Anleitung

Das KOMA-Script Paket

Frank Neukam Markus Kohm Axel Kielhorn

2001/08/17

Autoren der Anleitung:

Markus Kohm
Jens-Uwe Morawski
Enrico Kunz
Thomas Neumann

Haftungsausschluss:
Es wird keinerlei Haftung übernommen für irgendwelche Schäden, die aus der Benutzung der Bestandteile des hier beschriebenen Paketes resultieren. Siehe hierzu auch den Text der Datei `LEGALDE.TXT`, die zwingender Bestandteil des Paketes ist.

Inhaltsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Vorwort

KOMA-Script ist ein sehr komplexes Paket (engl. *bundle*). Dies ist schon allein darin begründet, dass es nicht nur aus einer einzigen Klasse (engl. *class*) oder einem einzigen Paket (engl. *package*), sondern einer Vielzahl derer besteht. Zwar sind die Klassen als Gegenstücke zu den Standardklassen konzipiert (siehe Kapitel 3), das heisst jedoch insbesondere nicht, dass sie nur über die Befehle, Umgebungen und Einstellmöglichkeiten der Standardklassen verfügen oder deren Aussehen als Standardeinstellung übernehmen. Näheres zur Geschichte entnehmen Sie bitte Abschnitt 1.3.

KOMA-Script bietet insgesamt eine Fülle von Klassen, Paketen, Befehlen, Umgebungen und Einstellmöglichkeiten. Diese reichen teilweise weit über die Fähigkeiten der Standardklassen hinaus. Manche davon sind auch als Ergänzung zu den Grundfähigkeiten des LaTeX-Kerns zu betrachten.

Allein aus dem Vorgenannten ergibt sich schon zwangsläufig, dass die Dokumentation zu KOMA-Script sehr umfangreich ausfällt. Hinzu kommt, dass KOMA-Script nirgendwo gelehrt wird. Das heisst, es gibt keinen Lehrer, der seine Schüler kennt und damit den Unterricht und das Unterrichtsmaterial entsprechend wählen und anpassen kann. Es wäre ein leichtes, die Dokumentation für irgendeine Zielgruppe zu verfassen. Die Schwierigkeit, der sich die Autoren gegenüber sehen, besteht jedoch darin, dass eine Anleitung für alle möglichen Zielgruppen benötigt wird. Wir haben uns bemüht, eine Anleitung zu erstellen, die für den Informatiker gleichermaßen geeignet ist wie für die Sekretärin des Fischhändlers. Wir haben uns bemüht, obwohl es sich dabei eigentlich um ein unmögliches Unterfangen handelt. Ergebnis sind zahlreiche Kompromisse. Wir bitten jedoch, die Problematik bei eventuellen Beschwerden zu berücksichtigen und bei der Verbesserung unserer derzeitigen Lösung zu helfen.

Trotz des Umfangs der Anleitung bitten wir außerdem darum, im Falle von Problemen zunächst die Dokumentation zu konsultieren. Hierzu gehören neben dem Guide auch alle Text-Dokumente, die Bestandteil des Pakets sind. Diese sind in `liesmich.txt` vollständig aufgeführt.

1.2 Dokumentaufbau

Dieser Guide enthält sehr wenige Informationen, die speziell für den LaTeX-Neuling geschrieben wurden. Es wird als zwingend vorausgesetzt, dass Dokumente wie [SKPH99], [Tea99a] gelesen und verstanden sind. Auch das Studium des einen oder anderen Buches zu LaTeX wird empfohlen. Literaturempfehlungen finden sich beispielsweise in [RH01]. Der Umfang von [RH01] ist ebenfalls erheblich. Dennoch wird darum gebeten, das Dokument nicht nur irgendwo vorliegen zu haben, sondern es mindestens einmal zu lesen und bei Problemen zu konsultieren. In [RH01] findest man übrigens auch einen Tipp, wie man ein A5-Dokument so auf A4-Papier drucken kann, dass man durch Falten in der Mitte ein Heft daraus machen kann. Der KOMA-Script-Guide liegt im A5-Format vor.

Im Gegensatz zu tiefgehender Anfängerinformation sind weiterführende Informationen und Begründungen in dieser Dokumentation reichlich vorhanden. Um solche Dokumentteile leichter erkennbar zu machen, sind sie in einer besonderen Schrift hervorgehoben. Damit ist es besonders ungeduldigen und uninteressierten Zeitgenossen möglich, solche Teile zu überspringen. Empfohlen wird dies aber ausdrücklich nicht. Gleichwohl ist es gut zu wissen, dass diese Teile nicht zwingend verstanden werden müssen, um KOMA-Script anzuwenden. Vor der Änderung und Kritisierung der Voreinstellungen von KOMA-Script, ist das Studium und Verständnis dieser Teile jedoch von erheblichem Vorteil.

Die Einteilung der Anleitung in Kapitel und Abschnitte soll ebenfalls dabei helfen, nur die Teile lesen zu müssen, die tatsächlich von Interesse sind. Um dies zu erreichen, sind die Informationen zu den einzelnen Klassen und Paketen nicht über das gesamte Dokument verteilt, sondern jeweils in einem Kapitel konzentriert. Querverweise in ein anderes Kapitel sind damit in der Regel auch Verweise auf einen anderen Teil des Gesamtpakets. Da die drei Hauptklassen in weiten Teilen übereinstimmen, sind sie in einem gemeinsamen Kapitel zusammengefasst. Die Unterschiede werden deutlich hervorgehoben. Soweit dies sinnvoll ist, geschieht es auch durch eine entsprechende Randnotiz.
scrartcl tiz. Wenn etwas nur die Klasse scrartcl betrifft, kann dies beispielsweise wie in diesem Abschnitt erfolgen.

1.3 Die Geschichte von KOMA-Script

Anfang der 90er Jahre wurde Frank Neukam damit beauftragt, ein Vorlesungsskript zu setzen. Damals war noch LaTeX2.09 aktuell und es gab keine Unterscheidung nach Klassen und Paketen, sondern alles waren Stile (engl. *styles*). Die Standarddokumentstile erschienen ihm für ein Vorlesungsskript nicht optimal und boten auch nicht alle Befehle und Umgebungen, die er benötigte.

Zur selben Zeit beschäftigte sich Frank auch mit Fragen der Typografie, insbesondere mit [Tsc87]. Damit stand für Frank fest, nicht nur irgendeinen Dokumentstil für

Skripten zu erstellen, sondern allgemein eine Stilfamilie, die den Regeln der europäischen Typografie folgt. Script war geboren.

Ich, Markus Kohm, traf auf Script ungefähr um den Jahreswechsel 1992/1993. Im Gegensatz zu Frank Neukam hatte ich häufig mit Dokumenten zu tun, für die ich das A5-Format bevorzuge. Zu jenem Zeitpunkt wurde A5 weder von den Standardklassen noch von Script unterstützt. Daher dauerte es nicht lange, bis ich erste Veränderungen an Script vornahm. Diese fanden sich auch in Script-2 wieder, das im Dezember 1993 von Frank veröffentlicht wurde.

Mitte 1994 erschien dann LaTeX 2_ε. Die damit einhergehenden Änderungen waren tiefgreifend. Daher blieb dem Anwender von Script-2 nur die Entscheidung, sich entweder auf den Kompatibilitätsmodus von LaTeX zu beschränken, oder auf Script zu verzichten. Wie viele andere wollte ich beides nicht. Also machte ich mich daran, einen Script-Nachfolger für LaTeX 2_ε zu entwickeln, der am 7. Juli 1994 unter dem Namen KOMA-Script erschienen ist. Ich will hier nicht näher auf die Wirren eingehen, die es um die offizielle Nachfolge von Script gab und warum mein Paket einen neuen Namen hat. Tatsache ist, dass auch aus Franks Sicht KOMA-Script der Nachfolger von Script-2 ist. Zu erwähnen ist noch, dass KOMA-Script ursprünglich ohne Briefklasse erschienen war. Diese wurde im Dezember 1994 von Axel Kielhorn beigesteuert. Noch etwas später erstellte Axel Sommerfeldt den ersten richtigen scrguide zu KOMA-Script.

Seither ist einiges an Zeit vergangen. LaTeX hat sich ein wenig verändert, die LaTeX-Landschaft erheblich. KOMA-Script hat sich weiterentwickelt. Es findet nicht mehr allein im deutschsprachigen Raum Anwender, sondern in ganz Europa, Nordamerika und Australien. Diese Anwenderklientel sucht bei KOMA-Script nicht allein nach einem typografisch ansprechenden Ergebnis. Zu beobachten ist vielmehr, dass bei KOMA-Script ein neuer Schwerpunkt entstanden ist: Flexibilisierung durch Variabilisierung. Unter diesem Schlagwort verstehe ich die Möglichkeit, in das Erscheinungsbild an vielen Stellen eingreifen zu können. Dies führte zu vielen neuen Makros, die mehr schlecht als recht in die existierende Anleitung integriert wurden. Irgendwann wurde es damit auch Zeit für eine komplett überarbeitete Anleitung, den neuen KOMA-Script-Guide.

1.4 Danksagung

Eine Danksagung in der Einleitung? Gehört die nicht vielmehr an den Schluss? Richtig! Eigentlich gehört die an das Ende. Mein Dank gilt hier jedoch nicht primär denjenigen, die diese Anleitung möglich gemacht haben. Für den Dank an die Guide-Autoren, mache ich den Leser zuständig! Mein persönlicher Dank gilt Frank Neukam, ohne dessen Script-Familie es vermutlich KOMA-Script nie gegeben hätte. Mein Dank gilt denjenigen, die an der Entstehung von KOMA-Script und den Anleitungen mitgewirkt haben. Mein Dank gilt auch allen, die mich immer wieder aufgemuntert haben, weiter zu machen und dieses oder jenes noch besser, weniger fehlerhaft oder schlicht zusätzlich zu implementieren.

Ganz besonderen Dank bin ich den Gründern von DANTE, Deutschsprachige Anwendervereinigung TEX e.V., schuldig, durch die letztlich die Verbreitung von TEX und LATEX und allen Paketen einschließlich KOMA-Script an einer zentralen Stelle überhaupt ermöglicht wird. In gleicher Weise bedanke ich mich bei den aktiven Helfern in der Usenet-Gruppe `de.comp.text.tex` und der Mailingliste `TEX-D-L`, die mir so manche Antwort auf Fragen zu KOMA-Script abnehmen.

1.5 Rechtliches

KOMA-Script steht unter der LATEX Project Public Licence. Eine nicht offizielle deutsche Übersetzung ist Bestandteil dieses Pakets. In allen Zweifelsfällen gilt im deutschsprachigen Raum der Text `LEGALDE.TXT` während in allen anderen Ländern der Text `LEGAL.TXT` anzuwenden ist.

Für die Korrektheit der Anleitung, Teile der Anleitung oder einer anderen in diesem Paket enthaltenen Dokumentation wird keine Gewähr übernommen.

1.6 Installation

Die Installation von KOMA-Script wird in den Dateien `liesmich.txt` und `INSTALLDE.TXT` beschrieben. Beachten Sie dazu auch die jeweilige Dokumentation zu der installierten TEX-Distribution.

1.7 Fehlermeldungen

Sollten Sie der Meinung sein, dass Sie einen Fehler in der Anleitung, einer der KOMA-Script-Klassen, einem der KOMA-Script-Pakete oder einem anderen Bestandteil von KOMA-Script gefunden haben, so sollten Sie wie folgt vorgehen. Prüfen Sie zunächst, ob inzwischen eine neue Version von KOMA-Script erschienen ist. Installieren Sie diese neue Version gegebenenfalls und kontrollieren Sie, ob der Fehler oder das Problem auch dann noch vorhanden ist.

Wenn es sich nicht um einen Fehler in der Dokumentation handelt und der Fehler oder das Problem nach einem Update noch immer auftritt, erstellen Sie bitte wie in [RH01] angegeben ein minimales Beispiel. Ein solches Beispiel enthält nur einen minimalen Text und nur die Pakete und Definitionen, die für die Verdeutlichung des Fehlers unbedingt notwendig sind. Auf exotische Pakete sollte möglichst ganz verzichtet werden.

Oft lässt sich ein Problem durch ein minimales Beispiel so weit eingrenzen, dass bereits vom Anwender selbst festgestellt werden kann, ob es sich um einen Anwendungsfehler handelt oder nicht. Auch ist so sehr häufig zu erkennen, welche Pakete oder Klassen konkret das Problem verursachen und ob es sich überhaupt um ein KOMA-Script-Problem handelt oder nicht. Dies können Sie gegebenenfalls zusätzlich überprüfen, indem Sie statt einer KOMA-Script-Klasse einen Test mit der entsprechenden Standardklasse vornehmen. Danach ist dann auch klar, ob der Fehlerbericht an den Autor von KOMA-Script oder an den Autor eines anderen Pakets zu richten ist. Sie sollten spätestens jetzt noch einmal gründlich die Anleitungen der entsprechenden Paket, Klassen und KOMA-Script-Bestandteile studieren sowie [RH01] konsultieren. Möglicherweise existiert ja bereits ein Lösung für Ihr Problem, so dass sich eine Fehlermeldung erübrigt.

Für die eigentliche Meldung sollte unbedingt das zu KOMA-Script gehörende interaktive LaTeX-Dokument `komabug.tex` verwendet werden. Dabei wird eine Nachricht generiert, die alle grundlegenden Informationen enthält. Im Dokument ist auch die Adresse angegeben, an die Sie die Meldung schicken können.

