The Obsolete Package scrpage2

Extract from former versions of the KOMA-Script Manual

Markus Kohm Jens-Uwe-Morakswi

2014-06-25

Contents

Adapting Page Headers and Footers with scrpage2

From KOMA-Script 3.12 the completely newly implemented package scrlayer-scrpage replaces the old scrpage2. The new package is a consequently developed extension of the design of scrpage2. In difference to scrpage2 it provides the extended option interface of the KOMA-Script classes. Because scrlayer-scrpage predominates scrpage2 it is recommended to not longer use scrpage2 but scrlayer-scrpage. Because of this the current version of scrpage2 is the final one. All development resources will go into scrlayer-scrpage. For more information about scrlayer-scrpage see the KOMA-Script manual.

In place of scrpage2 or scrlayer-scrpage you can of course make use of fancyhdr. However, scrpage2 and especially scrlayer-scrpage integrated markedly better with the KOMA-Script bundle. For this reason, and because at the time the forerunner to fancyhdr was missing many features, scrpage2 was developed. Naturally, scrpage2 and scrlayer-scrpage are not limited to use only with the KOMA-Script classes, but can just as easily be used with other document classes.

1.1 Basic Functionality

To understand the following description, an overview of LaTeX's fairly involved header and footer mechanism is needed. The LaTeX kernel defines the page styles `empty`, which produces a completely empty header and footer, and `plain`, which produces usually only a page number in the footer and an empty header. Apart from these, many document classes provide the style `headings`, which allows more complex style settings and running headings. The `headings` style often has a related variant, `myheadings`, which is similar except for switching off the running headings and reverting them to manual control by the user. A more detailed description is given in the page style section of the KOMA-Script manual where it is also noted that some LaTeX commands automatically switch to another page style — usually page style `plain` — for the current page.

Package scrpage2 does not distinguish between page styles with automatic, running headings and page styles with manual headings. The way to deal with automatic and manual headings is independent from the page style and so the page style is independent from the choice of automatic or manual headings. More information about this in section 1.1.2.

1.1.1 Predefined Page Styles

One of the basic features of scrpage2 is a set of predefined, configurable page styles.

`scrheadings`
`scrplain`

Package scrpage2 delivers its own page style, named `scrheadings`, which can be activated with the `\pagestyle{scrheadings}`. When this page style is in use, an appropriate `scrplain` page style is used for the plain page style. In this case *appropriate* means that this new

plain page style is also configureable by the commands introduced in section 1.1.3, which, for example, configure the header and footer width and complies within the basic layout. Neither the activation of `scrheadings` nor the attendant change to the appropriate plain page style, `scrplain`, influences the mode of manual or automatic headings (see section 1.1.2). The `scrplain` page style can also be activated directly with `\pagestyle`.

```
\lehead[scrplain-left-even]{scrheadings-left-even}
\cehead[scrplain-center-even]{scrheadings-center-even}
\rehead[scrplain-right-even]{scrheadings-right-even}
\lefoot[scrplain-left-even]{scrheadings-left-even}
\cefoot[scrplain-center-even]{scrheadings-center-even}
\refoot[scrplain-right-even]{scrheadings-right-even}
\lohead[scrplain-left-odd]{scrheadings-left-odd}
\cohead[scrplain-center-odd]{scrheadings-center-odd}
\rohead[scrplain-right-odd]{scrheadings-right-odd}
\lofoot[scrplain-left-odd]{scrheadings-left-odd}
\cofoot[scrplain-center-odd]{scrheadings-center-odd}
\rofoot[scrplain-right-odd]{scrheadings-right-odd}
\ihead[scrplain-inside]{scrheadings-inside}
\chead[scrplain-centered]{scrheadings-centered}
\ohead[scrplain-outside]{scrheadings-outside}
\ifoot[scrplain-inside]{scrheadings-inside}
\cfoot[scrplain-centered]{scrheadings-centered}
\ofoot[scrplain-outside]{scrheadings-outside}
```

The page style of scrpage2 are defined to have flexible configurable header and footer. To achieve this, the page styles include three boxes in both the header and the footer. The contents of these boxes may be modified easily. The commands modifying the content of these boxes can be seen in figure 1.1. Commands in the middle column modify the box contents on both odd and even pages. All of the commands have an optional and a mandatory argument. The option Argument influences the content of corresponding box of the plain page style, `scrplain`. The mandatory argument influences the content of the corresponding box of the page style `scrheadings`.

Example: If one wants the page number within `scrheadings` be placed in the middle of the footer, then following can be used:

```
\cfoot{\pagemark}
```

The next example shows how to place both running heading and page number in the header; the running heading inside and the page number outside:

```
\ohead{\pagemark}
\ihead{\headmark}
\cfoot{}
```

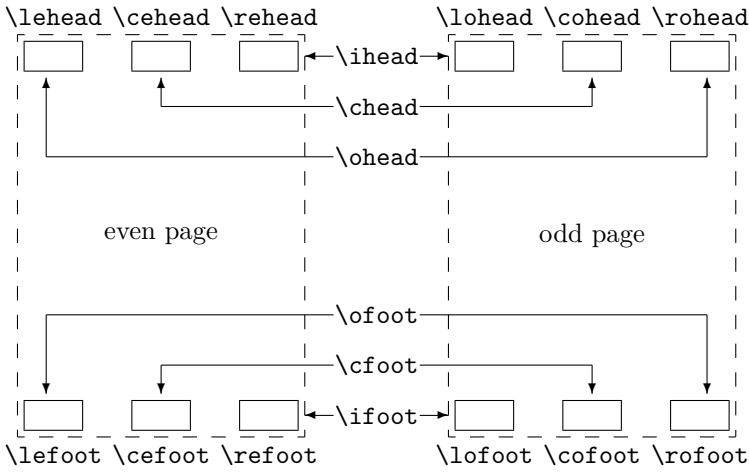


Figure 1.1: Commands for modification of page styles `scrheadings` and `scrplain` and their relationship to header and footer elements

The command `\cfoot{}` is only required in order to empty the item in the middle of the footer, which normally contains the page number.

The commands which are associated with only one item can be used for more advanced settings.

Example: Assuming one has the order to write an annual report of a company, one could use commands like this:

```
\ohead{\pagemark}
\rehead{Annual Report 2001}
\lohead{\headmark}
\cefoot{TheCompanyName Inc.}
\cofoot{Department: Development}
```

In order to keep the data in the footer synchronized with the content of the document, the footer has to be updated using `\cofoot` when a new department is discussed in the report.

As mentioned above, there is a new plain page style which corresponds to `scrheadings`. Since it should also be possible to customize this style, the commands support an optional argument with which the contents of the appropriate fields of this plain page style can be modified.

Example: The position of the page number for the page style `scrheadings` can be declared as follows:

```
\cfoot[\pagemark]{}
\ohead[]{\pagemark}
```

When the command `\chapter`, after it has started a new page, now switches to the page style `plain`, then the page number is centered in the footer.

`\clearscrheadings`
`\clearscrplain`
`\clearscrheadfoot`

If one wants to redefine both the page style `scrheadings` and the corresponding plain page style, frequently one must empty some already occupied page elements. Since one rarely fills all items with new content, in most cases several instructions with empty parameters are necessary. With the help of these three instructions the quick and thorough deletion is possible. While `\clearscrheadings` only deletes all fields of the page style `scrheadings`, and `\clearscrplain` deletes all fields of the corresponding plain page style, `\clearscrheadfoot` sets all fields of both page styles to empty.

Example: If one wants to reset the page style to the default KOMA-Script settings, independent of the actual configuration, only these three commands are sufficient:

```
\clearscrheadfoot
\ohead{\headmark}
\ofoot[\pagemark]{\pagemark}
```

Without the commands `\clearscrheadfoot`, `\clearscrheadings` and `\clearscrplain`, six commands with additional nine empty arguments would be required:

```
\ihead[]{}
\chead[]{}
\ohead[]{\headmark}
\ifoot[]{}
\cfoot[]{}
\ofoot[\pagemark]{\pagemark}
```

Of course, for a specific configuration, some of them could be dropped.

In the previous examples two commands were used which have not been introduced yet. The description of these commands follows.

`\leftmark`
`\rightmark`

These two instructions make it possible to access the running headings, which are normally meant for the left or for the right page. These two instruction are not made available by scrpage2, but directly by the LaTeX kernel. When in this section running headings of the left page or the right page are mentioned, this refers to the contents of `\leftmark` or `\rightmark`, respectively.

`\headmark`

This command gives access to the content of running headings. In contrast to `\leftmark` and `\rightmark`, one need not regard the proper assignment to left or right page.

`\pagemark`

This command returns the formatted page number. The formatting can be controlled by `\pnumfont`, introduced in section 1.1.3, page E.8, which `\pagemark` heeds automatically.

`useheadings`

The package scrpage2 is meant primarily for use of the supplied styles or for defining one's own styles. However, it may be necessary to shift back also to a style provided by the document class. It might appear that this should be done with `\pagestyle{headings}`, but this has the disadvantage that commands `\automark` and `\manualmark`, to be discussed shortly, do not function as expected. For this reason one should shift back to the original styles using `\pagestyle{useheadings}`, which chooses the correct page styles automatically for both manual and automatic running headings.

1.1.2 Manual and Running Headings

Usually there is a *my*-version of the `headings` page style. If such a page style is active, then the running headings are no longer updated no longer automatically and become manual headings. With scrpage2 a different path is taken. Whether the headings are running or manual is determined by the instructions `\automark` and `\manualmark`, respectively. The default can be set already while loading of the package, with the options `automark` and `manualmark` (see section 1.1.4, page E.14).

`\manualmark`

As the name suggests, `\manualmark` switches off the updating of the running headings and makes them manual. It is left to the user to update and provide contents for the headings. For that purpose the instructions `\markboth` and `\markright` are available.

`\automark[`*`right page`*`]{`*`left page`*`}`

v2.2

The macro `\automark` activates the automatic updating, that is, running headings. For the two parameters the designations of the document sectioning level whose title is to appear in appropriate place are to be used. Valid values for the parameters are: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, and `subparagraph`. For most of the classes use of `part` will not produce the expected result. So far only KOMA-Script classes from version 2.9s up are known to support this value. The optional argument *`right page`* is understandably meant only for double-sided documents. In the single-sided case one should normally not use it. With the help of the option `autooneside` one can also set that the optional argument in single-sided mode is ignored automatically (see section 1.1.4, page E.15).

Example: Assuming that the document uses a *book* class, whose topmost section level is *chapter*, then after a preceding `\manualmark`

```
\automark[section]{chapter}
```

restores the original behaviour. If one prefers lower section levels in running headings, the following can be used:

```
\automark[subsection]{section}
```

For the upper section level, the data of the headings is set by the command `\markboth`, while that for the lower section level by `\markright` or `\markleft`. These commands are called indirectly by the sectioning commands. The macro `\markleft` is provided by the package scrpage2 and is defined similarly to `\markright` in the LaTeX kernel. Although `\markleft` is not defined as an internal command, the direct use is not recommended.

1.1.3 Formatting of Header and Footer

The previous section concerned itself mainly with the contents of the header and footer. This is of course not sufficient to satisfy formative ambitions. Therefore we devote this section exclusively to this topic.

`\headfont`
`\footfont`
`\pnumfont`

The command `\headfont` contains the commands which determine the font of header and footer lines. Command `\footfont` contains the difference of the footer to that. The difference for the style of the page number is defined by the command `\pnumfont`.

