

# Das Seitenlayout

4.1 Geometrische Dimensionen des Layouts. . . . .	202
4.2 Verändern des Seitenlayouts. . . . .	205
4.3 Dynamische Seitendaten: Seitenzahlen und Textmarken . . . . .	223
4.4 Layouts für Kolumnentitel . . . . .	230
4.5 Visuelle Formatierung . . . . .	242
4.6 Layouts mit Klasse . . . . .	244

Unter einem Satzspiegel versteht man die Fläche auf dem Papier, die mit Text und Abbildungen gefüllt werden soll. Er ist in der Regel nicht auf dem Papier zentriert, und der freie Raum am Kopfsteg und am Bundsteg (Leerraum in der Mitte einer Doppelseite) ist normalerweise kleiner als der freie Raum am Fußsteg und am Seitensteg (Leerraum an der Außenseite). Ein klassisches Seitenverhältnis von *Bund*, *Kopf*, *Außenseite* und *Fuß* zueinander ist z.B. 2 : 3 : 4 : 6. In manchen Fällen enthält der Seitensteg kurze Texte, so genannte *Marginalien*, oder auch erläuternde Texte zu Abbildungen und Tafeln. Eine gesetzte Seite (bzw. Spalte) wird in der Fachwelt auch als *Kolumne* bezeichnet. Eventuelle Kopf- und Fußzeilen, welche die Seitennummer oder andere Informationen über die aktuelle Seite enthalten, nennt man deshalb auch *Kolumnentitel*.

Die Größe, Form und Position all dieser Felder sowie die Struktur der Kolumnentitel bilden zusammen das *Seitenlayout*. In diesem Kapitel wird gezeigt, wie sich verschiedene Seitenlayouts festlegen lassen. Häufig benötigt man schon für ein einziges Dokument unterschiedliche Seitenlayouts. So unterscheidet sich z.B. die erste Seite eines Kapitels, welche die Kapitelüberschrift enthält, meistens von den übrigen Kapitelseiten.

Zunächst werden die Dimensionsparameter vorgestellt, mit denen  $\LaTeX$  das Seitenlayout steuert, und es wird erklärt, wie man sie verändern und ihre Werte bildlich darstellen kann. Danach folgt eine ausführliche Besprechung der Pakete `typearea` und `geometry`, mit deren Hilfe sich die Konfiguration von Seitenlayouts sehr genau steuern lässt.

Im dritten Abschnitt wird erklärt, nach welchen Verfahren  $\LaTeX$  das Datenmaterial für die Kolumnentitel zusammenstellt. Der darauf folgende

Abschnitt legt anhand zahlreicher Beispiele, unter anderem mit dem Paket `fancyhdr`, dar, wie diese Elemente formatiert werden.

Der fünfte Abschnitt stellt Befehle vor, die hilfreich sind, wenn der Text nicht ins Layout passt und manuelle Änderungen erforderlich werden. Das Kapitel schließt mit einem kurzen Blick auf zwei generische Dokumentenklassen, die eine nahezu vollständige Kontrolle über den Prozess zum Festlegen des Seitenlayouts gewähren.

## 4.1 Geometrische Dimensionen des Layouts

Der Text eines Dokumentes nimmt normalerweise eine rechteckige Fläche auf dem Papier ein, den so genannten *Satzspiegel* oder *Textbereich*. Über und unter dem Text (am Kopf und Fuß der Seite) können sich *Kolumnentitel* befinden. Diese können aus einer oder mehreren Zeilen bestehen und die Seitennummer (Folio), Informationen über das aktuelle Kapitel, den Abschnitt, die Zeit und das Datum sowie möglicherweise weitere Informationen enthalten.

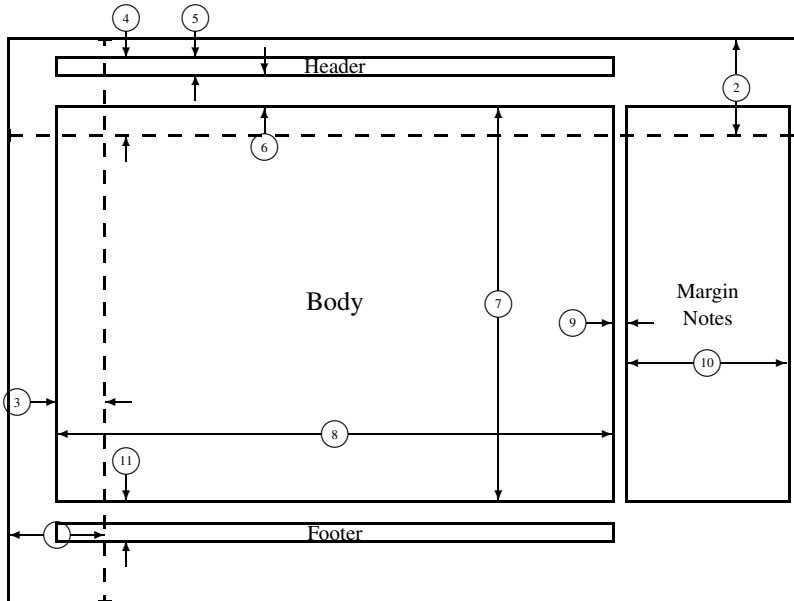
Man spricht von einem *lebenden Kolumnentitel*, wenn dieser Informationen enthält, die sich mit dem Inhalt der aktuellen Seite verändern, wie z.B. den Überschriftentext des derzeitigen Kapitels. Kolumnentitel, die nur aus einer Seitennummer oder aus unveränderlichem Text bestehen, bezeichnet man als *tote Kolumnentitel*. Wenn die Kolumnentitel sehr ins Auge fallen und eng mit dem Text verbunden sind, werden diese Elemente dem Satzspiegel zugerechnet. Das gilt häufig für lebende Kolumnentitel am Seitenkopf, insbesondere wenn diese unterstrichen sind. Andernfalls werden sie zum *Kopf-* oder *Fußsteg* (oberer und unterer Seitenrand) gezählt. Diese Unterscheidung ist wichtig für die Interpretation der Größenangaben.

Die Ränder links und rechts des Textbereiches sind normalerweise leer; in manchen Fällen enthalten sie jedoch kurze Texte, wie z.B. Anmerkungen oder Kommentare, die so genannten *Marginalien*. Im Allgemeinen spricht man hier vom inneren Rand oder *Bundsteg* und vom äußeren Rand oder *Seitensteg*. Beim zweiseitigen Druck ist der Bundsteg der mittlere Rand - d.h. auf rechten (ungeraden) Seiten der linke und auf linken (geraden) Seiten der rechte. Beim einseitigen Druck ist der Bundsteg immer der linke Rand. Bei den Doppelseiten eines Buches liegen ungerade Seiten immer rechts.

Mithilfe der Standarddokumentenklassen von  $\text{\LaTeX}$  lassen sich Dokumente für den Recto-Verso-Druck (zweiseitiger Druck) formatieren. Zweiseitige Layouts können entweder asymmetrisch oder symmetrisch ( $\text{\LaTeX}$ -Voreinstellung) sein. Bei einem symmetrischen Layout sind die Satzspiegel der rechten und linken Seiten deckungsgleich, wenn man ein bedrucktes Blatt gegen das Licht hält. Auch die Marginalien wechseln normalerweise ihre Position je nach linker oder rechter Seite.

Die Dimensionsparameter für das Seitenlayout sind in Abbildung 4.1 auf der nächsten Seite schematisch dargestellt.<sup>1</sup> Die Standardwerte dieser Parameter sind abhängig vom Papierformat. Die Klassendateien von  $\text{\LaTeX}$  unterstützen eine Anzahl verschiedener Optionen, mit denen sich Ausdrücke leichter auf andere Papierformate anpassen lassen. Sie setzen `\paperwidth` und

<sup>1</sup>Die graphische Darstellung wurde mit dem Paket `layouts` erzeugt, siehe Abschnitt 4.2.1.



1	<code>one inch + \hoffset</code>	2	<code>one inch + \voffset</code>
3	<code>\oddsidemargin = -36pt</code>	4	<code>\topmargin = -58pt</code>
5	<code>\headheight = 12pt</code>	6	<code>\headsep = 25pt</code>
7	<code>\textheight = 296pt</code>	8	<code>\textwidth = 418pt</code>
9	<code>\marginparsep = 11pt</code>	10	<code>\marginparwidth = 121pt</code>
11	<code>\footskip = 30pt</code>		<code>\marginparpush = 5pt (not shown)</code>
	<code>\hoffset = 0pt</code>		<code>\voffset = 0pt</code>
	<code>\paperwidth = 597pt</code>		<code>\paperheight = 423pt</code>

Bsp.  
4-1-1

`\paperheight` Papierhöhe.

`\paperwidth` Papierbreite.

`\textheight` Höhe des Satzspiegels (ohne Kolumnentitel).

`\textwidth` Breite des Satzspiegels.

`\columnsep` Spaltenzwischenraum bei Mehrspaltensatz (Zwischensteg).

`\columnseprule` Stärke der Spaltentrennlinie bei Mehrspaltensatz (Voreinstellung 0pt, d.h. unsichtbare Linie).

`\columnwidth` Spaltenbreite bei Mehrspaltensatz. Wird von  $\LaTeX$  aus `\textwidth` und `\columnsep` passend berechnet.

`\linewidth` Breite der aktuellen Zeile; hat normalerweise den gleichen Wert wie `\columnwidth`, kann sich aber in Umgebungen ändern, die andere Ränder setzen.

`\evensidemargin` Bei zweiseitigem Druck zusätzlicher linker Rand auf Verso-Seiten.

`\oddsidemargin` Bei zweiseitigem Druck zusätzlicher linker Rand auf Recto-Seiten, sonst zusätzlicher linker Rand auf allen Seiten.

`\footskip` Vertikaler Abstand zwischen der Grundlinie der letzten Textzeile und der Grundlinie der Fußzeile.

`\headheight` Höhe der Kopfzeile.

`\headsep` Vertikaler Abstand zwischen Kopfzeile und Textbereich.

`\topmargin` Zusätzlicher vertikaler Abstand oberhalb der Kopfzeile.

`\marginparpush` Vertikaler Mindestabstand zwischen zwei aufeinander folgenden Marginalien (nicht abgebildet).

`\marginparsep` Horizontaler Abstand zwischen Textbereich und Marginalien.

`\marginparwidth` Breite der Marginalien.

Abbildung 4.1: Seitenlayoutparameter und ihre Darstellung (mit layouts generiert)

letterpaper	$8\frac{1}{2} \times 11$	Zoll		
legalpaper	$8\frac{1}{2} \times 14$	Zoll		
executivepaper	$7\frac{1}{4} \times 10\frac{1}{2}$	Zoll		
a4paper	$\approx 8\frac{1}{4} \times 11\frac{3}{4}$	Zoll	210 × 297	mm
a5paper	$\approx 5\frac{7}{8} \times 8\frac{1}{4}$	Zoll	148 × 210	mm
b5paper	$\approx 7 \times 9\frac{7}{8}$	Zoll	176 × 250	mm

Tabelle 4.1: Standardoptionen für Papierformate in L<sup>A</sup>T<sub>E</sub>X

`\paperheight` auf die erforderliche Papiergröße und passen die davon abhängigen Werte (wie z.B. `\textheight` und `\textwidth`) entsprechend an.

Tabelle 4.1 zeigt die Optionen für Papierformate, die den L<sup>A</sup>T<sub>E</sub>X-Standardklassen bekannt sind, zusammen mit den entsprechenden Papiermaßen. Tabelle 4.2 auf der nächsten Seite zeigt die Parameterwerte des Seitenlayouts für die Papierformat-Option `letterpaper`, die voreingestellt ist, wenn nicht explizit ein anderes Format gewählt wurde. Die aufgeführten Werte sind für die drei L<sup>A</sup>T<sub>E</sub>X-Standarddokumentenklassen (`article`, `book` und `report`) gleich. Andere Papierformat-Optionen können natürlich andere Werte ergeben. Mit Hilfe des folgenden Befehls kann man z.B. auf DIN A4-Papier drucken: `\documentclass[a4paper]{article}`.

Für andere Klassen mögen weitere oder andere Optionen zur Verfügung stehen. Es erscheint jedoch nicht sehr sinnvoll, beispielsweise für die Dokumentenklasse `book` eine Option `a0paper` zu definieren, die unglaublich breite Zeilen erzeugt.

Die meisten Layoutparameter in den Klassendateien von L<sup>A</sup>T<sub>E</sub>X sind direkt von der tatsächlichen Papiergröße abhängig. Daher ändern sie sich automatisch, wenn die Größen `\paperwidth` oder `\paperheight` über eine der Optionen für Papierformate modifiziert werden. Wenn man diese beiden Größen in der Dokumentenpräambel ändert, hat das nicht die gleiche Wirkung, da die anderen Parameter schon berechnet sind, wenn die Präambel gelesen wird.

Standardrand von  
einem Zoll

Der Standard für dvi-Treiber sieht vor, dass der Referenzpunkt für T<sub>E</sub>X ein Zoll unterhalb und rechts der oberen linken Papierecke liegt. Dieser durch räumliche Verschiebung des Referenzpunktes erzeugte zusätzliche Rand legt den *Druckbereich* einer Seite fest. Der Referenzpunkt lässt sich über die Längen `\hoffset` und `\voffset` verschieben. Sie sind auf null voreingestellt und sollten im allgemeinen nicht verändert werden. Man kann jedoch mit ihrer Hilfe auf einfache Weise den gesamten Satzspiegel (Text, Kopf, Fuß und Marginalien) auf der Ausgabefläche verschieben, ohne das Layout zu beeinträchtigen. Die vom Treiber erzeugten Ränder stammen aus T<sub>E</sub>X und werden von L<sup>A</sup>T<sub>E</sub>X für die Parametrisierung des Seitenlayouts nicht benötigt. Über `\topmargin` verschiebt man den Satzspiegel vertikal, über `\oddsidemargin` oder `\evensidemargin` verschiebt man ihn horizontal.

Man sollte dabei allerdings beachten, dass einige dvi-Treiber den Satzspiegel um andere Werte verschieben. Um sicherzugehen, dass der Referenzpunkt richtig positioniert ist, kann man die Testdatei `testpage.tex` (von Leslie Lamport, mit Änderungen von Stephen Gildea) unter L<sup>A</sup>T<sub>E</sub>X mit dem fraglichen dvi-Treiber ausdrucken. Die ausgegebene Seite zeigt die Position des

Parameter	zweiseitiger Druck			einseitiger Druck		
	10pt	11pt	12pt	10pt	11pt	12pt
<code>\oddsidemargin</code>	44pt	36pt	21pt	63pt	54pt	39pt
<code>\evensidemargin</code>	82pt	74pt	59pt	63pt	54pt	39pt
<code>\marginparwidth</code>	107pt	100pt	85pt	90pt	83pt	68pt
<code>\marginparsep</code>	11pt	10pt	10pt	<i>wie links</i>		
<code>\marginparpush</code>	5pt	5pt	7pt	<i>wie links</i>		
<code>\topmargin</code>	27pt	27pt	27pt	<i>wie links</i>		
<code>\headheight</code>	12pt	12pt	12pt	<i>wie links</i>		
<code>\headsep</code>	25pt	25pt	25pt	<i>wie links</i>		
<code>\footskip</code>	30pt	30pt	30pt	<i>wie links</i>		
<code>\textheight</code>	$\underbrace{43 \quad 38 \quad 36}_{\times \backslash baselineskip}$			<i>wie links</i>		
<code>\textwidth</code>	345pt	360pt	390pt	<i>wie links</i>		
<code>\columnsep</code>	10pt	10pt	10pt	<i>wie links</i>		
<code>\columnseprule</code>	0pt	0pt	0pt	<i>wie links</i>		

Tabelle 4.2: Voreinstellungen der Seitenlayoutparameter (letterpaper)


Referenzpunktes relativ zu den Papierrändern. Diese Datei wurde von Rainer Schöpf für  $\LaTeX 2_{\epsilon}$  so umgeschrieben, dass man eine Papierformat-Option angeben kann.

## 4.2 Verändern des Seitenlayouts

Die Werte der Parameter für das Seitenlayout sollte man nur über die Befehle `\setlength` und `\addtolength` verändern. Dabei sollten Änderungen an den geometrischen Parametern nur in Klassen- oder Paketdateien und/oder in der Präambel (d.h. vor dem `\begin{document}`-Befehl) erfolgen. Auch wenn es nicht völlig unmöglich ist sie mitten in einem Dokument zu ändern, wird daraus doch mit ziemlicher Sicherheit nichts Gutes entstehen, da die internen  $\TeX$ -Abläufe sehr komplex und zeitlich voneinander abhängig sind. Wenn man z.B. den Wert von `\textwidth` ändert, kann sich das ganz unerwartet auf den lebenden Kolumnentitel auf der vorhergehenden Seite auswirken.

Vertikale Abstände sollte man möglichst als Vielfaches des  $\TeX$ -Parameters `\baselineskip` initialisieren. Dieser Parameter gibt den Abstand zwischen den Grundlinien zweier aufeinanderfolgender Zeilen (Durchschuss) in einem Absatz mit „normalem“ Dokumentenschriftgrad an. Daher kann `\baselineskip` auch als Höhe einer Zeile betrachtet werden. Somit bezieht sich die folgende Einstellung immer auf „zwei Zeilen Text“.

```
\normalsize % normales \baselineskip
\setlength\headheight{2\baselineskip} % Kolumnentitel einstellen
```

 Parameter nur  
in der Präambel  
ändern

Damit `\baselineskip` wirklich richtig eingestellt ist, sollte man zunächst (falls erforderlich) die in einem Dokument verwendeten Fonts festlegen und dann den Befehl `\normalsize` aufrufen, um den Schriftgrad auf die im Dokument verwendete Standardgröße, die so genannte Grund- oder Brotschrift, zu initialisieren.