Häufig werden Sie eine Frage zu KOMA-Script oder im Zusammenhang mit KOMA-Script lieber öffentlich, beispielsweise in `de.comp.text.tex` stellen wollen, als dem KOMA-Script-Autor zu schreiben. Auch in diesem Fall sollten Sie unbedingt die Version des LaTeX-Kerns, die Version der verwendeten Klassen und Pakete und ein minimales Beispiel angeben. Wenn Sie in der Präambel dieses minimalen Beispiels die Anweisung `\listfiles` verwenden, können Sie die Angaben zu den Versionen anschließend der `log`-Datei entnehmen.

1.8 Autoren

Unter dieser Überschrift ist am Ende jedes Kapitels der Dokumentation zu finden, welche Autoren daran mitgewirkt haben. Der jeweils zuständige Autor ist dabei fett hervorgehoben. Ist beim zuständigen Autor eine E-Mail-Adresse angegeben, können Sie diesem auch direkt schreiben. Bei diesem Kapitel ist das:

- **Markus Kohm** <Markus.Kohm@gmx.de>

2 Satzspiegelberechnung mit typearea.sty

2.1 Grundlagen der Satzspiegelkonstruktion

Betrachtet man eine einzelne Seite eines Buches oder eines anderen Druckwerkes, so besteht diese auf den ersten Blick aus den Rändern[1], einem Kopfbereich, einem Textkörper und einem Fußbereich. Genauer betrachtet kommt noch ein Abstand zwischen Kopfbereich und Textkörper sowie zwischen Textkörper und Fußbereich hinzu. Die Aufteilung dieser Bereiche, sowie ihre Anordnung zueinander und auf dem Papier nennt man *Satzspiegel*.

In der Literatur werden verschiedene Algorithmen und heuristische Verfahren zur Konstruktion eines guten Satzspiegels vorgeschlagen und diskutiert. Häufig findet man dabei ein Verfahren, das mit verschiedenen Diagonalen und Schnittpunkten arbeitet. Das gewünschte Ergebnis dabei ist, dass das Seitenverhältnis des Textbereichs dem Seitenverhältnis *der Seite* entspricht. Bei einem einseitigen Dokument sollen außerdem der linke und rechte Rand gleich breit sein, während der obere zum unteren Rand im Verhältnis $1 : 2$ stehen sollte. Bei einem doppelseitigen Dokument, beispielsweise einem Buch, ist hingegen zu beachten, dass der innere Rand genauso groß sein sollte wie jeder der beiden äußeren Ränder.

Im vorherigen Abschnitt wurde *die Seite* erwähnt und hervorgehoben. Irrtümlich wird oftmals angenommen, das Format der Seite wäre mit dem Format des Papiers gleichzusetzen. Betrachtet man jedoch ein gebundenes Druckerzeugnis, so ist zu erkennen, dass ein Teil des Papiers in der Bindung verschwindet und nicht mehr als Seite zu sehen ist. Für den Satzspiegel ist jedoch nicht entscheidend, welches Format das Papier hat, sondern, was der Leser für einen Eindruck vom Format der Seite bekommt. Damit ist klar, dass bei der Berechnung des Satzspiegels der Teil, der durch die Bindung versteckt wird, aus dem Papierformat herausgerechnet und dann zum inneren Rand hinzugefügt werden muss. Wir nennen diesen Teil *Bindekorrektur*.

Die Bindekorrektur ist vom jeweiligen Produktionsvorgang abhängig und kann nicht allgemein festgelegt werden. Es handelt sich dabei also um einen Parameter, der für jeden Produktionsvorgang neu festzulegen ist. Im professionellen Bereich spielt dieser Wert nur eine geringe Rolle, da ohnehin auf größere Papierbögen gedruckt und entsprechend geschnitten wird. Beim Schneiden wird dann wiederum sichergestellt, dass obige Verhältnisse für die sichtbare Doppelseite möglichst eingehalten sind.

Wir wissen nun also, wie die einzelnen Teile zueinander stehen. Was wir noch nicht

[1] Der Autor und der Lektor haben an dieser Stelle überlegt, ob eine Seite nicht nur einen umlaufenden Rand hat und daher von „dem Rand“ die Rede sein müsste. Da jedoch LaTeX diesen einen Rand logisch in mehrere Ränder unterteilt, die getrennt bestimmt werden, ist hier auch von „den Rändern“ die Rede.

wissen ist, wie breit und hoch der Textbereich ist. Kennen wir eines dieser beiden Maße, so ergeben sich zusammen mit dem Papierformat und dem Seitenformat oder der Bindekorrektur alle anderen Maße durch Lösung mehrerer mathematischer Gleichungen.

$$
\begin{aligned}
Textbereichshoehe : Textbreichsbreite &= Seitenhoehe : Seitenbreite \\
Seitenbreite &= Papierbreite - Bindekorrektur \\
oberer\ Rand + unterer\ Rand &= Seitenhoehe - Textbereichshoehe \\
oberer\ Rand : unterer\ Rand &= 1 : 2 \\
linker\ Rand : rechter\ Rand &= 1 : 1 \\
halber\ innerer\ Rand &= \frac{1}{2} aeusserer\ Rand + Bindekorrektur
\end{aligned}
$$

Dabei gibt es $linker\ Rand$ und $rechter\ Rand$ nur im einseitigen Druck, während es $innerer\ Rand$ und $aeusserer\ Rand$ nur im doppelseitigen Druck gibt. In den Gleichungen wird mit $halber\ innerer\ Rand$ gearbeitet, weil der ganze innere Rand zur Doppelseite gehört. Zu einer Seite gehört also nur die Hälfte des inneren Randes.

Die Frage nach der Breite des Textbereichs wird in der Literatur ebenfalls diskutiert. Die optimale Textbereichsbreite ist von verschiedenen Faktoren abhängig.

- Größe, Laufweite und Art der verwendeten Schrift
- Verwendeter Durchschuss
- Länge der Worte
- Verfügbarer Platz

Der Einfluss der Schrift wird deutlich, wenn man sich bewusst macht, wozu Serifen dienen. Serifen sind kleine Striche an den Linienenden der Buchstaben. Buchstaben, die mit vertikalen Linien auf die Grundlinie der Textzeile treffen, lösen diese eher auf, als dass sie das Auge auf der Linie halten. Genau bei diesen Buchstaben liegen die Serifen horizontal auf der Grundlinie und verstärken damit die Zeilenwirkung der Schrift. Das Auge kann der Textzeile nicht nur beim Lesen der Worte, sondern insbesondere auch beim schnellen Zurückspringen an den Anfang der nächsten Zeile besser folgen. Damit darf die Zeile bei einer Schrift mit Serifen genaugenommen länger sein als bei einer Schrift ohne Serifen.

Unter dem Durchschuss versteht man den Abstand zwischen Textzeilen. Bei LaTeX ist ein Durchschuss von etwa 20% der Schriftgröße voreingestellt. Mit Befehlen wie `\linespread` oder besser mit Hilfe von Paketen wie setspace (siehe [Tob00]) kann der Durchschuss verändert werden. Ein großer Durchschuss erleichtert dem Auge die Verfolgung einer Zeile. Bei sehr großem Durchschuss wird das Lesen aber dadurch gestört, dass das Auge zwischen den Zeilen weite Wege zurücklegen muss. Daneben wird sich der Leser des entstehenden Streifeneffekts sehr deutlich und unangenehm bewusst. Der Graueindruck der Seite ist in diesem Fall gestört. Dennoch können bei großem Durchschuss, die Zeilen länger sein.

Auf der Suche nach konkreten Werten für gute Zeilenlängen findet man in der Literatur je nach Autor unterschiedliche Angaben. Teilweise ist dies auch in der Muttersprache des Autors begründet. Das Auge springt nämlich üblicherweise von Wort zu Wort, wobei kurze Worte diese Aufgabe erleichtern. Über alle Sprachen und Schriftarten hinweg kann man sagen, dass eine Zeilenlänge von 60 bis 70 Zeichen, einschließlich Leer- und Satzzeichen, einen brauchbaren Kompromiss darstellen. Ein gut gewählter Durchschuss wird dabei vorausgesetzt. Bei den Voreinstellungen von LaTeX braucht man sich über letzteres normalerweise keine Sorgen zu machen.

Bevor wir uns an die konkrete Konstruktion machen, fehlen jetzt nur noch Kleinigkeiten, die man wissen sollte. LaTeX beginnt die erste Zeile des Textbereichs einer Seite nicht am oberen Rand des Textbereichs, sondern setzt die Zeile mit einem definierten Abstand zum oberen Rand des Textbereichs. Desweiteren verfügt LaTeX über die beiden Befehle `\raggedbottom` und `\flushbottom`. Der erste dieser Befehle legt fest, dass die letzte Zeile einer jeden Seite dort liegen soll, wo sie eben zu liegen kommt. Das kann dazu führen, dass sich die Position der letzten Zeile von Seite zu Seite vertikal um nahezu eine Zeile verändern kann. Im doppelseitigen Druck ist das in der Regel unerwünscht. Mit dem zweiten Befehl, `\flushbottom`, wird hingegen festgelegt, dass die letzte Zeile immer am unteren Rand des Textbereichs zu liegen kommt. Um dies zu erreichen, muss LaTeX gegebenenfalls dehnbare vertikale Abstände über das erlaubte Maß hinaus strecken. Ein solcher Abstand ist beispielsweise der Absatzabstand. Dies gilt auch, wenn man diesen auf Null gesetzt hat. Um nicht bereits auf normalen Seiten, auf denen der Absatzabstand das einzige dehnbare vertikale Maß darstellt, eine Dehnung zu erzwingen, sollte die Höhe des Textbereichs ein Vielfaches der Textzeilenhöhe zuzüglich des Abstands der ersten Zeile vom oberen Rand des Textbereichs sein.

Damit sind nun alle Grundlagen der Satzspiegelberechnung, die bei KOMA-Script eine Rolle spielen, zusammengetragen. Wir können also mit der konkreten Konstruktion beginnen.

2.2 Satzspiegelkonstruktion durch Teilung

Der einfachste Weg, um zu erreichen, dass der Textbereich dasselbe Verhältnis aufweist wie die Seite, ist folgender. Zunächst nimmt man an der Innenseite des Papiers den Teil $BCOR$, der für die Bindekorrektur benötigt wird ab und teilt die restliche Seite vertikal in eine Anzahl DIV gleich hoher Streifen. Dann teilt man die Seite horizontal in die gleiche Anzahl DIV gleich breiter Streifen. Nun verwendet man den obersten horizontalen Streifen als oberen und die beiden untersten horizontalen Streifen als unteren Rand. Im doppelseitigen Druck verwendet man außerdem den innersten vertikalen Streifen als inneren und die beiden äußersten vertikalen Streifen als äußeren Rand. Zum inneren Rand gibt man dann noch $BCOR$ hinzu. Was nun innerhalb der Seite noch übrig bleibt, ist der Textbereich. Die Breite bzw. Höhe der Ränder und des Textbereichs resultiert damit automatisch aus der Anzahl DIV der Streifen. Da für die Ränder insgesamt jeweils drei Streifen benötigt werden, muss DIV zwingend größer als drei sein.

Bei KOMA-Script ist diese Art der Konstruktion im Paket typearea realisiert. Dabei sind für A4-Papier je nach Schriftgröße unterschiedliche Werte voreingestellt, die Tabelle 2.2 zu entnehmen sind. Bei Verzicht auf Bindekorrektur, wenn also $BCOR = 0\,\mathrm{pt}$ gilt, ergeben sich in etwa die Satzspiegelmaße aus Tabelle 2.1.

Neben den voreingestellten Werten kann man *BCOR* und *DIV* direkt beim Laden des Pakets als Option angeben (siehe Abschnitt 2.4). Zusätzlich existiert ein Befehl, mit dem man einen Satzspiegel explizit berechnen kann und dem man die beiden Werte als Parameter übergibt (siehe ebenfalls Abschnitt 2.4).

Das typearea-Paket bietet außerdem die Möglichkeit, den optimalen *DIV*-Wert automatisch zu bestimmen. Dieser ist von der Schriftart abhängig, die zum Zeitpunkt der Satzspiegelberechnung eingestellt ist. Siehe hierzu ebenfalls Abschnitt 2.4.

2.3 Satzspiegelkonstruktion durch Kreisschlagen

Neben der zuvor beschriebenen Satzspiegelkonstruktion gibt es in der Literatur noch eine eher klassische Methode. Bei diesem Verfahren will man die gleichen Werte nicht nur in Form des Seitenverhältnisses wiederfinden, man geht außerdem davon aus, dass das Optimum dann erreicht wird, wenn die Höhe des Textbereichs der Breite der Seite entspricht. Das genaue Verfahren ist beispielsweise in [Tsc87] nachzulesen.

Als Nachteil dieses spätmittelalterlichen Buchseitenkanons ergibt sich, dass die Breite des Textbereichs nicht mehr von der Schriftart abhängt. Es wird also nicht mehr der zur Schrift passende Textbereich gewählt, stattdessen muss der Autor oder Setzer die zum Textbereich passende Schrift wählen. Dies ist als zwingend zu betrachten.

Im typearea-Paket wird diese Konstruktion dahingehend abgewandelt, dass durch Auswahl eines ausgezeichneten – normalweise unsinnigen – *DIV*-Wertes oder eine spezielle Paket-Option derjenige *DIV*-Wert ermittelt wird, bei dem der resultierende Satzspiegel dem spätmittelalterlichen Buchseitenkanon am nächsten kommt. Siehe hierzu ebenfalls Abschnitt 2.4.

