs a valid TEX measure unit (pt, mm, cm, in, ...), *DIV* must be an integer greater than 3.

Using scrartcl, scrreprt, or scrbook selects *DIV* as described in table 2.1 (without selecting a binding correction) and calls `\typearea` automatically. They will do so by inputting the package typearea which contains the `\typearea` macro.

It's also possible to let KOMA–Script compute a good value for *DIV*: if you choose one of the values 1 or 2 for *DIV* (which would make no sense in the normal context), KOMA–Script tries to find an appropriate type area assuming that an optimal legibility is given if 69 characters per line are used, additionally too small margins are avoided. A value 0 for *DIV* has a special meaning also: the predefined default values for the given paper and font size are used. If no default values exist, KOMA–Script will compute them. A value 3 for *DIV* has a special meaning too: see *DIVclassic* below.

You can use table 2.2 as a starting point for selecting *DIV* manually. The table values assume `a4paper` with twosided printing and $BCOR = 0$ pt (without considering the adjustment between the height of the text area and the font size dependent value of `\baselineskip` which can increase the text area by at most 14.5 pt $\simeq$ 5.1 mm).

`DIV`*factor*
`BCOR`*correction*

It is also possible to pass through the type area parameters *DIV* and *BCOR* as options to the classes scrartcl, scrreprt, and scrbook to the package typearea, then it's not necessary to call `\typearea` explicitly. *factor* and *correction* should follow the option directly.

Here a small example:

	text area		margins	
DIV	width [mm]	height [mm]	top [mm]	inner [mm]
6	105.00	148.50	49.50	35.00
7	120.00	169.71	42.43	30.00
8	131.25	185.63	37.13	26.25
9	140.00	198.00	33.00	23.33
10	147.00	207.90	29.70	21.00
11	152.73	216.00	27.00	19.09
12	157.50	222.75	24.75	17.50
13	161.54	228.46	22.85	16.15
14	165.00	233.36	21.21	15.00
15	168.00	237.60	19.80	14.00

Table 2.2: Type area values in relation to *DIV*

```
\documentclass[DIV14,BCOR0.5cm]{scrreprt}
```

Also you can use the option `DIVcalc` to let KOMA–Script calculate *DIV* values automatically (this is similar to `DIV0` provided no default values are given).

Finally you can use the option `DIVclassic` to let KOMA-Script calculate *DIV* for “spätmittelalterlichen Buchseitenkanon”. At the resulting typearea textheight is almost equal to paperwidth. Fontsize and linewidth (number of characters per line) are not considered, so typearea may be as awful as can be.

2.2.4 Uncommon Type Areas

```
\areaset[BCOR]{WIDTH}{HEIGHT}
```

To construct type areas with uncommon proportions or to specify type areas which have *exactly* the values of the previous table (without adjusting to an integer multiple of `\baselineskip`) it’s also possible to use `\areaset` instead of `\typearea`. `\areaset` sets the text width to `{`*WIDTH*`}` and the text height to *HEIGHT* (without any additional corrections) and computes the margins in a similar way as the `\typearea` command. The optional argument *BCOR* will be handled as described above.

Now we can easily construct type areas which are exactly quadratic or very narrow and high. If you don’t like the positioning chosen by `\areaset`, it’s of course possible to change the margins afterwards.

Here is the example which shows how to adjust the text area having exactly a width of 30 characters in `typewriter` type:

```
\newlength{\thirtytt}
```

```
\settowidth{\thirtytt}{\texttt{123456789012345678901234567890}}
\areaset{\thirtytt}{20cm} % text height: 20cm
```

`\areaset` is only allowed in the preamble.

2.2.5 Including Headers and Footers

Sometimes it makes sense to assume that the header and/or the footer line are part of the text area while computing the type area, as shown in this documentation. Here we have a horizontal line which separates the header from the text body, thus moving the header optically nearer.

```
headinclude
headexclude
```

If you switch on the `headinclude` option the commands `\typearea` and `\areaset` will include the header line to compute the type area. The opposite option is `headexclude`.

```
footinclude
footexclude
```

These are the analogous options for including and excluding the footer line.