In manchen Fällen ist es einfacher, die Parameter für das Seitenlayout in Anlehnung an bestimmte typographische Vorgaben zu errechnen. Die Vorgabe „der Text sollte 50 Zeilen enthalten“ kann z.B. durch den unten angegebenen Befehl ausgedrückt werden. Dabei geht man davon aus, dass die Höhe aller Zeilen außer der ersten dem Wert `\baselineskip` entspricht, während die Höhe der obersten Zeile des Textbereiches `\topskip` beträgt (das ist die  $\text{\TeX}$ -Entsprechung zu `\baselineskip` für die erste Zeile, mit einem Standardwert von 10pt).

```
\setlength\textheight{\baselineskip*49+\topskip}
```

Man beachte, dass in den Beispielen in diesem Kapitel das  $\text{\LaTeX}$ -Paket `calc` (welches die Notation von Berechnungen erleichtert) und die erweiterten Steuerfunktionen von  $\text{\LaTeX} 2_{\epsilon}$  verwendet werden (siehe Anhang A, Abschnitte A.3.1 und A.3.2).

Eine Vorgabe wie „der Text sollte eine Höhe von 198mm haben“ kann auf ähnliche Weise erfüllt werden. Die entsprechende Berechnung ist weiter unten aufgeführt. Zunächst ist die Anzahl der Zeilen zu berechnen, die ein Textbereich der gewünschten Höhe enthalten kann: Dazu muss ein Dimensionsparameter durch den anderen geteilt werden, um die entsprechende ganze Zahl zu erhalten. Da  $\text{\TeX}$  diese Operation jedoch nicht direkt durchführen kann, werden die Dimensionsparameter zunächst Zählerregistern zugeordnet. Die dabei benutzte Konvertierung ist sehr präzise, da intern `sp`-Einheiten (skalierete Punkte, siehe Tabelle A.1 auf Seite 888) verwendet werden.

```
\newcounter{tempc} \newcounter{tempcc} % zwei temp. Zähler
\setlength\textheight           % Zielhöhe ohne
    {198mm-\topskip}           % erste Zeile
\setcounter{tempc}{\textheight} % Zähler 1 zuweisen
\setcounter{tempcc}{\baselineskip} % Zähler 2 zuweisen
\setcounter{tempc}%           % Zähler teilen
    {\value{tempc}/\value{tempcc}}
\setlength\textheight{\baselineskip*\value{tempc}+\topskip}
```

Der vertikale Abstand `\topmargin` für den oberen Rand kann ebenfalls angepasst werden, wenn z.B. der obere Rand nur halb so groß sein soll wie der verbleibende Raum unterhalb des Textbereiches. Die folgende Berechnung zeigt, wie der gewünschte Wert für DIN A4-Papier errechnet wird (Papierhöhe ist 297 mm).

```
\setlength\topmargin
    {(297mm-\textheight)/3 - 1in - \headheight - \headsep}
```

Generell gilt: wenn man das Seitenlayout ändert, sollte man einige Grundregeln der Lesbarkeit beachten (siehe z.B. [151]). Untersuchungen gedruckter Texte im englischen Sprachraum haben gezeigt, dass eine Zeile nicht mehr als

zehn bis zwölf Wörter enthalten sollte, was einem Maximum von 60 bis 70 Zeichen pro Zeile entspricht.

Die Anzahl der Zeilen pro Seite ist vom verwendeten Schriftgrad abhängig. Der unten angegebene Code zeigt, wie man die Texthöhe `\textheight` in Abhängigkeit vom normalen Schriftgrad eines Dokumentes berechnet. Dabei ist die Tatsache hilfreich, dass in den meisten Dokumentenklassen der interne L<sup>A</sup>T<sub>E</sub>X-Befehl `\@ptsize` die jeweilige Grundschrift von 10pt, 11pt oder 12pt in der Form 0, 1 oder 2 kodiert. Dieser Befehl wird gesetzt, wenn man eine Option wie etwa 11pt wählt.

```
\ifthenelse{\@ptsize = 0}%      10 pt-Schrift als Grundschrift
  {\setlength\textheight{53\baselineskip}}{}
\ifthenelse{\@ptsize = 1}%      11 pt-Schrift als Grundschrift
  {\setlength\textheight{46\baselineskip}}{}
\ifthenelse{\@ptsize = 2}%      12 pt-Schrift als Grundschrift
  {\setlength\textheight{42\baselineskip}}{}
\addtolength\textheight{\topskip}
```

Ein weiterer wichtiger Parameter ist die Größe des Weißraums um den Text. Da Druckschriften sehr wahrscheinlich auch gebunden oder geheftet werden, sollte am inneren Rand, eben dem Bundsteg, immer genug Raum dafür gelassen werden. Wenn `\oddsidemargin` festgelegt ist, berechnet sich beim zweiseitigen Druck der Wert für `\evensidemargin` aus der Gleichung:

```
Papierbreite =
  1in + \oddsidemargin + \textwidth + \evensidemargin + 1in
```

In den meisten Klassen wird der zweiseitige Druck durch die Klassenoption `twoside` aktiviert, welche die boolesche Variable `@twoside` auf `true` (wahr) setzt. Mithilfe des Paketes `ifthen` kann man in Abhängigkeit vom Wert dieser Variablen und unter Berücksichtigung der gewählten Grundschrift weitere Parameter einstellen:

```
\ifthenelse{\@ptsize = 0}%      10 pt-Schrift als Grundschrift
  {\setlength\textwidth{5in}%
   \setlength\marginparwidth{1in}%
   \ifthenelse{\boolean{@twoside}}%
     {\setlength\oddsidemargin {0.55in}%      zweiseitig
      \setlength\evensidemargin{0.75in}}%
     {\setlength\oddsidemargin {0.55in}%      einseitig
      \setlength\evensidemargin{0.55in}}%
   }{}
\ifthenelse{\@ptsize = 1}{...}%  11 pt-Schrift als Grundschrift
\ifthenelse{\@ptsize = 2}{...}%  12 pt-Schrift als Grundschrift
```

Wenn ein Dokument viele Marginalien enthält, kann es außerdem angebracht sein, die Ränder des Layouts zu vergrößern. Das (veraltete) Paket `a4` enthält für diese Situation z.B. den Befehl `\WideMargins`. Es stellt die geometrischen Parameter so ein, dass auf Kosten der Textbreite ein Rand von 1,5 Zoll für die Marginalien freigehalten wird (Marginalsatzspalte).



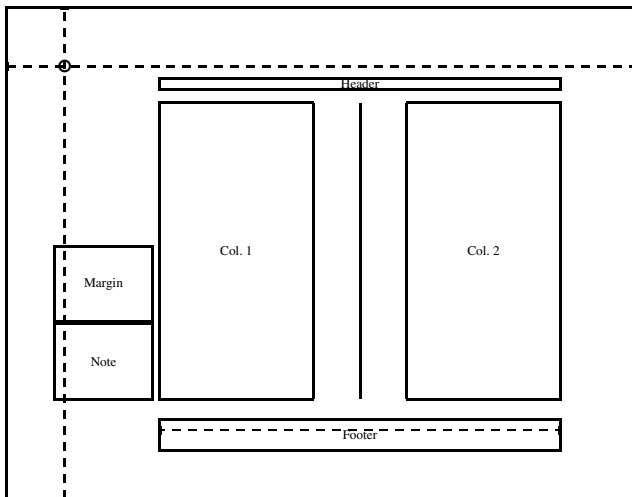


erzeugt eine „gerade Seite“ (voreingestellt sind ungerade Seiten), die Deklaration `\twocolumnlayouttrue` generiert einen Zweispaltensatz (voreingestellt ist einspaltiger Satz). Der Befehl `\reversemarginpartrue` ahmt die Auswirkungen des L<sup>A</sup>T<sub>E</sub>X-Befehls `\reversemarginpar` nach und `\marginparswitchfalse` bewirkt, dass Marginalien auf linken und rechten Seiten immer auf der gleichen Seite des Textbereiches bleiben (eine nützliche Einstellung für asymmetrische Layouts, die sich leicht mit dem Paket `geometry` erzeugen lassen; siehe Seite 217).

Um die Eingabe geeigneter Testwerte zu erleichtern, kann man zunächst den Befehl `\currentpage` nutzen. Er übernimmt die aktuell im Dokument verwendeten Werte als Testwerte und stellt auch die booleschen Schalter entsprechend ein.

Der Seitenfuß wird auf die Höhe einer Zeile voreingestellt, da L<sup>A</sup>T<sub>E</sub>X über keinen expliziten Parameter verfügt, mit dem sich die Boxgröße des unteren Kolumnentitels ändern lässt. Je nachdem welches Kolumnentitel-Layout verwendet wird, kann dieser Wert jedoch ungeeignet sein, wenn das Layout für die Box des unteren Kolumnentitels eine besonders große Tiefe vorgibt. Um in diesem Fall ein (einigermaßen) korrektes Diagramm zu erhalten, kann man mit `\setfootbox`, wie im folgenden Beispiel, die Höhe und Tiefe der Box explizit festlegen.

Das Beispiel zeigt auch, dass man dieses Paket mit dem Paket `calc` kombinieren kann, so dass arithmetische Ausdrücke in den Testdeklarationen möglich sind.



```
\usepackage{calc,layouts}
\setlayoutscales{0.3}
\currentpage
\oddpagelayoutfalse
\twocolumnlayouttrue

\trypaperwidth{11in}
\trypaperheight{8.5in}
\trytextwidth{500pt}
\trytextheight{\topskip
+ 30\baselineskip}
\trycolumnsep{120pt}
\trycolumnseprule{3pt}

\tryheadheight{12pt}
\tryheadsep{18pt}
\tryfootskip{40pt}

\tryevensidemargin{120pt}

\setfootbox{12pt}{24pt}

\setlabelfont{\tiny}
\drawdimensionsfalse
\printheadingsfalse
\pagedesign
```

Lengths are to the nearest pt.

page height = 614pt	page width = 795pt
\hoffset = 0pt	\voffset = 0pt
\evensidemargin = 120pt	\topmargin = 16pt
\headheight = 12pt	\headsep = 18pt
\textheight = 370pt	\textwidth = 500pt
\footskip = 40pt	\marginparsep = 11pt
\marginparpush = 5pt	\columnsep = 120pt
\columnseprule = 3.0pt	

Bsp.  
4-2-2

### Steuern der Darstellung

Die visuelle Darstellung der gedruckten Seitenlayouts wird von einer Reihe von Befehlen gesteuert, von denen einige schon im vorigen Beispiel verwendet wurden. Die wichtigsten von ihnen werden hier besprochen; weitere Erläuterungen findet man in der Dokumentation des Paketes.

Mit der Deklaration `\setlabelfont` kann man den Schriftgrad für die Textlabel ändern. Mit `\setparameterfont` lässt sich auf ähnliche Weise der Schriftgrad für angezeigte Parameter beeinflussen (z.B. Beispiel 4-2-1 auf Seite 208).

Der vorgegebene englische Text über dem Beispiel kann mit der Deklaration `\printheadingsfalse` unterdrückt werden. Der boolesche Schalter `\printparametersfalse` blendet die tabellarische Auflistung von Parameterwerten unterhalb des Diagramms aus. Eine ähnliche Tabelle kann mit dem Befehl `\pagevalues` einzeln generiert werden.

Mit `\drawdimensionstrue` werden Pfeile eingezeichnet, die zeigen, wo sich die Parameter auswirken (normalerweise ist diese Eigenschaft bei `\pagedesign` ein- und bei `\pagedesign` ausgeschaltet).

### Andere Layoutobjekte abbilden

Das Paket `layouts` beschränkt sich nicht auf das Seitenlayout. Es unterstützt auch die Darstellung anderer Objekte. Mithilfe von acht „Diagramm“-Befehlen können weitere  $\text{\LaTeX}$ -Layoutparameter visualisiert werden. Der Befehl `\listdiagram` bildet die Parameter ab, die bei der Darstellung von Listen Verwendung finden, (er wird in Abbildung 3.3 auf Seite 153 verwendet). Der Befehl `\tocdiagram` zeigt, welche Befehle Inhaltsverzeichnisse und Ähnliches beeinflussen und wie diese zusammenhängen. Parameter für Gleitobjekte werden über `\floatdiagram` und `\floatpagediagram` dargestellt. Parameter für Gliederungsbefehle lassen sich mit `\headingdiagram`, solche für Fußnoten und Absätze mit `\footnotediagram` und `\paragraphdiagram` veranschaulichen. Und der Befehl `\stockdiagram` schließlich erzeugt ein Seitenlayout-Diagramm, ähnlich wie `\pagediagram`, wobei jedoch nur die Parameter der Dokumentenklasse `memoir` und ihrer Abkömmlinge dargestellt werden (vgl. Abschnitt 4.6.2 auf Seite 245).

Es gibt außerdem entsprechende „Design“-Befehle, wie zum Beispiel `\listdesign`, `\tocdesign`, `\floatdesign`, `\floatpagedesign`, `\headingdesign` usw., mit deren Hilfe man verschiedene Parametereinstellungen ausprobieren kann. Jeder Parameter kann über eine Deklaration `\try{param}` eingestellt werden. Eine vollständige Liste der so unterstützten Parameter ist in der Dokumentation zum Paket enthalten. Wenn man die entsprechenden  $\text{\LaTeX}$ -Parameter bereits kennt (oder sie aus dem Ergebnis der „Diagramm“-Befehle abliest) kann man sofort anfangen zu experimentieren.

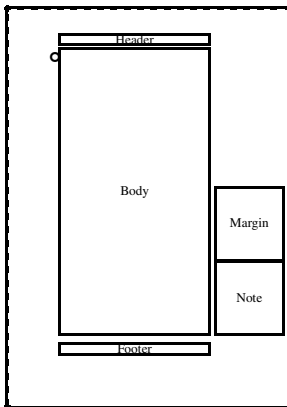
## 4.2.2 Eine Sammlung von Paketen für Seitenlayouts

Da die Originalklassen von  $\text{\LaTeX}$  auf amerikanischen Papierformaten basieren, haben europäische Anwender verschiedene Pakete entwickelt, mit denen die Seitenlayoutparameter auf metrische Formate angepasst werden können. Alle diese Pakete wurden durch das `typearea`- und das `geometry`-Paket (die in den nächsten zwei Abschnitten beschrieben werden) abgelöst. Neue Dokumente sollten immer mit diesen Paketen erstellt werden.

Da frühere Ansätze immer noch in den Archiven zu finden sind, werden sie hier kurz umrissen. Beispiele für derartige Pakete sind `a4` (das relativ

kleine Seiten generiert), das gut dokumentierte Paket `a4dutch` (von Johannes Braams und Nico Poppelier) sowie `a4wide` (von Jean-François Lamy), das etwas längere Zeilen erzeugt. Zudem existieren häufig lokal definierte Dateien unter diesen Namen, was zu einem heillosen Durcheinander führt. Für DIN A5-formatige Seiten gibt es die Pakete `a5` und `a5comb` von Mario Wolczko. Bei all diesen Paketen bestand das Problem darin, dass sie kaum oder gar nicht erlaubten, die Größe und Position des Textbereiches anzupassen. Von einigen gibt es sogar inkompatible Implementierungen.

Das Paket `vmargin` von Volker Kuhlmann folgt einem allgemeineren Ansatz. Sein Paket unterstützt eine Vielzahl von Papierformaten und ermöglicht dem Anwender eine Reihe von Layoutparametern mit einer einzigen Deklaration festzulegen. Die verbleibenden Werte werden aus diesen Angaben errechnet (es gibt verschiedene Deklarationsvarianten). Im folgenden Beispiel werden die Ränder festgelegt und der Satzspiegel berechnet.



Bsp.  
4-2-3

```
\usepackage{vmargin}
\setpapersize[portrait]{A5}
\setmarginsrb{80pt}{40pt}% links, oben
                        {120pt}{80pt}% rechts, unten
                        {12pt}{10pt}% Kopf: Höhe, Abst.
                        {12pt}{30pt}% Fuß: Höhe, Abst.
\setlength\marginparwidth{100pt}
% Programmcode um das erhaltene Layout darzustellen:
\usepackage{layouts}
\newcommand\showpage{%
  \setlayoutscale{0.25}\setlabelfont{\tiny}%
  \printheadingsfalse\printparametersfalse
  \currentpage\pagedesign}
\showpage
```

Das Paket hebt intern die voreingestellte Verschiebung des Referenzpunktes um einen Zoll (die normalerweise von den TeX-Ausgabetreibern hinzugefügt wird) auf, indem es negative Werte für `\hoffset` und `\voffset` verwendet. Das kann zu Überraschungen führen. Dieses Verhalten wird im Beispiel dadurch deutlich, dass die gestrichelte Linie, die diese Verschiebung normalerweise anzeigt, hinter dem Blattrand verschwunden ist und nur der Kreis bei (1 Zoll, 1 Zoll) verbleibt.

### 4.2.3 typearea – Ein traditioneller Ansatz

In Büchern zum Thema Typographie findet man normalerweise einen Abschnitt, der sich mit dem Seitenlayout befasst. Darin werden häufig Konstruktionsmethoden für die Positionierung des Textkörpers beschrieben und das ein oder andere Kriterium für Textbreite, Zeilenanzahl, Verhältnis der Seitenränder zueinander und weitere Gesichtspunkte aufgeführt.

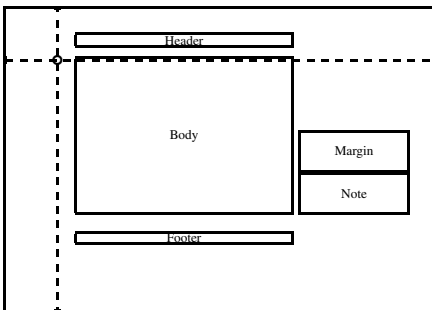
Das Paket `typearea` von Markus Kohm und Frank Neukam, das als Teil der KOMA-Script-Sammlung verteilt wird, stellt Funktionen zur Verfügung, mit deren Hilfe man auf einfache Weise eine der traditionelleren Konstruktionsmethoden für das Seitenlayout einsetzen kann, die seit den frühen Tagen des Buchdrucks in vielen Büchern zur Anwendung kam.

Das bedeutet, ein mithilfe von `typearea` generiertes Seitenlayout weist dem Textbereich das gleiche Größenverhältnis zu, wie es das Papierformat für die Ausgabe des Dokumentes vorgibt. Außerdem ist der Seitensteg doppelt so breit wie der Bundsteg und der Fußsteg doppelt so hoch wie der Kopfsteg.