Example: If, for example, one wants the header to be typeset in bold sans serif, the footer in non-bold sans serif, and the page number in a slanted serif style, then one can use the following definitions:

```
\renewcommand{\headfont}{\normalfont\sffamily\bfseries}
\renewcommand*{\footfont}{\normalfont\sffamily}
\renewcommand{\pnumfont}{\normalfont\rmfamily\slshape}
```

From version 2.8p of the KOMA-Script classes a new unified user interface scheme is implemented for font attributes. If scrpage2 is used together with one of these classes, then it is recommended to set up font attributes in the manner described in the KOMA-Script manual.

Example: Instead of `\renewcommand` the command `\setkomafont` should be used to configure the font attributes. The previous definitions can then be written as:

```
\setkomafont{pagehead}\normalfont\sffamily\bfseries}
\setkomafont{pagefoot}{\normalfont\sffamily}
\setkomafont{pagenumber}{\normalfont\rmfamily\slshape}
```

```
\setheadwidth[shift]{width}
\setfootwidth[shift]{width}
```

Normally the widths of header and footer lines correspond to the width of the text body. The commands `\setheadwidth` and `\setfootwidth` enable the user to adapt in a simple manner the widths to his needs. The mandatory argument *width* takes the value of the desired width of the page header or footer, while *shift* is a length parameter by which amount the appropriate item is shifted toward the outside page edge.

For the most common situations the mandatory argument *width* accepts the following symbolic values:

`paper` – the width of the paper

`page` – the width of the page

`text` – the width of the text body

`textwithmarginpar` – the width of the text body including margin

`head` – the current header width

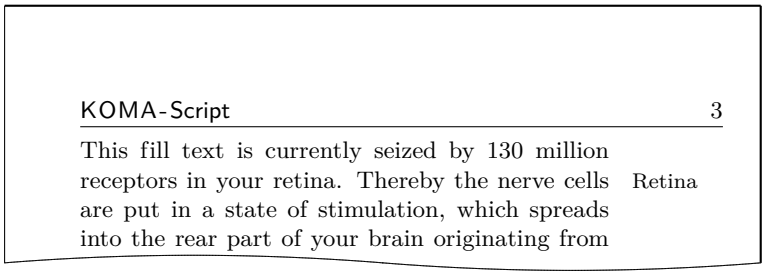
`foot` – the current footer width

The difference between `paper` and `page` is that `page` means the width of the paper less the binding correction if the package typearea is used (see the chapter about typearea in the KOMA-Script manual). Without typearea both values are identical.

Example: Assume that one wants a layout like that of *The LaTeX Companion*, where the header projects into the margin. This can be obtained with:

```
\setheadwidth[0pt]{textwithmarginpar}
```

which appears like this on an odd page:



If the footer line should have the same width and alignment, then two ways to set this up are possible. The first way simply repeats the settings for the case of the footer line:

```
\setfootwidth[0pt]{textwithmarginpar}
```

In the second way the symbolic value `head` is used, since the header already has the desired settings.

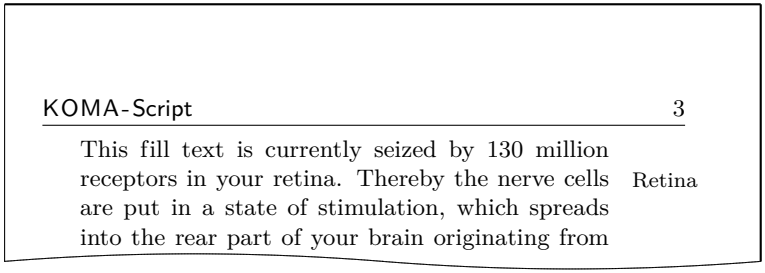
```
\setfootwidth[0pt]{head}
```

If no *shift* is indicated, i. e., without the optional argument, then the header or footer appears arranged symmetrically on the page. In other words, a value for the *shift* is determined automatically to correspond to the current page shape.

Example: Continuing with the previous example, we remove the optional argument:

```
\setheadwidth{textwithmarginpar}
```

which appears like this on an odd page:



As can be seen, the header is now shifted inward, while the header width has not changed. The shift is calculated in a way that the configuration of the typearea become visible also here.

```
\setheadtopline[length]{thickness}[commands]
\setheadsepline[length]{thickness}[commands]
\setfootsepline[length]{thickness}[commands]
\setfootbotline[length]{thickness}[commands]
```

Corresponding to the size configuration parameters of header and footer there are commands to modify the rules above and below the header and footer. But first of all the rules should be activated. See options `headtopline`, `headsepline`, `footsepline`, and `footbotline` in section 1.1.4, page E.13 for this.

`\setheadtopline` – configures the line above the header

`\setheadsepline` – configures the line below the header

`\setfootsepline` – configures the line above the footer

`\setfootbotline` – configures the line below the footer

The mandatory argument *thickness* determines how strongly the line is drawn. The optional argument *length* accepts the same symbolic values as *width* for `\setheadwidth`, as well as also a normal length expression. As long as the optional argument *length* is not assigned a value, the appropriate line length adapts automatically the width of the header or the footer.

Use `auto` in the length argument to restore this automation for the length of a line.

v2.2 The optional argument *commands* may be used to specify additional commands to be executed before the respective line is drawn. For example, such commands could be used for changing the color of the line. When using a KOMA-Script class you could also use `\setkomafont` to specify commands for one of the elements `headtopline`, `headsepline`, `footsepline`, `footbottomline`, or `footbotline`. These can then be extended via `\addtokomafont`. See the KOMA-Script manual for details on the `\setkomafont` and `\addtokomafont` commands.

```
\setheadtopline[auto]{current}
\setheadtopline[auto]{}
\setheadtopline[auto]{}[]
```

The arguments shown here for the command `\setheadtopline` are of course valid for the other three configuration commands too.

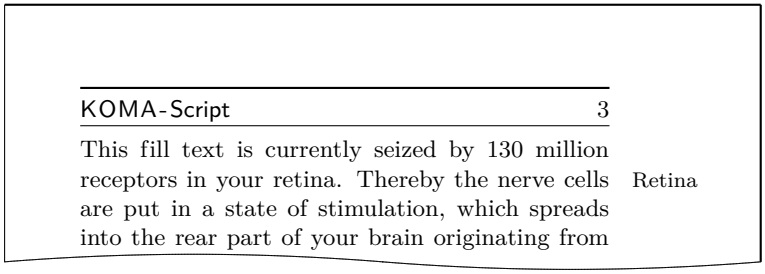
If the mandatory parameter has the value `current` or has been left empty, then the line thickness is not changed. This may be used to modify the length of the line without changing its thickness.

If the optional argument *commands* is omitted, then all command settings that might have been specified before will remain active, while an empty *commands* argument will revoke any previously valid commands.

Example: If the header, for example, is to be contrasted by a strong line of 2 pt above and a normal line of 0.4 pt between header and body, one can achieve this with:

```
\setheadtopline{2pt}
\setheadsepline{.4pt}
```

Additionally the options `headtopline` and `headsepline` have to be used preferably globally in the optional argument of `\documentclass`. In this case the result may be the following.



To specify that this line is to be drawn also, e. g., in red color, you would change the commands like this:

```
\setheadtopline{2pt}[\color{red}]
\setheadsepline{.4pt}[\color{red}]
```

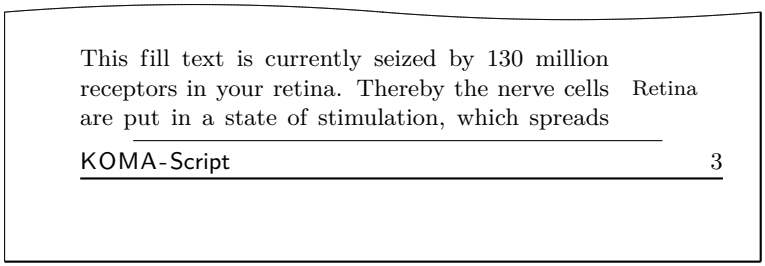
In this example, as well as in the following one, line color is activated by applying the syntax of the color package, so this package must of course be loaded. Since scrpage2 comes without built-in color handling, any package providing color support may be used.

KOMA-Script classes also support the following way of color specification:

```
\setheadtopline{2pt}
\setheadsepline{.4pt}
\setkomafont{headtopline}[\color{red}]
\setkomafont{headsepline}[\color{red}]
```

The automatic adjustment to the header and footer width is illustrated in the following example:

```
\setfootbotline{2pt}
\setfootsepline[text]{.4pt}
\setfootwidth[0pt]{textwithmarginpar}
```



Now not everyone will like the alignment of the line above the footer; instead, one would expect the line to be left-aligned. This can only be achieved with a global package option, which will be described together with other package options in the next section 1.1.4.

1.1.4 Package Options

In opposite to the KOMA-Script classes, where the most options may be changed using `\KOMAoptions` or `\KOMAoption` also after loading the class, package scrpage2 does not provide this feature currently. All options to scrpage2 have to be global options, i. e. be part of the optional argument of `\documentclass`, or package option, i. e. be part of the optional argument of `\usepackage`.

`headinclude`
`headexclude`
`footinclude`
`footexclude`

v2.3 Since KOMA-Script 3 this options should not be passed to scrpage2 any longer using `\PassOptionsToPackage` or the optional argument of `\usepackage`. Only for compatibility reason scrpage2 still declares them and pass them as `headinclude`, `headinclude=false`, `footinclude` und `footinclude=false` to package typearea weitergereicht.

`headtopline`
`plainheadtopline`
`headsepline`
`plainheadsepline`
`footsepline`
`plainfootsepline`
`footbotline`
`plainfootbotline`

Basic adjustment of the lines under and over header and footer can be made with these options. These adjustments are then considered the default for all page styles defined with scrpage2. If one of these options is used, then a line thickness 0.4 pt is set. Since there is a corresponding plain page style to the page style `scrheadings`, the corresponding line in the plain style can also be configured with the `plain...` options. These `plain` options do however work only if the corresponding options without `plain` are activated. Thus, `plainheadtopline` shows no effect without the `headtopline` option set.

With these options, it is to be noted that the appropriate page part, header or footer, is considered as a part of the text area for the calculation of the type area in case a line has been activated. This means that, if the separation line between header and text is activated with `headsepline`, then the package typearea calculates the type area in such a way that the page header is part of the text block automatically.

The conditions for the options of the preceding paragraph apply also to this automation. That means that the package typearea must be loaded after scrpage2, or that on use of a KOMA-Script class, the options `headinclude` and `footinclude` must be set explicitly with `\documentclass` in order to transfer header or footer line in the text area.

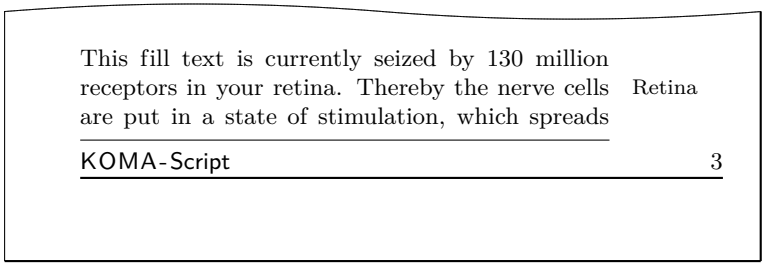
`ilines`
`clines`
`olines`

With the definition of the line lengths the case can arise where the lengths are set correctly, but the justification is not as desired because the line will be centered in the header or footer area. With the package options presented here, this specification can be modified for all page styles defined with scrpage2. The option `ilines` sets the justification in such a way that the lines align to the inside edge. The option `clines` behaves like the default justification, and `olines` aligns at the outside edge.