You can specify the options directly in the `\documentclass` macro which is easiest. If you specify the options in the `\usepackage` macros, don't forget to load the typearea package explicitly.

2.2.6 Different Paper Formats

As with the standard classes it is possible to select different paper formats using the options `a3paper`, `landscape` etc. Contrary to the original classes the computation of the type area will be done implicitly by calling the `typearea` package using the above described rules.

To select a paper format which is not defined in the typearea package you can choose between two possibilities. Either you change `\paperwidth` and `\paperheight` manually and call then the `\typearea` command (or `\areaset`), or you create additional options and write them into a file `typearea.cfg`. Such defined options can be used similar to standard options.

```
\SetDIVList{list}
```

Besides the paper size it is generally possible to define font size dependent *DIV* factors.

The command `\SetDIVList` creates such a list which contains a *DIV* value (in braces) for each font size, starting with 10 pt, followed by the value for 11 pt, then 12 pt and so forth. Not defining a list with `\SetDivList` is equal to `\SetDIVList{{8}{10}{12}}`. If you select a font size without a value in this list, the default value 10 will be used.

Example 1: A poem booklet should be printed on ISO/DIN A5 paper. It is sufficient to select the `a5paper` option:

```
\documentclass[a5paper]{scrreprt}
```

Example 2: The text area of a schedule using *landscape* ISO/DIN A4 paper should be exactly 270 mm × 190 mm. The following code implements this:

```
\documentclass[a4paper]{scrreprt}
\areaset{270mm}{190mm}
```

Additionally you need a (system dependent) command forcing the printer driver to use landscape format.

Example 3: The poem booklet of example 1 will now have a binding correction of 1.5 mm, and the type area should be a bit larger:

```
\documentclass[a5paper,DIV12,BCOR1.5mm]{scrreprt}
```

Example 4: We want to use endless paper with the dimensions $8\frac{1}{4}$ in × 12 in. To have a good–looking type area we could say

```
\paperwidth 8.25in
\paperheight 12in
\typearea{10}
```

which is best to be written as a new option into `typearea.cfg`, assuming that the name of this option is 'endless' we can now simply say

```
\documentclass[endless]{scrreprt}
```

Here a complete example of an entry in `typearea.cfg`:

```
\DeclareOption{endless}
  {\setlength\paperheight {8.25in}%
   \setlength\paperwidth  {12in}%
   \SetDIVList{{8}{10}{12}}}
```

Note that you can also use the typearea package together with the standard classes, however it is *not* a good idea to use typearea package with the scrlettr letter class which has a special, fixed type area (this class is only documented in the German version of this documentation).

2.3 Additional Changes

2.3.1 Default Fontsize

Without specifying a different default fontsize explicitly in the `\documentclass` command 11 pt fonts will be used as the default (as shown in this documentation) for all classes except scrlettr (which uses 12 pt). The LaTeX 2$_{\varepsilon}$ standard classes use 10 pt fonts.

2.3.2 Chapter Headings and Enumeration

Chapter headings of the classes scrbook and scrreprt will be typeset without a separate line for the word "chapter" (as implemented in book and report, KOMA–Script classes don't use this word), instead the chapter number will directly precede the caption text. To have a distinct separation between chapters and the appendix (which now only differs in the style of enumeration) it is highly recommended to insert `\part*{Appendix}` between the last chapter and the appendix.

```
bigheadings
normalheadings
smallheadings
```

Some people prefer smaller headings. Use one of the above three options to change the size, KOMA–Script uses `bigheadings` as the default.

```
\raggedsection
```

Starting with version 2.2h no hyphenation will be used in headings. If you want to allow hyphenation you should write

```
\let\raggedsection\relax
```

Usually `\raggedsection` is identical to `\raggedright`.

```
pointednumbers
pointlessnumbers
```

Starting with version 2.2c no final dot will be used after ordinary section numbers if you use Arabic numbers (e.g. 1, 1.1, 1.2.3 etc., see [Duden20, p. 18, rule 5]). If you use Roman numerals or letters also, a final dot will be used (e.g. I., I.1., A., A.1.1 etc, see [Duden20, p. 18, rule 6]).