Bei dieser Konstruktionsmethode wird das Papier horizontal und vertikal in  $n$  gleiche Stücke aufgeteilt, von denen jeweils eines für den Kopf- und den Bundsteg und zwei für den Fuß- und den Seitensteg verwendet werden. Normalerweise berechnet das Paket die Variable  $n$  automatisch. Sie kann aber auch mithilfe der Option `DIVcalc` explizit abgefragt werden (z.B. um die in der Datei `typearea.cfg` gespeicherte Konfiguration zu überschreiben). Diese Option prüft die Grundschrift und wählt für eine Seite im Hochformat einen Wert, der ungefähr 60–70 Zeichen pro Zeile zulässt. Alternativ dazu kann man, um eine vorgegebene Anzahl von Stücken zu erhalten, den Wert für  $n$  über die Option `DIVn` manuell festlegen. Die dritte Möglichkeit besteht in der Option `DIVclassic`, die ein Seitenlayout erzeugt, wie es in bestimmten mittelalterlichen Werken zu sehen ist.

Die Seitenhöhe, die sich aus dem gewählten oder berechneten DIV-Wert ergibt, wird automatisch auf eine ganze Anzahl von Zeilen gerundet. Damit dies funktioniert, muss vorher der effektive Wert von `\baselineskip` für das Dokument festgelegt werden. Wenn man also ein Paket wie `setspace` oder den Befehl `\linespread` einsetzt, dann muss das geschehen, bevor `typearea` geladen wird.

Als Papierformat lässt `typearea` alle entsprechenden Optionen der L<sup>A</sup>T<sub>E</sub>X-Standardklassen zu (siehe Tabelle 4.1 auf Seite 204), sowie alle ISO-A-, ISO-B- und ISO-C-Formate (z.B. `a0paper` oder `c5paper`). Die Orientierung des Textes kann, wie im folgenden Beispiel, mit `landscape` geändert werden.



```
\usepackage[a5paper,landscape,
DIVcalc]{typearea}
% um das erhaltene Layout darzustellen:
\usepackage{layouts}
\newcommand\showpage{%
\setlayoutscales{0.27}\setlabelfont{\tiny}%
\printheadingsfalse\printparametersfalse
\currentpage\pagedesign}
\showpage
```

Bsp.  
4-2-4

Der errechnete DIV-Wert wird zusammen mit den für die anderen Seitenparameter gewählten Werten in der `.log`-Datei des L<sup>A</sup>T<sub>E</sub>X-Laufs gespeichert. Im vorigen Beispiel war dieser Wert 7. Anstelle von `DIVcalc` hätte man also auch `DIV7` verwenden können.

#### Bestimmen des Satzspiegels

Bisher wurde erklärt, wie das Paket die Größe des Satzspiegels wählt und wie es diesen auf der Seite positioniert. Es wurde noch nicht besprochen, ob die Kolumnentitel bei dieser Berechnung berücksichtigt werden. Die Antwort auf diese Frage hängt von ihrem Inhalt ab. Wenn zum Beispiel der lebende Kolumnentitel am Kopf der Seite sehr umfangreich ist, vielleicht zusätzlich mit einer Linie unterstrichen, und damit erheblich zum Grauwert der Seite beiträgt, sollte er als Teil des Satzspiegels betrachtet werden. In anderen

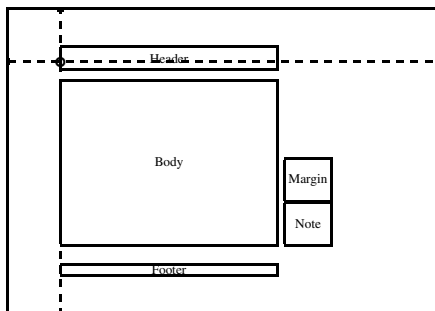
Fällen kann es zutreffender sein, ihn den Rändern zuzurechnen (z.B. wenn er aus sehr lichtem Text mit kleinem Schriftgrad besteht). Aus dem gleichen Grund sollte ein toter Kolumnentitel am Fuß der Seite, der nur eine Seitenzahl (genannt *Pagina*, *Folio* oder *Kolumnenziffer*) enthält, als außerhalb des Satzspiegels angesehen werden und damit nicht in die Berechnungen für dessen Positionierung einfließen.

Für ein bestimmtes Dokument können diese Optionen mit `headinclude`, `footinclude`, `headexclude` und `footexclude` explizit ausgewählt werden. Die letzten beiden Optionen entsprechen der Voreinstellung. Bei hohen DIV-Werten (d.h. schmalen Rändern) können die Kolumnentitel über die Seitenränder hinausgehen, wenn sie bei der Berechnung nicht berücksichtigt werden. In diesem Falle muss die ein oder andere Einstellung nachgebessert werden.

Genauso lässt sich der Marginalienbereich `\marginpar` (mithilfe von `mpinclude` und `mpexclude`) für den linken und rechten Rand ein- bzw. ausschließen. Er wird normalerweise ebenfalls nicht berücksichtigt, aber wenn ein Layout viele Marginalien enthält, kann es angebracht sein ihn einzubeziehen.

Der obere Kolumnentitel hat eine Standardhöhe von 1,25 Zeilen. Dieser Wert lässt sich mit einer Option der Art *numheadlines* einstellen, wobei *num* eine Dezimalzahl (z.B. 2.3) ist, die angibt, über wie viele Zeilen sich der Kopf erstrecken soll.

Im nächsten Beispiel sind der obere Kolumnentitel und die Marginalien einbezogen, wobei der obere Kolumnentitel auf 2,5 Zeilen vergrößert ist. Zum Vergleich empfiehlt sich das in Beispiel 4-2-4 auf der gegenüberliegenden Seite dargestellte Layout, bei dem Kolumnentitel und Marginalien aus dem Satzspiegel ausgeschlossen sind.



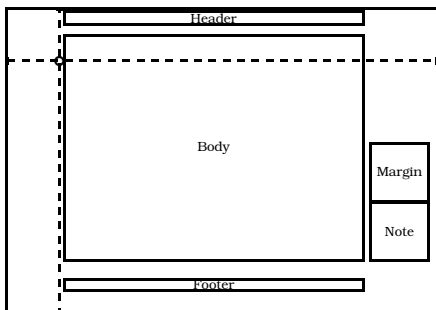
Bsp.  
4-2-5

```
\usepackage[a5paper,landscape,
2.5headlines,
headinclude,mpinclude,
DIVcalc]{typearea}
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Je nach Art der Bindung des fertigen Werkes wird der innere Rand, der Bundsteg, noch mehr oder weniger verschmälert. Um diesen Verlust an Weißraum auszugleichen unterstützt das Paket die Option `BCOR<val>`, wobei *val* (in einer beliebigen L<sup>A</sup>T<sub>E</sub>X-Einheit) der Raum ist, den die Bindung einnimmt. Ein Wert von `BCOR1.2cm` würde die Seitenbreite z.B. um 1,2cm reduzieren, bevor das Seitenlayout berechnet wird.

Neben der Anpassung des Layouts durch Paketooptionen kann man die Parameter auch mit dem Befehl `\typearea` berechnen; die KOMA-Script-Dokumentation enthält hierzu nähere Details. Diese Möglichkeit ist insbesondere dann nützlich, wenn eine Dokumentenklasse der KOMA-Script-Sammlung das `typearea`-Paket bereits lädt, und man in der Präambel des Dokumentes eine ungewöhnliche Schriftart als Grundschrift angeben möchte. In

diesem Fall muss das Layout unter Berücksichtigung des ausgewählten Fonts neu berechnet werden.



```
\usepackage[a5paper,landscape]{typearea}
\usepackage{bookman}

% Syntax: \typearea[<bindungs-korr.>]{<stücke>}
\typearea[10mm]{11}

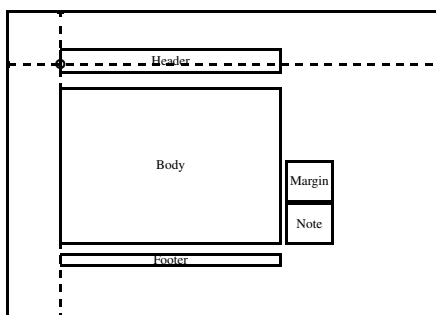
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Bsp.  
4-2-6

#### 4.2.4 geometry – Layouts mit Auto-Vervollständigung

Das geometry-Paket von Hideo Umeki bietet eine universelle und leicht zu bedienende Schnittstelle für alle geometrischen Belange des Seitenlayouts. Es nutzt das Paket keyval, so dass alle Parameter (und ihre Werte) als Optionen von `\usepackage` deklariert werden können.