Example: The next example illustrates the influence of the option `ilines`. Please compare to the example for `\setfootsepline` on page E.12.

```
\usepackage[ilines]{scrpage2}
\setfootbotline{2pt}
\setfootsepline[text]{.4pt}
\setfootwidth[0pt]{textwithmarginpar}
```

The mere use of the option `ilines` leads to the different result shown below:



In contrast to the default configuration, the separation line between text and footer is now left-aligned, not centered.

`automark`
`manualmark`

These options set at the beginning of the document whether to use running headings or manual ones. The option `automark` switches the automatic updating on, `manualmark` deactivates it. Without the use of one of the two options, the setting which was valid when the package was loaded is preserved.

Example: You load the package scrpage2 directly after the document class scrreprt without any package options:

```
\documentclass{scrreprt}
\usepackage{scrpage2}
```

Since the default page style of scrreprt is `plain`, this page style is also now still active. Futhermore, `plain` means manual headings. If one now activates the page style `scrheadings` with

```
\pagestyle{scrheadings}
```

then the manual headings are nevertheless still active.

If you instead use the document class scrbook, then after

```
\documentclass{scrbook}
\usepackage{scrpage2}
```

the page style `headings` is active and the running headings are updated automatically. Switching to the page style `scrheadings` keeps this setting active. The marking commands of scrbook continue to be used.

However, the use of

```
\usepackage[automark]{scrpage2}
```

activates running headings independently of the used document class. The option does not of course affect the used page style `plain` of the class scrreprt. The headings are not visible until the page style is changed to `scrheadings`, `useheadings` or another user-defined page style with headings.

`autooneside`

This option ensures that the optional parameter of `\automark` will be ignored automatically in one-sided mode. See also the explanation of the command `\automark` in section 1.1.2, page E.7.

`komastyle`
`standardstyle`

These options determine the look of the predefined page styles `scrheadings` and `scrplain`. The option `komastyle` configures a look like that of the KOMA-Script classes. This is the default for KOMA-Script classes and can in this way also be set for other classes.

The option `standardstyle` configures a page style as it is expected by the standard classes. Furthermore, the option `markuppercase` will be activated automatically, but only if option `markusedcase` is not given.

`markuppercase`
`markusedcase`

In order to achieve the functionality of `\automark`, the package scrpage2 modifies internal commands which are used by the document structuring commands to set the running headings. Since some classes, in contrast to the KOMA-Script classes, write the headings in uppercase letters, scrpage2 has to know how the used document class sets the headings.

Option `markuppercase` shows scrpage2 that the document class uses uppercase letters. If the document class does not set the headings in uppercase letters, then the option `markusedcase` should be given. These options are not suitable to force a representation; thus, unexpected effects may occur if the given option does not match the actual behaviour of the document class.

`nouppercase`

In the previous paragraph dealing with `markuppercase` and `markusedcase`, it has been already stated that some document classes set the running headings in uppercase letters using the commands `\MakeUppercase` or `\uppercase`. Setting the option `nouppercase` allows disabling both these commands in the headers and footers. However, this is valid only for page styles defined by scrpage2, including `scrheadings` and its corresponding plain page style.

The applied method is very brutal and can cause that desired changes of normal letters to uppercase letters do not occur. Since these cases do not occur frequently, the option `nouppercase` usually affords a useful solution.

Example: Your document uses the standard class book, but you do not want the uppercase headings but mixed case headings. Then the preamble of your document could start with:

```
\documentclass{book}
\usepackage[nouppercase]{scrpage2}
\pagestyle{scrheadings}
```

The selection of the page style `scrheadings` is necessary, since otherwise the page style `headings` is active, which does not respect the settings made by option `nouppercase`.

In some cases not only classes but also packages set the running headings in uppercase letters. Also in these cases the option `nouppercase` should be able to switch back to the normal mixed case headings.

1.2 Defining Own Page Styles

1.2.1 The Interface for Beginners

Now one would not like to remain bound to only the provided page styles, but may wish to define one's own page styles. Sometimes there will be a special need, since a specific *Corporate Identity* may require the declaration of its own page styles. The easiest way to deal with this is:

`\deftripstyle{`*name*`}[`*LO*`][`*LI*`]{`*HI*`}{`*HC*`}{`*HO*`}{`*FI*`}{`*FC*`}{`*FO*`}`

The individual parameters have the following meanings:

name – the name of the page style, in order to activate it using the command `\pagestyle{`*name*`}`

LO – the thickness of the outside lines, i. e., the line above the header and the line below the footer (optional)

LI – the thickness of the separation lines, i. e., the line below the header and the line above the foot (optional)

HI – contents of the inside box in the page header for two-sided layout or left for one-sided layout

HC – contents of the centered box in the page header

HO – contents of the outside box in the page header for two-sided layout or right for one-sided layout

FI – contents of the inside box in the page footer for two-sided layout or left for one-sided layout

FC – contents of the centered box in the page footer

FO – contents of the outside box in the page footer for two-sided layout or right for one-sided layout

The command `\deftripstyle` definitely represents the simplest possibility of defining page styles. Unfortunately, there are also restrictions connected with this, since in a page range using a page style defined via deftripstyle, no modification of the lines above and below header and footer can take place.

Example: Assume a two-sided layout, where the running headings are placed on the inside. Furthermore, the document title, here "Report", shall be placed outside in the header, the page number shall be centered in the footer.

```
\deftripstyle{TheReport}%
              {\headmark}{}{Report}%
              {}{\pagemark}{}
```

If moreover the lines above the header and below the footer shall be drawn with a thickness of 2 pt, and the text body be separated from header and footer with 0.4 pt lines, then the definition has to be extended:

```
\deftripstyle{TheReport}[2pt][.4pt]%
              {\headmark}{}{Report}%
              {}{\pagemark}{}
```

See figure 1.2 for the result.

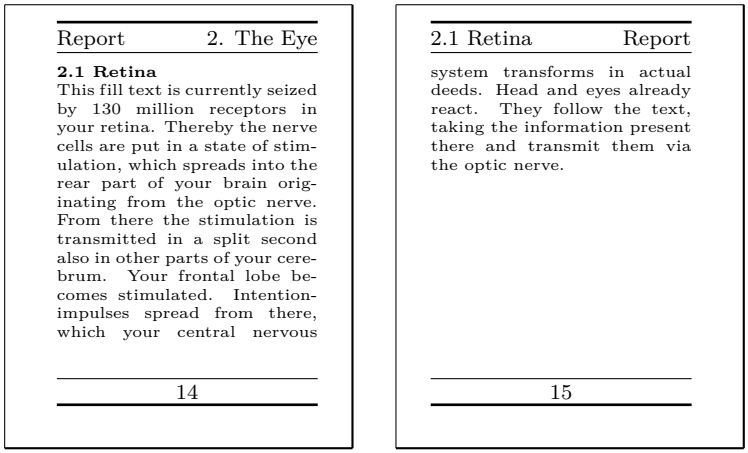


Figure 1.2: example of a user defined, line dominated page style with a static and a running heading at the page header and a page number centered at the footer

1.2.2 The Interface for Experts

Simple page styles, as they can be defined with `\deftripstyle`, are fairly rare according to experience. Either a professor requires that the thesis looks like his or her own — and who seriously wants to argue against such a wish? — or a company would like that half the financial accounting emerges in the page footer. No problem, the solution is:

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

These four commands give full access to the capabilities of scrpage2 to define page styles. Their structure is indentical, they differ only in the manner of working.

`\defpagestyle` – defines a new page style. If a page style with this name already exists it will be overwritten.

`\newpagestyle` – defines a new page style. If a page style with this name already exists a error message will be given.

`\renewpagestyle` – redefines a page style. If a page style with this name does not exist a error message will be given.

`\providepagestyle` – defines a new page style only if there is no page style with that name already present.

Using `\defpagestyle` as an example, the syntax of the four commands is explained below.

name – the name of the page style for `\pagestyle{`*name*`}`

header definition – the declaration of the header, consisting of five element; elements in round parenthesis are optional:
(ALL,ALT){EP}{OP}{OS}(BLL,BLT)

footer definition – the declaration of the footer, consisting of five element; elements in round parenthesis are optional:
(ALL,ALT){EP}{OP}{OS}(BLL,BLT)

As can be seen, header and footer declaration have identical structure. The individual parameters have the following meanings:

ALL – above line length: (header = outside, footer = separation line)

ALT – above line thickness

EP – definition for *even* pages

OP – definition for *odd* pages

OS – definition for *one-sided* layout

BLL – below line length: (header = separation line, footer = outside)

BLT – below line thickness

If the optional line-parameters are omitted, then the line behaviour remains configurable by the commands introduced in section 1.1.3, page E.10.

The three elements *EP*, *OP* and *OS* are boxes with the width of page header or footer, as appropriate. The corresponding definitions are set left-justified in the boxes. To set something left- *and* right-justified into the boxes, the space between two text elements can be stretched using `\hfill`:

```
{\headmark\hfill\pagemark}
```

If one would like a third text-element centered in the box, then an extended definition must be used. The commands `\rlap` and `\llap` simply write the given arguments, but for LaTeX they take up no horizontal space. Only in this way is the middle text really centered.

```
{\rlap{\headmark}\hfill centered text\hfill\llap{\pagemark}}
```

Example: This examples uses the document class scrbook, which means that the default page layout is two-sided. The package scrpage2 is loaded with options `automark` and `headsepline`. The first switches on the automatic update of running headings, the second determines that a separation line between header and text body is drawn in the `scrheadings` page style.

```
\documentclass{scrbook}
\usepackage[automark,headsepline]{scrpage2}
```

The expert interface is used to define two page styles. The page style `withoutLines` does not define any line parameters. The second page style `withLines` defines a line thicknes of 1 pt for the line above the header and 0 pt for the separation line between header and text.

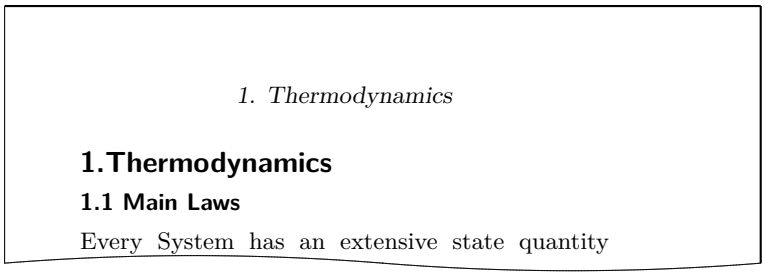
```
\defpagestyle{withoutLines}{%
  {Example\hfill\headmark}{\headmark\hfill without lines}
  {\rlap{Example}\hfill\headmark\hfill%
   \llap{without lines}}
}{%
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
}

\defpagestyle{withLines}{%
  (\textwidth,1pt)
  {with lines\hfill\headmark}{\headmark\hfill with lines}
  {\rlap{\KOMAScript}\hfill \headmark\hfill%
   \llap{with lines}}
  (0pt,0pt)
}{%
  (\textwidth,.4pt)
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
  (\textwidth,1pt)
}
```

Right at the beginning of the document the page style `scrheadings` is activated. The command `\chapter` starts a new chapter and automatically sets the page rstyle for this page to `plain`. Even though not a prime example, the command `\chead` shows how running headings can be created even on a plain page. However, in principle running headings on chapter start-pages are to be avoided, since otherwise the special character of the `plain` page style is lost. It is more important to indicate that a new chapter starts here than that a section of this page has a special title.