This behaviour can be controlled with `pointednumbers` and `pointlessnumbers`: the former option will always force and the latter will always suppress a final dot. Note that the opposite of `pointednumbers` is *not* `pointlessnumbers`!

Currently you have to run LaTeX twice to assure correct behaviour of the final dot.

2.3.3 Headings and Page Numbers

The srcbook class uses `headings` as the default page style, scrreprt and scrartcl have `plain` as the default.

If you use the option `twoside`, page numbers will be set into the outer margins, otherwise they will be centered. scrbook has `twoside` as the default.

In contrast to the standard classes headings will not be set in uppercase, and the default font will be *slanted* (was *italics*). Chapter headings will not use the word "CHAPTER", and the page number is always in the footer (which would be empty otherwise). An advantage of these changes is the possibility to have much longer chapter headings without truncation.

```
headsepline
headnosepline
footsepline
footnosepline
```

To have a horizontal line below the header line use the class option `headsepline` to turn it on (`headnosepline` to turn it off). This only works for the page styles `headings` and `myheadings` since `plain` has no header line, `headsepline` automatically activates `headinclude` (and similarly `headnosepline` sets `headexclude`). Default for scrbook, scrreprt and scrartcl is `headnosepline`.

The analogous options for footers are `footinclude`, `footexclude`, `footsepline`, and `footnosepline`. This also works for the `plain` page style.

Example: In the documentation you are just reading we have

```
\documentclass[headsepline]{...}
```

to get a horizontal line below the page headers (this also activates `headinclude`).

2.3.4 Some Additional Cosmetic Changes

A small horizontal region (1 em) is reserved for the footnote number alone, i.e. the footnote text always starts with this offset from the left margin. In multiline footnotes the following lines will be indented 0.5 em again to improve reading.

Paragraphs will be output without vertical space, being indented 1 em (this is 1 `\quad`).

`\frenchspacing` will be activated in spite of the fact that the german package starting with version 2.4 activates it too. This is subject to change.

The second level of the `enumerate` environment will be 'a)' instead of '(a)', the third level of the `itemize` environment is now '$\triangleright$' instead of '$*$'.

Author and date entries in titles will not be `\large` but `\Large`.

Multiline captions (as they can happen in `table` or `figure` environments) will be indented so that "Figure . . . :" corresponds to a hanging indentation of that length.

There are many further changes which will be hardly noticed, e.g. the space between the columns in `twocolumn` mode has been enlarged to 1 cc (1 Cicero $\simeq$ 12 pt). Compare the `.dtx`–files of the original and the SCRIPT classes if you interested in details.

3 Additional Commands

3.1 Enhancements of Abstract and Titlepage

```
abstractoff
abstracton
```

You can suppress the ‘Abstract’ header with the option `abstractoff`, this is the default for scrartcl and scrreprt. scrbook has no abstract. The opposite option is `abstracton`.

```
\extratitle{extratitle}
\titlehead{text}
\subject{text}
\publishers{publisher}
\uppertitleback{text}
\lowertitleback{text}
\dedication{dedication}
```

The following commands enhance `\maketitle` of the Script main classes:

- `\extratitle` creates a separate page with the extratitle[1] on it. It will be printed as normal text by default, and the user has completely freedom to customize. The backside (even if twoside printing is on) is always empty.
- `\titlehead` prints some additional information in the head part of the main title page, e.g. an address of an institute.
- `\subject` prints directly above the main title an additional subject (e.g. “dissertation”) in a somewhat larger font.
- `\publishers` creates the publisher’s name in the lower part of the main title page.
- If twosided printing is activated it is possible to write some information on the back page of the title page which would be otherwise empty. `\uppertitleback`

[1] In former times the hard cover was sometimes missing, thus the very first page, the extratitle, protected the book so that it didn’t become dirty. Today it will be frequently used as an extrapage before the main title to hold the publisher’s name, the ISDN number etc.

and `\lowertitleback` print text near the upper and lower margin respectively (cf. the “Haftungsausschluß” in this documentation).