2.4 Optionen und Makros zur Beeinflussung des Satzspiegels

Das Paket typearea bietet zwei unterschiedliche Benutzerschnittstellen, um auf die Satzspiegelkonstruktion Einfluss zu nehmen. Die erste Möglichkeit ist, beim Laden des Pakets entsprechende Optionen anzugeben. Wie man Pakete lädt und Paketoptionen übergibt entnehmen Sie bitte der Literatur zu LaTeX, beispielsweise [SKPH99] und [Tea99a], oder den hier aufgeführten Beispielen. Da bei der Verwendung der KOMA-Script-Hauptklassen das Paket typearea automatisch geladen wird, können die entsprechenden Paket-Optionen bei diesen Klassen auch direkt als Klassen-Optionen übergeben werden (siehe Abschnitt 3.1).

`BCOR`*Korrektur*

Mit Hilfe der Option `BCOR`*Korrektur* geben Sie den absoluten Wert der Bindekorrektur an, also die Breite des Bereichs der durch die Bindung von der Papierbreite verloren geht. Dieser Wert wird in der Satzspiegelkonstruktion automatisch berücksichtigt und bei der Ausgabe wieder dem inneren beziehungsweise linken Rand zugeschlagen. Als *Korrektur* können Sie jedes übliche TEX-Maß angeben.

Beispiel: Angenommen Sie erstellen einen Finanzbericht. Das ganze soll einseitig in A4 gedruckt und anschließend in eine Klemmmappe geheftet werden. Die Klemme der Mappe verdeckt 7,5 mm. Der Papierstapel ist sehr dünn, deshalb gehen beim Knicken und Blättern durchschnittlich höchstens weitere 0,75 mm verloren. Sie schreiben dann also:

```
\documentclass[a4paper]{report}
\usepackage[BCOR8.25mm]{typearea}
```

oder bei Verwendung einer KOMA-Script-Klasse:

```
\documentclass[a4paper,BCOR8.25mm]{scrreprt}
```

`DIV`*Faktor*

Mit Hilfe der Option `DIV`*Faktor* wird festgelegt, in wieviele Steifen die Seite horizontal und vertikal bei der Satzspiegelkonstruktion eingeteilt wird. Die genaue Konstruktion ist Abschnitt 2.2 zu entnehmen. Wichtig zu wissen ist, dass gilt: Je größer der *Faktor* desto größer wird der Textbereich und desto kleiner die Ränder. Als *Faktor* kann jeder ganzzahlige Wert ab 4 verwendet werden. Bitte beachten Sie jedoch, dass sehr große Werte dazu führen können, dass Randbedingungen der Satzspiegelkonstruktion, je nach Wahl der weiteren Optionen, verletzt werden. So kann die Kopfzeile im Extremfall auch außerhalb der Seite liegen. Bei Verwendung der Option `DIV`*Faktor* sind Sie für die Einhaltung der Randbedingungen sowie eine nach typografischen Gesichtspunkten günstige Zeilenlänge selbst verantwortlich.

In Tabelle 2.1 findet Sie für das Seitenformat A4 und ohne Bindekorrektur die aus einigen Faktoren *DIV* resultierenden Satzspiegelgrößen. Dabei werden die weiteren, von der Schriftgröße abhängigen Nebenbedingungen nicht berücksichtigt.

Beispiel: Angenommen Sie schreiben ein Sitzungsprotokoll. Sie verwenden dafür die Klasse protokol[2]. Das ganze soll doppelseitig werden. In Ihrer Firma wird die Schriftart Bookman in 12 pt verwendet. Diese

[2] Die Klasse protokol ist eine hypothetische Klasse. Diese Anleitung geht von dem Indealfall aus, dass für jede Aufgabe eine dafür passende Klasse vorhanden ist.

	Textbereich		Ränder	
DIV	Breite [mm]	Höhe [mm]	oben [mm]	innen [mm]
6	105,00	148,50	49,50	35,00
7	120,00	169,71	42,43	30,00
8	131,25	185,63	37,13	26,25
9	140,00	198,00	33,00	23,33
10	147,00	207,90	29,70	21,00
11	152,73	216,00	27,00	19,09
12	157,50	222,75	24,75	17,50
13	161,54	228,46	22,85	16,15
14	165,00	233,36	21,21	15,00
15	168,00	237,60	19,80	14,00

Tabelle 2.1: Satzspiegelmaße in Abhängigkeit von *DIV* bei A4

Schriftart, die zu den Standard-Postscript-Schriften gehört, wird in LaTeX mit der Anweisung `\usepackage{bookman}` aktiviert. Die Schriftart Bookman läuft sehr weit, das heißt, die einzelnen Zeichen sind im Verhältnis zur Höhe relativ breit. Deshalb ist Ihnen die Voreinstellung für den *DIV*-Wert in typearea zu gering. Statt einem Wert von 12 sind Sie nach gründlichem Studium dieses Kapitels einschließlich der weiterführenden Abschnitte überzeugt, dass ein Wert 15 angebracht ist. Das Protokoll wird nicht gebunden, sondern gelocht und in einen Ordner abgeheftet. Eine Bindekorrektur ist deshalb nicht notwendig. Sie schreiben also:

```
\documentclass[a4paper,twoside]{protokol}
\usepackage{bookman}
\usepackage[DIV15]{typearea}
```

Als Sie fertig sind, macht man Sie darauf aufmerksam, dass die Protokolle neuerdings gesammelt und am Quartalsende alle zusammen als Buch gebunden werden. Die Bindung erfolgt im Copyshop als einfache Leimbindung, weil den Band ohnehin nie wieder jemand anschaut und nur wegen ISO 9000 angefertigt wird. Für die Bindung einschließlich Biegefalz werden durchschnittlich 12 mm benötigt. Sie ändern die Optionen von typearea also entsprechend ab und verwenden die Klasse für Protokolle nach ISO 9000:

```
\documentclass[a4paper,twoside]{iso9000p}
\usepackage{bookman}
\usepackage[DIV15,BCOR12mm]{typearea}
```

Grundschriftgröße:	10 pt	11 pt	12 pt
DIV:	8	10	12

Tabelle 2.2: *DIV*-Voreinstellungen für A4

Natürlich können Sie auch hier wieder eine KOMA-Script-Klasse verwenden:

```
\documentclass[a4paper,twoside,DIV15,BCOR12mm]{scrartcl}
\usepackage{bookman}
```

`DIVcalc`
`DIVclassic`

Wie bereits in Abschnitt 2.2 erwähnt, gibt es nur für das Papierformat A4 feste Voreinstellungen für den *DIV*-Wert. Diese sind Tabelle 2.2 zu entnehmen. Wird ein anderes Papierformat gewählt, so berechnet typearea selbst einen guten *DIV*-Wert. Natürlich können Sie diese Berechnung auch für A4 wählen. Hierzu verwenden Sie einfach die Option `DIVcalc` an Stelle von `DIV`*Faktor*. Selbstverständlich können Sie diese Option auch explizit bei allen anderen Papierformaten angeben. Wenn Sie die automatische Berechnung wünschen, ist diese Angabe sogar sinnvoll, da die Möglichkeit besteht, in einer Konfigurationsdatei andere Voreinstellungen zu setzen (siehe Abschnitt 2.7), die dann mit dieser Option außer Kraft gesetzt werden könnten.

Die in Abschnitt 2.3 erwähnte klassische Konstruktion, der mittelalterliche Buchseitenkanon, mit der Abweichung, dass ein dazu möglichst gut passender *DIV*-Wert ermittelt wird, ist ebenfalls auswählbar. Verwenden Sie in diesem Fall an Stelle von `DIV`*Faktor* oder `DIVcalc` einfach die Option `DIVclassic`.

Beispiel: In dem bei der Option `DIV`*Faktor* aufgeführten Beispiel mit der Schriftart Bookman, gab es ja genau das Problem, dass man einen zur Schriftart besser passenden *DIV*-Wert haben wollte. Man könnte also in Abwandlung des ersten Beispiels auch einfach die Ermittlung dieses Wertes typearea überlassen:

```
\documentclass[a4paper,twoside]{protokol}
\usepackage{bookman}
\usepackage[DIVcalc]{typearea}
```

`\typearea[`*BCOR*`]{`*DIV*`}`

Wenn Sie bis hier die Beispiele aufmerksam verfolgt haben, werden Sie sich fragen, wie man die Berechnung eines *DIV*-Wertes in Abhängigkeit von der gewählten Schrift erreicht, wenn eine KOMA-Script-Klasse verwendet wird, also die Optionen für typearea vor dem Laden beispielsweise des bookman-Pakets erfolgen müsste. In diesem Fall könnte typearea nur einen Satzspiegel für die Standardschrift nicht jedoch für die dann tatsächlich verwendete Schrift Bookman berechnen. Nach der Auswertung der Optionen berechnet das typearea-Paket den Satzspiegel mit Hilfe des Befehls `\typearea[`*BCOR*`]{`*DIV*`}`. Dabei wird der gewählte *BCOR*-Wert als optionaler Parameter übergeben und der *DIV*-Wert als Parameter. Bei der Option `DIVcalc` wird dabei als *DIV* der eigentlich ungültige Wert 1 übergeben. Bei der Option `DIVclassic` der eigentlich ungültige Wert 3. Den Befehl `\typearea` kann man auch explizit in der Präambel aufrufen.

Beispiel: Gehen wir wieder davon aus, dass für die Schriftart Bookman ein Satzspiegel mit guter Zeilenlänge berechnet werden soll. Gleichzeitig wird eine KOMA-Script-Klasse verwendet. Dies ist unter Verwendung des `\typearea`-Befehls mit dem `DIVcalc`-Wert 1 als *DIV*-Parameter möglich:

```
\documentclass[a4paper,BCOR12mm,DIVcalc,twoside]{scrartcl}
\usepackage{bookman}
\typearea[12mm]{1}% entspricht obigen Optionen
```

Nun ist es möglicherweise etwas unpraktisch, wenn man bei der `DIV`-Option die Möglichkeit hat, `DIVcalc` und `DIVclassic` anzuwenden, beim `\typearea`-Befehl aber mit irgendwelchen Pseudowerten jonglieren soll. Deshalb versteht `\typearea` auch folgende symbolische Angaben für den Parameter *DIV*:

`current` – Satzspiegelberechnung mit dem aktuell gültigen *DIV*-Wert erneut durchführen.

`default` – Satzspiegelberechnung mit dem Standardwert für das aktuelle Seitenformat und die aktuelle Schriftgröße erneut durchführen. Falls kein Standardwert existiert `calc` anwenden.

`calc` – Satzspiegelberechnung einschließlich Ermittlung eines guten *DIV*-Wertes erneut durchführen.

`classic` – Satzspiegelberechnung nach dem mittelalterlichen Buchseitenkanon (Kreisberechnung) erneut durchführen.

Jetzt wäre es natürlich äußerst unpraktisch, wenn man zwar eine Satzspiegelberechnung mit dem aktuellen *DIV*-Wert erneut durchführen könnte, jedoch dabei den *BCOR*-Wert neu angeben müsste. deshalb versteht `\typearea` auch die folgende symbolsiche Angabe für den Parameter *BCOR*:

`current` – Satzspiegelberechnung mit dem aktuell gültigen *BCOR*-Wert erneut durchführen.

Beispiel: Gehen wir wieder wieder davon aus, dass für die Schriftart Bookman ein Satzspiegel mit guter Zeilenlänge berechnet werden soll. Gleichzeitig wird eine KOMA-Script-Klasse verwendet. unter Verwendung der symbolischen Parameterwerte für *BCOR* und *DIV* ist dies mit dem `\typearea`-Befehl einfach möglich:

```
\documentclass[a4paper,BCOR12mm,DIVcalc,twoside]{scrartcl}
\usepackage{bookman}
\typearea[current]{calc}
```

Soll hingegen die neuerliche Berechnung mit einem festen *DIV*-Wert durchgeführt werden, so gibt es neben der Möglichkeit:

```
\documentclass[a4paper,BCOR12mm,DIV11,twoside]{scrartcl}
\usepackage{bookman}
\typearea[current]{current}
```

natürlich auch noch die alte Methode:

```
\documentclass[a4paper,twoside]{scrartcl}
\usepackage{bookman}
\typearea[12mm]{11}
```

Letztlich ist es eine Frage des Geschmacks, welche Lösung man lieber verwendet.

Häufig wird die Satzspiegelneuberechnung im Zusammenhang mit der Veränderung des Zeilenabstandes (*Durchschuss*) benötigt. Da der Satzspiegel unbedingt so berechnet werden sollte, dass eine ganze Anzahl an Zeilen in den Textbereich passt, muss bei Verwendung eines anderen Durchschusses als dem normalen der Satzspiegel für diesen Zeilenabstand neu berechnet werden.

Beispiel: Angenommen für eine Diplomarbeit wird bei eine Schriftgröße von 10 pt bei eineinhalbzeiligem Satz zwingend gefordert. LaTeX setzt normalwerweise bei 10 pt mit 2 pt Durchschuss, also 1,2-zeilig. Deshalb muss als zusätzlicher Dehnfaktor der Wert 1,25 verwendet werden. Gehen wir außerdem davon aus, dass eine Bindekorrektur von 12 mm benötigt wird. Dann könnte die Lösung dies Problems wie folgt aussehen:

```
\documentclass[10pt,BCOR12mm,DIVcalc,twoside]{scrreprt}
\linespread{1.25}\selectfont
\typearea[current]{calc}
```

Der Befehl `\selectfont` wird benötigt, damit der geänderte Durchschuss auch tatsächlich vor der Neuberechnung aktiviert wird.

Das gleiche Beispiel sähe unter Verwendung des setspace-Pakets (siehe [Tob00]) wie folgt aus:

```
\documentclass[10pt,BCOR12mm,DIVcalc,twoside]{scrreprt}
\usepackage{setspace}
\onehalfspacing
\typearea[current]{calc}
```

Wie man sieht, spart man sich mit dem setspace-Paket nicht nur das Wissen um den korrekten Dehnungswert, sondern auch das Wissen, dass `\selectfont` benötigt wird.