Anders als das typearea-Paket verwendet geometry kein bestimmtes typographisches Konzept, sondern setzt die eingegebenen Werte wie gewünscht um. Es kennt jedoch das Verhältnis verschiedener Seitenparameter zueinander und kann bei unvollständigen Angaben die restlichen Parameter automatisch berechnen. Das nächste Beispiel zeigt ein Layout, das dem mit typearea erzeugten Beispiel 4-2-5 auf der vorherigen Seite sehr ähnelt. Hier werden eine Reihe von Werten explizit festgelegt (z.B. für den Kopf- und den Bundsteg), der Satzspiegel wird jedoch automatisch aus dem Papierformat (a5paper), den Werten für den Kopfsteg (tmargin) und den Bundsteg (lmargin) und einem expliziten Verhältnis der Seitenränder von 1:2 (marginratio) berechnet.



```
\usepackage[marginratio=1:2,
paper=a5paper,landscape=true,
tmargin=52pt,lmargin=74pt,
headheight=30pt,marginparwidth=62pt,
includehead,includemp]{geometry}
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Bsp.  
4-2-7

Das Beispiel zeigt außerdem, dass boolesche Optionen auch ohne Wert verwendet werden können (so dass die Voreinstellung `=true` greift); für alle anderen Optionen müssen Werte angegeben werden.

Im Folgenden werden die verschiedenen Aspekte des Seitenlayouts erklärt, die von geometry unterstützt werden. In den meisten Fällen führen mehrere Wege zum gleichen Ergebnis, da einige Parameter bestimmten Beziehungen gerecht werden müssen. Verstößt eine Spezifikation gegen eine solche

Beziehung, dann gibt `geometry` eine Warnung aus und ignoriert dann die ein oder andere Einstellung.

Das Papierformat kann mit der Option `paper` eingestellt werden, welche die Werte `a0paper` bis `a6paper`, sowie `b0paper` bis `b6paper` akzeptiert. Daneben können auch die Werte `letterpaper`, `legalpaper` und `executivepaper` verwendet werden. Als Kurzform kann man auch einfach das gewünschte Papierformat als Optionsnamen verwenden; die Option `a5paper` entspricht zum Beispiel der Einstellung `paper=a5paper`.

*Papierformate*

Für die Ausgabe auf dem Bildschirm gibt es die Option `screen`. Bei Formaten, die keinem Standard entsprechen, können die Maße des Papiers explizit mit den Optionen `paperwidth` und `paperheight` spezifiziert werden.

An allgemeinen Seiteneigenschaften unterstützt das `geometry`-Paket die booleschen Optionen `twoside` (zweiseitiger Satz), `landscape` (Papierhöhe und -breite vertauschen), und `portrait`. Dabei ist `portrait=false` das gleiche wie `landscape`.

*Allgemeine Seiteneigenschaften*

Wenn durch die Bindemethode ein Teil der Seite verschwindet, kann man diesen Verlust an Weißraum über die Option `bindingoffset` ausgleichen. Der angegebene Wert wird zum Bundsteg hinzugefügt.

Die boolesche Option `twocolumn` bewirkt, dass der Satzspiegel für Zweispaltensatz eingerichtet wird. In diesem Falle wird der Spaltensteg (der Abstand zwischen den Spalten) mithilfe der Option `columnsep` festgelegt.

In Abschnitt 4.2.3, in dem das `typearea`-Paket beschrieben wird, wurde erklärt, dass es je nach Art eines Dokumentes angebracht sein kann, die lebenden Kolumnentitel (und in manchen Fällen sogar die Marginalsatzspalte) dem Satzspiegel zuzurechnen. Das Paket `geometry` ist so voreingestellt, dass es Kolumnentitel und Marginalien nicht mit einbezieht. Da diese Einstellungen das Verhältnis von Textbereich und Rändern zueinander verändern und sich damit auf automatisch berechnete Werte auswirken, sollten sie passend festgelegt werden. Die Voreinstellungen lassen sich mithilfe einer Reihe boolescher Optionen<sup>1</sup> ändern: `includemp` schließt die Marginalien mit ein, was jedoch selten erforderlich ist; `includehead` wird bei schweren lebenden Kolumnentiteln am Seitenkopf verwendet; `includefoot` ist kaum je erforderlich, da der Kolumnentitel am Fuß der Seite normalerweise nur die Kolumnenziffer (Seitenzahl) enthält; und `includeheadfoot` und `includeall` sind Abkürzungen für Kombinationen der anderen Optionen.

*Bestandteile des Satzspiegels*

Fußnoten werden immer als Teil des Satzspiegels angesehen. Mit der Option `footnotesep` wird nur der Abstand zwischen der letzten Textzeile und den Fußnoten festgelegt; die Berechnung der Ränder bleibt davon unbeeinträchtigt.

Die Größe des Satzspiegels lässt sich nach mehreren Methoden festlegen; es ist weitgehend eine Frage der persönlichen Vorliebe, welche dieser Methoden man verwendet. Man kann sie bestimmen, indem man explizit Werte für `textwidth` und `textheight` vorgibt. In diesem Fall muss `textheight` normalerweise eine ganzzahlige Anzahl an Textzeilen enthalten, damit es bei Seiten, die ausschließlich Text enthalten, nicht zu „Underfull box“-Warnungen kommt. Für diesen Zweck lässt sich sehr gut die `lines`-Option einsetzen, die mithilfe von `\baselineskip` und `\topskip` einen geeigneten Wert für `\textheight` errechnet.

*Satzspiegel*

<sup>1</sup>Das Paket `typearea` bietet die gleiche Funktionalität mit ähnlichen (aber effektiv doch anderen) Optionsnamen, wie `headinclude` anstatt `includehead`.



Alternativ kann man die boolesche Option `heightrounded` verwenden, die `geometry` veranlasst, den Wert von `\textheight` passend zu runden. Diese boolesche Option ist besonders nützlich, wenn der Satzspiegel automatisch berechnet wird – zum Beispiel, wenn man nur für einige der Ränder Werte angibt und die Größe der übrigen dem Paket überlässt.

Mit den zuvor genannten Optionen wird die Größe des Textbereiches vorgegeben und das Paket errechnet durch Hinzufügen der Kolumnentitel und/oder Marginalien daraus die Werte für den Satzspiegel. Stattdessen kann man auch Werte für den gesamten Satzspiegel vorgeben und das Paket daraus den Textbereich berechnen lassen. Dazu werden die Optionen `width` und `height` verwendet (dieser Ansatz macht natürlich nur einen Unterschied zum vorhergehenden, wenn man die Kolumnentitel in den Satzspiegel einschließt). Bei dieser Methode sollte man das Paket mithilfe von `heightrounded` den erforderlichen Wert für `\textheight` berechnen lassen.

Wenn man nicht gerne feste Werte vorgibt, sondern lieber den Satzspiegel vom Papierformat abhängig macht, kann man dazu die Optionen `hscale` und `vscale` verwenden. Sie bestimmen den horizontalen bzw. vertikalen Anteil der Seitengröße, der vom Satzspiegel eingenommen werden sollte.

Die Größe der Ränder kann mithilfe der Optionen `lmargin`, `rmargin`, `tmargin` und `bmargin` (für den Bund-, Seiten-, Kopf- bzw. Fußsteg) explizit festgelegt werden. Wenn die boolesche Option `twoside` den Wert `true` hat, beziehen sich `lmargin` und `rmargin` auf den mittleren und den äußeren Rand, so dass die Optionsnamen etwas irreführend sind. Um dem gerecht zu werden unterstützt das Paket alternativ auch die Optionsnamen `inner` und `outer` – sie beziehen sich jedoch auf die gleichen Optionen. Zusammen mit der Option `asymmetric` für asymmetrische Layouts, die im Folgenden beschrieben wird, wären sie erneut missverständlich. Um dem Anwender noch mehr freie Auswahl zu bieten, gibt es zusätzlich einen weiteren Satz von Optionsnamen: `left`, `right`, `top` und `bottom`. Wenn man nur Vorgaben für Verso-Seiten (linke oder gerade Seiten) machen will (und die Recto-Seiten [rechte oder ungerade Seiten] automatisch mithilfe der Optionen `twoside` oder `asymmetric` erzeugen lässt), ist die erste oder die letzte Gruppe von Optionsnamen wahrscheinlich die beste Wahl.

Wenn keine oder nicht alle Werte für die Ränder vorliegen, werden die fehlenden errechnet. Mit der Gleichung

$$\text{paperwidth} = \text{left} + \text{width} + \text{right} \quad (4.1)$$

$$\text{paperheight} = \text{top} + \text{height} + \text{bottom} \quad (4.2)$$

lässt sich aus zwei Werten auf der rechten Seite der dritte Wert bestimmen (anstatt mit `width` oder `height` kann der Satzspiegel auch mit einer der zuvor besprochenen Methoden festgelegt werden). Wenn auf der rechten Seite nur ein Wert vorliegt, verwendet das Paket zwei weitere Gleichungen um die Anzahl der freien Variablen zu reduzieren:

$$\text{left/right} = \text{hmarginratio} \quad (4.3)$$

$$\text{top/bottom} = \text{vmarginratio} \quad (4.4)$$

Dabei ist die Option `hmarginratio` auf 2:3 eingestellt, wenn `twoside true` ist, sonst auf 1:1. Voreinstellung für `vmarginratio` ist immer 2:3.

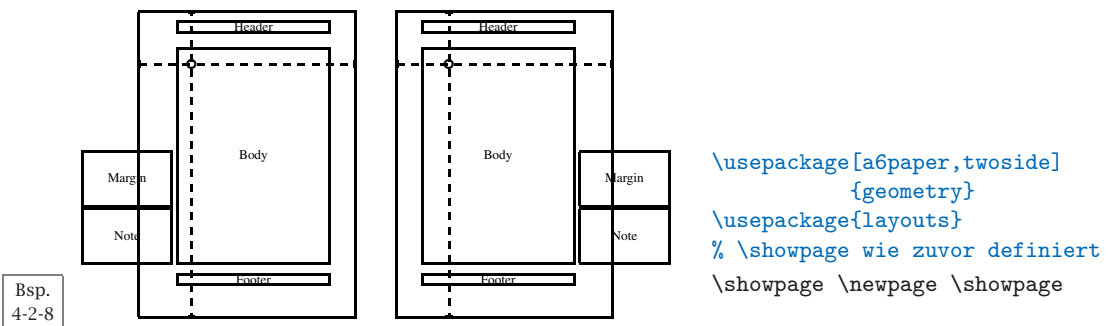


Die Werte für diese „Verhältnis“-Optionen unterliegen folgenden Beschränkungen: Beide Werte müssen positive ganze Zahlen unter 100 sein, die durch einen Doppelpunkt getrennt sind; man schreibt also beispielsweise 4:5, nicht jedoch 1:1.25.

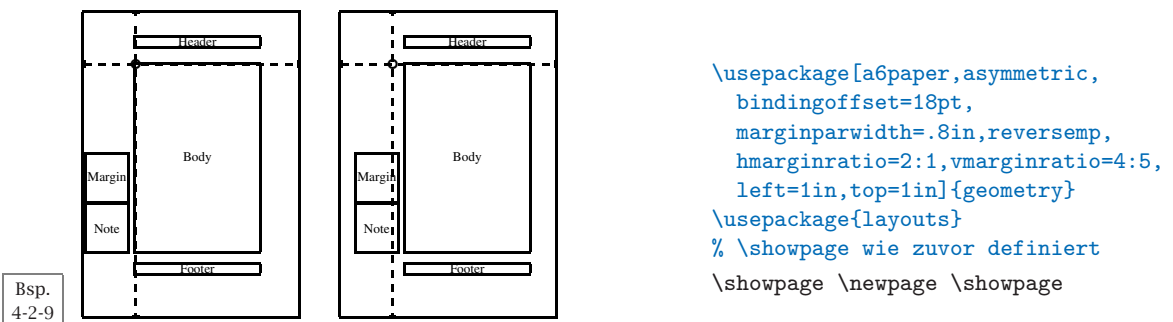
Mit der Option `centering` wird der Satzspiegel zentriert. Das ist ein schneller Weg `hmarginratio` und `vmarginratio` jeweils auf 1:1 zu setzen.

In den L<sup>A</sup>T<sub>E</sub>X-Standardklassen erfüllt die Option `twoside` zwei Funktionen: Sie bereitet die Kolumnentitel auf unterschiedliche Inhalte für gerade und ungerade Seiten vor und stellt automatisch ein symmetrisches Layout ein, bei dem auf geraden Seiten der linke und rechte Rand (inklusive der Marginalien) vertauscht werden. Das Ergebnis ist aus dem nächsten Beispiel ersichtlich, das auch zeigt, dass die `geometry`-Voreinstellungen einen sehr großen Textbereich erzeugen, dabei jedoch die Größe der Marginalboxen nicht an den verbleibenden Rand anpassen.

*Asymmetrische und symmetrische Layouts*



Das `geometry`-Paket erlaubt es, asymmetrische Layouts einfach mithilfe der Option `asymmetric` festzulegen. Der Einsatz von `bindingoffset` im nächsten Beispiel beweist, dass wirklich ein asymmetrisches zweiseitiges Layout erzeugt wird, da der innere und nicht immer der linke Rand verändert wird, obwohl die Marginalien immer links erscheinen. Da der größere Rand links sein soll, muss `hmarginratio` entsprechend geändert werden. Auf den ersten Blick erscheint der rechte Rand der Verso-Seiten im Verhältnis zur Marginalsatzspalte bei einem Wert von 2:1 für `hmarginratio` vielleicht als zu groß; das liegt jedoch daran, dass dort der zusätzliche Bundsteg von `bindingoffset` hinzugefügt wird.



*Lebende Kolumnentitel*

Die Maße des oberen Kolumnentitels und sein Abstand zum Text können mit den Optionen `headheight` und `headsep` festgelegt werden. Der Abstand zwischen Textbereich und unterem Kolumnentitel wird über `footskip` gesteuert. Außerdem kann man die entsprechenden Werte mithilfe der booleschen Optionen `nohead`, `nofoot` und `noheadfoot` auf null setzen. In den meisten Fällen ist es jedoch besser `ignorehead` und die verwandten Optionen zu verwenden, da diese erlauben, auf einzelnen Seiten Kolumnentitel hinzuzufügen, ohne dass die Berechnung der Ränder davon beeinträchtigt wird.

*Marginalien*

Da die meisten Dokumente nur wenige Marginalien enthalten, wird der Raum, den sie einnehmen, normalerweise bei der Berechnung der Ränder nicht berücksichtigt. Dieser Raum lässt sich mit `marginparwidth` und sein Abstand zum Text mit `marginparsep` festlegen. Solange `includemp` nicht verwendet wird, liegt es beim Anwender sicherzustellen, dass dieser Bereich in den errechneten oder festgelegten Rand passt.

Die Marginalien sind so voreingestellt, dass sie im Seitensteg erscheinen. Diese Einstellung lässt sich mit der booleschen Option `reversemp` umkehren.

*Verschiedene Funktionen*

Anstelle eines externen Paketes wie `layouts` kann `geometry` die erzeugten Seiten auch selbst über die eingebaute Option `showframe` darstellen. Alle Einstellungen, einschließlich der berechneten Werte, werden normalerweise in der Protokolldatei des aktuellen  $\LaTeX$ -Laufes gespeichert. Mithilfe der booleschen Option `verbose` werden diese Einstellungen außerdem zusätzlich am Bildschirm angezeigt.

Einige  $\TeX$ -Erweiterungen oder Gerätetreiber wie `pdf $\TeX$`  oder `V $\TeX$`  benötigen die Maße des Papierformates, auf dem die Ausgabe erfolgen soll. Dem trägt das `geometry`-Paket durch die Optionen `pdftex`, `vtex`, `dvipdfm` und `dvips` Rechnung, von denen eigentlich immer eine ausgewählt werden sollte. Wird ein Dokument mit dem Programm `pdf $\TeX$`  generiert, so wird automatisch die Option `pdftex` gewählt (und alle anderen ausgeschaltet).

Wie die meisten modernen Pakete unterstützt `geometry` die erweiterte Syntax des Paketes `calc`, wenn letzteres vor `geometry` geladen wird.

Um ein ungewöhnliches Druckerverhalten auszugleichen, verfügt  $\LaTeX$  über die zwei Dimensionsparameter `\hoffset` und `\voffset`, welche die Ausgabe (für alle Seiten) um den angegebenen Wert horizontal nach rechts und vertikal nach unten verschieben. Das Paket spricht diese Variablen über die Optionen `hoffset` und `voffset` an. Sie haben keinerlei Auswirkungen auf die Berechnung der übrigen Seitenmaße.

*Vergrößerung*

$\TeX$  verfügt über eine Vergrößerungsfunktion, die alle verwendeten Maße und Fonts um einen angegebenen Faktor vergrößert. In  $\LaTeX$  ist diese Funktion normalerweise deaktiviert, doch das `geometry`-Paket stellt sie dem Anwender über die Option `mag` wieder zur Verfügung. Ihr Wert muss eine ganze Zahl sein, wobei 1000 keine Vergrößerung bedeutet. Der Wert `mag=1414` würde bei dem Papierformat `a5paper` einen Ausdruck im Format `a4paper` erzeugen, da alle Maße um  $1,414(=\sqrt{2})$  vergrößert werden, also um den Faktor, um den sich die aufeinanderfolgenden Formate der ISO-A-Serie voneinander unterscheiden. Dies kann nützlich sein, wenn man zum Beispiel den Ausdruck nachträglich photomechanisch verkleinern möchte, um eine höhere Auflösung für den Druck zu erhalten. Da diese Option Fonts skaliert anstatt Originalfonts in der entsprechenden Entwurfsgröße einzusetzen, ist es normalerweise nicht sinnvoll, die Vergrößerung als Endergebnis weiter zu verwenden.

Beim Vergrößern kann man  $\TeX$  anweisen, bestimmte Dimensionen in ihrer Originalgröße zu belassen, indem man den Maßeinheiten die Zeichenfolge `true` voranstellt. Die Eingabe `left=1truein` würde beispielsweise einen linken Rand in der exakten Größe von einem Zoll belassen, egal welcher Vergrößerungsfaktor angegeben wurde. Implizit festgelegte Maße (wie z.B. Papierformate, die über die Option `paper` definiert wurden) werden normalerweise ebenfalls vergrößert, es sei denn, man wählt die Option `trueedimen`.

Mit den zuvor beschriebenen Optionen kann man individuelle Werte festlegen. Für die häufigsten Fälle bietet `geometry` außerdem kombinierte Optionen. Mit ihrer Hilfe lassen sich mehrere Werte auf einmal einstellen: Entweder durch einen einzelnen Wert (der mehrfach verwendet wird) oder als eine durch Kommas getrennte Liste von Werten (die in geschweiften Klammern stehen muss, damit die Kommas nicht fälschlicherweise als Begrenzungszeichen für Optionen interpretiert werden).

*Kurzbefehle*

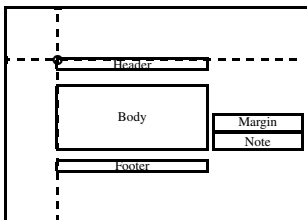
Die Option `papersize` erhält eine Liste von zwei Maßen für die horizontalen und vertikalen Abmessungen der Seite.

Die Option `hmargin` stellt den Bund- und den Seitensteg ein, entweder auf denselben Wert (wenn nur einer angegeben ist) oder auf unterschiedliche Werte (wenn eine Liste angegeben ist). Auf die gleiche Weise stellt `vmargin` den Kopf- und Fußsteg ein. Dieses Vorgehen kann mit der Option `margin` manchmal noch mehr verkürzt werden. Sie gibt ihren Wert (oder die Liste) an `hmargin` und `vmargin` weiter. Genauso gibt `marginratio` seinen Wert zur weiteren Verarbeitung an `hmarginratio` und `vmarginratio` weiter.

Die Abmessungen des Textbereiches können mit der Option `body` festgelegt werden, die einen oder zwei Werte erhält, mit denen sie `textwidth` und `textheight` einstellt. Alternativ dazu kann man auch die Option `total` verwenden, die `width` und `height` bestimmt. Außerdem kann man mit der Option `scale` einen oder zwei Skalierungsfaktoren für `hscale` und `vscale` zur Verfügung stellen.

Wenn das Paket `geometry` als Teil einer Klasse zum Einsatz kommt, möchte man vielleicht einige seiner Einstellungen in der Präambel des eigenen Dokumentes überschreiben. Hier ist die `\usepackage`-Option keine große Hilfe, da das Paket ja bereits geladen ist. Für diese Fälle hält es den Befehl `\geometry` bereit, dessen Argument eine durch Kommas getrennte Liste von Optionen erhält. Der Befehl kann immer wieder aufgerufen werden und überschreibt jedes Mal die zuvor getroffenen Einstellungen. Im nächsten Beispiel wird gezeigt, wie man ihn einsetzt: Zunächst wird das Paket geladen, alle Ränder auf einen Zoll gesetzt, und die Kolumnentitel und Marginalien als Teil des Satzspiegels festgelegt; danach wird der rechte Rand auf zwei Zoll geändert und die Marginalien werden aus der Berechnung herausgenommen.

*Verwendung in der Präambel*



```
\usepackage[a6paper,landscape,
margin=1in,includeall]{geometry}

% Überschreiben einiger Werte:
\geometry{right=2in,ignoremp}

\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Zwei weitere Optionen können sich beim Umgang mit der `\geometry`-Schnittstelle als nützlich erweisen: Mit `reset` stellt man die Voreinstellungen des Paketes wieder her und bei `pass` wird das Paket deaktiviert.

### 4.2.5 lscape – Setzen einzelner Seiten im Querformat

Bei den meisten Dokumenten ist die längere Papierseite die Vertikale (beim so genannten *Hochformat*). Für manche Dokumente, wie z.B. Folien oder Tabellen, ist jedoch das *Querformat*, bei dem die längere Seite horizontal liegt, besser geeignet. Moderne Drucker und `dvi`-Treiber können normalerweise beide Seitenformate drucken.

Hoch- und Querformat benötigen unterschiedliche Seitenlayouts, und Pakete wie `geometry` liefern die Werkzeuge, um diese passend zu erzeugen. Manchmal ist es jedoch wünschenswert, nur für einzelne Seiten zwischen Hoch- und Querformat zu wechseln. In diesem Fall helfen die bisher besprochenen Pakete nicht weiter, da sie ein Seitenlayout für das gesamte Dokument festlegen.

Stattdessen kann man hier das Paket `lscape` von David Carlisle einsetzen, das die Umgebung `landscape` definiert, mit deren Hilfe man eine ausgewählte Gruppe von Seiten im Querformat setzen kann, ohne dass die Kolumnentitel ihre Position verändern. Es beendet zunächst mit `\clearpage` die aktuelle Seite, so dass zuerst alle aufgelaufenen Gleitobjekte gesetzt werden. Dann vertauscht es intern die Werte von `\textheight` und `\textwidth` und dreht alle erzeugten Satzspiegel in seinem Geltungsbereich um 90 Grad. Für die Drehung verwendet es das Paket `graphics`, so dass es mit jedem von diesem Paket unterstützten Gerätetreiber funktioniert, der Rotationen ausführen kann. Wenn die Umgebung endet, wird ein weiterer `\clearpage`-Befehl ausgegeben, bevor wieder zum Hochformat zurückgekehrt wird.

Um Gleitobjekte mit oder ohne ihre Legenden zu drehen, ist das Paket `rotating` die bessere Wahl. Es wird in Abschnitt 6.3.3 beschrieben.

### 4.2.6 crop – Erzeugen von Beschnittmarken

Wenn man eine reprofähige Vorlage erzeugt, wird der endgültige Druck normalerweise auf überformatigem Papier, dem sogenannten „Rohbogen“, aufgebracht. In diesem Fall muss das bedruckte Papier noch zugeschnitten werden, bevor es gebunden werden kann. Damit der Schnitt exakt wird, benötigen Druckereien normalerweise so genannte Beschnittmarken auf jeder Seite. Wenn bei Mehrfarbdrucken mehrere logische (einfarbige) Seiten zu einer mehrfarbigen Seite übereinander gedruckt werden, benötigt man ebenfalls Markierungen, die in diesem Fall Passermarken genannt werden.

Das Paket `crop` von Melchior Franz bietet für diese Anforderungen eine einfache Schnittstelle, mit der verschiedene Arten von Beschnitt- und Passermarken gesetzt werden können. Mit seiner Hilfe lassen sich auch wahlweise nur der Text oder die Abbildungen eines Dokumentes drucken, und die Ausgabe unter anderem invertieren, spiegeln oder drehen – Funktionen, die in dieser Phase des Druckprozesses nützlich sein können.

Beschnitt- oder Passermarken können mit einer der nachfolgend beschriebenen Optionen angefordert werden.

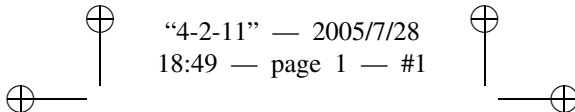
`cam` Erzeugt acht Marken, welche die Maße der Nettoseite anzeigen ohne sie zu berühren (siehe Beispiel 4-2-11). Sie dienen hauptsächlich der Ausrichtung der Kamera.


`cross` Erzeugt vier große Kreuze an den Ecken der logischen Seite, welche die Kanten der logischen Seite berühren.

`frame` Erzeugt einen Rahmen um die Nettoseite; dieser soll hauptsächlich die Abmessungen der Seite verdeutlichen.

Das Paket geht davon aus, dass `\paperheight` und `\paperwidth` die korrekten Maße der *logischen* Seite (Nettoseite) wiedergeben. Die *tatsächliche* Papiergröße (des Rohbogens) für den Druck wird dem Paket als Option übergeben. Diese Optionen sind `a0`, `a1`, `a2`, `a3`, `a4`, `a5`, `a6`, `b0`, `b1`, `b2`, `b3`, `b3`, `b4`, `b5`, `b6`, `executive`, `legal` und `letter`. Wenn das Druckpapier im Querformat verwendet wird, kann man dies zusätzlich mit der Option `landscape` angeben. Wenn keine dieser Optionen zu den tatsächlichen Papiermaßen passt, lassen sich die genauen Maße mit den Optionen `width` und `height` festlegen, die beide Dimensionswerte aufnehmen.

Das nächste Beispiel erzeugt mithilfe des `geometry`-Paketes eine unnatürlich kleine Seite (die in den Beispielbereich in diesem Buch passt) und zentriert diese auf einer Rohseite im Format DIN A5. Da aber alle Beispiele nachträglich auf ihre „sichtbare“ Größe zugeschnitten werden und die Ränder der DIN A5-Seite aus ersichtlichem Grund nicht Teil des Beispiels werden, kann man nicht sehen, dass die Beispielseite in einem Arbeitsgang korrekt zentriert wurde – man muss es entweder glauben oder selbst ausprobieren.



Text für den Satzspiegel  
 um sein Verhältnis zu den Beschnittmarken aufzuzeigen.

```
\usepackage{graphicx,geometry}
\geometry{paperwidth=2in,
paperheight=1.3in,
margin=5mm}
\usepackage[cam,a5,center]{crop}
Text für den Satzspiegel
\includegraphics[width=8mm]
{cat.ps}
um sein Verhältnis zu den
Beschnittmarken aufzuzeigen.
```



Bsp.  
4-2-11

Die Beschreibung und das Beispiel sollten deutlich machen, dass das `crop` erst *nach* Festlegung des Layouts für das Dokument geladen werden sollte.

Der Informationstext zwischen den oberen Beschnittmarken wird automatisch eingefügt. Er kann mit der Option `noinfo` unterdrückt werden, aber normalerweise ist es nützlich ihn beizubehalten. Er enthält sowohl die Seitenanzahl (wie sie  $\LaTeX$  bekannt ist) und einen Seitenindex, der mit 1 beginnt und für jede gedruckte Seite hochgezählt wird. Vor allem bei umfangreichen Publikationen, die verschiedene Methoden zur Seitennummerierung verwenden, kann dieser Ausdruck verhindern, dass die Seiten durcheinander geraten.