- `\dedication` creates an own dedication page, with *dedication* centered and with a slightly larger font. The back page is always empty.

Example:

```
\titlehead{Institute for Sinology \hfill SS1993
           \\12 University Street
           \\Unitown 3456}
\subject{Dissertation}
\title{About Chinese Food}
\author{Wun Tun}
\lowertitleback{This paper was created with the
                text formatting system \TeX.}
\maketitle
```

Usually the title starts on page 1. With an optional argument (e.g. `\maketitle[-1]`) it’s possible to adjust the starting page number.

3.2 Table of Contents

Starting with version 2.3h some additional options are available to change the appearance of the table of contents.

```
liststotoc
```

Usually there is no entry for the lists of figures and lists of tables in the table of contents. Specifying the option `liststotoc` changes this to get an entry (without a number).

```
bibtotoc
bibtotocnumbered
```

`bibtotoc` is similar to `liststotoc`. `bibtotocnumbered` will additionally treat the bibliography as an own chapter having a number.

```
idxtotoc
```

Similar to `bibtotoc`, but will never be an own chapter. The entry in the table of contents should only indicate that an index exists.

variable	affected area	default
`\capfont`	caption texts	`\normalfont`
`\caplabelfont`	caption labels	`\normalfont`
`\descfont`	item texts of the `description` environment	`\sffamily\bfseries`
`\headfont`	column titles	`\slshape`
`\pnumfont`	pagination	`\normalfont`
`\sectfont`	all headers and titles	`\sffamily\bfseries`

Table 3.1: Commands to adjust fonts, its meaning and default values

3.3 The “`\xxxxfont`” Variables

```
\capfont
\caplabelfont
\descfont
\headfont
\pnumfont
\sectfont
```

Starting with version 2.0 of the Script package a bunch of variables to control the global font changes for column titles, pagination, captions, headers and the `description` environment are implemented. This enables the user to change e.g. the header font from `\sffamily\bfseries` to `\rmfamily` with only one command—it's only necessary to write `\renewcommand{\sectfont}{\rmfamily}` (usually in the preamble). In a similar way all other fonts can be changed as shown in table 3.1. Changes of the font size are of course also possible within the above commands.

3.4 The Marginline

```
\marginline{text}
```

This command is the oneline analogon to the multiline `\marginpar` macro. It has the advantage that in `twoside` mode on left pages the margin text will be aligned with the outer text margin (i.e. flush right), contrary to `\marginpar` which treats the line as the start of a paragraph, thus printing the text flush left in the margin. Use this command for cross references and similar text snippets.

3.5 The \addchap and \addsec Command

```
\addchap[index entry]{header}
\addchap*{header}
\addsec[index entry]{header}
\addsec*{header}
```

To create an unnumbered section like a preface or a greeting you only could use the `\chapter*` or `\section*` command of the standard classes. But they neither create an index entry nor a running head which you must add manually if you want them which is not in the sense of the LaTeX philosophy.

Thus the Script package offers the new commands `\addchap` and `\addchap*` (for scrbook and scrreprt) together with `\addsec` and `\addsec*` (for scrartcl) which behaves like `\chapter` and `\section` except the missing enumeration. The starred commands are very similar to the starred original commands with the difference that running heads will be cleared additionally. As with the original commands you can create index entries different from the header with an optional argument.

3.6 The \minisec Command

```
\minisec{Header}
```

Sometimes it is desirable to have a kind of header which is connected to the surrounding text without too great vertical spaces.

Appearance of the new \minisec command:
The command `\minisec` ("mini section") will do this job, the argument will be printed with `\sffamily\bfseries`, and a vertical space of 1.5 ex will be added before the header (and none after). Mini sections produce no entry in the index.

3.7 The labeling Environment

```
\begin{labeling}[separator]{longest title}
\item[title] ...
   ⋮
\end{labeling}
```

Appearance: As you can see here in an example for the `labeling` environment, text will be printed like an ordinary paragraph but will have a title in front of it (here: "Appearance").