An dieser Stelle erscheint es mir angebracht, darauf hinzuweisen, dass der Zeilenabstand für die Titelseite wieder auf den normalen Wert zurückgesetzt werden sollte. Ein vollständiges Beispiel wäre also:

```
\documentclass[10pt,BCOR12mm,DIVcalc,twoside]{scrreprt}
\usepackage{setspace}
\onehalfspacing
\typearea[current]{calc}
\begin{document}
\title{Title}
\author{Markus Kohm}
\begin{spacing}{1}
  \maketitle
  \tableofcontents
\end{spacing}
\chapter{Ok}
\end{document}
```

Siehe hierzu auch die Anmerkungen in Abschnitt 2.8.

Der Befehl `\typearea` ist derzeit so definiert, dass es theoretisch auch möglich wäre, mitten in einem Dokument den Satzspiegel zu wechseln. Dabei werden allerdings Annahmen über den Aufbau des LaTeX-Kerns gemacht und interne Definitionen und Größen des LaTeX-Kerns verändert. Es gibt zwar eine gewisse Wahrscheinlichkeit aber keine Garantie, dass dies in zukünftigen Versionen von LaTeX 2_ε noch funktionieren wird. Es ist anzunehmen, dass es bei LaTeX3 nicht mehr zu einem korrekten Ergebnis führt. Aber als Autor von KOMA-Script gehe ich derzeit davon aus, dass der Umstieg zu LaTeX3 mit sehr viel mehr Inkompatiblilitäten einher gehen wird.

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

Bisher wurde zwar erklärt, wie die Satzspiegelkonstruktion funktioniert und in welchem Verhältnis die Ränder zueinander und der Textkörper zur Seite steht, aber eine entscheidene Frage blieb ausgeklammert. Es handelt sich dabei um die Frage, was denn eigentlich unter dem Rand zu verstehen ist. Auf den ersten Blick wirkt diese Frage trivial: der Rand ist der Teil der Seite, der oben, unten, links und rechts frei bleibt. Doch das ist nur die halbe Wahrheit. Der äußere Rand ist keineswegs immer leer. Teilweise findet man darin noch gesetzte Randnotizen (siehe den Befehl `\marginpar` beispielsweise in [SKPH99] bzw. Abschnitt 3.5.5).

Beim oberen und unteren Rand stellt sich die Frage, wie Kopf- und Fußzeile zu behandeln sind. Gehören diese beiden zum Textkörper oder zum jeweiligen Rand? Die Frage ist nicht einfach zu beantworten. Eindeutig ist, dass ein leerer Fuß und ein leerer Kopf zum Rand zu rechnen ist. Schließlich kann er nicht vom restlichen Rand unterschieden werden. Ein Fuß, der nur die Paginierung[3] enthält, wirkt optisch ebenfalls eher wie Rand und sollte deshalb zu diesem gerechnet werden. Für die optische Wirkung ist dabei unwesentlich, ob der Fuß beim Lesen oder Überfliegen leicht als Fuß erkannt werden kann oder nicht. Entscheidend ist, wie eine wohlgefüllte Seite bei *unscharfer Betrachtung* wirkt. Dazu bedient man sich beispielsweise seiner altersweitsichtigen Großeltern, denen man die Brille stibitzt und dann die Seite etwa einen halben Meter von der Nasenspitze entfernt hält. In Ermangelung erreichbarer Großerltern kann man sich auch damit behelfen, dass man die eigenen Augen auf Fernsicht stellt die Seite aber nur mit ausgestreckten Armen hält. Brillenträger sind hier deutlich im Vorteil. Hat man eine Fußzeile, die neben der Paginierung weitere, weitschweifige Angaben enthält, beispielsweise einen Copyrighthinweis, so wirkt die Fußzeile eher wie ein etwas abgesetzter Teil des Textkörpers. Bei der Berechnung des Satzspiegels sollte das berücksichtigt werden.

Bei der Kopfzeile sieht es noch schwieriger aus. In der Kopfzeile wird häufig der Kolumnentitel[4] gesetzt. Arbeitet man mit einem lebenden Kolumnentitel, also der Wiederholung der ersten bzw. zweiten Gliederungsebene in der Kopfzeile, und hat gleichzeitig sehr lange Überschriften, so erhält man automatisch sehr lange Kopfzeilen. In diesem Fall wirkt der Kopf wiederum wie ein abgesetzter Teil des Textkörpers und weniger wie leerer Rand. Verstärkt wird dieser Effekt noch, wenn neben dem Kolumnentitel auch die Paginierung im Kopf erfolgt. Dadurch erhält man einen links und rechts abgeschlossenen Bereich, der kaum noch als leerer Rand wirkt. Schwieriger ist es bei Paginierung im Fuß

[3] Unter der Paginierung versteht man die Angabe der Seitenzahl.

[4] Unter dem Kolumnentitel versteht man in der Regel die Wiederholung einer Überschrift mit Titelcharakter.

und Überschriften, deren Länge sehr stark schwankt. Hier kann der Kopf der einen Seite wie Textkörper wirken, der Kopf der anderen Seite jedoch eher wie Rand. Keinesfalls sollte man die Seiten jedoch unterschiedliche behandeln. Das würde zu vertikal springenden Köpfen führen und ist nicht einmal für ein Daumenkino geeignet. Ich rate in diesem Fall dazu, den Kopf zum Textkörper zu rechnen.

Ganz einfach fällt die Entscheidung, wenn der Kopf oder Fuß durch eine Linie vom eigentlichen Textkörper abgetrennt ist. Dadurch erhält man eine geschlossene Wirkung und der Kopf bzw. Fuß sollte unbedingt zum Textkörper gerechnet werden. Wie gesagt, die durch die Trennlinie verbesserte Erkennung des Kopfes oder Fußes ist hier unerheblich. Entscheidend ist die unscharfe Betrachtung.

Das typearea-Paket trifft die Entscheidung, ob ein Kopf oder Fuß zum Textkörper gehört oder davon getrennt zum Rand gerechnet werden muss, nicht selbst. Stattdessen kann mit den Optionen `headinclude` und `footinclude` der Kopf bzw. Fuß explizit zum Textkörper gezählt werden, während mit den Optionen `headexclude` und `footexclude` der Kopf bzw. Fuß zum Rand gerechnet wird. Wenn Sie unsicher sind, was die richtige Einstellung ist, lesen Sie bitte obige Erläuterungen. Voreingestellt sind normalerweise `headexclude` und `footexclude`. Dies kann sich jedoch bei den KOMA-Script-Klassen je nach Klassenoption oder bei Verwendung anderer KOMA-Script-Pakete generell ändern (siehe Abschnitt 3.1 und 4).

Wert`headlines`

Es ist nun also bekannt, wie man Satzspiegel mit dem typearea-Paket berechnet und wie man dabei angibt, ob der Kopf oder Fuß zum Textkörper oder zum Rand gehört. Insbesondere für den Kopf fehlt aber noch die Angabe, wie hoch er denn eigentlich sein soll. Hierzu dient die Option `headlines`, der man die Anzahl der Kopfzeilen voranstellt. Normalerweise arbeitet das typearea-Paket mit 1,25 Kopfzeilen. Dieser Wert stellt einen Kompromiss dar. Zum einen ist er groß genug, um auch für eine unterstrichene Kopfzeile (siehe Abschnitt 3.1) Platz zu bieten, zum anderen ist er klein genug, um das Randgewicht nicht zu stark zu verändern, wenn mit einer einfachen, nicht unterstrichenen Kopfzeile gearbeitet wird. Damit ist der voreingestellte Werte in dem meisten Standardfällen ein guter Wert. In einigen Fällen will oder muss man aber die Kopfhöhe genauer den tatsächlichen Erfordernissen anpassen.

Beispiel: Angenommen es soll ein Text mit einem zweizeiligen Kopf erstellt werden. Normalerweise würde dies dazu führen, dass auf jeder Seite eine Warnung „`overfull \vbox`“ von LaTeX ausgegeben würde. Um dies zu verhindern, wird das typearea-Paket angewiesen einen entsprechenden Satzspiegel zu berechnen:

```
\documentclass[a4paper]{article}
```

```
\usepackage[2.1headlines]{typearea}
```

Bei Verwendung einer KOMA-Script-Klasse, kann die Option auch wieder direkt an die Klasse übergeben werden:

```
\documentclass[a4paper,2.1headlines]{scrartcl}
```

Ein Werkzeug, mit dem dann der Inhalt der zweizeiligen Kopfzeile definiert werden kann, ist in Kapitel 4 zu finden.

`\areaset[`*BCOR*`]{`*Breite*`}{`*Höhe*`}`

Bis hier wurde nun eine Menge darüber erzählt, wie man einen guten oder sogar sehr gute Satzspiegel für Standardanwendungen erstellt bzw. wie das typearea-Paket dem Anwender diese Arbeit weitgehend abnimmt, ihm aber gleichzeitig Möglichkeiten der Einflussnahme bietet. Es gibt jedoch auch Fälle, in denen der Textkörper eine bestimmte Größe exakt einhalten soll, ohne dass dabei auf gute Satzspiegelkonstruktion oder auf weitere Nebenbedingungen zu achten ist. Trotzdem sollen die Ränder so gut wie möglich verteilt und dabei gegebenenfalls auch eine Bindekorrektur berücksichtigt werden. Das typearea-Paket bietet hierfür den Befehl `\areaset`, dem man neben der optionalen Bindekorrektur als Parameter die Breite und Höhe des Textbereichs übergibt. Die Ränder und deren Verteilung werden dann automatisch berechnet, wobei gegebenenfalls auch die Paketoptionen `headinclude`, `headexclude`, sowie `footinclude` und `footexclude` berücksichtigt werden.

Beispiel: Angenommen ein Text auf A4-Papier soll genau die Breite von 60 Zeichen in der Typewriter-Schrift haben und exakt 30 Zeilen je Seite besitzen. Dann könnte mit folgender Präambel gearbeitet werden:

```
\documentclass[a4paper,11pt]{article}
\usepackage{typearea}
\newlength{\CharsDX}% Breite von 60 Zeichen
\newlength{\LinesXXX}% Höhe von 30 Zeilen
\settowidth{\CharsDX}{\texttt{1234567890}}
\setlength{\CharsDX}{6\CharsDX}
\setlength{\LinesXXX}{\topskip}
\addtolength{\LinesXXX}{30\baselineskip}
\areaset{\CharsDX}{\LinesXXX}
```

Soll stattdessen ein Gedichtband gesetzt werden, bei dem es nur darauf ankommt, dass der Textbereich genau quadratisch mit einer Seitenlänge von 15 cm ist, wobei ein Binderand von 1 cm zu berücksichtigen ist, so kann dies wie folgt erreicht werden:

```
\documentclass{gedichte}
\usepackage{typearea}
\areaset[1cm]{15cm}{15cm}
```

2.5 Optionen und Makros zur Auswahl des Papierformats

Die LaTeX-Standardklassen unterstützen mit den Optionen `a4paper`, `a5paper`, `b5paper`, `letterpaper`, `legalpaper` und `executivepaper` die Papierformate A4 und A5 aus der ISO-A-Reihe, B5 aus der ISO-B-Reihe, sowie die englischen Formate letter, legal und executive.

```
letterpaper
legalpaper
executivepaper
aXpaper
bXpaper
cXpaper
dXpaper
landscape
\isopaper[Reihe]{Formatnummer}
```

Die drei englischen Formate werden vom typearea-Paket in gleicher Weise unterstützt. Darüber hinaus werden jedoch alle Formate der ISO-A-, ISO-B-, ISO-C- und ISO-D-Reihe durch Ableitung aus den jeweiligen Grundgrößen A0, B0, C0 und D0 unterstützt. Diese können ebenfalls direkt durch entsprechende Optionen `a0paper`, `a1paper` usw. ausgewählt werden. Genau wie bei den Standardklassen ist es mit dem typearea-Paket möglich, durch zusätzliche Verwendung der Paketoption `landscape` das jeweilige Querformat zu wählen.

Alternativ kann beim typearea-Paket die Papiergröße mit Hilfe des Befehls `\isopaper` eingestellt werden. Danach muss allerdings der Satzspiegel mit Hilfe des Befehls `\typearea` oder `\areaset` neu berechnet werden. Ich rate deshalb von der Verwendung des Befehls `\isopaper` ab.

Beispiel: Angenommen, es soll eine Karteikarte im Format ISO-A8 quer bedruckt werden. Dabei sollen die Ränder sehr klein gewählt werden. Außerdem wird auf eine Kopf- und eine Fußzeile verzichtet.

```
\documentclass{article}
\usepackage[headexclude,footexclude,
            a8paper,landscape]{typearea}
\areaset{7cm}{5cm}
\pagestyle{empty}
\begin{document}
\section*{Papieroptionen}
letterpaper, legalpaper, executivepaper, a0paper,
a1paper \dots\ b0paper, b1paper \dots\ c0paper,
c1paper \dots\ d0paper, d1paper \dots
\end{document}
```

`\paperwidth`
`\paperheight`

Für besonders exotische Papierformate, die weder durch die oben angegebenen englischen Formate noch durch eines der Formate der vier ISO-Reihen abgedeckt sind, können direkt die Längen `\paperwidth` und `\paperheight` gesetzt werden. Danach muss allerdings der Satzspiegel für dieses Format mit einem der Befehle `\typearea` oder `\areaset` neu berechnet werden.