Verschiedene Optionen des `crop`-Paketes verlassen sich auf Unterstützung durch den Druckertreiber. Wenn eine explizite Angabe zum Treiber fehlt, versucht das Paket den Druckertreiber aus den Installationseinstellungen für das `graphics`- oder das `color`-Paket zu bestimmen. Es ist aber auch möglich, den Treiber mit Optionen wie `dvips`, `pdflatex` oder `vtex` explizit anzugeben. Ist eine dieser Optionen ausgewählt, dann wird das Papierformat an den externen Treiber übergeben; das ist wichtig, wenn man das Dokument mit `ghostview` oder ähnlichen Programmen ansehen möchte.

Wenn man Graphiken gesondert ausdrucken will – zum Beispiel, weil der Ausdruck des gesamten Dokumentes auf einem Farbdrucker nicht sinnvoll ist – kann man unterschiedliche Versionen des gleichen Dokumentes erzeugen: eine Version, die nur Text und keine Graphiken enthält (genauer gesagt, ohne Graphiken, die mit `\includegraphics` eingebunden sind) und eine, die nur die Graphiken enthält (da der gesamte Text in der Farbe „weiß“ formatiert wurde). Diese Effekte werden mit den Optionen `nographics` bzw. `notext` erreicht. Die letzte Option kann natürlich nur genutzt werden, wenn der verwendete Treiber Farbbefehle unterstützt, da intern das `color`-Paket eingesetzt wird. Das nächste Beispiel<sup>1</sup> zeigt, wie sich die Optionen `nographics` und `cross` auswirken; vergleiche die Ausgabe von Beispiel 4-2-11.

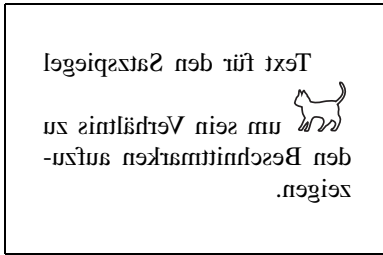
	<p>“4-2-12” — 2005/7/28 18:49 — page 1 — #1</p>	
	<p>Text für den Satzspiegel</p> <p>um sein Verhältnis zu den Beschnittmarken aufzuzeigen.</p>	<pre>\usepackage{graphicx,geometry} \geometry{paperwidth=2in, paperheight=1.3in, margin=5mm} \usepackage[<b>cross</b>,a5,<b>nographics</b>] {<b>crop</b>}</pre>
		<p>Text für den Satzspiegel</p> <pre>\includegraphics[width=8mm] {cat.ps}</pre> <p>um sein Verhältnis zu den Beschnittmarken aufzuzeigen.</p>

Bsp.  
4-2-12

Drei weitere Optionen erfordern, dass der Gerätetreiber die erweiterten Befehle der Pakete `graphics` und `color` zum Drehen und Spiegeln und zum Einfärben des Hintergrundes unterstützt. Mithilfe der Option `rotate` werden die Seiten um 180 Grad gedreht. Die Option `mirror` spiegelt jede Seite, wie im nächsten Beispiel. Und die Option `invert` vertauscht weiß und schwarz, so dass der Text weiß auf schwarzem Hintergrund erscheint.

<sup>1</sup>Die kreuzförmigen Beschnittmarken sehen in dieser Größe zugegebenermaßen ziemlich seltsam aus.

8517\2005 — “4-2-13”  
 I# — I eage — 04:18



```
\usepackage{graphicx,geometry}
\geometry{paperwidth=2in,
          paperheight=1.3in,
          margin=5mm}
\usepackage[frame,a5,mirror]{crop}
Text für den Satzspiegel
\includegraphics[width=8mm]
                {cat.ps}
um sein Verhältnis zu den
Beschnittmarken aufzuzeigen.
```

Bsp.  
4-2-13

## 4.3 Dynamische Seitendaten: Seitenzahlen und Textmarken

Die Ausgaberroutine von  $\LaTeX$ , welche die formatierten Seiten erzeugt, arbeitet asynchron. Das heißt,  $\LaTeX$  sammelt genug Material um gut eine Seite zu füllen, bereitet es auf und baut daraus die Seite auf. Dabei bleibt normalerweise etwas Material übrig, das für die nächste(n) Seite(n) weiter verwendet wird. Während  $\LaTeX$  also Überschriften, Absätze und andere Elemente aufbereitet, ist normalerweise noch nicht abzusehen, auf welcher Seite diese letztendlich erscheinen werden, da sich möglicherweise noch herausstellen wird, dass nicht alles auf die aktuelle Seite passt. Dieses Problem wurde bereits im Abschnitt 3.2.2 im Zusammenhang mit der seitenweisen Nummerierung von Fußnoten angesprochen.

Wenn die fertige Seite gesetzt wird, möchte man vielleicht einige Informationen aus dem Seiteninhalt in lebende Kolumnentitel aufnehmen (z.B. die aktuelle Überschrift), um dem Leser eine zusätzliche Orientierungshilfe zu geben. Solange das Material gesammelt wird, lassen sich keine korrekten Informationen in Befehlen abspeichern: In dieser Phase liest  $\LaTeX$  oft zu weit voraus und der Befehl könnte Daten enthalten, die gar nicht auf der fertigen Seite auftauchen.  $\LaTeX$  löst dieses Problem mithilfe eines Textmarkenmechanismus, mit dessen Hilfe sich „interessante“ Daten markieren lassen. Alle Marken der Seite werden in der Ausgaberroutine gesammelt, welche dann die erste und die letzte der Marken bereitstellt. Dieser Abschnitt erklärt, wie das genau funktioniert, und verweist auf einige nützliche Erweiterungspakete.

### 4.3.1 Seitenzahlen in $\LaTeX$

Die Seitenzahlen werden durch einen Zähler namens `page` gesteuert. Dieser Zähler wird von  $\LaTeX$  automatisch hochgezählt, sobald eine Seite fertiggestellt ist – also *nachdem* er bereits verwendet wurde. Daher muss er mit 1 initialisiert werden, während die meisten anderen  $\LaTeX$ -Zähler bei 0 beginnen und erst kurz vor ihrer Verwendung hochgezählt werden.

Entsprechend der  $\LaTeX$ -Standardkonventionen lautet der Befehl zur typographischen Darstellung der Seitenzahl `\thepage`. Es gibt jedoch noch einen weiteren feinen Unterschied zu anderen  $\LaTeX$ -Zählern: Der Befehl `\thepage` wird nicht vom  $\LaTeX$ -Kern definiert. Er entsteht in dem Moment, wenn zum



ersten Mal die Deklaration `\pagenumbering` ausgeführt wird, was normalerweise in der Dokumentenklassendatei geschieht.

Der beste (wenn vielleicht auch nicht bequemste) Weg, mitten im Text der aktuellen Seite an die Seitenzahl zu gelangen, ist eine Kombination der Befehle `\label` und `\pageref`, die direkt hintereinander stehen sollten, so dass kein Seitenumbruch zwischen ihnen erfolgen kann.

Hier ist Seite 6. Diese Kodierung erzeugt immer ein korrektes Ergebnis, während „Seite 6“ zwar hier funk-

6

tionieren würde, jedoch nicht hier: „Seite 6“, da  $\LaTeX$  beschlossen hat, den Absatz über drei Seiten zu umbrechen.

7

Hier ist Seite `\label{p1}\pageref{p1}`. Diese Kodierung erzeugt immer ein korrektes Ergebnis, während "Seite `\thepage{}`" zwar hier funktionieren würde, jedoch nicht hier: "Seite `\thepage`", da  $\LaTeX$  beschlossen hat, den Absatz über drei Seiten zu umbrechen.

Bsp.  
4-3-1

Aufgrund des asynchronen Verfahrens der Ausgaberroutine kann man `\thepage` innerhalb des Dokumentes nicht ohne Risiko einsetzen. Der Befehl lässt sich nur in den Deklarationen zuverlässig verwenden, die das Erscheinungsbild der von der Ausgaberroutine fertiggestellten Seite beeinflussen.

```
\pagenumbering{layout}
```

Der Befehl `\pagenumbering` setzt den Zähler `page` auf 1 zurück und definiert den Befehl `\thepage` in `\layout{page}` um. Gebrauchsfertige Layouts für Seitenzahlen sind: `Alph`, `alph`, `Roman`, `roman` und `arabic` (siehe auch Abschnitt A.1.4).

Bei Büchern etwa ist es üblich, die Seiten des Vorspanns mithilfe von `roman` in römischen Zahlen zu setzen. Für das erste Kapitel des Hauptteils beginnt die Seitennummerierung dann in arabischen Ziffern von vorne (mit `arabic`). Diesen Effekt kann man auch manuell erzielen, indem man den Befehl `\pagenumbering` zweimal einsetzt; die Befehle `\frontmatter` und `\mainmatter` der Klasse `book` setzen dieses Verfahren implizit im Hintergrund ein.

### 4.3.2 lastpage – Verweise auf die letzte Seite

Standard- $\LaTeX$  kennt keinen Weg, auf die Anzahl der Seiten eines Dokumentes zu verweisen. Man kann also nicht schreiben: „Dieses Dokument besteht aus 6 Seiten“ oder den Text „Seite 5 von 10“ generieren, ohne zuvor die Seiten von Hand zu zählen. Das Paket `lastpage` von Jeffrey Goldberg umgeht dieses Problem, indem es automatisch auf der letzten Seite ein Label namens `LastPage` verwendet, auf dessen Seitenzahl man sich dann mit `\pageref{LastPage}` beziehen kann. Beispiel 4-4-5 auf Seite 234 zeigt, wie man es verwendet.

Die Zeichenfolge, die durch den Aufruf von `\pageref` erzeugt wird, entspricht dem Inhalt von `\thepage` wie er auf der letzten Seite erscheinen würde. Wenn innerhalb des Dokumentes die Nummerierung neu begonnen wird – z.B. weil der Vorspann separat nummeriert ist – gibt die Zeichenfolge nicht die Anzahl aller Seiten an.



Das Paket generiert das Label mithilfe von `\AtEndDocument` um sicherzugehen, dass zunächst alle aufgelaufenen Gleitobjekte platziert werden. Da dieser Befehl jedoch auch von anderen Paketen dazu verwendet werden kann, Textmaterial ans Ende des Dokumentes zu setzen, bleibt immer noch das Risiko, dass das Label zu früh platziert wird. In diesem Fall kann man versuchen, `lastpage` nach dem Paket zu laden, das dieses zusätzliche Material erzeugt.

### 4.3.3 chappg – Kapitelweise Nummerierung der Seiten

Manche Werke erfordern, dass die Seiten in jedem Kapitel separat nummeriert werden und die Seitenzahl zusammen mit der Kapitelnummer auf jeder Seite steht. Das lässt sich mit den bereits verfügbaren Befehlen erreichen, indem man folgenden Code nach jedem `\chapter`-Befehl wiederholt:

```
% Kapitelweise Nummerierung (nach jedem \chapter-Befehl wiederholen:
\pagenumbering{arabic} % zuerst die Seitenzahlen zurücksetzen und dann das ...
\renewcommand\thepage{\thechapter--\arabic{page}} % ... Layout überschreiben
```

Das ist jedoch sehr umständlich und zwingt dazu, sehr viel Layoutinformationen in das Dokument einzubringen, was man besser vermeiden sollte.

Das Paket `chappg`, ursprünglich von Max Hailperin und später von Robin Fairbairns reimplementiert und erweitert, bietet einen besseren Weg. Es unterstützt jede Dokumentenklasse, die über den Befehl `\chapter` verfügt, und führt für die gewünschte Seitenummerierung das Layout `bychapter` ein. Außerdem erweitert es den `\pagenumbering`-Befehl um ein optionales Argument, mit dessen Hilfe man der Seitenzahl ein beliebiges Präfix anstelle der Kapitelnummer voranstellen kann. Das ist zum Beispiel im Vorspann nützlich, da die Überschriften dort normalerweise nicht nummeriert sind.

Bsp.  
4-3-2

... hier befinden wir  
uns mitten im Vorspann,

Vorwort-1

wo Kapitel normaler-  
weise nicht nummeriert

Vorwort-2

```
\usepackage{chappg}
```

```
% \chapter*{Vorwort} % -- nicht gezeigt
\pagenumbering[Vorwort]{bychapter}
\ldots hier befinden wir uns mitten im
Vorspann, wo Kapitel normalerweise
nicht nummeriert sind.
```

Mit etwas Fingerspitzengefühl kann man dieses Paket sogar mit Dokumentenklassen verwenden, die nicht über einen `\chapter`-Befehl verfügen. Angenommen die höchste Gliederungsebene ist `\section` und jeder Abschnitt beginnt automatisch auf einer neuen Seite (das ist eine wichtige Voraussetzung). Dann bewirkt die Deklaration

```
\makeatletter \@addtoreset{page}{section} \makeatother
\pagenumbering[\thesection]{bychapter}
```

dass die Seiten abschnittsweise nummeriert werden. Wenn jedoch nicht jeder Abschnitt auf einer neuen Seite beginnt, kann dieses Verfahren scheitern, da  $\LaTeX$  möglicherweise bereits den Anfang eines neuen Abschnitts gefunden

und den `section`-Zähler hochgezählt hat, ohne die vorherige Abschnittsnummer auf der aktuellen Seite zu vermerken. Dadurch ergibt sich das gleiche Problem, das bereits zuvor für `\thepage` beschrieben wurde.

Auch das Zeichen zwischen dem Präfix und der Seitenzahl lässt sich anpassen, da er mit dem Befehl `\chappgsep` erzeugt wird. Damit generiert

```
\renewcommand\chappgsep{/}
```

in Kapitel „2“ Seiten mit der Nummerierung 2/1, 2/2, 2/3 und so weiter.

#### 4.3.4 Textmarkenbefehle

In Paketdateien, die sich mit Seitenlayouts oder Ausgaberroutinen befassen, findet sich der  $\text{T}_\text{E}\text{X}$ -Basisbefehl `\mark`. Er ist dafür zuständig, irgendwelche Informationen (sein Argument) mit einer Position auf einer Seite zu verknüpfen (d.h. mit der Position, an welcher der `\mark`-Befehl ausgeführt wurde). Wenn die fertige Seite erzeugt wird, speichert  $\text{T}_\text{E}\text{X}$  die erste Marke auf der zusammengestellten Seite in `\firstmark`, die letzte in `\botmark` und die `\botmark` der vorigen Seite als `\topmark`. Wenn die Seite keine Marken enthält, übernehmen auch `\firstmark` und `\botmark` den Wert der vorherigen `\botmark`. Wenn also jeder Gliederungsbeleg intern mit `\mark` eine Marke setzt, die als Argument den Text der Überschrift aufnimmt, könnte man mithilfe dieser Befehle die erste oder letzte Überschrift der Seite in einem lebenden Kolumnentitel anzeigen.

Diese Befehle können jedoch *nicht* direkt in  $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$  eingesetzt werden, da  $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$  ein darauf aufbauendes Protokoll verwendet, welches die Argumente der Befehle intern strukturiert, um mehr als eine Marke zu ermöglichen. Sie werden hier nur erwähnt, um den grundlegenden Mechanismus dahinter zu erklären; ein direkter Einsatz dieser Befehle würde sehr wahrscheinlich zu seltsamen Fehlermeldungen führen.

Anstelle des `\mark`-Befehls stellt Standard- $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$  die beiden folgenden Befehle zum Erzeugen von Textmarkenpaaren zur Verfügung:

```
\markboth{hauptmarke}{untermarke}   \markright{untermarke}
```

Der erste Befehl platziert ein Textmarkenpaar an der aktuellen Stelle im Dokument. Der zweite erzeugt intern ebenfalls ein Markenpaar, wobei er aber nur die *untermarke* verändert, während er die *hauptmarke* vom vorherigen `\markboth`-Befehl übernimmt.

Ursprünglich wollte man damit einigermaßen voneinander unabhängige Marken erzeugen – zum Beispiel Kapitelüberschriften als *hauptmarken* und Abschnittsüberschriften als *untermarken*. Der Befehlsname `\markright` deutet jedoch schon darauf hin, dass Leslie Lamport ein ganz bestimmtes Markierungsschema vor Augen hatte, als er diese Befehle entwarf. Das wird noch deutlicher, wenn man sich die Befehle ansieht, mit deren Hilfe die Ausgaberroutine die Werte der Marken ausliest.

In der Ausgaberroutine enthält `\leftmark` das Argument *hauptmarke* des letzten `\markboth`-Befehls vor dem Ende der Seite. Der Befehl `\rightmark` enthält die *untermarke* des ersten `\markright`- oder `\markboth`-Befehls auf

Keine Low-Level  
 $\text{T}_\text{E}\text{X}$ -Marken in  $\text{L}^{\text{A}}\text{T}_\text{E}\text{X}$   
verwenden

Textmarkenbefehle	Markenpaar	verwendete Marken	
		\leftmark	\rightmark
\markboth{L1}{}	{L1}{}		
\newpage% -- Seitenumbruch --		L1	
\markright{R1.1}	{L1}{R1.1}		
\markboth{L2}{}	{L2}{}		
\markright{R2.1}	{L2}{R2.1}		
\newpage% -- Seitenumbruch --		L2	R1.1
\markright{R2.2}	{L2}{R2.2}		
\markright{R2.3}	{L2}{R2.3}		
\markright{R2.4}	{L2}{R2.4}		
\newpage% -- Seitenumbruch --		L2	R2.2
\markboth{L3}{}	{L3}{}		
\markright{R3.1}	{L3}{R3.1}		
\newpage% -- Seitenumbruch --		L3	
\newpage% -- Seitenumbruch --		L3	R3.1
\markright{R3.2}	{L3}{R3.2}		
\markboth{L4}{}	{L4}{}		
\markboth{L5}{}	{L5}{}		
\newpage% -- Seitenumbruch --		L5	R3.2
\markright{R5.1}	{L5}{R5.1}		
\end{document}		L5	R5.1

Abbildung 4.2: Arbeitsweise des Textmarkenmechanismus von L<sup>A</sup>T<sub>E</sub>X

der Seite, sofern dort einer dieser Befehle vorkommt; ansonsten enthält es diejenige, die zuletzt definiert wurde.