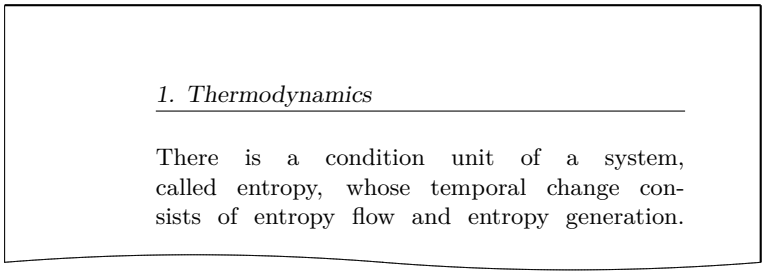
```
\begin{document}
\pagestyle{scrheadings}
\chapter{Thermodynamics}
\chead[\leftmark]{}
\section{Main Laws}
Every system has an extensive state quantity called
```

```
Energy. In a closed system the energy is constant.
```



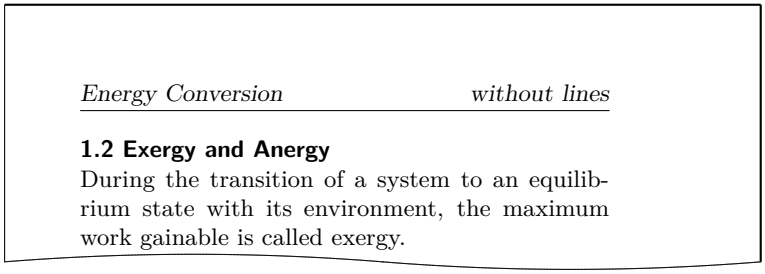
After starting a new page the page style `scrheadings` is active and thus the separation line below the header is visible.

```
There is a state quatity of a system, called entropy, whose temporal
change consists of entropy flow and entropy generation.
```



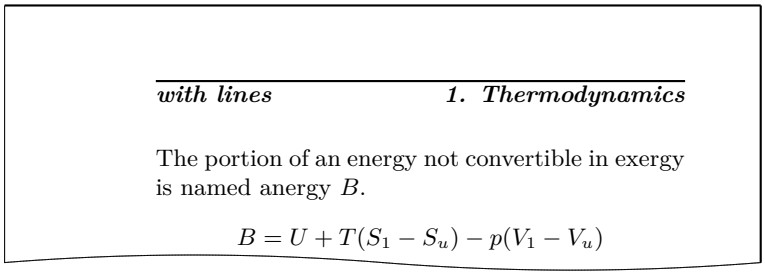
After switching to the next page, the automatic update of the running headings is disabled using `\manualmark`, and the page style `withoutLines` becomes active. Since no line parameters are given in the definition of this page style, the default configuration is used, which draws a separation line between header and text body because scrpage2 was called with `headsepline`.

```
\manualmark
\pagestyle{withoutLines}
\section{Exergy and Anergy}\markright{Energy Conversion}
During the transition of a system to an equilibrium state
with its environment, the maximum work gainable is called
exergy.
```



At the next page of the document, the page style `withLines` is activated. The line settings of its definition are taken in account and the lines are drawn accordingly.

```
\pagestyle{mitLinien}
\renewcommand{\headfont}{\itshape\bfseries}
The portion of an energy not convertible in exergy
is named anergy \Var{B}.
\[ B = U + T (S_1 - S_u) - p (V_1 - V_u)\]
\end{document}
```



1.2.3 Managing Page Styles

Before long the work with different page styles will establish a common set of employed page styles, depending on taste and tasks. In order to make the management of page styles easier and avoid time-consuming copy operations each time a new project is started, scrpage2 reads the file `scrpage.cfg` after initialisation. This file can contain a set of user-defined page styles which many projects can share.

Das obsolete Paket scrpage2

Auszug aus früheren Versionen der KOMA-Script-Anleitung

Markus Kohm Jens-Uwe-Morawski

2014-06-25

Kopf- und Fußzeilen mit scrpage2

Mit KOMA-Script 3.12 wurde das komplett neu implementierte Paket scrlayer-scrpage vorgestellt, das eine konsequente Weiterführung des Designs von scrpage2 mit anderen Mitteln darstellt. Siehe dazu das entsprechende Kapitel der KOMA-Script-Anleitung. Während scrpage2 die erweiterte Optionen-Schnittstelle der KOMA-Script-Klassen nicht unterstützt, kommt diese in scrlayer-scrpage selbstverständlich zum Einsatz. Da der Ansatz von scrlayer-scrpage gegenüber scrpage2 als überlegen betrachtet wird, wird somit empfohlen, scrlayer-scrpage an Stelle von scrpage2 zu verwenden. Die aktuelle Version von scrpage2 ist daher auch als final zu betrachten. Sämtliche Entwicklungsresourcen im Bereich Kopf- und Fußzeilen werden zukünftig in scrlayer-scrpage einfließen.

An Stelle von scrpage2 oder scrlayer-scrpage kann natürlich auch fancyhdr verwendet werden. scrpage2 und insbesondere scrlayer-scrpage harmonieren jedoch mit den KOMA-Script-Klassen deutlich besser. Genau deshalb und weil der Vorläufer von fancyhdr damals viele Möglichkeiten vermissen lies, ist scrpage2 entstanden. Natürlich ist das Paket scrpage2 ebenso wie das Paket scrlayer-scrpage nicht an eine KOMA-Script-Klasse gebunden, sondern kann auch sehr gut mit anderen Klassen verwendet werden.

1.1 Grundlegende Funktionen

Um die nachfolgende Beschreibung zu verstehen, muss noch einiges zu LaTeX gesagt werden. Im LaTeX-Kern sind die Standardseitenstile `empty`, welcher eine völlig undekorierte Seite erzeugt, und `plain`, welcher meist nur die Seitenzahl enthält, definiert. In vielen Klassen ist der Stil `headings` zu finden, welcher eine komplexe Seitendekoration mit automatischen Kolumnentitel erzeugt. Die Variante `myheadings` gleicht `headings`. Die Kolumnentitel müssen dabei aber manuell gesetzt werden. Ausführlicher wird das im Abschnitt über Seitenstile der KOMA-Script-Anleitung beschrieben. Dort wird auch erläutert, dass auf einigen Seiten der Seitenstil automatisch – in der Regel zu `plain` – wechselt.

Das Paket scrpage2 unterscheidet nicht mehr zwischen Seitenstilen mit automatischem und mit manuellem Kolumnentitel. Die Wahl des Seitenstils erfolgt unabhängig davon, ob mit automatischem oder manuellem Kolumnentitel gearbeitet wird. Näheres dazu finden Sie in Unterabschnitt 1.1.2.

1.1.1 Vordefinierte Seitenstile

Zu den grundlegenden Funktionen von scrpage2 gehören unter anderem vordefinierte, konfigurierbare Seitenstile.

scrheadings
scrplain

Das Paket scrpage2 liefert für Seiten mit Kolumnentitel einen eigenen Seitenstil namens `scrheadings`. Dieser Seitenstil kann mittels `\pagestyle{scrheadings}` aktiviert werden. Wird dieser Seitenstil benutzt, dann wird gleichzeitig der plain-Stil durch den dazu passenden Stil `scrplain` ersetzt. Passend bedeutet, dass auch der plain-Stil auf in Unterabschnitt 1.1.3 vorgestellte Befehle, die beispielsweise die Kopfbreite ändern, reagiert und im Grundlayout übereinstimmt. Die Aktivierung des Seitenstils `scrheadings` oder des zugehörigen plain-Stils, `scrplain`, hat keine Auswirkung darauf, ob mit manuellen oder automatischen Kolumnentiteln gearbeitet wird (siehe Unterabschnitt 1.1.2). Der Seitenstil `scrplain` kann auch direkt per `\pagestyle` aktiviert werden.

```
\lehead[scrplain-links-gerade]{scrheadings-links-gerade}
\cehead[scrplain-mittig-gerade]{scrheadings-mittig-gerade}
\rehead[scrplain-rechts-gerade]{scrheadings-rechts-gerade}
\lefoot[scrplain-links-gerade]{scrheadings-links-gerade}
\cefoot[scrplain-mittig-gerade]{scrheadings-mittig-gerade}
\refoot[scrplain-rechts-gerade]{scrheadings-rechts-gerade}
\lohead[scrplain-links-ungerade]{scrheadings-links-ungerade}
\cohead[scrplain-mittig-ungerade]{scrheadings-mittig-ungerade}
\rohead[scrplain-rechts-ungerade]{scrheadings-rechts-ungerade}
\lofoot[scrplain-links-ungerade]{scrheadings-links-ungerade}
\cofoot[scrplain-mittig-ungerade]{scrheadings-mittig-ungerade}
\rofoot[scrplain-rechts-ungerade]{scrheadings-rechts-ungerade}
\ihead[scrplain-innen]{scrheadings-innen}
\chead[scrplain-zentriert]{scrheadings-zentriert}
\ohead[scrplain-außen]{scrheadings-außen}
\ifoot[scrplain-innen]{scrheadings-innen}
\cfoot[scrplain-zentriert]{scrheadings-zentriert}
\ofoot[scrplain-außen]{scrheadings-außen}
```

Die Seitenstile von scrpage2 sind so definiert, dass ihr Kopf und Fuß flexibel angepasst werden kann. Hierzu sind sowohl im Kopf als auch im Fuß drei Felder vorhanden, deren Inhalt modifiziert werden kann. Die Befehle zur Modifikation sind in Abbildung 1.1 verdeutlicht. Die in der Mitte dargestellten Befehle modifizieren sowohl die Felder der linken als auch der rechten Seite. Alle Befehle haben sowohl ein optionales als auch ein obligatorisches Argument. Das optionale Argument bestimmt jeweils das durch den Befehl festgelegte Feld im plain-Seitenstil, `scrplain`. Das obligatorische Argument definiert das entsprechende Feld im Seitenstil `scrheadings`.

Beispiel: Angenommen, man möchte bei `scrheadings` zentriert im Seitenfuß die Seitenzahl dargestellt haben, dann benutzt man einfach:

```
\cfoot{\pagemark}
```

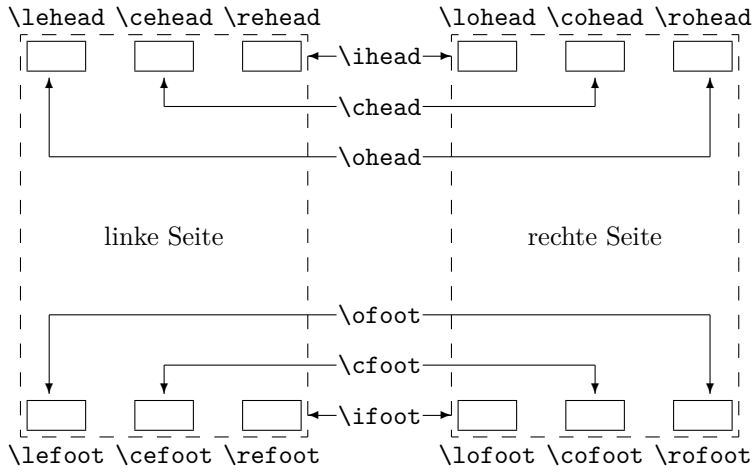


Abbildung 1.1: Zuordnung der Befehle zur Manipulation der beiden Seitenstile `scrheadings` und `scrplain` zu den Feldern im Kopf und Fuß der Seiten

Sollen die Seitenzahlen im Kopf außen und die Kolumnentitel innen stehen, dann erfolgt dies mit:

```
\ohead{\pagemark}
\ihead{\headmark}
\cfoot{}
```

Das `\cfoot{}` ist nur notwendig, um eine möglicherweise in der Mitte des Fußes vorhandene Seitenzahl zu entfernen.