Beispiel: Angenommen, es soll auf Endlospapier mit den Maßen $8\frac{1}{4}$ inch $\times$ 12 inch gedruckt werden. Dieses Papierformat wird von typearea nicht direkt unterstützt. Das Papierformat muss daher vor der Berechnung des Satzspiegels definiert werden:

```
\documentclass{article}
\usepackage{typearea}
\setlength{\paperwidth}{8.25in}
\setlength{\paperheight}{12in}
\typearea{1}
```

`dvips`
`pdftex`
`pagesize`

Die oben genannten Mechanismen zur Auswahl des Papierformats haben genaugenommen nur insofern einen Einfluss auf die Ausgabe, dass gewisse interne LaTeX-Maße so gesetzt werden, dass bestimmte Bereiche der Seite, wie Kopf, Textkörper und Fuß so angeordnet und von typearea so berechnet werden, dass sie auf entsprechendes Papier ausgedruckt werden können. Die Spezifikation des DVI-Formats sieht aber an keiner

Stelle Angaben zum Papierformat vor. Wird direkt aus dem DVI-Format in eine Low-Level-Druckersprache wie PCL[5] oder ESC/P2[6] ausgegeben, spielt dies normalerweiser keine Rolle, da auch bei diesen Ausgaben der 0-Bezugspunkt wie bei DVI links oben liegt. Wird aber in Sprachen wie Postscript oder PDF übersetzt, bei denen der 0-Bezugspunkt an anderer Stelle liegt und außerdem das Papierformat in der Ausgabedatei angegeben werden kann, so fehlt diese Information in der DVI-Datei. Als Lösung des Problems verwendet der entsprechende Treiber eine voreingestellte Papiergröße, die der Anwender entweder per Option oder durch entsprechende Angabe in der TEX-Quelldatei verändern kann. Bei Verwendung des DVI-Treibers `dvips` kann diese Angabe in Form einer `\special`-Anweisung erfolgen. Bei pdfTEX werden stattdessen zwei Längen entsprechend gesetzt.

Mit der Option `dvips` wird erreicht, dass die Papiergröße als `\special` in die DVI-Datei geschrieben wird. Dieses `\special` wird beispielsweise von `dvips` ausgewertet. Demgegenüber schreibt die Option `pdftex` die Papiergröße am Anfang des Dokuments in die pdfTEX-Seitenregister, so dass später beim Betrachten der erzeugten PDF-Datei das korrekte Format angegeben wird. Die Option `pagesize` verhält sich flexibler und verwendet je nachdem, ob eine PDF- oder eine DVI-Datei ausgegeben wird den Mechanismus der Option `dvips` oder der Option `pdftex`.

Beispiel: Angenommen es soll ein Dokument sowohl als DVI-Datei verwendet werden, als auch eine Online-Version im PDF-Format erstellt werden. Dann könnte die Präambel beispielsweise so beginnen:

```
\documentclass{article}
\usepackage[a4paper,pagesize]{typearea}
```

Wird nun für die Bearbeitung pdfTEX verwendet *und* die PDF-Ausgabe aktiviert, so werden die beiden Größen `\pdfpagewidth` und `\pdfpageheight` entsprechend gesetzt. Wird jedoch eine DVI-Datei erzeugt – egal ob mit LATEX oder pdfLATEX –, so wird ein `\special` an den Anfang dieser Datei geschrieben.

2.6 Kleinigkeiten ohne direkten Bezug zum Satzspiegel

`\ifpdfoutput{`*Dann-Teil*`}{`*Sonst-Teil*`}`

Manchmal ist es wünschenswert, in einem Dokument abhängig vom Ausgabeformat bestimmte Dinge anders zu machen. Normalerweise verwendet TEX

[5] PCL ist die Druckersprache, die HP für seine Tinten- und Laserdrucker verwendet.

[6] ESC/P2 ist die Druckersprache, die EPSON für seine Nadel-, Tinten- und Laserdrucker erfunden hat.

das Ausgabeformat DVI. Mit pdfTEX ist aber die Wahlmöglichkeit hinzugekommen, statt einer DVI-Datei eine PDF-Datei direkt zu erzeugen. Der Befehl `\ifpdfoutput` stellt eine Verzweigung dar. Wurde die PDF-Ausgabe aktiviert, so wird der *Dann-Teil* ausgeführt. Wurde die PDF-Ausgabe nicht aktiviert oder wird überhaupt kein pdfTEX verwendet, so wird der *Sonst-Teil* ausgeführt.

Beispiel: Bekanntlich gibt pdfLATEX in der Voreinstellung ebenfalls eine DVI-Datei aber keine PDF-Datei aus. Erst, wenn der Zähler `\pdfoutput` einen von 0 verschiedenen Wert erhält, wird auf PDF-Ausgabe umgeschaltet. Nun setzten manche Pakete diese Variable einfach auf 1, wenn sie existiert. Manchmal ist das aber gar nicht erwünscht. Da andererseits `\pdfoutput` unbekannt ist, wenn LATEX statt pdfLATEX verwendet wird, kann man `\pdfoutput` auch nicht einfach generell auf 0 setzen. Eine einfache Lösung ist nun nach dem Laden eines solchen Pakets folgende Zeile:

```
\ifpdfoutput{\pdfoutput=0}{}
```

Selbstverständlich muss dafür das typearea-Paket geladen sein.

2.7 Lokale Einstellungen durch die Datei typearea.cfg

Noch vor der Abarbeitung der Paketoptionen prüft typearea, ob eine Datei `typearea.cfg` existiert und lädt diese gegebenenfalls. Es ist daher möglich, in dieser Konfigurationsdatei beispielsweise zusätzliche Optionen für weitere Papierformate zu definieren.

`\SetDIVList{`*Liste*`}`

Ebenfalls zur Verwendung in dieser Konfigurationsdatei war ursprünglich der Befehl `\SetDIVList` vorgesehen. Bevor die Option `DIVcalc` existierte, war dies die einzige Möglichkeit, für unterschiedliche Schriftgrößen und Papierformate unterschiedliche Voreinstellungen für den *DIV*-Wert zu definieren. Die Liste besteht dabei aus einer Reihe von Werten in geschweiften Klammern. Der Wert ganz links ist für die Schriftgröße 10 pt, der nächste für 11 pt, der dritte für 12 pt usw. vorgesehen. Wird keine Liste mit `\SetDIVList` gesetzt, so entspricht dies `\SetDIVList{{8}{10}{12}}`. Ist für eine Schriftgröße kein Standardwert gesetzt, so wird stattdessen 10 verwendet.

Dieser Befehl sollte nicht mehr verwendet werden. Stattdessen wird empfohlen, einen günstigen Satzspiegel automatisch berechnen zu lassen (siehe Abschnitt 2.4).

2.8 Tips

Insbesondere für die Erstellung von schriftlichen Arbeiten während des Studiums findet man häufig Vorschriften, die einer typografischen Begutachtung nicht nur in keinster Weise standhalten, sondern massiv gegen alle Regeln der Typografie verstoßen. Ursache für solche Regeln sind teilweise typografische Inkompetenz derjenigen, die sie herausgeben, teilweise auch der Ausgangspunkt. Mit einer Schreibmaschine oder eine Textverarbeitung von 1980 ist es ohne erheblichen Aufwand kaum möglich, typografisch perfekte Ergebnisse zu erzielen. Also wurden einst Vorschriften erlassen, die leicht erfüllbar schienen und dem Korrektor trotzdem entgegen kommen. Dazu zählen dann Randeinstellungen, die für einseitigen Druck mit einer Schreibmaschine zu brauchbaren Zeilenlängen führen. Um nicht extrem kurze Zeilen zu erhalten, die durch Flattersatz zudem verschlimmert werden, werden die Rändern schmal gehalten und für Korrekturen stattdessen ein großer Durchschuss in Form von eineinhalbzeiligem Satz vorgeschrieben. Bevor moderne Textverarbeitungssysteme verfügbar wurden, wäre – außer mit TEX – einzeiliger Satz die einzige Alternative gewesen. Dabei wäre dann selbst das Anbringen von Korrekturzeichen schwierig geworden. Als die Verwendung von Computern für die Erstellung schriftlicher Arbeiten üblicher wurde, hat sich manches Mal auch der Spieltrieb des einen oder anderen Studenten gezeigt, der durch Verwendung einer Schmuckschrift, seine Arbeit aufpeppen und so eine bessere Note mit weniger Einsatz herausschinden wollte. Nicht bedacht hat er dabei, dass solche Schriften schlechter zu lesen und deshalb für den Zweck ungeeignet sind. Damit hielten zwei Brotschriften Einzug in die Vorschriften, die weder zusammenpassen noch im Falle von Times wirklich gut geeignet sind. Times ist eine relativ enge Schrift, die Anfang des 20. Jahrhunderts speziell für schmale Spalten im englischen Zeitungssatz entworfen wurde. In modernen Schnitten ist dies etwas entschärft. Dennoch passt die häufig vorgeschriebene Times meist nicht zu den gleichzeitig gegebenen Randvorgaben.

LATEX setzt bereits von sich aus mit ausreichendem Durchschuss. Gleichzeitig sind die Ränder auch bei sinnvollen Zeilenlängen groß genug, um Platz für Korrekturen zu bieten. Dabei wirkt die Seite trotz einer Fülle von Text großzügig angelegt.

Teilweise sind die typografisch mehr als fragwürdigen Satzvorschriften mit LATEX auch außerodentlich schwierig umzusetzen. So kann eine feste Anzahl von „Anschlägen“ nur dann eingehalten werden, wenn keine proportionale Schrift verwendet wird. Es gibt nur wenig gute nichtproportionale Schriften. Kaum ein Text, der mit einer derartigen Schrift gesetzt ist, wirkt wirklich gut. So wird häufig versucht, durch ausladende Ligaturen beispielsweise beim kleinen „i“ oder „l“ die unterschiedliche Breite der Zeichen auszugleichen. Dies kann nicht funktionieren. Im Ergebnis wirkt der Text unruhig oder zerrissen. Außerdem verträgt sich eine solche Schrift kaum mit dem im deutschen Sprachraum üblichen und allgemein vorzuziehenden Blocksatz. Gewisse Vorgaben können daher bei Verwendung von LATEX nur ignorriert oder großzügig ausgelegt werden, etwa indem man „60 Anschläge pro Zeile“ nicht als feste, sondern als durchschnittliche oder maximale Angabe interpretiert.

Wie ausgeführt, sind Satzvorschriften meist dazu gedacht, ein brauchbares Ergebnis zu erhalten, auch wenn der Ausführende selbst nicht weiß, was dabei zu beachten ist. Brauchbar bedeutet häufig: lesbar und korrigierbar. Nach meiner Auffassung wird ein mit LaTeX und dem typearea-Paket gesetzter Text, bezüglich des Satzspiegels diesen Anforderungen von vornherein gerecht. Wenn Sie also mit Vorschriften konfrontiert sind, die offensichtlich erheblich davon abweichen, so empfehle ich, dem Betreuer einen Textauszug vorzulegen und nachzufragen, ob es gestattet ist, die Arbeit trotz der Abweichungen in dieser Form zu liefern. Gegebenenfalls kann durch Veränderung der Option `DIV` der Satzspiegel moderat angepasst werden. Von der Verwendung von `\areaset` zu diesem Zweck rate ich jedoch ab. Schlimmstenfalls verwenden Sie das nicht zu KOMA-Script gehörende geometry-Paket (siehe [Ume00]) oder verändern Sie die Satzspiegelparameter von LaTeX selbst. Die von typearea ermittelten Werte finden Sie in der `log`-Datei Ihres Dokuments. Damit sollten moderate Anpassungen möglich sein. Achten Sie jedoch unbedingt darauf, dass die Proportionen des Textbereichs mit denen der Seite unter Berücksichtigung der Bindekorrektur annähernd übereinstimmen.

Sollte es unbedingt erforderlich sein, den Text eineinhalbzeilig zu setzen, so definieren Sie keinesfalls `\baselinestretch` um. Dieses Vorgehen wird zwar allzu häufig empfohlen, ist aber seit der Einführung von LaTeX 2_ε im Jahre 1994 obsolet. Verwenden Sie schlimmstenfalls den Befehl `\linespread`. Ich empfehle das Paket setspace, das nicht zu KOMA-Script gehört (siehe [Tob00]). Auch sollten Sie typearea nach der Umstellung des Zeilenabstandes den Satzspiegel für diesen Abstand berechnen lassen jedoch für den Titel, besser auch für die Verzeichnisse – sowie das Literaturverzeichnis und den Index – wieder auf normalen Satz umschalten. Das setspace-Paket bietet dafür eine spezielle Umgebung und eigene Befehle.

Das typearea-Paket berechnet auch bei der Option `DVIcalc` einen sehr großzügigen Textbereich. Viele konservative Typografen werden feststellen, dass die resultierende Zeilenlänge noch zu groß ist. Der berechnete *DIV*-Wert ist ebenfalls in der `log`-Datei zum jeweiligen Dokument zu finden. Sie können also leicht nach dem ersten LaTeX-Lauf einen kleineren Wert wählen.