Diese Textmarkenbefehle funktionieren ziemlich gut, wenn die rechten Marken in Abhängigkeit von den linken Marken nummeriert sind – daher auch die Namen (wenn also z.B. die linken Marken durch den Befehl `\chapter` und die rechten durch den Befehl `\section` geändert werden). Die Ergebnisse werden jedoch etwas ungewöhnlich, wenn einem `\markboth`-Befehl auf der gleichen Seite ein anderer Markenbefehl vorangeht – vgl. die Seiten mit L2 R1.1 und L5 R3.2 in Abbildung 4.2. Diese Abbildung zeigt schematisch, welche linken und rechten Marken für die Ausgabeseiten erzeugt werden. Bei einigen Arten von Kolumnentiteln wäre die Anzeige der ersten *hauptmarke* oder der letzten *untermarke* besser geeignet. Zu diesem Zweck lässt sich das Paket `extramarks`, das im Folgenden noch beschrieben wird, heranziehen, da Standard-L<sup>A</sup>T<sub>E</sub>X diese Möglichkeit nicht bietet. Es sollte auch beachtet werden, dass man keine *hauptmarke* setzen kann, ohne gleichzeitig eine *untermarke* zu setzen und damit zu überschreiben.

Bei Layouts, deren lebende Kolumnentitel aus dem Text von Überschriften erzeugt werden, wäre es schön, wenn die Marken automatisch aus den entsprechenden Gliederungsbefehlen generiert werden. Glücklicherweise gibt es eine Schnittstelle, mit deren Hilfe man festlegen kann, welche Befehle Marken setzen und welcher Text dabei an die Marke übergeben wird. Das funktioniert folgendermaßen: Alle Standard-Gliederungsbefehle rufen intern den Befehl `\namemark` auf, wobei *name* der Name des Gliederungsbefehls ist (z.B. `\chaptermark`, `\sectionmark`). Diese Befehle besitzen ein Argument, das

den Text der Überschrift oder seine Kurzform aus dem optionalen Argument des Gliederungsbefehls aufnimmt.

Sie haben normalerweise keinerlei Auswirkung. Wenn sie jedoch in geeigneter Weise umdefiniert werden, können sie ein Textmarkenpaar erzeugen, wie L<sup>A</sup>T<sub>E</sub>X es benötigt. In der Klasse `book` werden diese Befehle z.B. (ungefähr) folgendermaßen definiert:

```
\renewcommand\chaptermark[1]{\markboth{\chaptername\
\thechapter. #1}{}}
\renewcommand\sectionmark[1]{\markright{\thesection. #1}}
```

Im Falle eines Kapitels wird dadurch der Inhalt von `\chaptername` (Voreinstellung „Chapter“<sup>1</sup>), gefolgt von der Gliederungsnummer des Kapitels (die in `\thechapter` gespeichert ist), sowie der Inhalt (oder die Kurzversion) der Kapitelüberschrift als Argument *hauptmarke* von `\markboth` gespeichert; gleichzeitig wird der Inhalt der *untermarke* gelöscht. Bei einem Abschnitt wird die Abschnittsnummer (aus dem Zähler `section`), gefolgt vom Inhalt (oder der Kurzversion) der Abschnittsüberschrift, an den Befehl `\markright` übergeben, der daraus ein Markenpaar mit einer neuen *untermarke* erzeugt.

### 4.3.5 extramarks – Eine neue Art von Marken

Wie bereits zu sehen war, wurde der Textmarkenmechanismus von L<sup>A</sup>T<sub>E</sub>X für ein ganz bestimmtes Layout entworfen und ist daher für andere Anwendungen nur bedingt geeignet. Daher wurden einige Versuche unternommen ihn durch Funktionen zu erweitern oder zu ersetzen, die komplexere Markensysteme unterstützen.

Teilweise liegen die Grenzen in T<sub>E</sub>X selbst, das nur eine Art von Marken kennt und es damit schwierig (wenn auch nicht unmöglich) macht, verschiedene, voneinander unabhängige Marken zu implementieren. Dieses Problem ist in eT<sub>E</sub>X behoben, das unabhängige Textmarkenklassen zu Verfügung stellt. Da dieses Programm jedoch bis vor kurzem noch nicht sehr weit verbreitet war, gibt es bisher noch keine Pakete, welche die neuen Möglichkeiten des erweiterten Textmarkenmechanismus verwenden.

Das Paket `extramarks` von Piet van Oostrum (das als Teil des `fancyhdr`-Paketes verteilt wird) stellt einen erweiterten Mechanismus innerhalb des zentralen L<sup>A</sup>T<sub>E</sub>X-Modells zur Verfügung. Es bietet zwei zusätzliche, (teilweise) voneinander unabhängige Marken, sowie weitere Steuerungsmöglichkeiten für die L<sup>A</sup>T<sub>E</sub>X-Standardmarken, indem es ermöglicht, dass man auf jeder Seite für beide Markentypen *hauptmarke* und *untermarke* sowohl die erste als auch die letzte Textmarke ansprechen kann.

Das Paket bietet die Befehle `\firstleftmark` und `\lastleftmark`, um die erste bzw. letzte *hauptmarke* auf einer Seite anzusprechen. Auf die gleiche Weise kann man mit `\firstrightmark` und `\lastrightmark` auf die erste bzw. letzte *untermarke* zugreifen.<sup>2</sup> Beispiel 4-4-9 auf Seite 237 zeigt, wie sie verwendet werden.

<sup>1</sup>Für Sprachanpassungen siehe Tabelle 9.2 auf Seite 561.

<sup>2</sup>Genaugenommen sind die Befehle `\lastleftmark` und `\firstrightmark` Synonyme für die L<sup>A</sup>T<sub>E</sub>X-Befehle `\leftmark` und `\rightmark`, nur dass ihre Namen ihre Funktion genauer beschreiben.

```
\extramarks{linke-xmarke}{rechte-xmarke}
```

Mithilfe des Befehls `\extramarks` des Paketes lassen sich zusätzliche Textmarken in das Dokument einfügen. Er besitzt zwei obligatorische Argumente: die Texte für zwei Marken an der aktuellen Position. Der Befehl `\firstleftxmark` verweist auf die erste *linke-xmarke*, `\lastleftxmark` auf die letzte. Genauso können `\firstrightxmark` und `\lastrightxmark` in der Ausgaberroutine verwendet werden, um auf die *rechte-xmarke* zuzugreifen.

Das nächste Beispiel zeigt die praktische Anwendung dieser Befehle. Mithilfe des Paketes `fancyhdr` (das in Abschnitt 4.4.2 beschrieben ist) wird ein Seitenlayout erzeugt, bei dem die erste *linke-xmarke* am Kopf einer Seite und die letzte *rechte-xmarke* rechts unten auf jeder Seite dargestellt wird. Dabei ist der Einsatz des Befehls `\extramarks` besonders interessant. Das Beispiel beginnt mit einem `\extramarks`-Befehl, der den Text „Eine Geschichte“ als *linke-xmarke* und eine leere *rechte-xmarke* besitzt. Unmittelbar darauf folgt ein weiteres Markenpaar mit den Werten „... geht weiter“ und „bitte umblättern“. Das führt dazu, dass die erste *linke-xmarke* auf der ersten Seite den Text „Eine Geschichte“ enthält, auf den folgenden Seiten jedoch „... geht weiter“. Die letzte *rechte-xmarke* auf jeder Seite enthält immer den Text „bitte umblättern“. Solange die Geschichte weitergeht erhält man also immer die richtigen Fortsetzungsmarken am Kopf und Fuß jeder Seite. Am Ende der Geschichte sollte jedoch nicht „bitte umblättern“ stehen. Dazu enthält das Beispiel am Ende einen weiteren `\extramarks`-Befehl, dessen *rechte-xmarke* leer ist. Seine *linke-xmarke* enthält weiterhin den Text „... geht weiter“, damit auch am Kopf der letzten Seite der richtige Text steht.

Eine Geschichte
Text für unsere Seite, der immer wieder verwendet wird. Text für unsere Seite, der immer wieder
bitte umblättern

... geht weiter
verwendet wird.

```
\usepackage{fancyhdr,extramarks}
\pagestyle{fancy} \cfoot{}
\lhead{\firstleftxmark}
\rfoot{\lastrightxmark}
\newcommand\sample{ Text für unsere Seite,
der immer wieder verwendet wird.}
\extramarks{Eine Geschichte}{}
\extramarks{\ldots geht weiter}
{bitte umblättern}
\sample \sample
\extramarks{\ldots geht weiter}{}

```

Bsp.  
4-3-3

Die zusätzlichen Marken können mit den Standardmarken kombiniert werden, die  $\LaTeX$  mit Gliederungsbefehlen oder mit `\markboth` und `\markright` erzeugt. Dabei ist zu beachten, dass sie nicht völlig unabhängig voneinander sind. Immer wenn der Befehl `\extramarks` oder einer der Standard-Textmarkenbefehle von  $\LaTeX$  verwendet wird, generiert  $\LaTeX$  alle vier Marken (wobei die Werte für nicht explizit gesetzte Marken beibehalten werden). Daher kann es sein, dass die erste Marke einer bestimmten Art einen unerwarteten Inhalt hat. Wenn ein Dokument z.B. mit einem `\extramarks`-Befehl anfängt, generiert es implizit eine leere *hauptmarke* und eine leere *untermarke*.

Der Textmarkenmechanismus von  $\TeX$  kennt noch einen dritten Basisbefehl namens `\topmark`, der normalerweise nicht in  $\LaTeX$  verfügbar ist. Dieser

enthält den Wert des Befehls `\botmark` der vorhergehenden Seite, so dass er im Grunde die Marken unmittelbar zu Beginn der Seite wiedergibt – daher auch sein Name. In  $\LaTeX$  ist er normalerweise nicht verfügbar, da er dort mit den Mechanismen für Gleitobjekte und mit `\marginpar` Konflikte erzeugt. Anders ausgedrückt löst jedes dieser Objekte intern die Ausgaberroutine aus, so dass die `\topmark`-Werte für die aktuelle Seite verschwinden.

Wenn aber weder Gleitobjekte noch `\marginpar` zum Einsatz kommen, kann man die Informationen von `\topmark` durchaus verwenden, und genau für diese Fälle bietet `extramarks` eine entsprechende Schnittstelle. Wenn man Verwendung für eine solche Textmarke hat, kann man mit den Befehlen `\topleftxmark` und `\toprightxmark` auf die mit `\extramarks` erzeugte *linke-xmarke* und *rechte-xmarke* zugreifen.

## 4.4 Layouts für Kolumnentitel

Während der Seitenspiegel bei fast allen Seiten gleich bleibt, kann sich das Format der lebenden Kolumnentitel im Verlauf eines Dokumentes ändern. In  $\LaTeX$  werden Layouts für Kolumnentitel *page style* genannt. Sie enthalten verschiedene Formatierungen und ihre Namen wie `empty` oder `plain` lassen schon auf ihre Verwendung schließen.

Mit den Befehlen `\pagestyle` oder `\thispagestyle` können neue Layouts ausgewählt werden. Beide erwarten den Namen des Layouts als obligatorisches Argument. Der erste Befehl legt das Layout für die aktuelle und alle folgenden Seiten fest, der zweite nur für die aktuelle Seite.

Bei kürzeren Dokumenten ist es normalerweise nicht erforderlich, zwischen verschiedenen Layouts für die Kolumnentitel zu wechseln. Normalerweise reicht hier das voreingestellte Layout der Dokumentenklasse aus. Bei umfangreicheren Dokumenten, wie etwa ganzen Büchern, kann man durch typographische Gepflogenheiten, Vorgaben der Verlage oder aus anderen Gründen gezwungen sein, das Layout der Kolumnentitel an bestimmten Stellen manuell zu verändern.

$\LaTeX$  verfügt über vier grundlegende Kolumnentitel-Layouts; besondere Pakete oder Dokumentenklassen können weitere bereitstellen.

`empty` Beide Kolumnentitel sind leer.

`plain` Der obere Kolumnentitel ist leer und der untere enthält die Seitenzahl (Folio).

`headings` Der obere Kolumnentitel enthält durch die Dokumentenklasse festgelegte Informationen sowie die Seitenzahl; der untere ist leer.

`myheadings` Ähnlich wie `headings`, jedoch kann der obere Kolumnentitel vom Anwender bestimmt werden.

Die Standardklassen verwenden die ersten drei Layouts. Normalerweise wird die Titelseite intern mit dem Befehl `\thispagestyle{empty}` gestaltet. Für die erste Seite der Hauptgliederungsbefehle (wie `\part` oder `\chapter`, aber auch `\maketitle`), verwenden die  $\LaTeX$ -Standardklassen den Befehl `\thispagestyle{plain}`. Deshalb erhält man weiterhin Seitenzahlen auf

*$\LaTeX$ -Standardlayouts  
für Kolumnentitel*

*Alle Seitenzahlen  
unterdrücken*

	Befehl	Dokumentenklasse	
		book, report	article
Zweiseitiger Druck	<code>\markboth<sup>a</sup></code>	<code>\chapter</code>	<code>\section</code>
	<code>\markright</code>	<code>\section</code>	<code>\subsection</code>
Einseitiger Druck	<code>\markright</code>	<code>\chapter</code>	<code>\section</code>

<sup>a</sup>Legt eine leere rechte Marke an (siehe Abbildung 4.2 auf Seite 227).

Tabelle 4.3: Layout-Befehle für Kolumnentitel in  $\LaTeX$

den Seiten, die durch Befehle wie `\chapter` oder `\maketitle` gebildet werden, auch wenn am Anfang des Dokumentes das Layout `\pagestyle{empty}` eingestellt ist. Um die Seitenzahlen überall zu unterdrücken, muss entweder auf jeden dieser Befehle die Anweisung `\thispagestyle{empty}` folgen, oder das Layout `plain` muss in einem eigenen Anpassungspaket des Anwenders durch den Befehl `\let\ps@plain=\ps@empty` auf das Layout `empty` umdefiniert werden.

Im Layout `headings` erzeugen die Gliederungsbefehle die Kolumnentitel automatisch mithilfe von `\markboth` und `\markright`, wie in Tabelle 4.3 angezeigt.

Das Standardlayout `myheadings` ähnelt `headings`, wobei der Anwender hier die Informationen für die Kopfzeile mithilfe der oben beschriebenen Befehle `\markboth` und `\markright` selbst angibt. Außerdem lassen sich damit auch Überschriften von anderen Elementen wie Inhalts- und Abbildungs- oder Stichwortverzeichnissen übernehmen. Diese Befehle (`\tableofcontents`, `\listoffigures` und `\listoftables`) und Umgebungen (`thebibliography` und `theindex`) verwenden eigentlich den `\chapter*`-Befehl, der nicht `\chaptermark` aufruft, sondern stattdessen den Befehl `\@mkboth` verwendet. Das Layout `headings` definiert `\@mkboth` als `\markboth`, im Layout `myheadings` wird `\@mkboth` jedoch nicht aktiviert, so dass die Entscheidung dem Anwender überlassen bleibt.

#### 4.4.1 Die Low-Level-Schnittstelle für Kolumnentitel-Layouts

Der  $\LaTeX$ -Kern verwendet vier interne Befehle als Schnittstelle für das Kolumnentitel-Layout. Zwei dieser Befehle werden auf jeder Seite dazu verwendet, die lebenden Kolumnentitel zu formatieren. Durch Umdefinieren dieser Befehle kann man verschiedene Aktionen anstoßen.

`\@oddhead` erzeugt bei zweiseitigem Druck den oberen Kolumnentitel für ungerade Seiten, ansonsten für alle Seiten.

`\@oddfoot` erzeugt bei zweiseitigem Druck den unteren Kolumnentitel für ungerade Seiten, ansonsten für alle Seiten.

`\@evenhead` erzeugt bei zweiseitigem Druck den oberen Kolumnentitel für gerade Seiten und wird bei einseitigem Druck ignoriert.

`\@evenfoot` erzeugt bei zweiseitigem Druck den unteren Kolumnentitel für gerade Seiten und wird bei einseitigem Druck ignoriert.



Ein benanntes Layout besteht lediglich aus geeigneten Umdefinitionen für diese Befehle, die in einem Makro namens `\ps@{stil}` gespeichert werden; man muss also dieses Makro umdefinieren, um das Verhalten eines Layouts *stil* zu verändern. Beispielsweise sieht die Definition für das Layout `plain`, das lediglich eine zentrierte Seitenzahl am Fuß der Seite ausgibt, im  $\text{\LaTeX}$ -Kern ungefähr folgendermaßen aus:

```
\newcommand\ps@plain{%
  \renewcommand\@oddhead{}%           % leerer Recto-Kopf
  \let\@evenhead\@oddhead           % leerer Verso-Kopf
  \renewcommand\@evenfoot{\hfil
    \normalfont \textrm{\thepage}\hfil}% % zentrierte
  \let\@oddfoot\@evenfoot           %           Seitenzahl
}
```

#### 4.4.2 fancyhdr – Anpassen von Kolumnentitel-Layouts

Da sich das Layout der Kolumnentitel in Standard- $\text{\LaTeX}$  nur über interne Befehle ändern lässt, ist es nicht überraschend, dass eine Reihe von Paketen für besondere Layouts entwickelt wurden. So ändert z.B. das Paket `rplain` das Layout `plain` dahingehend, dass die Seitenzahl rechts und nicht zentriert erscheint. Es gibt auch komplexere Pakete. Beispielsweise lohnt es sich, die entsprechenden Deklarationsmöglichkeiten des Paketes `titlesec` (zur Definition von Gliederungsbefehlen, siehe Abschnitt 2.2.6) näher zu erkunden.

Ein gut eingeführtes, eigenständiges Paket in diesem Bereich ist `fancyhdr`<sup>1</sup> von Piet van Oostrum, mit dessen Hilfe sich Kolumnentitel auf einfache Weise anpassen lassen. Das Standardlayout von `fancyhdr` heißt `fancy`. Es sollte mit `\pagestyle` aktiviert werden – gegebenenfalls *nach* dem Ändern bzw. Einstellen von `\textwidth`, da `fancyhdr` die Breite der Kolumnentitel in Abhängigkeit vom aktuellen Wert dieser Länge initialisiert.

*Basisschnittstelle*

Das Erscheinungsbild des Layouts `fancy` wird durch sechs Deklarationen bestimmt, die festlegen, welches Material in den Kolumnentiteln links, zentriert und rechts erscheint. Der Befehl `\lhead` legt z.B. fest, was links im oberen Kolumnentitel erscheint, während `\cfoot` die Mitte des unteren Kolumnentitels definiert. Im nächsten Beispiel werden die Ergebnisse aller sechs Deklarationen dargestellt.