Die Befehle, die direkt nur einem Feld zugeordnet sind, können für anspruchsvollere Vorhaben genutzt werden.

Beispiel: Angenommen, man hat den Auftrag, einen Jahresbericht einer Firma zu erstellen, dann könnte das so angegangen werden:

```
\ohead{\pagemark}
\rehead{Jahresbericht 2001}
\lohead{\headmark}
\cefoot{Firma WasWeißIch}
\cofoot{Abteilung Entwicklung}
```

Natürlich muss man hier dafür sorgen, dass mittels `\cofoot` der Fuß der rechten Seite aktualisiert wird, wenn eine neue Abteilung im Bericht besprochen wird.

Wie oben dargestellt, gibt es einen zu `scrheadings` korrespondierenden plain-Seitenstil. Da es auch möglich sein soll, diesen Stil anzupassen, unterstützen die Befehle ein optionales Argument. Damit kann der Inhalt des entsprechenden Feldes im plain-Seitenstil modifiziert werden.

Beispiel: Um für die Nutzung von `scrheadings` die Position der Seitenzahlen festzulegen, kann man folgendes benutzen:

```
\cfoot[\pagemark]{}
\ohead[]{\pagemark}
```

Wird anschließend der Stil `plain` genutzt, beispielsweise weil `\chapter` eine neue Seite beginnt und darauf umschaltet, dann steht die Seitenzahl zentriert im Seitenfuß.

`\clearscrheadings`
`\clearscrplain`
`\clearscrheadfoot`

Will man sowohl den Seitenstil `scrheadings` als auch den dazu gehörenden plain-Seitenstil von Grund auf neu definieren, muss man häufig zusätzlich einige der bereits belegten Seitenelemente löschen. Da man selten alle Elemente mit neuem Inhalt füllt, sind dazu in den meisten Fällen mehrere Befehle mit leeren Parametern notwendig. Mit Hilfe dieser drei Befehle ist das Löschen schnell und gründlich möglich. Während `\clearscrheadings` lediglich alle Felder des Seitenstils `scrheadings` und `\clearscrplain` alle Felder des zugehörigen plain-Seitenstils löscht, werden von `\clearscrheadfoot` alle Felder beider Seitenstile auf leere Inhalte gesetzt.

Beispiel: Sie wollen unabhängig davon, wie die Seitenstile derzeit aussehen, die Standardform der KOMA-Script-Klassen bei zweiseitigem Satz erreichen. Dies ist mit nur drei Befehlen möglich:

```
\clearscrheadfoot
\ohead{\headmark}
\ofoot[\pagemark]{\pagemark}
```

Ohne die Befehle `\clearscrheadings`, `\clearscrplain` und `\clearscrheadfoot` wären doppelt so viele Anweisungen und neun weitere leere Argumente notwendig:

```
\ihead[]{}
\chead[]{}
\ohead[]{\headmark}
\ifoot[]{}
\cfoot[]{}
\ofoot[\pagemark]{\pagemark}
```

Einige davon könnten natürlich entfallen, wenn man von einer konkreten Vorbelegung ausginge.

In den vorausgehenden Beispielen wurden schon zwei Befehle benutzt, die noch gar nicht besprochen wurden. Das soll jetzt nachgeholt werden.

`\leftmark`
`\rightmark`

Diese beiden Befehle erlauben es, auf die Kolumnentitel zuzugreifen, die normalerweise für die linke bzw. die rechte Seite gedacht sind. Diese beiden Befehle werden nicht von scrpage2, sondern direkt vom LaTeX-Kern zur Verfügung gestellt. Wenn in diesem Kapitel vom Kolumnentitel der linken Seite oder vom Kolumnentitel der rechten Seite die Rede ist, dann ist damit eigentlich der Inhalt von `\leftmark` und `\rightmark` gemeint.

`\headmark`

Dieser Befehl ermöglicht es, auf die Inhalte der Kolumnentitel zuzugreifen. Im Gegensatz zu den originalen LaTeX-Befehlen `\leftmark` und `\rightmark` braucht man nicht auf die richtige Zuordnung zur linken oder rechten Seite zu achten.

`\pagemark`

Dieser Befehl ermöglicht den Zugriff auf die Seitenzahl. Im Unterabschnitt 1.1.3, Seite D.7 wird der Befehl `\pnumfont` zur Formatierung der Seitenzahl vorgestellt, den `\pagemark` automatisch berücksichtigt.

`useheadings`

Das Paket scrpage2 ist in erster Linie dafür gedacht, dass die bereitgestellten Stile benutzt oder eigene Stile definiert werden. Jedoch kann es notwendig sein, auch auf einen von der Dokumentklasse zur Verfügung gestellten Stil zurückzuschalten. Es wäre nahe liegend, dieses mit `\pagestyle{headings}` vorzunehmen. Das hätte aber den Nachteil, dass die nachfolgend besprochenen Befehle `\automark` und `\manualmark` nicht wie erwartet funktionierten. Daher sollte mit `\pagestyle{useheadings}` auf die originalen Stile umgeschaltet werden. Eine solche Umschaltung hat dann keine Auswirkung darauf, ob mit manuellen oder automatischen Kolumnentiteln gearbeitet wird.

1.1.2 Manuelle und automatische Kolumnentitel

Gewöhnlich gibt es zu einem `headings`-Stil eine *my*-Variante. Ist ein solcher Stil aktiv, dann werden die Kolumnentitel nicht mehr automatisch aktualisiert. Bei scrpage2 wird ein anderer Weg beschritten. Ob die Kolumnentitel lebend sind oder nicht, bestimmen die Befehle `\automark` und `\manualmark`. Die Voreinstellung kann auch bereits beim Laden des Paketes über die Optionen `automark` und `manualmark` beeinflusst werden (siehe Unterabschnitt 1.1.4, Seite D.14).

`\manualmark`

Wie der Name bereits verdeutlicht, schaltet `\manualmark` die Aktualisierung der Kolumnentitel aus. Es bleibt somit dem Nutzer überlassen, für eine Aktualisierung bzw. für den Inhalt der

Kolumnentitel zu sorgen. Dazu stehen die Befehle `\markboth` und `\markright` aus dem LaTeX-Kern bereit. Diese Anweisungen sind im Abschnitt über Seitenstile in der KOMA-Script-Anleitung erklärt.

`\automark[`*rechte Seite*`]{`*linke Seite*`}`

Die Anweisung `\automark` aktiviert die automatische Aktualisierung des Kolumnentitels. Für die beiden Parameter sind die Bezeichnungen der Gliederungsebenen einzusetzen, deren Titel an entsprechender Stelle erscheinen soll. Gültige Werte für die Parameter sind: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph` und `subparagraph`. Der Wert `part` führt bei Verwendung der meisten Klassen nicht zu dem gewünschten Ergebnis. Bisher ist nur von den KOMA-Script-Klassen ab Version 2.9s bekannt, dass dieser Wert unterstützt wird. Das optionale Argument *rechte Seite* ist verständlicherweise nur für zweiseitigen Satz gedacht. Im einseitigen Satz sollten Sie normalerweise darauf verzichten. Mit Hilfe der Option `autooneside` können Sie auch einstellen, dass das optionale Argument im einseitigen Satz automatisch ignoriert wird (siehe Unterabschnitt 1.1.4, Seite D.15).

v2.2

Beispiel: Wird beispielsweise mit einer *book*-Klasse gearbeitet, deren höchste Gliederungsebene *chapter* ist, dann stellt nach einem vorhergehenden `\manualmark` der Befehl

```
\automark[section]{chapter}
```

den Originalzustand wieder her. Bevorzugt man stattdessen, die tieferen Gliederungsebenen angezeigt zu bekommen, dann erfolgt dies mit:

```
\automark[subsection]{section}
```

Die Markierung der jeweils höheren Gliederungsebene wird mit Hilfe von `\markboth` gesetzt. Die Markierung der tieferen Gliederungsebene wird mit `\markright` bzw. `\markleft` gesetzt. Der entsprechende Aufruf erfolgt indirekt über die Gliederungsbefehle. Die Anweisung `\markleft` wird von scrpage2 bereitgestellt und ist vergleichbar zu `\markright` aus dem LaTeX-Kern definiert. Obwohl sie nicht als internes Makro definiert ist, wird von einem direkten Gebrauch abgeraten.

1.1.3 Formatierung der Kopf- und Fußzeilen

Im vorherigen Abschnitt ging es hauptsächlich um inhaltliche Dinge. Das genügt natürlich nicht, um die gestalterischen Ambitionen zu befriedigen. Deshalb soll es sich in diesem Abschnitt ausschließlich darum drehen.

`\headfont`
`\footfont`
`\pnumfont`

Die Schriftformatierung für den Seitenkopf und -fuß übernimmt der Befehl `\headfont`, `\footfont` die Abweichung davon für den Fuß und `\pnumfont` wiederum die Abweichung davon für die Seitenzahl.

Beispiel: Um beispielsweise den Kopf in fetter, serifenloser Schrift und den Fuß in nicht fetter, serifenloser Schrift zu setzen und die Seitenzahl geneigt mit Serifen erscheinen zu lassen, nutzt man folgende Definitionen:

```
\renewcommand*{\headfont}{%
  \normalfont\sffamily\bfseries}
\renewcommand*{\footfont}{%
  \normalfont\sffamily}
\renewcommand*{\pnumfont}{%
  \normalfont\rmfamily\slshape}
```

Ab Version 2.8p der KOMA-Script-Klassen wurde die Schnittstelle für Schriftattribute vereinheitlicht. Wird scrpage2 in Verbindung mit einer dieser Klassen verwendet, dann sollte die Zuweisung in der Art erfolgen, wie sie in der KOMA-Script-Anleitung beschrieben wird.

Beispiel: Statt `\renewcommand` wird bei Verwendung einer KOMA-Script-Klasse vorzugsweise der Befehl `\setkomafont` verwendet. Die vorhergehenden Definitionen lauten damit:

```
\setkomafont{pagehead}{%
  \normalfont\sffamily\bfseries}
\setkomafont{pagefoot}{%
  \normalfont\sffamily}
\setkomafont{pagenumber}{%
  \normalfont\rmfamily\slshape}
```

`\setheadwidth[`*Verschiebung*`]{`*Breite*`}`
`\setfootwidth[`*Verschiebung*`]{`*Breite*`}`

Normalerweise entsprechen die Breiten von Kopf- und Fußzeile der Breite des Textbereichs. Die beiden Befehle `\setheadwidth` und `\setfootwidth` ermöglichen dem Anwender, auf einfache Weise die Breiten seinen Bedürfnissen anzupassen. Das obligatorische Argument *Breite* nimmt den Wert der Breite des Kopfes bzw. des Fußes auf, *Verschiebung* ist ein Längenmaß für die Verschiebung des entsprechenden Elements in Richtung des äußeren Seitenrandes.

Für die möglichen Standardfälle akzeptiert das obligatorische Argument *Breite* auch folgende symbolische Werte:

`paper` – die Breite des Papiers

`page` – die Breite der Seite

`text` – die Breite des Textbereichs

`textwithmarginpar` – die Breite des Textbereichs inklusive Seitenrand

`head` – die aktuelle Breite des Seitenkopfes

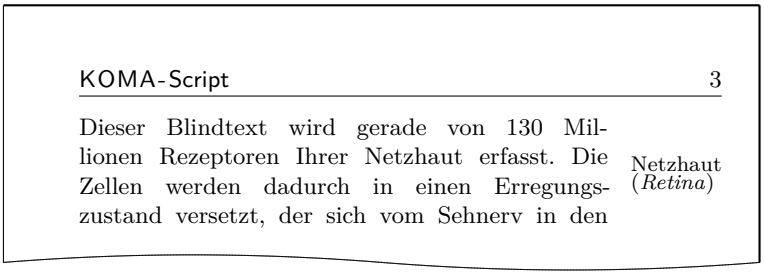
`foot` – die aktuelle Breite des Seitenfußes

Der Unterschied zwischen `paper` und `page` besteht darin, dass `page` die Breite des Papiers abzüglich der Bindekorrektur ist, falls das typearea-Paket verwendet wird (siehe das Kapitel über typearea in der KOMA-Script-Anleitung). Ohne Verwendung von typearea sind `paper` und `page` identisch.