Nicht selten wird mir die Frage gestellt, warum ich eigentlich kapitelweise auf einer Satzspiegelberechnung herumreite, während es sehr viel einfacher wäre, nur ein Paket zur Verfügung zustellen, mit dem man die Ränder wie bei einer Textverarbeitung einstellen kann. Oft wird auch behauptet, ein solches Paket wäre ohnehin die bessere Lösung, da jeder selbst wisse, wie gute Ränder zu wählen seien, und die Ränder von KOMA-Script wären ohnehin nicht gut. Ich erlaube mir zum Abschluss dieses Kapitels ein passendes Zitat aus [WF00]:

> *Das Selbermachen ist längst üblich, die Ergebnisse oft fragwürdig, weil Laien-Typografen nicht sehen, was nicht stimmt und nicht wissen können, worauf es ankommt. So gewöhnt man sich an falsche*

> *und schlechte Typografie.* [...] *Jetzt könnte der Einwand kommen, Typografie sei doch Geschmacksache. Wenn es um Dekoration ginge, könnte man das Argument vielleicht gelten lassen, da es aber bei Typografie in erster Linie um Information geht, können Fehler nicht nur stören, sondern sogar Schaden anrichten.*

2.9 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- Frank Neukam
- **Markus Kohm** <Markus.Kohm@gmx.de>
- Axel Sommerfeldt

3 Die Hauptklassen scrbook, scrreprt, scrartcl

Die Hauptklassen des KOMA-Script-Pakets sind als Äquivalent zu den LaTeX-Standardklassen angelegt. Das bedeutet, dass zu den drei Standardklassen book, report und article im KOMA-Script-Paket Entsprechungen zu finden sind. Daneben ist auch für die Standardklasse letter eine Entsprechung vorhanden. Der Briefklasse in KOMA-Script ist jedoch ein eigenes Kapitel gewidmet, da sie sich von den drei Hauptklassen grundsätzlich unterscheidet (siehe Kapitel 6). Die Namen der KOMA-Script-Klassen sind aus dem Präfix „`scr`" und den gekürzten Namen der Standardklassen zusammengesetzt. Um die Länge des Namens auf acht Buchstaben zu beschränken, werden dabei notfalls von hinten her die Vokale weggelassen. Eine Gegenüberstellung zeigt Tabelle 3.1. Dort sind auch die Script-Stile für LaTeX2.09 aufgeführt, die Eingang in KOMA-Script gefunden haben.

Die einfachste Möglichkeit, an Stelle einer Standardklasse eine KOMA-Script-Klasse zu verwenden, ist die Ersetzung des Klassennamens in der Anweisung `\documentclass` entsprechend Tabelle 3.1. Normalerweise sollte das Dokument danach ebenso fehlerfrei mit LaTeX bearbeitbar sein, wie dies vor der Ersetzung der Fall war. Das Aussehen ändert sich allerdings. Außerdem gibt es mit den KOMA-Script-Klassen weitere Möglichkeiten und Optionen, die in den nachfolgenden Abschnitten beschrieben sind.

Standard-Klasse	KOMA-Script-Klasse	Script-Stil (LaTeX2.09)
article	scrartcl	`script_s`
report	scrreprt	`script`
book	scrbook	`script`
letter	scrlettr	`script_l`

Tabelle 3.1: Gegenüberstellung der Standardklassen, der KOMA-Script-Klassen und der Script-Stile

3.1 Die Optionen

Dieser Abschnitt behandelt alle Klassenoptionen der drei Hauptklassen. Die Mehrzahl der Optionen ist in genau der gleichen Form auch in den Standardklassen zu finden. Da die Erfahrung jedoch zeigt, dass viele Optionen der Standardklassen unbekannt sind, wurde deren Beschreibung hier aufgenommen. Dies stellt eine Abweichung von dem Grundsatz dar, im `scrguide` nur die Dinge zu beschreiben, die abweichend von den Standardklassen implementiert sind.

In Tabelle 3.2 finden Sie diejenigen Optionen aufgeführt, die bei mindestens einer KOMA-Script-Klasse voreingestellt sind. Dabei ist für jede der drei KOMA-Script-Hauptklassen angegeben, ob die Option voreingestellt ist oder nicht, oder ob sie für die jeweilige Klasse überhaupt nicht definiert ist. Eine nicht definierte Option kann natürlich weder voreingestellt sein, noch vom Anwender gewählt werden.

Option	scrbook	scrreprt	scrartcl
`11pt`	Voreinstellung	Voreinstellung	Voreinstellung
`a4paper`	Voreinstellung	Voreinstellung	Voreinstellung
`abstractoff`	*nicht definiert*	Voreinstellung	Voreinstellung
`bigheadings`	Voreinstellung	Voreinstellung	Voreinstellung
`final`	Voreinstellung	Voreinstellung	Voreinstellung
`footnosepline`	Voreinstellung	Voreinstellung	Voreinstellung
`headnosepline`	Voreinstellung	Voreinstellung	Voreinstellung
`nochapterprefix`	Voreinstellung	Voreinstellung	*nicht definiert*
`notitlepage`			Voreinstellung
`onecolumn`	Voreinstellung	Voreinstellung	Voreinstellung
`oneside`		Voreinstellung	Voreinstellung
`openany`		Voreinstellung	Voreinstellung
`openright`	Voreinstellung		
`parindent`	Voreinstellung	Voreinstellung	Voreinstellung
`tablecaptionbelow`	Voreinstellung	Voreinstellung	Voreinstellung
`titlepage`	Voreinstellung	Voreinstellung	
`twoside`	Voreinstellung		

Tabelle 3.2: Die voreingestellten Optionen der KOMA-Script-Klassen

Lassen Sie mich der Erläuterung der Optionen noch eine Bemerkung vorausschicken. Oft ist man sich am Anfang eines Dokuments unsicher, welche Einstellungen für dieses konkret zu wählen sind. Bei einigen Optionen, wie der Auswahl des Papierformats, mögen sie bereits vorab feststehen. Aber schon die Frage nach dem *DIV*-Wert für den Satzspiegel könnte im Voraus schwer zu beantworten sein. Andererseits sollten diese Angaben für die Haupttätigkeiten des Autors: Entwurf der Gliederung, Schreiben des Textes, Zusammenstellen von Abbildungen, Tabellen und Verzeichnissen zunächst auch unerheblich sein. Konzentrieren Sie sich als Autor erst einmal auf den Inhalt. Wenn der dann steht, können Sie sich um die Feinheiten der Form kümmern. Neben der Auswahl der Optionen gehören dazu dann auch Dinge wie die Korrektur der Trennung und möglicherweise dezente Eingriffe in den Seitenumbruch oder die Verteilung von Abbildungen und Tabellen. Als Beispiel hierfür sei Tabelle 3.2 genannt, die ich mehrfach vom Anfang des Abschnitts an das Ende verschoben habe und umgekehrt. Die Wahl der tatsächlichen Position wird erst bei Fertigstellung des Dokuments feststehen.

3.1.1 Satzspiegeloptionen

Während bei den Standardklassen der Satzspiegel allein durch das Laden der Optionendateien `size10.clo`, `size11.clo`, `size12.clo` beziehungsweise `bk10.clo`, `bk11.clo`, `bk12.clo` und festen, in den Klassen definierten Werten bestimmt wird, verwenden die KOMA-Script-Klassen keinen festen Satzspiegel, sondern einen, der von Papierformat und Schriftgröße abhängt. Dazu stützen sich alle drei Hauptklassen auf das typearea-Paket (siehe Kapitel 2). Das Paket wird von den Klassen automatisch geladen. Es ist also nicht notwendig, es mit `\usepackage{typearea}` selbst einzuladen.

`letterpaper`
`legalpaper`
`executivepaper`
`a`*X*`paper`
`b`*X*`paper`
`c`*X*`paper`
`d`*X*`paper`
`landscape`

Die grundlegenden Optionen zur Auswahl des Papierformats werden nicht direkt von den Klassen ausgeführt. Stattdessen werden diese als globale Optionen automatisch vom typearea-Paket behandelt (siehe Abschnitt 2.4). Die Optionen `a5paper`, `a4paper`, `letterpaper`, `legalpaper` und `executivepaper` entsprechen den gleichnamigen Optionen der Standardklassen und ergeben die gleichen Papierformate. Der aus dem jeweiligen Format berechnete Satzspiegel ist dann jedoch ein anderer.

Tatsächlich werden die Optionen für das A-, B-, C- oder D-Format vom typearea-Paket nicht ausgewertet, weil es sich um globale Optionen handelt, sondern nur deshalb, weil die KOMA-Script-Klassen sie explizit an das typearea-Paket weiterreichen. Dies hat seine Gründe in der Implementierung der Optionen im typearea-Paket.

`BCOR`*Korrektur*
`DIV`*Faktor*
`DIVcalc`
`DIVclassic`
Wert`headlines`

Die Optionen für den Divisor und die Bindekorrektur werden direkt an das typearea-Paket weitergeleitet (siehe Abschnitt 2.4). Dies geschieht in Abweichung zu den Standardklassen, bei denen eine entsprechende Weiterleitung nicht stattfindet. Dies gilt in gleicher Weise für die Option zur Einstellung der Anzahl der Kopfzeilen.

3.1.2 Layoutoptionen

In diesem Unterabschnitt werden alle Optionen zusammengefasst, die sich im entfernten Sinne auf das Layout und nicht nur auf den Satzspiegel auswirken. Genaugenommen sind natürlich alle Satzspiegeloptionen (siehe Abschnitt 3.1.1) Layoutoptionen. Teilweise gilt dies auch umgekehrt.

`oneside`
`twoside`

Diese beiden Optionen funktionieren genau wie bei den Standardklassen. Bei der Option `oneside` wird ein einseitiges Layout mit einem einseitigen Satzspiegel gewählt. Dies bedeutet insbesondere auch, dass normalweise mit einem flatternden unteren Rand gearbeitet wird.

Bei der Option `twoside` wird ein doppelseitiges Layout mit einem doppelseitigen Satzspiegel gewählt. Dazu wird der LaTeX-Befehl `\flushbottom` ausgeführt, der dafür sorgt, dass am Seitenende umbrochener Seiten kein variabler Leerraum zu finden ist. Will man stattdessen einen flatternden unteren Rand, so kann man dies mit Hilfe des LaTeX-Befehls `\raggedbottom` erreichen.

`onecolumn`
`twocolumn`

Diese beiden Optionen funktionieren genau wie bei den Standardklassen. Mit ihrer Hilfe wird zwischen einem einspaltigen und einem zweispaltigen Layout umgeschaltet. Die in LaTeX eingebauten Möglichkeiten für mehrspaltigen Satz reichen oft nur für einfache Aufgaben aus. Es gibt jedoch das Standardpaket multicol, mit dem sehr viel flexiblerer Mehrspaltensatz möglich ist (siehe [Mit00]).

`openany`
`openright`

scrbook, scrreprt

Diese beiden Optionen funktionieren genau wie bei den Standardklassen. Sie haben Auswirkungen auf die Auswahl der Seite, auf der ein Kapitel beginnen kann und existieren daher bei scrartcl nicht, da dort die oberste Gliederungsebene unter dem Teil der Abschnitt ist. Die Gliederungsebene Kapitel gibt es bei scrartcl nicht.

Ein Kapitel beginnt immer auf einer neuen Seite. Mit der Option openany kann dies jede neue Seite sein. Bei Wahl der Einstellung openright wird jedoch zwingend eine neue rechte Seite vorgeschrieben. Dazu wird gegebenenfalls eine linke Seite eingefügt. Diese leeren Seiten werden durch implizite Ausführung des LaTeX-Befehls \cleardoublepage erzeugt.

Da zwischen rechten und linken Seiten nur im doppelseitigen Layout unterschieden wird, hat die Option openright im einseitigen Layout keinerlei Auswirkungen. Sie sollten sie deshalb zusammen mit der Option twoside einsetzen.

```
cleardoublestandard
cleardoubleplain
cleardoubleempty
```

Will man, dass mit der \cleardoublepage-Anweisung eingefügte Leerseiten keinen Kolumnentitel, sondern nur eine Seitenzahl oder auch weder Kolumnentitel noch Seitenzahl enthalten, so bleibt bei den Standardklassen nur, die Anweisung entsprechend umzudefinieren. KOMA-Script bietet Optionen, um dem Anwender dies abzunehmen. Mit der Option cleardoublestandard hat man das Standardverhalten von \cleardoublepage. Bei Verwendung der Option cleardoubleplain wird die leere, linke Seite hingegen mit dem Seitenstil plain ausgegeben. Bei Verwendung der Option cleardoubleempty findet entsprechend der Seitenstil empty Anwendung. Die Seitenstile werden in Abschnitt 3.2 näher erläutert.

```
headsepline
headnosepline
footsepline
footnosepline
```

Wird unter Kolumnentiteln eine horizontale Linie gewünscht, so lässt sich diese mit der Option headsepline einschalten. Die Option headnosepline stellt die Umkehrung dar. Bei den Seitenstilen empty und plain hat dies selbstverständlich keine Auswirkung, da hier auf einen Seitenkopf ausdrücklich verzichtet werden soll. Typografisch betrachtet hat eine solche Linie immer die Auswirkung, dass der Kopf optisch näher an den Text heranrückt. Dies bedeutet nun nicht, dass der Kopf räumlich weiter vom Textkörper weggerückt werden müsste. Stattdessen sollte der Kopf dann bei der Berechnung des Satzspiegels als zum Textkörper gehörend betrachtet werden. Dies wird bei KOMA-Script dadurch erreicht, dass bei Verwendung der Klassenoption headsepline automatisch die Paketoption headinclude an das typearea-Paket weitergereicht wird.

Für eine Trennlinie über dem Seitenfuß gibt es die analog funktionierenden Optionen `footsepline` und `footnosepline`. Bei Aktivierung der Trennlinie mit `footsepline` wird automatisch die Option `footinclude` an das typearea-Paket weitergereicht. Im Gegensatz zu `headsepline` wirkt sich die Option `footsepline` auch beim Seitenstil `plain` aus, da `plain` eine Seitenzahl im Fuß ausgibt.

`titlepage`
`notitlepage`

Die beiden Optionen funktionieren genau wie bei den Standardklassen. Mit der Option `titlepage` wird erreicht, dass für die gesamte Titelei eigene Seiten verwendet werden. Diese Seiten werden innerhalb von `titlepage`-Umgebungen gesetzt und erhalten somit normalerweise weder Seitenkopf noch Seitenfuß. Bei KOMA-Script wurde die Titelei gegenüber den Standardklassen stark erweitert (siehe dazu Abschnitt 3.5.2).