The text is not restricted to one paragraph.

Syntax: `labeling` is an ordinary environment with one parameter and an optional argument. *longest title* is the longest argument including all font change commands. If you don't need a separator you can omit the optional argument *separator*.

Example: The `labeling` environment you are just reading starts with

```
\begin{labeling}{Appearance:}
\item[Appearance:] As you can see here in an example for the
    ;labeling; environment, ...
\item[Syntax:] ;labeling; is an ordinary...
    .
    .
\end{labeling}
```

3.8 Preamble texts for Bibliography and Index

`\setindexpreamble{`*preamble text*`}`
`\setbibpreamble{`*preamble text*`}`

Sometimes you may want to insert some additional text after the index and bibliography headings, for example to explain to your readers the meaning of some special formatting of the entries. From version 2.6c on, the SCRIPT package provides the commands `\setindexpreamble` and `\setbibpreamble` for this purpose. They default to insert nothing.

The contents of `{`*preamble text*`}` will be inserted after the headings with no additional formatting, so if you wish to insert some vertical space between the preamble text and the following items, you should put something like `\par\bigskip` at the end of `{`*preamble text*`}`. Note that `\indexpreamble` will *not* wrap across pages.

4 Internationalization

In cooperation with Werner Lemberg (a7971428@unet.univie.ac.at) the language interface of the standard classes has been modified in a compatible way to support languages which do not follow English grammatical or syntactical rules. Even in English not all possibilities of labeling e.g. a figure are supported: cf. “Figure 1” (supported) and “First Figure” (not supported).[1] Additionally some languages will suppress spaces between the running number and the \figurename or set them differently.

Three modification levels have been introduced starting with KOMA–Script version 2.2c (dated 25–May–1995) to gain a great flexibility without loosing simplicity.

Level 1 are the standard macros like \figurename as defined in the standard classes. german.sty, as an example, redefines this bunch of names with German equivalents. Related to these macros are the font changing commands \sectfont and its relatives.

Level 2 is the modification of the counters itself (e.g. \thesection could use Chinese characters instead of Arabic numbers).

Level 3 finally allows for complete control over the ordering of counters, titles, and spaces. It consists of the commands \chaptermarkformat, \sectionmarkformat, \subsectionmarkformat, \partformat, \chapterformat, \figureformat, \tableformat, and \captionformat.

Here a simplified example for a Chinese \part command (Chinese characters have been replaced with a transcription):

```
\newcommand\prepartname{di}
\newcommand\postpartname{bu}
\renewcommand\partformat{\prepartname~\thepart~\postpartname}
```

It must be stated that some of the new features are not used within the Script package by default since most sectioning commands don’t use \chaptername etc. The letter class scrlettr.cls doesn’t accept the \...format commands, scrpage.cls does (see section 5).

[1] With ‘not supported’ it is meant that you must do tricky things or even modify some of the user invisible low level macros to achieve the wanted results.

5 The scrpage Package

The scrpage package defines a new interface to make the page layout much more flexible to define as it would be with \pagestyle. It has some similarities with the fancyheadings package [POost94] but tries new solutions of old problems. As an example, there is a simple interface for the novice and a more sophisticated one for the experienced user.

scrpage works with the standard LaTeX classes also.

5.1 The Simple Interface

```
\headfont
\pnumfont
```

\headfont selects the font for the headers and footers, \pnumfont selects the page number font. Default definition for both commands is \normalfont. Here an example of use:

```
\renewcommand{\headfont}{\normalfont\slshape}
```

```
\headmark
```