Beispiel: Angenommen, man möchte ein Seitenlayout wie im *LaTeX-Begleiter*, bei dem die Kopfzeile in den Rand ragt, dann geschieht das ganz einfach mit:

```
\setheadwidth[0pt]{textwithmarginpar}
```

und sieht dann auf einer rechten Seite folgendermaßen aus:



Soll der Seitenfuß die gleiche Breite und Ausrichtung haben, dann hat man jetzt zwei Wege. Der erste ist, man wiederholt das Gleiche für den Seitenfuß mit:

```
\setfootwidth[0pt]{textwithmarginpar}
```

oder man greift auf den anderen symbolischen Wert `head` zurück, da der Kopf bereits die gewünschte Breite hat.

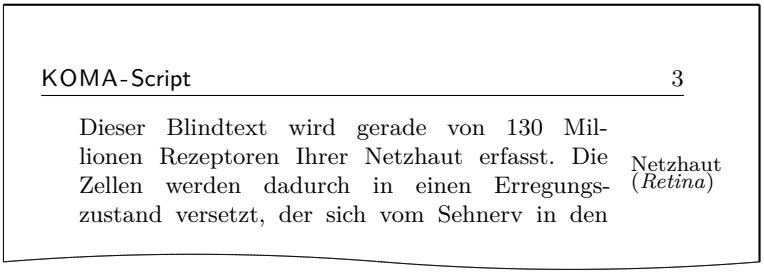
```
\setfootwidth[0pt]{head}
```

Wird keine Verschiebung angegeben, das heißt auf das optionale Argument verzichtet, dann erscheint der Kopf bzw. der Fuß symmetrisch auf der Seite angeordnet. Es wird somit ein Wert für die Verschiebung automatisch ermittelt, der der aktuellen Seitengestalt entspricht.

Beispiel: Entsprechend dem vorherigen Beispiel wird hier auf das optionale Argument verzichtet:

```
\setheadwidth{textwithmarginpar}
```

und sieht dann auf einer rechten Seite folgendermaßen aus:



Wie zu sehen, ist der Kopf jetzt nach innen verschoben, wobei die Kopfbreite sich nicht geändert hat. Die Verschiebung ist so berechnet, dass die Seitenproportionen auch hier sichtbar werden.

```
\setheadtopline[Länge]{Dicke}[Anweisungen]
\setheadsepline[Länge]{Dicke}[Anweisungen]
\setfootsepline[Länge]{Dicke}[Anweisungen]
\setfootbotline[Länge]{Dicke}[Anweisungen]
```

Entsprechend den Größenparametern für die Kopf- und Fußzeile gibt es auch Befehle, die die Dimensionen der Linien im Kopf und Fuß modifizieren können. Dazu sollten diese Linien aber zunächst erst einmal eingeschaltet werden. Siehe hierzu die Optionen `headtopline`, `headsepline`, `footsepline`, `footbotline` in Unterabschnitt 1.1.4, Seite D.13.

`\setheadtopline` – modifiziert die Parameter für die Linie über dem Seitenkopf

`\setheadsepline` – modifiziert die Parameter für die Linie zwischen Kopf und Textkörper

`\setfootsepline` – modifiziert die Parameter für die Linie zwischen Text und Fuß

`\setfootbotline` – modifiziert die Parameter für die Linie unter dem Seitenfuß

Das obligatorische Argument *Dicke* bestimmt, wie stark die Linie gezeichnet wird. Das optionale Argument *Länge* akzeptiert die gleichen symbolischen Werte wie *Breite* bei `\setheadwidth`, also auch einen normalen Längenausdruck. Solange im Dokument dem optionalen Argument *Länge* kein Wert zugewiesen wurde, passt sich die entsprechende Linienlänge automatisch der Breite des Kopfes bzw. des Fußes an.

Möchte man diesen Automatismus für die Länge einer Linie wieder restaurieren, dann nutzt man im Längenargument den Wert `auto`.

v2.2 Mit dem optionalen Argument *Anweisungen* können zusätzliche Anweisungen definiert werden, die vor dem Zeichnen der jeweiligen Linie auszuführen sind. Das können beispielsweise Anweisungen sein, um die Farbe der Linie zu ändern. Bei Verwendung einer KOMA-Script-Klasse können diese Anweisungen auch über `\setkomafont` für eines der Elemente

`headtopline`, `headsepline`, `footsepline`, `footbottomline` oder auch `footbotline` gesetzt und mit `\addtokomafont` erweitert werden. Die beiden Anweisungen `\setkomafont` und `\addtokomafont` sind in der KOMA-Script-Anleitung näher beschrieben.

```
\setheadtopline[auto]{current}
\setheadtopline[auto]{}
\setheadtopline[auto]{}[]
```

Die hier am Befehl `\setheadtopline` illustrierten Argumente sind natürlich auch für die anderen drei Längenbefehle gültig.

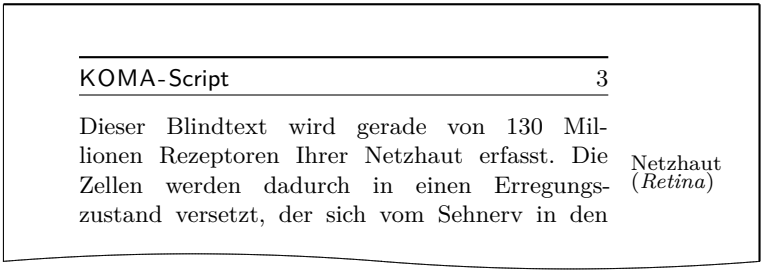
Enthält das obligatorische Argument den Wert `current` oder wird leer gelassen, dann wird die Dicke der Linie nicht verändert. Das kann genutzt werden, wenn die Länge der Linie, aber nicht die Dicke modifiziert werden soll.

Wird das optionale Argument *`Anweisungen`* weggelassen, so bleiben eventuell zuvor gesetzte Anweisungen erhalten. Wird hingegen ein leeres Argument *`Anweisungen`* gesetzt, so werden eventuell zuvor gesetzte Anweisungen wieder gelöscht.

Beispiel: Soll beispielsweise der Kopf mit einer kräftigen Linie von 2 pt darüber und einer normalen von 0,4 pt zwischen Kopf und Text abgesetzt werden, dann erfolgt das mit:

```
\setheadtopline{2pt}
\setheadsepline{.4pt}
```

Zusätzlich sind unbedingt die Optionen `headtopline` und `headsepline` vorzugsweise global im optionalen Argument von `\documentclass` zu setzen. Das Ergebnis könnte dann wie folgt aussehen.



Sollen diese Linien zusätzlich in roter Farbe gesetzt werden, dann sind die Anweisungen beispielsweise wie folgt zu ändern:

```
\setheadtopline{2pt}[\color{red}]
\setheadsepline{.4pt}[\color{red}]
```

In diesem und auch dem folgenden Beispiel wurde für die Aktivierung der Farbe die Syntax des color-Pakets verwendet, das dann natürlich auch geladen werden muss.

scrpage2 selbst bietet keine direkte Farbunterstützung. Damit ist jedes beliebige Farbunterstützungspaket verwendbar.

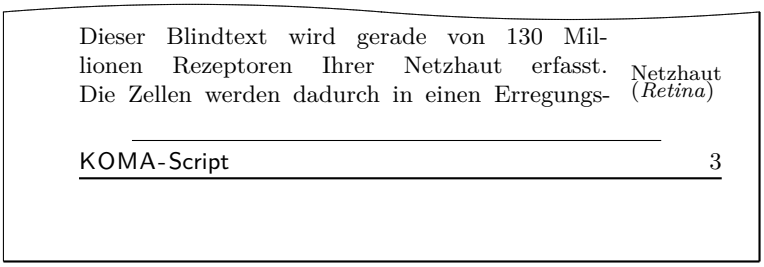
Mit einer KOMA-Script-Klasse kann alternativ

```
\setheadtopline{2pt}
\setheadsepline{.4pt}
\setkomafont{headtopline}{\color{red}}
\setkomafont{headsepline}{\color{red}}
```

verwendet werden.

Die automatische Anpassung an die Kopf- und Fußbreiten illustriert folgendes Beispiel, für das die Optionen `footbotline` und `footsepline` gesetzt sein sollten:

```
\setfootbotline{2pt}
\setfootsepline[text]{.4pt}
\setfootwidth[0pt]{textwithmarginpar}
```



Nun mag nicht jedem die Ausrichtung der Linie über der Fußzeile gefallen, sondern es wird in einem solchen Fall erwartet, dass sie wie der Kolumnentitel linksbündig zum Text ist. Diese Einstellung kann nur global in Form einer Paketoption erfolgen und wird im folgenden Unterabschnitt 1.1.4 mit anderen Optionen beschrieben.

1.1.4 Optionen beim Laden des Paketes

Während bei den KOMA-Script-Klassen die Mehrzahl der Optionen auch noch nach dem Laden der Klasse mit `\KOMAoptions` und `\KOMAoption` geändert werden kann, trifft dies für das Paket scrpage2 derzeit noch nicht zu. Alle Optionen für dieses Paket müssen als globale Optionen, also im optionalen Argument von `\documentclass`, oder als Paketoptionen, also im optionalen Argument von `\usepackage`, angegeben werden.

`headinclude`
`headexclude`
`footinclude`
`footexclude`

v2.3

Diese Optionen sollten bei Verwendung von KOMA-Script 3 nicht mehr beispielsweise per optionalem Argument von `\usepackage` oder per `\PassOptionsToPackage` direkt an scrpage2 übergeben werden. Lediglich aus Gründen der Kompatibilität sind sie noch in scrpage2 deklariert und werden von diesem als `headinclude`, `headinclude=false`, `footinclude` und `footinclude=false` an das Paket typearea weitergereicht.

`headtopline`
`plainheadtopline`
`headsepline`
`plainheadsepline`
`footsepline`
`plainfootsepline`
`footbotline`
`plainfootbotline`

Eine Grundeinstellung für die Linien unter und über den Kopf- und Fußzeilen kann mit diesen Optionen vorgenommen werden. Diese Einstellungen gelten dann als Standard für alle mit scrpage2 definierten Seitenstile. Wird eine von diesen Optionen verwendet, dann wird eine Linienstärke von 0,4 pt eingesetzt. Da es zum Seitenstil `scrheadings` einen entsprechenden plain-Stil gibt, kann mit den `plain...`-Optionen auch die entsprechende Linie des plain-Stils konfiguriert werden. Diese `plain`-Optionen wirken aber nur, wenn auch die korrespondierende Option ohne `plain` aktiviert wurde. Somit zeigt die Option `plainheadtopline` ohne `headtopline` keine Wirkung.

Bei diesen Optionen ist zu beachten, dass der entsprechende Seitenteil in den Textbereich des Satzspiegels mit übernommen wird, wenn eine Linie aktiviert wurde. Wird also mittels `headsepline` die Trennlinie zwischen Kopf und Text aktiviert, dann wird automatisch mittels typearea der Satzspiegel so berechnet, dass der Seitenkopf Teil des Textblocks ist.