Demgegenüber wird mit der Option `notitlepage` erreicht, dass ein sogenannter *in-page* Titel gesetzt wird. Das heißt, die Titelei wird lediglich speziell hervorgehoben, auf der Titelseite kann aber nachfolgend weiteres Material, beispielsweise eine Zusammenfassung oder ein Abschnitt gesetzt werden.

`chapterprefix`
`nochapterprefix`

scrbook, scrreprt

Bei den Standardklassen book und report werden Kapitelüberschriften in der Form ausgegeben, dass zunächst in einer Zeile „Kapitel“[1] gefolgt von der Kapitelnummer steht. Erst ab der nächsten Zeile wird dann die Überschrift in linksbündigem Flattersatz ausgegeben. Bei KOMA-Script kann dieses Verhalten mit der Klassenoption `chapterprefix` ebenfalls erreicht werden. Voreingestellt ist jedoch das Verhalten der Option `nochapterprefix`. Die Optionen wirken sich außerdem auf das Aussehen der automatischen Kolumnentitel für Kapitel aus (siehe Abschnitt 3.2).

`appendixprefix`
`noappendixprefix`

scrbook, scrreprt

Zuweilen kommt es vor, dass man die Kapitelüberschriften im Hauptteil durchaus in der einfachen Form von `nochapterprefix` setzten möchte. Gleichzeitig sollen die Überschriften im Anhang jedoch davon abweichend mit einer Präfixzeile, „Anhang“ gefolgt vom Buchstaben des Anhangs, versehen werden. Dies

[1] Bei Verwendung einer anderen Sprache als Deutsch wird „Kapitel“ selbstverständlich in der jeweiligen Sprache gesetzt.

ist mit der Klassenoption `appendixprefix` möglich. Da sich jedoch dadurch ein inkonsistentes Layout ergibt, rate ich von der Verwendung ab.

Lediglich aus Gründen der symmetrischen Vollständigkeit gibt es auch die Umkehrungsoption `noappendixprefix`. Mir ist jedoch keine sinnvolle Anwendung dieser Option bekannt.

```
parskip
parskip*
parskip+
parskip-
halfparskip
halfparskip*
halfparskip+
halfparskip-
parindent
```

Die Standardklassen setzen Absätze normalerweise mit Absatzeinzug und ohne Absatzabstand. Bei Verwendung eines normalen Satzspiegels, wie ihn typearea bietet, ist dies die vorteilhafteste Absatzauszeichnung. Würde man ohne Einzug und Abstand arbeiten, hätte der Leser als Anhaltspunkt nur die Länge der letzten Zeile. Im Extremfall kann es sehr schwer sein, zu erkennen, ob eine Zeile voll ist oder nicht. Desweiteren stellt der Typograf fest, dass die Auszeichnung des Absatzendes am Anfang der nächsten Zeile leicht vergessen ist. Demgegenüber ist eine Auszeichnung am Absatzanfang einprägsamer. Der Absatzabstand hat den Nachteil, dass er in verschiedenem Zusammenhang leicht verloren geht. So wäre nach einer abgesetzten Formel nicht mehr festzustellen, ob der Absatz fortgesetzt wird oder ein neuer beginnt. Auch am Seitenanfang müsste zurückgeblättert werden, um feststellen zu können, ob mit der Seite auch ein neuer Absatz beginnt. All diese Probleme sind beim Absatzeinzug nicht gegeben. Eine Kombination von Absatzeinzug und Absatzabstand ist wegen der übertriebenen Redundanz abzulehnen. Der Einzug alleine ist deutlich genug. Der einzige Nachteil des Absatzeinzuges liegt in der Verkürzung der Zeile. Damit gewinnt der Absatzabstand bei ohnehin kurzen Zeilen, etwa im Zeitungssatz, seine Berechtigung.

Unabhängig von obigen Erläuterungen wird hin und wieder ein Layout mit Absatzabstand an Stelle des Absatzeinzugs gefordert. KOMA-Script bietet hierfür mit `parskip`, `parskip-`, `parskip*`, `parskip+` sowie `halfparskip`, `halfparskip-`, `halfparskip*` und `halfparskip+` eine ganze Reihe von Optionen.

Die vier `parskip`-Optionen setzen jeweils einen Absatzabstand von einer Zeile. Die vier `halfparskip`-Optionen verwenden nur eine halbe Zeile. Um zu verhindern, dass beispielsweise beim Seitenwechsel ein Absatzwechsel unerkannt bleibt, wird bei jeweils drei Varianten dafür gesorgt, dass die letzte Zeile eines Absatzes nicht komplett gefüllt wird. Bei der Variante ohne Plus und Stern bleibt mindestens ein Leeraum von 1 em. Bei der Plus-Variante bleibt mindestens ein Drittel einer normalen Zeile und bei der Stern-Variante mindestens ein Viertel einer normalen Zeile frei. Bei der Minus-Variante werden keine Vorkehrungen für die letzte Zeile eines Absatzes getroffen.

Alles acht Optionen zur Auswahl eines Absatzabstandes verändern außerdem den Abstand vor, nach und innerhalb von Listenumgebungen. Dadurch wird verhindert, dass diese Umgebungen oder Absätze innerhalb dieser Umgebungen stärker vom Text abgesetzt werden als die Absätze des normalen Textes voneinander. Darüber hinaus wird dafür gesorgt, dass im Inhalts-, Abbildungs- und Tabellenverzeichnis ohne zusätzlichem Absatzabstand gearbeitet wird.

Voreingestellt ist bei KOMA-Script das Verhalten der Option `parindent`. Hierbei gibt es keinen Absatzabstand, sondern einen Absatzeinzug von 1 em.

3.1.3 Schriftoptionen

Schriftoptionen sind Optionen, die sich auf die Größe der Grundschrift oder der Schrift einzelner Teile auswirken. Theoretisch wären auch Optionen, die sich auf die Schriftart auswirken, Schriftoptionen. Solche Optionen gibt es bei KOMA-Script derzeit jedoch nicht.

```
10pt
11pt
12pt
Xpt
```

Die Optionen `10pt`, `11pt` und `12pt` entsprechen den Standardoptionen. Im Gegensatz zu den Standardklassen kann bei KOMA-Script jedoch auch eine andere Schriftgröße eingestellt werden. Da bei LaTeX jedoch nur Klassenoptionsdateien für 10 pt, 11 pt und 12 pt mitgeliefert werden und KOMA-Script keine entsprechenden Dateien mitliefert, müssen Sie selbst für diese Dateien sorgen. Das Paket extsizes hilft Ihnen dabei weiter (siehe [Kil99]). Bei sehr großen Schriften kann es jedoch zu einem arithmetischen Überlauf bei der Satzspiegelberechnung im typearea-Paket kommen.

```
smallheadings
normalheadings
bigheadings
```

Die Überschriften werden sowohl bei den Standardklassen als auch bei KOMA-Script normalerweise recht groß gesetzt. Dies gefällt nicht jedem und wirkt insbesondere bei kleinen Papiergrößen oft störend. Daher stehen bei KOMA-Script neben den mit der Option `bigheadings` sehr groß eingestellen Überschriften die beiden Optionen `normalheadings` und `smallheadings` zur Verfügung, mit denen man insgesamt kleinere Überschriften erhält. Die Abstände vor und nach Kapitelüberschriften werden von diesen Optionen ebenfalls beeinflusst. Auf Kapitelüberschriften wirken sich außerdem die Layoutoptionen `chapterprefix` und `nochapterprefix` sowie im Anhang `appendixprefix` und `noappendixprefix` aus, die in Abschnitt 3.1.2 beschrieben worden sind. scrbook, scrreprt

3.1.4 Inhaltsverzeichnisoptionen

Bei KOMA-Script gibt es mehrere Optionen, die sich auf den Inhalt des Inhaltsverzeichnisses auswirken. Die Form des Inhaltsverzeichnisses ist hingegen fest und kann nicht durch Optionen beeinflusst werden.

```
liststotoc
idxtotoc
bibtotoc
bibtotocnumbered
```

Normalerweise erscheinen Tabellen- und Abbildungsverzeichnis, Index und Literaturverzeichnis nicht im Inhaltsverzeichnis. Im klassischen Buchdruck wird auf entsprechende Einträge bewusst verzichtet, weil stillschweigend davon ausgegangen wird, dass Tabellen- und Abbildungsverzeichnis nach dem Inhaltsverzeichnis, der Index ganz am Ende und das Literaturverzeichnis vor dem Index zu finden sind, wenn die entsprechenden Teile überhaupt verwendet werden. Bücher, die all diese Verzeichnisse aufweisen, sind außerdem häufig mit diesen praktischen Bändern gebunden, die man an den entsprechenden Stellen ins Buch legen kann, so dass besagte Verzeichnisse höchstens einmal gesucht werden müssen.

Neuerdings ist es fast üblich geworden, dass Tabellen- und Abbildungsverzeichnis sowie das Literaturverzeichnis, seltener der Index im Inhaltsverzeichnis zu finden sind. Dies hat sicher auch mit der neuen Mode zu tun, Abbildungs- und Tabellenverzeichnis ans Buchende zu stellen. Beide Verzeichnisse haben von Aufbau und Intention eine deutliche Ähnlichkeit mit dem Inhaltsverzeichnis. Daher betrachte ich die Entwicklung skeptisch. Da es keinen Sinn hat, nur das Tabellen- oder nur das Abbildungsverzeichnis ohne das jeweils andere ins Inhaltsverzeichnis aufzunehmen, gibt es nur eine Option `liststotoc`, mit der beide Verzeichnisse gemeinsam ins Inhaltsverzeichnis aufgenommen werden. Dabei werden auch Verzeichnisse berücksichtigt, die mit Hilfe des float-Pakets ab Version 1.2e erstellt werden (siehe [Lin01]). Als Verzeichnisse, die den Inhalt anderer Abschnitte des Werks aufführen, erhalten Tabelle-, Abbildung- und die mit dem float-Paket erzeugten Verzeichnisse grundsätzlich keine Nummer.

Der Index erhält mit der Option `idxtotoc` einen Eintrag ins Inhaltsverzeichnis. Da der Index ebenfalls nur Verweise auf den Inhalt anderer Abschnitte enthält, wird auch er grundsätzlich nicht nummeriert.

Das Literaturverzeichnis stellt eine etwas andere Art von Verzeichnis dar. Hier wird nicht der Inhalt des vorliegenden Werks aufgelistet, sondern auf externe Inhalte verwiesen. Mit dieser Begründung könnte man argumentieren, dass das Literaturverzeichnis ein eigenes Kapitel bzw. einen eigenen Abschnitt darstellt, und somit eine Nummer verdient. Die Option `bibtotocnumbered` führt genau dazu, einschließlich des dann fälligen Eintrags in das Inhaltsverzeichnis. Ich selbst bin der Meinung, dass bei dieser Argumentation auch ein klassisches, kommentiertes Quellenverzeichnis ein eigenes Kapitel wäre. Außerdem ist das Literaturverzeichnis letztlich nichts, was man selbst geschrieben hat. Deshalb verdient es allenfalls einen nicht nummerierten Eintrag ins Inhaltsverzeichnis, was mit der Option `bibtotoc` erreicht wird.

3.1.5 Formatierungsoptionen

Formatierungsoptionen sind alle Optionen, welche die Form oder Formatierung einer Ausgabe beeinflussen und nicht in einen anderen Abschnitt eingeordnet werden können. Es sind also sozusagen die *sonstigen Optionen*.

```
abstracton
abstractoff
```

scrreprt, scrartcl

Bei den Standardklassen setzt die `abstract`-Umgebung noch den zentrierten Text „Zusammenfassung“ vor die Zusammenfassung. Früher war dies durchaus üblich. Inzwischen sind wir durch das Zeitunglesen darin geübt, einen entsprechend hervorgehobenen Text am Anfang eines Artikels oder Berichts als Zusammenfassung zu erkennen. Dies gilt umso mehr, wenn dieser Text noch vor dem Inhaltsverzeichnis steht. Zudem verwundert es, wenn ausgerechnet diese Überschrift klein und zentriert ist. KOMA-Script bietet mit den Optionen `abstracton` und `abstractoff` die Möglichkeit, die Überschrift über der Zusammenfassung ein- oder auszuschalten.

Bei Büchern wird in der Regel eine andere Art der Zusammenfassung verwendet. Dort setzt man ein entsprechendes Kapitel an den Anfang oder Ende des Werks. Oft wird diese Zusammenfassung entweder mit der Einleitung oder einem weiteren Ausblick verknüpft. Daher gibt es bei scrbook überhaupt keine `abstract`-Umgebung. Bei Berichten im weiteren Sinne, etwa einer Studien- oder Diplomarbeit, ist ebenfalls eine Zusammenfassung in dieser Form zu empfehlen.

```
pointednumbers
pointlessnumbers
```

Nach DUDEN steht in Gliederungen, in denen ausschließlich arabische Ziffern für die Nummerierung verwendet werden, am Ende der Gliederungsnummern kein abschließender Punkt (siehe [DUD96, R 3]). Wird hingegen innerhalb der Gliederung auch mit römischen Zahlen oder Groß- oder Kleinbuchstaben gearbeitet, so steht am Ende aller Gliederungsnummer ein abschließender Punkt (siehe [DUD96, R 4]). In KOMA-Script ist ein Automatismus eingebaut, der diese etwas komplexe Regel zu erfüllen versucht. Der Automatismus wirkt sich so aus, dass normalerweise bei Verwendung des Gliederungsbefehls `\part` oder eines Anhangs (`\appendix`) auf Gliederungsnummer mit abschließendem Punkt umgeschaltet wird. Diese Information wird in der aux-Datei gespeichert und wirkt sich dann beim nächsten LaTeX-Lauf auf das gesamte Dokument aus.