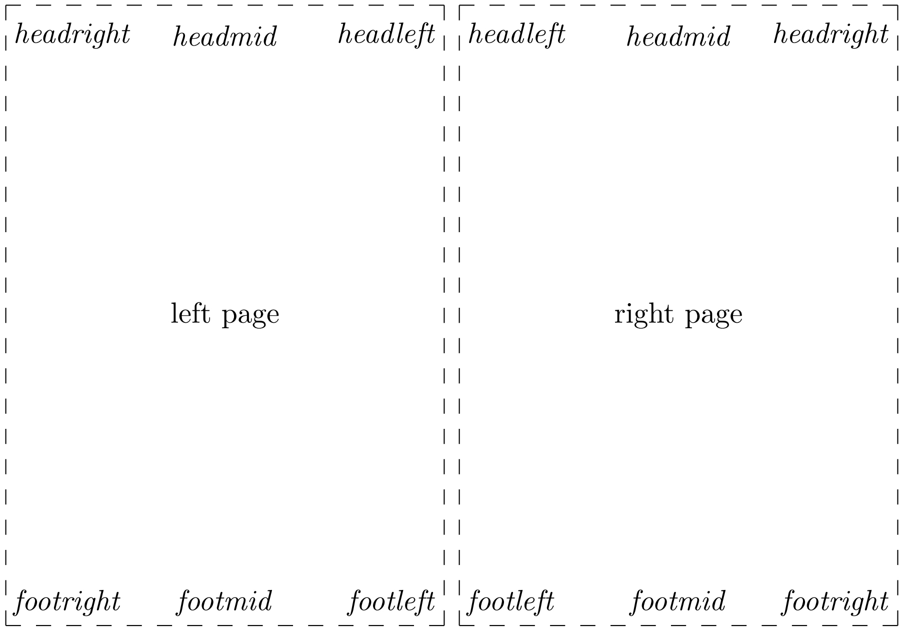
If you choose the layout option twoside, \headmark is equivalent to the LaTeX commands \leftmark on even pages and \rightmark on odd pages. If you select onesided printing, \headmark is always the same as \rightmark. Thus the scrpage package itself controls whether we are on a even or odd page, \headmark always contains the actual header line.

```
\pagemark
```

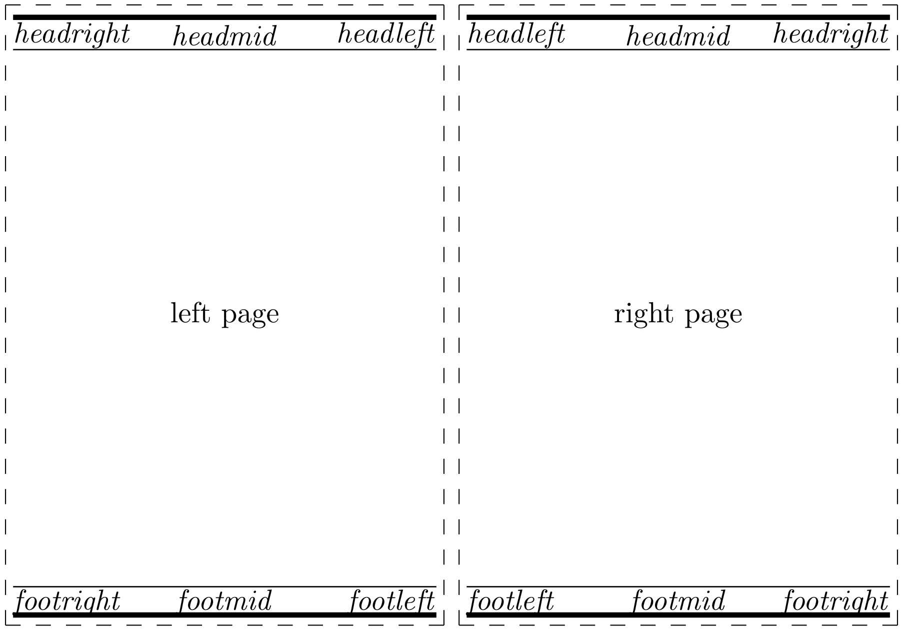
\pagemark contains the actual page number string. It will also select the correct font (using \pnumfont), contrary to the standard LaTeX command \thepage.

```
\deftripstyle{name}[olw][ilw]
              {headleft}{headmid}{headright}
              {footleft}{footmid}{footright}
```

Use this command to define a new page layout with the name *name*. The arguments *headleft*, *headmid*, *headright*, *footleft*, *footmid*, and *footright* will appear on the following places:



To create separation lines for the header and the footer respectively use the optional parameters *ilw* (inner line width) and *olw* (outer line width), if only one optional argument is given *ilw* is set (similar to class option `headsepline`). The values of the parameters are the line thicknesses. Here an example with *olw*=2 pt, *ilw*=0.5 pt:



A newly defined or redefined page style will be only active after a `\pagestyle` or `\thispagestyle` command.