LINKS	MITTE	RECHTS	
Text für unsere Seite, der immer und immer wieder verwendet wird.	Text für unsere Seite, der immer und immer wieder verwendet wird.	links-ganz-ganz-ganz-ganz-ganz-lang	\usepackage{fancyhdr} \pagestyle{fancy}
			\lhead{LINKS} \chead{MITTE} \rhead{RECHTS}
			\lfoot{links-ganz-ganz-ganz-ganz-lang} \cfoot{}
			\rfoot{rechts-sehr-lang}
			\renewcommand\headrulewidth{2pt}
			\renewcommand\footrulewidth{0.4pt}
			\newcommand\sample{ Text für unsere Seite,
			der immer und immer wieder verwendet wird.}
			\sample \par \sample

Bsp.  
4-4-1

<sup>1</sup>In diesem Buch wird Version 2.0 von `fancyhdr` beschrieben. Frühere Versionen waren unter dem Namen `fancyheadings` bekannt.





*Genauere Steuerung*

Mithilfe der bisher beschriebenen Deklarationen lässt sich das Layout der Kolumnentitel nicht in Abhängigkeit von der Art der aktuellen Seite ändern. Dies wird jedoch mit den allgemeineren Deklarationen `\fancyhead` und `\fancyfoot` möglich. Sie besitzen ein zusätzliches optionales Argument, mit dem man festlegen kann, für welchen Seitentyp und für welchen Bereich der Kolumnentitel die Deklaration angewendet werden soll. Der Seitentyp wird mit O oder E für ungerade bzw. gerade Seiten ausgewählt; der Bereich mit L, C oder R. Wenn kein Seitentyp oder Bereich ausgewählt ist, gilt die Deklaration für alle Seitentypen oder Bereiche. Das Kürzel LO steht also für den linken Bereich auf ungeraden Seiten, während C für den mittleren Bereich auf allen Seiten steht. Anders ausgedrückt waren die bisher besprochenen Deklarationen Kurzformen für die hier beschriebenen.

Wie das nächste Beispiel zeigt, können die Kürzel sogar aneinandergereiht werden. So bedeutet etwa RO,LE, dass diese Deklaration für den rechten Bereich auf ungeraden und den linken Bereich auf geraden Seiten gilt.

6	Memo
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Autor: Frank	

Memo	7
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Autor: Frank	

```
\usepackage{fancyhdr}\pagestyle{fancy}
\fancyhead{} % Kopfzeilen leeren
\fancyhead[RO,LE]{\thepage}
\fancyhead[LO,RE]{Memo}
\fancyfoot{} % Fußzeilen leeren
\fancyfoot[L]{Autor: Frank}
\renewcommand\headrulewidth{0.4pt}
\renewcommand\footrulewidth{0.4pt}

% \sample definiert wie zuvor
\sample \par \sample
```

Bsp.  
4-4-4

Tatsächlich sind `\fancyhead` und `\fancyfoot` nur von einer noch allgemeineren Deklaration, nämlich `\fancyhf`, abgeleitet. Die Syntax ist ähnlich, wobei es eine weitere Art von Kürzeln gibt. Das optionale Argument kann zusätzlich H oder F für Kopf- oder Fußzeilen enthalten. Damit ist `\fancyfoot[LE]` gleichbedeutend mit `\fancyhf[FLE]`, wobei die zweite Form wohl schlechter zu lesen ist, weshalb sie in diesem Buch normalerweise nicht verwendet wird. Allerdings ist die Deklaration `\fancyhf` von Vorteil, wenn man alle Bereiche gleichzeitig leeren möchte.

Das nächste Beispiel zeigt eine Anwendung des `lastpage`-Paketes: Im unteren Kolumnentitel wird die aktuelle Seitenzahl und die Gesamtzahl aller Seiten angezeigt.

1	EIN TEST
<b>1 Ein Test</b>	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Seite 6 von 7	

1	EIN TEST
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Seite 7 von 7	

```
\usepackage{fancyhdr,lastpage}
\pagestyle{fancy}
\fancyhf{} % -- alle Bereiche leeren
\fancyhead[RO,LE]{\leftmark}
\fancyfoot[C]{Seite \thepage\
von \pageref{LastPage}}

% \sample definiert wie zuvor
\section{Ein Test}
\sample \par \sample
```

Bsp.  
4-4-5

Die Kolumnentitel werden in Boxen gesetzt, die auf eine Breite von `\textwidth` voreingestellt sind. Ihre Breite kann soweit erforderlich mithilfe



Die Voreinstellungen  
von fancyhdr

Die bisherigen Beispiele ließen auf die ein oder andere Voreinstellung von fancyhdr schließen. Im nächsten Beispiel werden alle verwendet. Um sie besser erkennen zu können, sind sie im Programmcode des Beispiels in Kommentaren aufgeführt. Die Standardeinstellungen setzen eine schmale Linie unter der Kopfzeile und keine Linie über der Fußzeile; die Seitenzahl steht im Fuß und ist zentriert und der Kopf zeigt sowohl \leftmark als auch \rightmark an, wobei ihre Reihenfolge durch den Seitentyp festgelegt wird.

1 TEST
<h1>1 Test</h1> <h2>1.1 B-1</h2> <p>Text für unsere Seite, der immer und immer wieder verwendet wird.</p>
6

1 TEST	1.2 B-2
<h2>1.2 B-2</h2> <p>Text für unsere Seite, der immer und immer wieder verwendet wird.</p>	
7	

```
\usepackage{fancyhdr}
\pagestyle{fancy}
%\fancyhead[LE,R0]
%      {\slshape\rightmark}
%\fancyhead[LO,RE]
%      {\slshape\leftmark}
%\fancyfoot[C]{\thepage}
%\renewcommand\headrulewidth{0.4pt}
%\renewcommand\footrulewidth{0pt}
% \sample definiert wie zuvor
\section{Test}
\subsection{B-1} \sample
\subsection{B-2} \sample
```

Bsp.  
4-4-7

Der Abstand zwischen Zahl und Text im oberen Kolumnentitel ist sicher zu groß. Dies liegt an dem extrem schmalen Satzspiegel im Beispiel, daher soll dieses Problem zunächst vernachlässigt werden. Wie nützlich sind, davon einmal abgesehen, diese Voreinstellungen? Wie bereits erwähnt, wurden die L<sup>A</sup>T<sub>E</sub>X-Befehle \leftmark und \rightmark in erster Linie mit Blick auf „Abschnitte in Kapiteln“ entwickelt – also für den Fall, dass \leftmark immer mit einer Überschrift verknüpft ist, die eine neue Seite einleitet. Andernfalls kann man, wie im nächsten Beispiel, recht seltsame Kolumnentitel erhalten.

Ein Abschnitt wurde auf Seite 5 platziert (die Seite wird nicht angezeigt) und erstreckt sich bis auf Seite 6. Dadurch erscheint im oberen Kolumnentitel von Seite 6 der Unterabschnitt 1.1 zusammen mit Abschnitt 2, ähnlich wie auf Seite 7, wo man Abschnitt 3 zusammen mit Unterabschnitt 2.1 erhält.

1.1 B-1	2 A-2
<h2>1.1 B-1</h2> <p>Text für die Seite, der wiederverwendet wird.</p> <h2>2 A-2</h2> <p>Text für die Seite, der wiederverwendet wird.</p>	
6	

3 A-3	2.1 B-2
<h2>2.1 B-2</h2> <p>Text für die Seite, der wiederverwendet wird.</p> <h2>3 A-3</h2> <p>Text für die Seite, der wiederverwendet wird.</p>	
7	

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\newcommand\sample{ Text für die
Seite, der wiederverwendet wird.}

\setcounter{page}{5}
\section{A-1} \newpage
% Code oben setzt einen Abschnitt
% auf Seite 5 (nicht angezeigt)
\subsection{B-1} \sample
\section{A-2} \sample
\subsection{B-2} \sample
\section{A-3} \sample
```

Bsp.  
4-4-8

Um dieses Verhalten nachvollziehen zu können, muss man sich noch einmal vor Augen führen, dass sich `\leftmark` auf die letzte von `\markboth` erzeugte Marke auf dieser Seite bezieht, und `\rightmark` auf die erste von `\markright` oder `\markboth` erzeugte.

Wer häufig Seiten wie die oben dargestellten erzeugt, z.B. in einem Dokument mit vielen Unterabschnitten, dem ist mit den Standardeinstellungen des `fancyhdr`-Paketes sehr wahrscheinlich nicht gedient. Stattdessen sollte man sie auf die ein oder andere Weise überschreiben, wie in den meisten Beispielen dieses Abschnitts geschehen. Man muss sich folgende Frage stellen: Welche Informationen möchte ich dem Leser in einem solchen Kolumnentitel geben? Will man beispielsweise bei geraden (linken) Seiten die Situation zu Beginn der Seite und bei ungeraden Seiten den Status am Ende der Seite darstellen, dann bietet sich der Einsatz von `\firstleftmark` und `\lastrightmark` des `extramarks`-Paketes als mögliche Lösung an.

<i>1.1</i>	<i>B-1</i>	1	<i>A-1</i>
<hr/>			
<b>1.1</b>	<b>B-1</b>	Text für die Seite, der wiederverwendet wird.	
<b>2</b>	<b>A-2</b>	Text für die Seite, der wiederverwendet wird.	
6			

3	<i>A-3</i>
<hr/>	
<b>2.1</b>	<b>B-2</b>
Text für die Seite, der wiederverwendet wird.	
<b>3</b>	<b>A-3</b>
Text für die Seite, der wiederverwendet wird.	
7	

```
\usepackage{extramarks}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[RO]{\lastrightmark}
\fancyhead[RE]{\firstleftmark}
% \sample definiert wie zuvor
\setcounter{page}{5}
\section{A-1} \newpage
% Code oben setzt einen Abschnitt
% auf Seite 5 (nicht angezeigt)
\subsection{B-1} \sample
\section{A-2} \sample
\subsection{B-2} \sample
\section{A-3} \sample
```

Bsp.  
4-4-9

Wer prüfen will, ob der Mechanismus nun klar ist, sollte erklären, warum Seite 7 jetzt nur den Titel „A-3“ anzeigt und versuchen sich vorzustellen, was passieren würde, wenn die Überschrift „B-1“ (nicht aber der Text des ganzen Abschnitts) schon auf Seite 5 aufgetaucht wäre.

Trotz der zuvor aufgestellten Behauptung, alle Voreinstellungen gezeigt zu haben, nimmt das `fancy`-Layout doch noch zwei weitere Voreinstellungen vor. Da sie sozusagen im Verborgenen wirken, wurden sie bisher übergangen. Es wurde noch nicht erklärt, wie `\leftmark` und `\rightmark` ihre Werte erhalten; aus den vorherigen Beispielen sollte ersichtlich sein, dass sie Daten erhalten. Wie in Abschnitt 4.3.4 erläutert, geben die Gliederungsbefehle ihr Titelargument an Befehle wie `\sectionmark` weiter. Diese werden über `\markboth` oder `\markright` veranlasst, Marken oder eben keine Marken zu erzeugen. Das Kolumnentitel-Layout `fancy` definiert nun zwei derartige Befehle: `\chaptermark` und `\sectionmark`, wenn die aktuelle Klasse einen `\chapter`-Befehl definiert, oder andernfalls `\sectionmark` und `\subsectionmark`. Wenn man also einen anderen Textmarkenmechanismus verwenden möchte, oder selbst nur ein etwas abgewandeltes Kolumnentitel-Layout (z.B. ohne Gliederungsnummern in der Kopfzeile oder ohne `\MakeUppercase` für den Markentext), dann muss man diese Befehle unter Umständen selbst definieren.

Das nächste Beispiel ist eine Kopie von Beispiel 4-4-7 auf Seite 236, nur dass dieses Mal selbst definierte `\sectionmark`- und `\subsectionmark`-Befehle zum Einsatz kommen, die den Abstand zwischen der Zahl und dem Text verringern und auf den Befehl `\MakeUppercase` verzichten.

<i>1 Test</i>
<hr/>
<b>1 Test</b>
<b>1.1 Titel B</b>
Text für unsere Seite, der immer und immer wieder verwendet wird.
6

<i>1 Test</i>	<i>1.2 Titel B2</i>
<hr/>	<hr/>
<b>1.2 Titel B2</b>	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
7	

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand\sectionmark[1]
{\markboth{\thesection\ #1}{}}
\renewcommand\subsectionmark[1]
{\markright{\thesubsection\ #1}}
% \sample wie zuvor definiert
\section{Test}
\subsection{Titel B} \sample
\subsection{Titel B2}\sample
```

Bsp.  
4-4-10

„Benannte“ Layouts  
definieren

Bisher wurde das Layout fancy in allen Beispielen immer aufs Neue angepasst. Mit dem Paket fancyhdr kann man jedoch auch eigene Anpassungen unter einem Namen speichern und diese dann über die Befehle `\pagestyle` oder `\thispagestyle` auswählen. Das geschieht mithilfe der Deklaration `\fancypagestyle`. Sie verfügt über zwei Argumente: den Namen des Kolumnentitel-Layouts und die Anpassungen, die vorgenommen werden sollen, wenn das Layout später aufgerufen wird. Nicht definierte (oder gelöschte) Felder, sowie die Einstellungen für Linienstärken werden aus den fancyhdr-Standards übernommen. Das erklärt, warum zunächst alle Felder mit `\fancyhf` gelöscht werden.

6	Memo
<hr/>	<hr/>
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Text für unsere Seite,	
28. Juli 2005	

Memo	7
<hr/>	<hr/>
te, der immer und immer wieder verwendet wird.	
Text für unsere Seite, der immer und immer wieder	
28. Juli 2005	

```
\usepackage{fancyhdr}
\fancypagestyle{memo}{\fancyhf{}}%
\fancyhead[RO,LE]{\thepage}%
\fancyhead[LO,RE]{Memo}%
\fancyfoot[R]{\scriptsize\today}%
\renewcommand\headrulewidth{1pt}}
\pagestyle{memo}
% \sample definiert wie zuvor
\sample \par \sample\sample
```

Bsp.  
4-4-11

Einige L<sup>A</sup>T<sub>E</sub>X-Befehle, wie `\chapter` und `\maketitle`, wechseln mithilfe von `\thispagestyle` automatisch auf das Layout plain, und ignorieren damit das aktuell gültige Layout. Um die Kolumnentitel für solche Seiten anzupassen, kann man entweder diese Befehle entsprechend umdefinieren (was ziemlichen Ärger machen kann) oder das plain-Layout über `\fancypagestyle` abändern. Es ist genaugenommen nicht wirklich der richtige Ansatz, ein Standardlayout wie plain zu verändern. Der eigentliche Fehler liegt aber bei den L<sup>A</sup>T<sub>E</sub>X-Standardklassen selbst,<sup>1</sup> die es versäumen, für solche Fälle speziell benannte Layouts zu verwenden, und stattdessen einfach den

<sup>1</sup>Die KOMA-Script-Klassen verwenden für solche speziellen Kolumnentitel-Layouts beispielsweise Befehle wie `\chapterpagestyle`, die sich leicht anpassen lassen.

wahrscheinlichsten Kandidaten einsetzen. In der Praxis funktioniert die von fancyhdr bereitgestellte Lösung aber recht gut.

Manchmal ist es wünschenswert, das Layout in Abhängigkeit von Gleitobjekten auf der aktuellen Seite zu verändern. Für diesen Zweck sieht das Paket fancyhdr eine Reihe von Steuerbefehlen vor. Sie können in den Layoutdeklarationen eingesetzt werden, wodurch das Layout auf die An- oder Abwesenheit von Fußnoten (`\iffootnote`) sowie Gleitobjekten im oberen (`\iftopfloat`) oder unteren Bereich (`\ifbottomfloat`) auf der aktuellen Seite reagieren kann. Jeder von ihnen besitzt zwei Argumente: Das erste wird aktiv, wenn die Bedingung eintritt, das zweite, wenn nicht.

Im nächsten Beispiel fällt durch Umdefinieren von `\headrulewidth` die Linie unter der Kopfzeile weg, wenn im oberen Bereich Gleitobjekte erscheinen. Außerdem enthält der obere Kolumnentitel unterschiedliche Texte, je nachdem ob oben auf der Seite Gleitobjekte erscheinen oder nicht.

*Von Gleitobjekten abhängige Kolumnentitel-Layouts*

<p><b>SPEZIAL</b></p> <div style="border: 1px solid black; padding: 5px; margin: 10px auto; width: 80px; text-align: center;">Abbildung oben</div> <p>Text für unsere Seite, der immer und immer wieder verwendet wird.</p> <p style="text-align: center;">6</p>	<p><b>NORMAL</b></p> <hr style="border: 0.4pt solid black;"/> <p>Text für unsere Seite, der immer und immer wieder verwendet wird.</p> <p style="text-align: center;">7</p>
--	---

```

\usepackage{fancyhdr}
\pagestyle{fancy} \fancyhf{}
\thead{\iftopfloat{SPEZIAL}{NORMAL}}
\cfoot{\thepage}
\renewcommand\headrulewidth
  {\iftopfloat{0pt}{0.4pt}}

% \sample definiert wie zuvor
\sample
\begin{figure}[t]
  \centering
  \fbox{Abbildung oben}
\end{figure}
\sample

```

Bsp.  
4-4-12

Zum Anpassen des Layouts für reine Gleitobjektseiten gibt es einen ähnlichen Befehl, `\iffloatpage` - um z.B. die oberen Kolumnentitel auf solchen Seiten zu unterdrücken. Wenn das Layout von mehreren Bedingungen abhängig sein soll, können die Steuerbefehle verschachtelt werden, auch wenn das mit der Zeit etwas unübersichtlich wird. Wenn z.B. die Linie unter der Kopfzeile auf allen Seiten, die nur Gleitobjekte oder die Gleitobjekte im oberen Bereich enthalten, ausgeblendet werden soll, müsste man `\headrulewidth` folgendermaßen definieren:

*Layout für Gleitobjektseiten*

```

\renewcommand\headrulewidth
  {\iftopfloat{0pt}{\iffloatpage{0pt}{0.4pt}}}

```

In Wörterbüchern und ähnlichen Werken enthält der lebende Kolumnentitel häufig das erste und das letzte Wort, die auf einer Seite erklärt werden, um das Auffinden der Informationen zu erleichtern. Dieses Schema lässt sich leicht verwirklichen, indem man einen geeigneten Befehl definiert, der für jeden Wörterbucheintrag eine Marke setzt. Im nächsten Beispiel verwenden wir L<sup>A</sup>T<sub>E</sub>Xs *right-mark* um solche Marken zu speichern und greifen mit `\firstrightmark` und `\lastrightmark` aus dem Paket `extramarks` darauf

*Kolumnentitel für Wörterbücher*



zu. Auf Seiten, die nur einen einzigen Eintrag behandeln, erfolgt nur ein Eintrag im Kolumnentitel. Dazu wird mithilfe des `ifthen`-Paketes geprüft, ob beide Befehle den gleichen Wert enthalten. Die lebenden Kolumnentitel für den Index dieses Buches wurden mit einem ähnlichen Mechanismus erzeugt.