Die Bedingungen für die Optionen des vorhergehenden Abschnitts gelten auch für diesen Automatismus. Das bedeutet, dass das Paket typearea nach scrpage2 geladen werden muss, beziehungsweise, dass bei Verwendung einer KOMA-Script-Klasse die Optionen `headinclude` und `footinclude` explizit bei `\documentclass` gesetzt werden müssen, um Kopf- bzw. Fußzeile in den Textblock zu übernehmen.

`ilines`
`clines`
`olines`

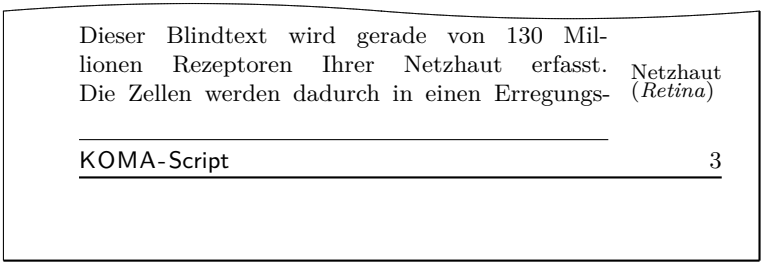
Bei der Festlegung der Linienlängen kann es vorkommen, dass die Linie zwar die gewünschte Länge, aber nicht die erwünschte Ausrichtung hat, da sie im Kopf- bzw. Fußbereich zentriert

wird. Mit den hier vorgestellten Paketoptionen kann global für alle mit scrpage2 definierten Seitenstile diese Vorgabe modifiziert werden. Dabei setzt `ilines` die Ausrichtung so, dass die Linien an den inneren Rand verschoben werden. Die Option `clines` verhält sich wie die Standardeinstellung und `olines` richtet am äußeren Rand aus.

Beispiel: Hier gilt es, das Beispiel zu `\setfootsepline` auf Seite D.12 mit dem folgenden zu vergleichen, um die Wirkung der Option `ilines` zu sehen.

```
\usepackage[ilines,footsepline,footbotline]
           {scrpage2}
\setfootbotline{2pt}
\setfootsepline[text]{.4pt}
\setfootwidth[0pt]{textwithmarginpar}
```

Allein die Verwendung der Option `ilines` führt dabei zu der geänderten Ausgabe, die nachfolgend veranschaulicht wird:



Die Trennlinie zwischen Text und Fuß wird bündig innen im Fußteil gesetzt und nicht wie bei der Standardeinstellung zentriert.

`automark`
`manualmark`

Diese Optionen bestimmen gleich zu Beginn des Dokuments, ob eine automatische Aktualisierung der Kolumnentitel erfolgt. Die Option `automark` schaltet die automatische Aktualisierung ein, `manualmark` deaktiviert sie. Ohne Verwendung einer der beiden Optionen bleibt die Einstellung erhalten, die beim Laden des Paketes gültig war.

Beispiel: Sie laden das Paket scrpage2 unmittelbar nach der Klasse scrreprt und ohne weitere Optionen. Dazu schreiben Sie:

```
\documentclass{scrreprt}
\usepackage{scrpage2}
```

Da bei scrreprt der Seitenstil `plain` voreingestellt ist, ist dies auch jetzt noch der Fall. Außerdem entspricht die Voreinstellung `plain` manuellen Kolumnentiteln. Wenn Sie also anschließend mit

```
\pagestyle{scrheadings}
```

auf den Seitenstil `scrheadings` umschalten, sind noch immer manuelle Kolumnentitel eingestellt.

Verwenden Sie stattdessen die Dokumentklasse scrbook, so ist nach

```
\documentclass{scrbook}
\usepackage{scrpage2}
```

der Seitenstil `headings` mit automatischen Kolumnentiteln aktiviert. Bei anschließender Umschaltung auf den Seitenstil `scrheadings` bleiben automatische Kolumnentitel eingeschaltet. Dabei werden dann weiterhin die Markierungsmakros von scrbook verwendet.

Verwenden Sie hingegen

```
\usepackage[automark]{scrpage2}
```

so wird unabhängig von der verwendeten Klasse auf automatische Kolumnentitel umgeschaltet, wobei die Markierungsmakros von scrpage2 genutzt werden. Natürlich wirkt sich dies auf den Seitenstil `plain` von scrreprt nicht aus. Die Kolumnentitel werden erst sichtbar, wenn auf den Seitenstil `scrheadings` oder `useheadings` oder einen selbst definierten Seitenstil mit Kolumnentiteln umgeschaltet wird.

`autooneside`

Mit dieser Option wird das optionale Argument von `\automark` im einseitigen Satz automatisch ignoriert. Siehe hierzu auch die Erläuterung zum Befehl `\automark` in Unterabschnitt 1.1.2, Seite D.7.

`komastyle`
`standardstyle`

Diese Optionen bestimmen, wie die beiden vordefinierten Seitenstile `scrheadings` und `scrplain` gestaltet sind. Bei `komastyle` wird eine Definition vorgenommen, wie sie den KOMA-Script-Klassen entspricht. Bei den KOMA-Script-Klassen ist dies die Voreinstellung und kann somit auch für andere Klassen gesetzt werden.

Die Option `standardstyle` definiert die beiden Seitenstile wie es von den Standardklassen erwartet wird. Außerdem wird hier automatisch `markuppercase` aktiviert, es sei denn, `markusedcase` wird ebenfalls als Option übergeben.

`markuppercase`
`markusedcase`

Für die Funktionalität von `\automark` modifiziert scrpage2 interne Befehle, die die Gliederungsbefehle benutzen, um die lebenden Kolumnentitel zu setzen. Da einige Klassen,

im Gegensatz zu den KOMA-Script-Klassen, die Kolumnentitel in Großbuchstaben schreiben, muss scrpage2 wissen, wie die genutzte Dokumentklasse die lebenden Kolumnentitel darstellt.

Die Option `markuppercase` zeigt scrpage2, dass die benutzte Klasse die Großschreibweise benutzt. Die Option `markusedcase` sollte angegeben werden, wenn die benutzte Dokumentklasse keine Großschreibweise verwendet. Die Optionen sind nicht geeignet, eine entsprechende Darstellung zu erzwingen. Es kann somit zu unerwünschten Effekten kommen, wenn die Angabe nicht dem Verhalten der Dokumentklasse entspricht.

`nouppercase`

Wie in obiger Erklärung zu `markuppercase` und `markusedcase` bereits ausgeführt wurde, gibt es Klassen und auch Pakete, die beim Setzen der lebenden Kolumnentitel mit Hilfe einer der Anweisungen `\uppercase` oder `\MakeUppercase` den gesamten Eintrag in Großbuchstaben wandeln. Mit der Option `nouppercase` können diese beiden Anweisungen im Kopf und im Fuß außer Kraft gesetzt werden. Das gilt aber nur für Seitenstile, die mit Hilfe von scrpage2 definiert werden. Dazu zählen auch `scrheadings` und der zugehörige plain-Seitenstil.

Die verwendete Methode ist äußerst brutal und kann dazu führen, dass auch erwünschte Änderungen von Klein- in Großbuchstaben unterbleiben. Da diese Fälle nicht sehr häufig sind, stellt `nouppercase` aber meist eine brauchbare Lösung dar.

Beispiel: Sie verwenden die Standardklasse book, wollen aber, dass die lebenden Kolumnentitel nicht in Großbuchstaben, sondern in normaler gemischter Schreibweise gesetzt werden. Die Präambel Ihres Dokuments könnte dann wie folgt beginnen:

```
\documentclass{book}
\usepackage[nouppercase]{scrpage2}
\pagestyle{scrheadings}
```

Die Umschaltung auf den Seitenstil `scrheadings` ist notwendig, weil sonst der Seitenstil `headings` verwendet wird, der von der Option `nouppercase` nicht behandelt wird.

In einigen Fällen setzen nicht nur Klassen, sondern auch Pakete lebende Kolumnentitel in Großbuchstaben. Auch in diesen Fällen hilft `nouppercase` meist, um zu gemischter Schreibweise zurückzuschalten.

1.2 Seitenstile selbst gestalten

1.2.1 Die Anwenderschnittstelle

Nun möchte man ja nicht immer an die vorgegebenen Seitenstile gebunden sein, sondern auch seiner Kreativität freien Lauf lassen. Manchmal ist man auch dazu gezwungen, weil eine bestimmte *Corporate Identity* einer Firma es verlangt. Der einfachste Weg damit umzugehen ist

`\deftripstyle{`*Name*`}[`*LA*`][`*LI*`]{`*KI*`}{`*KM*`}{`*KA*`}{`*FI*`}{`*FM*`}{`*FA*`}`

Die einzelnen Felder haben folgende Bedeutung:

Name – die Bezeichnung des Seitenstils, die dann bei der Aktivierung mit `\pagestyle{`*Name*`}` oder `\thispagestyle{`*Name*`}` verwendet wird

LA – die Dicke der äußeren Linien, d. h. der Linien über der Kopfzeile und unter der Fußzeile (optional)

LI – die Dicke der inneren Linie, d. h. der Linien die Kopf und Fuß vom Textkörper trennen (optional)

KI – Inhalt des Feldes im Kopf innenseitig oder bei einseitigem Layout links

KM – Inhalt des Feldes im Kopf zentriert

KA – Inhalt des Feldes im Kopf außenseitig oder bei einseitigem Layout rechts

FI – Inhalt des Feldes im Fuß innenseitig oder bei einseitigem Layout links

FM – Inhalt des Feldes im Fuß zentriert

FA – Inhalt des Feldes im Fuß außenseitig oder bei einseitigem Layout rechts

Der Befehl `\deftripstyle` stellt sicherlich die einfachste Möglichkeit dar, Seitenstile zu definieren. Leider sind damit auch Einschränkungen verbunden, da in einem Seitenbereich mit einem durch `\deftripstyle` deklarierten Seitenstil keine Änderung der Kopf- und Fußlinien erfolgen kann.

Beispiel: Vorgegeben sei ein doppelseitiges Layout, bei dem die Kolumnentitel innen erscheinen sollen. Weiterhin soll der Dokumenttitel, in diesem Fall kurz „Bericht“, an den Außenrand in den Kopf, die Seitenzahl soll zentriert in den Fuß.

```
\deftripstyle{DerBericht}%
              {\headmark}{}{Bericht}%
              {}{\pagemark}{}
```

Sollen weiterhin die Linien über dem Kopf und unter dem Fuß mit 2 pt erscheinen und der ganze Textkörper mit dünnen Linien von 0,4 pt von Kopf und Fuß abgesetzt werden, dann erweitert man vorherige Definition.

```
\deftripstyle{DerBericht}[2pt][.4pt]%
              {\headmark}{}{Bericht}%
              {}{\pagemark}{}
```

Das Ergebnis ist in Abbildung 1.2 zu sehen.