To redefine the `myheadings` page style you should use the following macro:

```
\deftripstyle*{name}[olw][ilw]
               {headleft}{headmid}{headright}
               {footleft}{footmid}{footright}
```

This command creates a *my* version of a page style. The difference to the unstarred form is that sectioning commands like `\chapter` will not alter the page headings, only `\markleft`, `\markright` and `\markboth` will do that.

5.2 The Experts' Interface

```
\defpagestyle{name}{header definition}{footer definition}
\newpagestyle{name}{header definition}{footer definition}
\renewpagestyle{name}{header definition}{footer definition}
\providepagestyle{name}{header definition}{footer definition}
```

All these commands define a new page style with the following structure of the header and footer definition:

```
(ull,ulw){evenpage}{oddpage}{onepage}(lll,llw)
```

ulw and *llw* are the widths of the upper and the lower line of the header or footer respectively, *ull* and *lll* are the lengths of the upper and lower line (`\deftripstyle` and `\defstripstyle*` always use `\textwidth` as the length thus these parameters are omitted). *evenpage* and *oddpage* define the corresponding headers or footers and will be used if twosided output is selected, *onepage* will be used for onesided output.

The various flavours of the commands are similar to the variants of the `\newcommand` macro: `\defpagestyle` will create the page style *name* and overwrites an existing page style of this name without warning, `\newpagestyle` outputs a warning if a page style with this name already exists, `\renewpagestyle` will do the opposite, i.e. outputs a warning if a page style with this name does not already exist, and `\providepagestyle` defines only a new page style if no page style with this name has been defined before (it will issue an entry in the log file if *name* already exists).

Finally an example which was used to realize the header and footer lines of this chapter:

```
\renewpagestyle{plain}{(\textwidth,0pt)%         header line
                       {\hfill}%
                       {\hfill}%
                       {\hfill}%
                       (\textwidth,0pt)}%
                      {(\textwidth,.4pt)%        footer line
                       {\pagemark\hfill}%
                       {\hfill\pagemark}%
                       {\hfill\pagemark\hfill}%
                       (\textwidth,1pt)}
\renewpagestyle{headings}{(\textwidth,1pt)%         header
                          {\headmark\hfill}%
                          {\hfill\headmark}%
                          {\hfill\headmark\hfill}%
                          (\textwidth,.4pt)}%
                         {(\textwidth,.4pt)%        footer
                          {\pagemark\hfill%
                           Copyright \copyright\ Markus Kohm, 1994--1996}%
                          {Package \texttt{scrpage}\hfill\pagemark}%
                          {\rlap{Package \texttt{scrpage}}\hfill%
                           Copyright \copyright\ Markus Kohm, 1994--1996%
                           \hfill\llap\pagemark}%
                          (\textwidth,1pt)}
\pagestyle{headings} % this starts our new page style
```

Bibliography

[Duden20] Duden “Die deutsche Rechtschreibung” (*The German Grammatic*), volume 1, 20th printing, Dudenverlag, 1991

[CJK] Werner Lemberg: “The CJK package for LaTeX 2$_\varepsilon$”, to be found at the CTAN hosts and its mirrors

[POost94] Piet van Oostrum: ”‘Page headers and footers in LaTeX”’, available at the CTAN hosts and its mirrors

[JTsch87] Jan Tschichold: “Ausgewählte Aufsätze über die Gestalt des Buches und der Typographie” (*Selected Papers on book design and typography*), 2nd printing, Birkhäuser Verlag, Basel 1987

Index