Fahne—Marke
<b>Fahne</b> Formatierter Text, nicht in Seiten aufgeteilt.
<b>Kolumne</b> Textspalte.
<b>Marke</b> Steht in der
6

OR
Fahne, um mit der OR zu kommunizieren.
<b>OR</b> Output-Routine.
7

```
\usepackage{ifthen,fancyhdr,extramarks}
\pagestyle{fancy} \fancyhf{}
\newcommand\combinemarks{\ifthenelse
  {\equal{\firstrightmark}%
    {\lastrightmark}}%
  {\firstrightmark} gleiche Werte
  {\firstrightmark---\lastrightmark}}
\thead{\combinemarks} \cfoot{\thepage}
\newcommand\idxitem[1]{\par\vspace{8pt}%
  \textbf{#1}\markright{#1}%
  \quad\ignorespaces}
\idxitem{Fahne} Formatierter Text,
  nicht in Seiten aufgeteilt.
\idxitem{Kolumne} Textspalte.
\idxitem{Marke} Steht in der Fahne,
  um mit der OR zu kommunizieren.
\idxitem{OR} Output-Routine.
```

Bsp.  
4-4-13

#### Probleme beim Zweispaltensatz

Wörterbücher werden oft mit zwei oder mehr Spalten pro Seite gesetzt. Unglücklicherweise behandelt der  $\LaTeX$ -Standardmodus `twocolumn` Marken fehlerhaft - `\leftmark` bezieht sich stets auf die zweite Spalte, anstatt die erste Marke der ersten Spalte wiederzugeben. Wenn das ein Problem darstellt, sollte man die neue Version aus dem Paket `fixltx2e` verwenden. Oder man setzt das Paket `multicol` ein, das die Marken ebenfalls korrekt verarbeitet.

### 4.4.3 truncate - Texte auf eine bestimmte Länge kürzen

Der begrenzte verfügbare Platz ist immer ein potentielles Problem für lebende Kolumnentitel: Ist der Text zu lang, dann überschreibt er unter Umständen die Seitennummer oder anderes Material. Für solche Fälle eignet sich das Paket `truncate` von Donald Arseneau. Es stellt einen Befehl bereit, mit dem sich ein vorgegebener Text auf eine bestimmte Länge kürzen lässt.

```
\truncate[zeichen]{breite}{text}
```

Wenn das Argument *text* für die angegebene *breite* zu groß ist, wird es abgeschnitten und ein oder mehrere Fortführungszeichen an sein Ende gesetzt. Wenn das optionale Argument *zeichen* fehlt, werden in `\TruncateMarker` gespeicherte Standardzeichen eingesetzt (das Paket verwendet den Wert `\, \dots`).

Normalerweise werden Kürzungen nur an Wortgrenzen vorgenommen, und nur da, wo Wörter nicht mit „~“, d.h. durch einen geschützten Leerraum verbunden sind. Aus diesem Grunde wird der Text im folgenden Beispiel nach dem Wort *hier* abgeschnitten. Das Beispiel veranschaulicht auch den



Gebrauch eines *zeichen*-Argumentes, das extra in geschweifte Klammern gesetzt werden muss, um die eckigen Klammern, die als Teil des Textes erscheinen sollen, zu verstecken. Damit man sich den Platz, der von dem gekürzten Text eingenommen werden darf, besser vorstellen kann, wurden links und rechts |-Zeichen angefügt.

Bsp.  
4-4-14

```

\usepackage{truncate}
|Der Text hier wird~gekapt|

|\truncate{50pt}
{Der Text hier wird~gekapt}|

|\truncate[{\, [. . .]}]{100pt}
|Der Text hier [...] |
{Der Text hier wird~gekapt}|

```

Man kann den Text auch innerhalb eines Wortes abschneiden, wenn man eine der Paketooptionen *hyphenate*, *breakwords* oder *breakall* angibt. Die ersten beiden kappen Wörter entsprechend ihrer Silbentrennung, mit dem Unterschied, dass *breakwords* das Trennzeichen unterdrückt (die gängigere Variante). Mit der dritten Option kann man den Text an jeder beliebigen Stelle abschneiden. Mithilfe dieser Optionen würde das vorige Beispiel folgendes Ergebnis liefern:

Der Text hier wird ge-[]	(hyphenate)
Der Text hier wird ge[]	(breakwords)
Der Text hier wird gekap[]	(breakall)

Der Text (ob gekürzt oder nicht) wird entsprechend der Voreinstellungen linksbündig in einer Box mit der angegebenen *breite* gesetzt. Die Paketooption *fit* sorgt dafür, dass der formatierte Text seine natürliche Breite bis zum Maximum der vorgegebenen *breite* behält.

Im nächsten Beispiel wird das Paket *truncate* mit *fancyhdr* kombiniert. Man beachte den Einsatz der Option *fit*. Ohne diese Option wäre die Kopfzeile immer linksbündig (um den Effekt deutlicher zu machen, wurde `\headwidth` etwas verkleinert).

Bsp.  
4-4-15

1 ABSCHNITT ...	1 ABSCHNITT ...	
<b>1 Abschnitt mit langem Titel</b>	wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.	<pre> \usepackage[fit]{truncate} \usepackage{fancyhdr} \pagestyle{fancy} \fancyhf{} % -- alle Bereiche leeren \fancyhead[R0,LE]{\truncate {.95\headwidth}{\leftmark}} \fancyfoot[C]{\thepage} % \sample definiert wie zuvor \section{Abschnitt mit langem Titel} \sample \par \sample </pre>
Text für unsere Seite, der immer und immer		
6	7	

## 4.5 Visuelle Formatierung

In der letzten Bearbeitungsphase für ein wichtiges Dokument ist es oft notwendig, einige manuelle Formatierungen vorzunehmen, um ungünstige Seitenumbrüche zu vermeiden. Für diesen Zweck stellt L<sup>A</sup>T<sub>E</sub>X die Befehle `\pagebreak`, `\nopagebreak`, `\newpage` und `\clearpage` sowie die Deklaration `\samepage` zur Verfügung, wobei letztere in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> als veraltet angesehen wird. Durch die Deklaration `\samepage` in Verbindung mit einer geeigneten Anzahl von `\nobreak`-Befehlen kann man erreichen, dass bestimmte Teile des Dokumentes zusammengehalten werden. Unglücklicherweise sind die Ergebnisse oft nicht zufriedenstellend; so verlängert L<sup>A</sup>T<sub>E</sub>X z.B. niemals eine Seite über ihre nominale Höhe (`\textheight`) hinaus, sondern bewegt eher alles, was sich im Gültigkeitsbereich der `\samepage`-Deklaration befindet, auf die nächste Seite. Die Alternative ist der L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>-Befehl `\enlargethispage*`, der im Folgenden beschrieben wird.

Bei der Buchproduktion ist es üblich, eine gewisse Anzahl von Seiten (normalerweise Doppelseiten) zu verkürzen oder zu verlängern, um dadurch spätere ungünstige Seitenumbrüche zu vermeiden. Das bedeutet, dass die nominale Höhe dieser Seiten um ein bestimmtes Maß, zum Beispiel eine Zeile (`\baselineskip`), vergrößert oder verkleinert wird. L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> unterstützt diese Praxis durch den Befehl `\enlargethispage{größe}`.

```
\enlargethispage{größe}
```

Wenn man also die Länge einiger Seiten um eine Zeile verlängern oder verkürzen möchte, kann man folgende Definition verwenden:

```
\newcommand\longpage[1][1]{\enlargethispage{#1\baselineskip}}
\newcommand\shortpage[1][1]{\enlargethispage{-#1\baselineskip}}
```

Diese Befehle müssen zwischen zwei Absätzen auf den betreffenden Seiten eingefügt werden.<sup>1</sup> Der Befehl `\enlargethispage` vergrößert die Höhe `\textheight` der aktuellen Seite, ändert ansonsten jedoch keinen der Formatierungsparameter. Wenn also `\flushbottom` eingestellt ist, füllt der Text die gesamte Länge der fraglichen Seite aus, falls erforderlich auch durch Dehnen oder Stauchen der vertikalen Leerräume auf der Seite. Dadurch setzen die obigen Definitionen genau eine Zeile Text mehr bzw. weniger auf eine Seite, während die Position der übrigen Zeilen unverändert bleibt. Das ist wichtig, denn auf diese Weise bleibt das einheitliche Erscheinungsbild gewahrt.

```
\enlargethispage*{größe}
```

Die Sternform des Befehls, `\enlargethispage*`, vergrößert oder verkleinert ebenfalls die Seitenlänge. Dabei wird jedoch die resultierende Seite so weit wie möglich gestaucht (d.h. soweit es der verfügbare Weißraum auf der Seite

<sup>1</sup>Aufgrund der vielen Beispiele in diesem Buch war dieser Trick einige Male notwendig, um halbleere Seiten zu vermeiden. So wurden z.B. vier Seiten dieses Kapitels ab Seite 228 um eine Zeile verlängert oder die vorherige Doppelseite um eine Zeile gekürzt. Das erwies sich aufgrund der vielen (umfangreichen) Beispiele in Abschnitt 4.4.2 als erforderlich – alle anderen Formatierungen führten irgendwo zu einer halbleeren Seite.

zulässt). Dieses Verfahren kann hilfreich sein, wenn man einen bestimmten Teil seines Dokumentes auf einer Seite zusammenhalten möchte, auch wenn die Seite dadurch etwas zu lang wird. (Ansonsten kann man einfach die Umgebung `minipage` verwenden.) Der Trick dabei ist, im Voraus ausreichend zusätzlichen Platz anzufordern und den Seitenumbruch dann dort, wo er erfolgen soll, zu erzwingen. Zum Beispiel:

```
\enlargethispage*{100cm}      % absurde Forderung
\begin{center}
  \begin{tabular}{l1111}      % etwas zu lange
    ....                      % Tabelle
  \end{tabular}
\end{center}
\pagebreak                    % erzwungener Seitenumbruch
```

Aus der obigen Beschreibung ist ersichtlich, dass beide Befehle nur in der letzten Bearbeitungsphase verwendet werden sollten, da jegliche spätere Änderung im Text (im unglücklichsten Fall selbst das Hinzufügen oder Löschen eines einzigen Wortes) die manuelle Formatierung ad absurdum führen und damit sehr hässliche Seiten erzeugen kann.

Es kann hilfreich sein, sich die Gründe vor Augen zu führen, warum  $\TeX$  Seiten an einer bestimmten Stelle umbricht, und wie viel Spielraum für bestimmte Seiten verfügbar ist, wenn man endgültige Seitenumbrüche manuell korrigiert. Dies gilt z.B. für Veröffentlichungen wie dieses Buch (das durch seine vielen Beispiele, in denen kein Seitenumbruch erfolgen darf, recht anspruchsvoll ist). Eine Beschreibung der Werkzeuge für diesen Zweck findet sich in Anhang B.3.2.

### 4.5.1 `nextpage` – Erweiterungen für `\clearpage`

Standard- $\LaTeX$  beendet den aktuellen Absatz und die laufende Seite mithilfe der Befehle `\clearpage` und `\cleardoublepage`, nachdem alle aufgelaufenen Gleitobjekte gesetzt wurden. (Falls erforderlich wird dazu eine Reihe von Gleitobjektseiten erzeugt.) Bei zweiseitigem Druck sorgt `\cleardoublepage` außerdem dafür, dass die nächste Seite immer eine rechte (ungerade) Seite ist, auch wenn dazu eine Seite ohne Text eingefügt werden muss. Diese zusätzliche Seite enthält immer noch die Kolummentitel des momentan gültigen Layouts, was nicht immer wünschenswert ist.

1
<p><b>1 Ein Test</b></p> <p><b>1.1 Ein Abschnitt</b></p> <p>Etwas Text für die Seite.</p>

2	1 <i>EIN TEST</i>
---	-------------------

```
\pagestyle{headings}
% rechte Seite, in diesem Beispiel
% links, aufgrund von:
\setcounter{page}{1}

\section{Ein Test}
\subsection{Ein Abschnitt}
Etwas Text für die Seite.
\cleardoublepage
\section{Noch ein Abschnitt}
Dieser gehört auf Seite 3.
```

Bsp.  
4-5-1

Das Paket `nextpage` von Peter Wilson erweitert diesen Ansatz durch die Befehle `\cleartoevenpage` und `\cleartooddpage`. Beide verfügen über ein optionales Argument, das Text aufnehmen kann, welcher auf einer gegebenenfalls erzeugten Seite stehen soll. Im nächsten Beispiel wird mit ihrer Hilfe der Befehl `\myclearpage` definiert, der LEERE SEITE auf solche generierten Seiten schreibt.

1	2
<p><b>1 Ein Test</b></p> <p><b>1.1 Ein Abschnitt</b></p> <p>Etwas Text für die Seite.</p>	<p><i>1 EIN TEST</i></p> <p>LEERE SEITE</p>

```
\usepackage{nextpage}
\pagestyle{headings}
\newcommand\myclearpage{%
  \cleartooddpage
  [\vspace*{\fill} \centering
  LEERE SEITE \vspace*{\fill}]}
\setcounter{page}{1} %rechte Seite
\section{Ein Test}
\subsection{Ein Abschnitt}
Etwas Text für die Seite.
\myclearpage
\section{Noch ein Abschnitt}
Dieser gehört auf Seite 3.
```

Bsp.  
4-5-2

Mit dieser Befehlsfolge erscheint immer noch ein lebender Kolumnentitel, aber mittlerweile ist ja bekannt, wie sich das beheben lässt: Man muss der Definition lediglich `\thispagestyle{empty}` hinzufügen.

Das Paket `nextpage` stellt außerdem ähnliche Funktionen mit den Befehlen `\movetoevenpage` und `\movetooddpage` zur Verfügung - nur werden hier aufgelaufene Gleitobjekte nicht ausgegeben.

## 4.6 Layouts mit Klasse

Das Seitenlayout wird normalerweise durch die Dokumentenklasse festgelegt, so dass es nicht überraschen sollte, dass die Verfahren und Pakete, die in diesem Kapitel beschrieben werden, normalerweise im Verborgenen zur Anwendung kommen (innerhalb einer Dokumentenklasse).

Die Standardklassen bedienen sich unmittelbar der  $\LaTeX$ -Parameter und -Schnittstellen, um Seitenverhältnisse, lebende Kolumnentitel und andere Elemente zu definieren. Neuere Klassen behandeln jedoch auch häufig bestimmte Aspekte des Seitenlayouts mithilfe von Paketen wie `geometry`.

In diesem Abschnitt werden zwei derartige Lösungen vorgestellt. Das CTAN-Archiv enthält möglicherweise noch mehr dieser Schätze.

### 4.6.1 KOMA-Script - Ein Ersatz für `article` et al.

Die KOMA-Script-Klassen von Markus Kohm, auf der Basis früherer Entwicklungen von Frank Neukam, sind ein passender Ersatz für die Standardklassen `article/report/book`. Sie arbeiten mit den typographischen Regeln nach Tschichold. Die Klasse `article` wird zum Beispiel zu `scartcl`.

Die KOMA-Script-Klassen erzeugen ihr Seitenlayout mithilfe des Paketes `typearea` (siehe Abschnitt 4.2.3), wobei sie dessen Paketoptionen als Klassenoptionen anbieten. Ein erweitertes Seitendesign ist mit dem Paket `scpage2`

möglich, das über ähnliche Funktionen verfügt wie das fancyhdr-Paket. Genau wie typearea kann es ebenfalls eigenständig in Kombination mit einer der Standardklassen eingesetzt werden. Layouteinstellungen, wie Fontkontrolle, Layout der Legenden usw., wurden um Anpassungsmöglichkeiten erweitert, die in der Präambel eines Dokumentes zum Einsatz kommen.

Neben allen Funktionen der Standardklassen bieten die KOMA-Script-Klassen dem Anwender zusätzliche Steuerungsmöglichkeiten im Vor- und Nachspann, sowie eine Reihe anderer nützlicher Erweiterungen.

Die Sammlung ist gut dokumentiert. Das Handbuch, das in deutscher und englischer Sprache vorliegt, erläutert ausführlich alle Funktionen. Die deutsche Dokumentation wurde von der deutschen T<sub>E</sub>X-Anwendergruppe DANTE auch als nett gesetztes Buch [102] veröffentlicht.

### 4.6.2 memoir – Setzen komplexer Werke

Die Klasse memoir von Peter Wilson wurde ursprünglich als Alternative für die Standardklasse book entwickelt. Sie umfasst viele Funktionen, die man sonst nur als Erweiterungspakete findet. Die aktuelle Version eignet sich auch als Ersatz für die Klasse article und kann daher für alle Arten von Veröffentlichungen – von kurzen Vermerken bis hin zu komplexen Büchern – eingesetzt werden.

Es unterstützt unter anderem einen erweiterten Satz an Größen für Grundschriften (von 9pt bis 17pt), sowie die konfigurierbare Gestaltung von Abschnittsüberschriften, Kolummentiteln und Legenden. Für alle diese Elemente gibt es vordefinierte Layouts und neue können nach Bedarf deklariert werden. Die Klasse unterstützt Deklarationen, mit denen sich alle Größeneinstellungen für Seiten, Texte und Seitenränder vornehmen lassen, sowie die Verwendung von Beschnittmarken. Viele Komponenten der Klasse sind für Anwender, die andere Klassen um bestimmte Funktionen erweitern wollen, auch als eigenständige Pakete verfügbar (z.B. für Epigraphe oder zum Formatieren von Legenden).

Wie die KOMA-Script-Klassen, enthält auch die Klasse memoir ein ausgezeichnetes, fast 200 Seiten umfassendes Handbuch, das sich mit allen Fragen des Dokumentendesigns befasst und zeigt, wie sich potentielle Schwierigkeiten mit memoir beheben lassen.