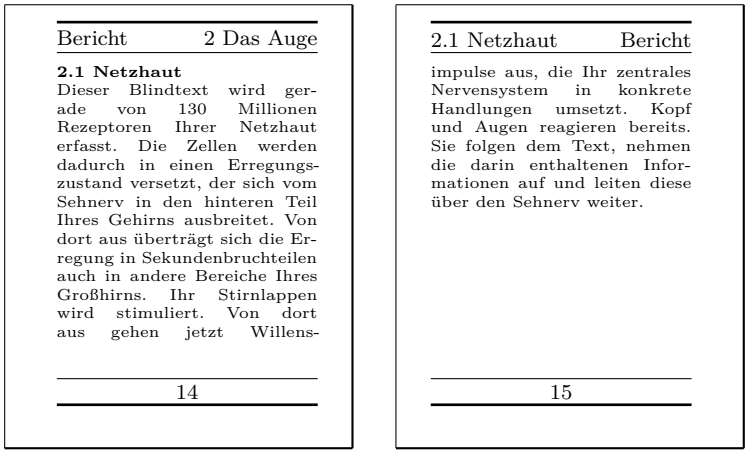


Abbildung 1.2: Beispiel für einen selbst definierten, von Linien dominierten Seitenstil mit einem statischen und einem lebenden Kolumnentitel im Kopf und der Seitenzahl in der Mitte des Fußes.

1.2.2 Die Expertenschnittstelle

Einfache Seitenstile, wie sie mit `\deftripstyle` deklariert werden können, sind erfahrungsgemäß selten. Entweder verlangt ein Professor, dass die Diplomarbeit so aussieht wie seine eigene – und wer will ihm da *ernsthaft* widersprechen – oder eine Firma möchte, dass die halbe Finanzbuchhaltung im Seitenfuß auftaucht. Alles kein Problem, denn es gibt noch:

```
\defpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\newpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\renewpagestyle{Name}{Kopfdefinition}{Fußdefinition}
\providepagestyle{Name}{Kopfdefinition}{Fußdefinition}
```

Dies sind die Befehle, die die volle Kontrolle über die Gestaltung eines Seitenstils ermöglichen. Der Aufbau ist bei allen vier Definitionen gleich, sie unterscheiden sich nur hinsichtlich der Wirkungsweise.

`\defpagestyle` – definiert einen neuen Seitenstil. Existiert bereits einer mit diesem Namen, wird dieser überschrieben.

`\newpagestyle` – definiert einen neuen Seitenstil. Wenn schon einer mit diesem Namen existiert, wird ein Fehler ausgegeben.

`\renewpagestyle` – definiert einen bestehenden Seitenstil um. Wenn noch keiner mit diesem Namen existiert, wird ein Fehler ausgegeben.

`\providepagestyle` – definiert einen neuen Seitenstil nur dann, wenn dieser vorher noch nicht existiert.

Am Beispiel von `\defpagestyle` soll die Syntax der Definitionen im Folgenden erläutert werden.

Name – die Bezeichnung des Seitenstils

Kopfdefinition – die Deklaration des Seitenkopfes bestehend aus fünf Teilen, wobei die in runden Klammern stehenden Angaben optional sind:
(OLL,OLD){GS}{US}{ES}(ULL,ULD)

Fußdefinition – die Deklaration des Seitenfußes bestehend aus fünf Teilen, wobei die in runden Klammern stehenden Angaben optional sind:
(OLL,OLD){GS}{US}{ES}(ULL,ULD)

Wie zu sehen ist, haben Kopf- und Fußdefinition identischen Aufbau. Die einzelnen Parameter haben folgende Bedeutung:

OLL – obere Linienlänge: Kopf = außen, Fuß = Trennlinie

OLD – obere Liniendicke

GS – Definition für die *gerade* Seite

US – Definition für die *ungerade* Seite

ES – Definition für *einseitiges* Layout

ULL – untere Linienlänge Kopf = Trennlinie, Fuß = außen

ULD – untere Liniendicke

Werden die optionalen Linienargumente nicht gesetzt, dann bleibt das Verhalten weiterhin durch die in Unterabschnitt 1.1.3, Seite D.10 vorgestellten Linienbefehle konfigurierbar.

Die drei Felder *GS*, *US* und *ES* entsprechen Boxen, die die Breite des Kopf- bzw. Fußteils haben. Die entsprechenden Definitionen erscheinen in diesen Boxen linksbündig. Um somit etwas links- *und* rechtsseitig in den Boxen zu platzieren, kann der Zwischenraum mit `\hfill` gestreckt werden:

```
{\headmark\hfill\pagemark}
```

Um zusätzlich etwas zentriert erscheinen zu lassen, ist eine erweiterte Definition notwendig. Die Befehle `\rlap` und `\llap` setzen die übergebenen Argumente. Für LaTeX erscheint es aber so, dass diese Texte eine Breite von Null haben. Nur so erscheint der mittlere Text auch wirklich zentriert.

```
{\rlap{\headmark}\hfill zentriert\hfill\llap{\pagemark}}
```

Beispiel: Angenommen es wird die Dokumentklasse scrbook genutzt. Damit liegt ein zweiseitiges Layout vor. Für das Paket scrpage2 wird festgelegt, dass mit automatisch aktualisierten Kolumnentiteln gearbeitet wird und dass im Seitenstil `scrheadings` eine Trennlinie zwischen Kopf und Text gezogen wird.

```
\documentclass{scrbook}
\usepackage[automark,headsepline]{scrpage2}
```

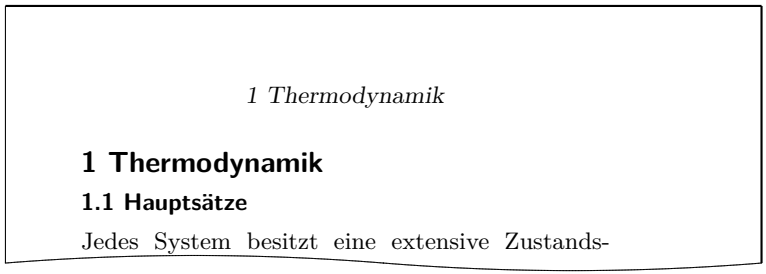
Mit Hilfe der Expertenschnittstelle werden zwei Seitenstile definiert. Der erste legt keine Linienargumente fest, im zweiten wird die Linie über dem Kopf mit einer Dicke von 1 pt und die Linie unter dem Kopf mit 0 pt festgelegt.

```
\defpagestyle{ohneLinien}{%
  {Beispiel\hfill\headmark}
  {\headmark\hfill ohne Linien}
  {\rlap{Beispiel}\hfill\headmark\hfill%
   \llap{ohne Linien}}
}{%
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
}
\defpagestyle{mitLinien}{%
  (\textwidth,1pt)
  {mit Linien\hfill\headmark}
  {\headmark\hfill mit Linien}
  {\rlap{\KOMAScript}\hfill \headmark\hfill%
   \llap{mit Linien}}
  (0pt,0pt)
}{%
  (\textwidth,.4pt)
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
  (\textwidth,1pt)
}
```

Gleich zu Beginn wird der Seitenstil `scrheadings` aktiviert. Mit `\chapter` wird ein neues Kapitel begonnen. Weiterhin wird automatisch durch `\chapter` der Seitenstil für diese Seite auf `plain` gesetzt. Das folgende `\chead` zeigt, wie durch Modifikation des plain-Stils ein Kolumnentitel erzeugt werden kann. Grundsätzlich sollte jedoch davon Abstand genommen werden, da sonst der Markierungscharakter der plain-Seite verloren geht. Es ist wichtiger anzuzeigen, dass hier ein neues Kapitel beginnt, als dass ein Abschnitt dieser Seite einen bestimmten Titel trägt.

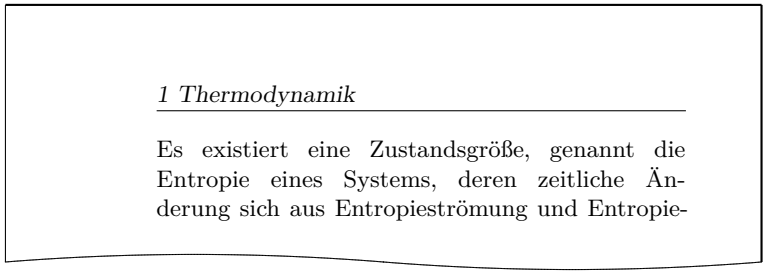
```
\begin{document}
\pagestyle{scrheadings}
\chapter{Thermodynamik}
\chead[\leftmark]{}
\section{Hauptsätze}
Jedes System besitzt eine extensive Zustandsgröße
```

```
Energie. Sie ist in einem abgeschlossenen System
konstant.
```



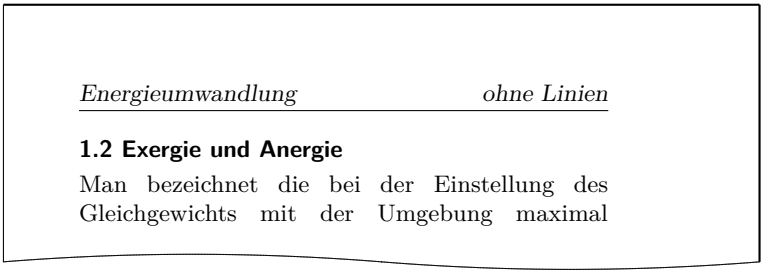
Nach dem Seitenwechsel ist der Seitenstil `scrheadings` aktiv, und somit auch die Trennlinie aus den Paketoptionen sichtbar.

```
Es existiert eine Zustandsgröße, genannt die
Entropie eines Systems, deren zeitliche Änderung
sich aus Entropieströmung und Entropieerzeugung
zusammensetzt.
```



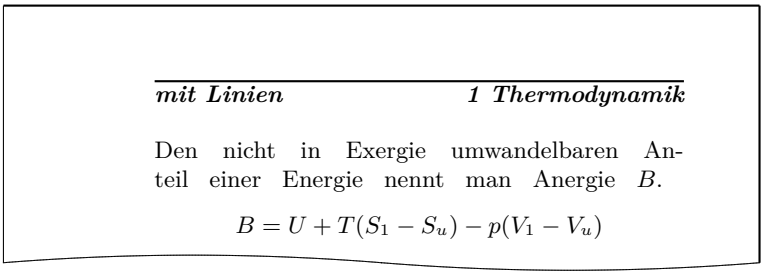
Wiederum nach einem Seitenwechsel wird auf manuelle Kolumnentitel gewechselt und der Seitenstil `ohneLinien` aktiviert. Da keine Linienargumente bei der Definition dieses Stils genutzt wurden, wird die Standard-Linienkonfiguration verwendet. Diese zeichnet hier eine Linie zwischen Kopf und Text, da `headsepline` als Argument für scrpage2 angegeben wurde.

```
\manualmark
\pagestyle{ohneLinien}
\section{Exergie und Anergie}
\markright{Energieumwandlung}
Man bezeichnet die bei der Einstellung des
Gleichgewichts mit der Umgebung maximal gewinnbare
Arbeit als Exergie.
```



Nach dem Wechsel auf die folgende linke Seite wird der Seitenstil `mitLinien` aktiviert. Die Linieneinstellungen werden hier nun angewendet und entsprechend der Definition dargestellt.

```
\pagestyle{mitLinien}
\renewcommand{\headfont}{\itshape\bfseries}
Den nicht in Exergie umwandelbaren Anteil einer
Energie nennt man Anergie \Var{B}.
\[ B = U + T (S_1 - S_u) - p (V_1 - V_u)\]
\end{document}
```



1.2.3 Seitenstile verwalten

Bei längerer Arbeit mit verschiedenen Seitenstilen wird sich, je nach Geschmack und Aufgabenstellung, ein fester Satz an benutzten Stilen etablieren. Um nicht bei jedem neuen Projekt eine große Kopieraktion von den Daten eines Projekts zum neuen Projekt starten zu müssen, liest scrpage2 am Ende seiner Initialisierungsphase die Datei `scrpage.cfg` ein. In dieser Datei können dann Seitenstile definiert sein, die viele Projekte gemeinsam nutzen können.