Manchmal versagt der Automatismus zum Setzen oder Weglassen des abschließenden Punktes in der Gliederungsnummer oder andere Sprachen sehen andere Regeln vor. Deshalb ist es mit der Option `pointednumbers` möglich, den Punkt manuell einzuschalten, oder mit der Option `pointlessnumbers` zu verbieten.

Es ist zu beachten, dass der Automatismus immer erst für den nächsten LaTeX-Lauf die Verwendung des abschließenden Punktes ein- oder ausschaltet. Bevor also versucht wird, die korrekte Darstellung über Verwendung einer der Optionen zu erzwingen, sollte grundsätzlich ein weiterer LaTeX-Lauf ohne Dokumentänderung durchgeführt werden.

Richtig, die korrekten Name für diese Optionen wären `dottednumbers` und `dotlessnumbers` oder ähnliches. Wie das Leben so spielt, war mir die Bedeutung der statt dessen gewählten Namen vor ein paar Jahren bei der Implementierung nicht klar.

`leqno`

Gleichungen werden normalerweise auf der rechten Seite nummeriert. Mit Hilfe der Standardoption `leqno` wird die Standardoptionsdatei `leqno.clo` geladen. Dadurch erfolgt die Nummerierung von Gleichungen links.

`fleqn`

Gleichungen werden normalerweise horizontal zentriert ausgegeben. Mit Hilfe der Standardoption `fleqn` wird die Standardoptionsdatei `fleqn.clo` geladen. Dadurch erfolgt die Ausgabe von Gleichungen linksbündig.

`tablecaptionbelow`
`tablecaptionabove`

Wie in Abschnitt 3.5.6 erläutert wird, verhält sich `\caption` bei Abbildungen immer wie `\captionbelow`. Bei Tabellen ist das Verhalten hingegen von zwei Optionen abhängig. Mit der Voreinstellung `tablecaptionbelow` verhält sich `\caption` auch bei Tabellen wie `\captionbelow`. Mit der Option `tablecaptionabove` verhält sich `\caption` jedoch wie `\captionabove`.

Bitte beachten Sie, dass bei Verwendung des float-Pakets die Optionen float tablecaptionbelow und tabelcaptionabove nicht mehr funktionieren, sobald Sie \refloatstyle auf Tabellen anwenden. Näheres zum float-Paket und \refloatstyle entnehmen Sie bitte [Lin01].

origlongtable

longtable

Beim Paket longtable (siehe [Car98]) werden Tabellenüberschriften intern mit dem Befehl \LT@makecaption gesetzt. Damit diese Tabellenüberschriften zu denen normaler Tabellen passen definieren die KOMA-Script-Klassen diese Anweisung normalerweise um. Siehe hierzu auch Abschnitt 3.5.6. Diese Umdefinierung erfolgt mit Hilfe von \AtBeginDocument erst während \begin{document}. Ist das Paket caption2 (siehe [Som95]) geladen, unterbleibt die Umdefinierung in KOMA-Script um caption2 nicht in die Quere zu kommen.

Falls die Tabellenüberschriften des longtable-Pakets von den KOMA-Script-Klassen nicht umdefiniert werden sollen, kann die Option origlongtable gesetzt werden.

openbib

Die Standardoption openbib schaltet auf eine alternative Formatierung des Literaturverzeichnisses um. Dabei wird zum einen die erste Zeile der Literaturangabe, die normalerweise den Autor enthält, weniger stark eingerückt. Zum anderen wird der Befehl \newblock so umdefiniert, dass er einen Absatz einfügt. Ohne die Option fügt \newblock lediglich einen dehnbaren horizontalen Abstand ein.

draft
final

Die beiden Standardoptionen draft und final werden normalerweise verwendet, um zwischen Dokumenten im Entwurfsstadium und fertigen Dokumenten zu unterscheiden. Insbesondere werden mit der Option draft kleine schwarze Kästchen aktiviert, die im Falle von überlangen Zeilen am Zeilenende ausgegeben werden. Diese Kästchen erleichtern dem ungeübten Auge, Absätze ausfindig zu machen, die manueller Nachbearbeitung bedürfen. Demgegenüber erscheinen mit der Option final keine solchen Kästchen.

Die beiden Optionen werden übrigens auch von verschiedenen Paketen, ausgewertet und beeinflussen deren Eigenschaften. So verzichtet das Paket graphics oder graphicx bei Verwendung der Option draft auf die Ausgabe der Grafiken. Stattdessen werden lediglich Rahmen in der entsprechenden Größe und die Dateinamen der Grafiken ausgegeben (siehe [Car99b]).

3.2 Seitenstil

```
\pagestyle{empty}
\pagestyle{plain}
\pagestyle{headings}
\pagestyle{myheadings}
\thispagestyle{lokaler Seitenstil}
```

Eine der allgemeinen Eigenschaften eines Dokuments ist der Seitenstil. Bei LaTeX versteht man unter dem Seitenstil in erster Linie den Inhalt der Kopf- und Fußzeilen. Üblicherweise wird zwischen vier verschiedenen Seitenstilen unterschieden.

empty ist der Seitenstil, bei dem Kopf- und Fußzeile vollständig leer bleiben. Dies ist bei KOMA-Script vollkommen identisch zu den Standardklassen.

plain ist der Seitenstil, bei dem keinerlei Kolumnentitel verwendet, sondern nur eine Seitenzahl ausgegeben wird. Bei den Standardklassen wird diese Seitenzahl immer mittig im Fuß ausgegeben. Bei KOMA-Script erfolgt die Ausgabe stattdessen im doppelseitigen Layout außen im Fuß. Der einseitige Seitenstil entspricht bei KOMA-Script dem der Standardklassen.

headings ist der Seitenstil für lebende Kolumnentitel. Das sind Kolumnentitel, bei denen Überschriften automatisch in den Seitenkopf übernommen werden. Im Internet oder in Beschreibungen zu LaTeX-Paketen findet man auch häufig die englische Bezeichnung „*running headline*“. Bei den Klassen scrbook und scrreprt werden dabei im doppelseitigen Layout die Überschriften der Kapitel und der Abschnitte in der Kopfzeile wiederholt – bei KOMA-Script jeweils außen, bei den Standardklassen innen. Die Seitenzahl wird bei KOMA-Script im Fuß außen, bei den Standardklassen im Kopf außen gesetzt. Im einseitigen Layout werden nur die Überschriften der Kapitel verwendet und bei KOMA-Script zentriert im Kopf ausgegeben. Die Seitenzahlen werden bei KOMA-Script dann zentriert im Fuß gesetzt. Bei scrartcl wird entsprechend verfahren, jedoch eine Ebene tiefer bei Abschnitt und Unterabschnitt angesetzt, da die Gliederungsebene Kapitel hier nicht existiert.

scrbook, scrreprt

scrartcl

Während die Standardklassen automatische Kolumnentitel immer in Versalien – also Großbuchstaben – setzen, verwendet KOMA-Script die Schreibweise, die in der Überschrift vorgefunden wurde. Dies hat verschiedene typografische Gründe. So sind Versalien als Auszeichnung eigentlich viel zu mächtig. Verwendet man sie trotzdem, sollten sie um einen Punkt kleiner gesetzt und leicht gesperrt werden. All dies findet bei den Standardklassen keine Beachtung.

myheadings entspricht weitgehend dem Seitenstil `headings`, allerdings werden die Kolumnentitel nicht automatisch erzeugt, sondern liegen in der Verantwortung des Anwenders. Er verwendet dazu die Anweisungen `\markboth` und `\markright`.

Die Form der Seitenstile `headings` und `myheadings` wird außerdem durch die jede der vier Klassenoptionen `headsepline`, `headnosepline`, `footsepline` und `footnosepline` (siehe Abschnitt 3.1.2) beeinflusst. Der Seitenstil ab der aktuellen Seite wird mit der Anweisung `\pagestyle` umgeschaltet. Demgegenüber verändert `\thispagestyle` nur den Seitenstil der aktuellen Seite

```
\titlepagestyle
\partpagestyle
\chapterpagestyle
\indexpagestyle
```

Auf einigen Seiten wird mit Hilfe von `\thispagestyle` automatisch ein anderer Seitenstil gewählt. Welcher Seitenstil dies ist, wird diesen vier Makros entnommen. In der Voreinstellung ist der Seitenstil immer `plain`.

`\titlepagestyle` ist der Seitenstil der Seite mit der Titelei bei *in-page*-Titeln.

`\partpagestyle` ist der Seitenstil der Seiten mit `\part`-Titeln.

scrbook, scrreprt

`\chapterpagestyle` ist der Seitenstil auf Kapitelanfangsseiten.

`\indexpagestyle` ist der Seitenstil der ersten Indexseite.

Die Seitenstile können mit Hilfe von `\renewcommand` umdefiniert werden.

Beispiel: Angenommen, Sie wollen, dass die Seiten mit der `\part`-Überschrift nicht mit einer Nummer versehen werden. Dann setzen Sie folgende Anweisung beispielsweise in der Präambel Ihres Dokuments:

```
\renewcommand*{\partpagestyle}{empty}
```

Wie Sie oben erfahren haben, ist der Seitenstil `empty` genau das, was in diesem Beispiel verlangt wird. Natürlich können Sie auch einen selbstdefinierten Seitenstil verwenden.

Angenommen, Sie haben mit dem Paket scrpage2 (siehe Kapitel 4) einen eigenen Seitenstil für Kapitelanfangsseiten definiert. Diesem

Seitenstil haben Sie den passenden Namen `chapter` gegeben. Um diesen nun auch tatsächlich zu verwenden, definieren Sie das Makro `\chapterpagestyle` entsprechend um:

```
\renewcommand*{\chapterpagestyle}{chapter}
```

Die Umdefinierungen der beiden anderen Seitenstilvorgaben funktioniert in gleicher Weise.

```
\headfont
\pnumfont
```

Die Schriftart, in der Kopf und Fuß gesetzt werden, ist im Makro `\headfont` festgelegt. Vordefiniert ist dabei die schräge Variante der normalen Schrift: `\slshape`. Dies liegt einerseits in der Historie von KOMA-Script begründet, andererseits hat es auch einen praktischen Sinn. Ungeachtet dessen wird häufig die kursive Variante verlangt. Sie können selbiges leicht erreichen, indem Sie das Schriftmakro beispielsweise in der Dokumentpräambel umdefinieren:

```
\renewcommand*{\headfont}{\itshape}
```

Es ist an dieser Stelle jedoch nicht möglich, Versalien für die automatischen Kolumnentitel zu erzwingen. Wenn Sie dies wünschen, verwenden Sie bitte das scrpage2-Paket (siehe Kapitel 4).

Die Schriftart der Seitenzahl kann abweichend von der generellen Schriftart von Kopf und Fuß gewählt werden. Sie ist im Makro `\pnumfont` abgelegt. Voreingestellt ist hier die Standardschrift: `\normalfont`.

Beispiel: Angenommen, sie wollen Kopf und Fuß einen Schriftgrad kleiner und kursiv setzen. Die Seitenzahl soll jedoch nicht kursiv sondern fett gesetzt werden. Davon abgesehen, dass das Ergebnis grauenvoll aussehen wird, können Sie dies wie folgt erreichen:

```
\renewcommand*{\headfont}{\small\itshape}
\renewcommand*{\pnumfont}{\normalfont\bfseries}
```

Die Erklärung zu den Schriftanweisungen (`\normalfont`, `\small`, `\slshape`, `\itshape`, `\bfseries` etc.) entnehmen Sie bitte beispielsweise [SKPH99] oder [Tea00]. Erklärungen zu Standardfarbe (`\normalcolor`) sind in [Car99b] zu finden.

Der Seitenstil kann jederzeit mit Hilfe der `\pagestyle`-Anweisung gesetzt werden und gilt dann ab der nächsten Seite, die ausgegeben wird. Üblicherweise setzt man den Seitenstil jedoch nur einmal zu Beginn des Dokuments oder in der Präambel. Für eine Änderung des Seitenstils nur der aktuellen Seite verwendet man stattdessen die Anweisung `\thispagestyle`. Dies geschieht auch an einigen Stellen im Dokument automatisch. Beispielsweise wird bei allen Kapitelanfangsseiten implizit die Anweisung `\thispagestyle{plain}` ausgeführt.

```
\clearpage
\cleardoublepage
\cleardoublestandardpage
\cleardoubleplainpage
\cleardoubleemptypage
```

Im LaTeX-Kern existiert die Anweisung `\clearpage`, die dafür sorgt, dass alle noch nicht ausgegebenen Fließumgebungen ausgegeben werden und anschließend eine neue Seite begonnen wird. Außerdem existiert die Anweisung `\cleardoublepage`, die wie `\clearpage` arbeitet, durch die aber im doppelseitigen Layout (siehe Layoutoption `twoside` in Abschnitt 3.1.2) eine neue rechte Seite begonnen wird. Dazu wird gegebenenfalls eine leere linke Seite im aktuellen Seitenstil ausgegeben.

Bei KOMA-Script arbeitet `\cleardoublestandardpage` genau in der soeben beschriebene Art und Weise. Die Anweisung `\cleardoubleplainpage` ändert demgegenüber den Seitenstil der leeren linken Seite zusätzlich auf `plain`, um den Kolumnentitel zu unterdrücken. Analog dazu wird bei der Anweisung `\cleardoubleemptypage` der Seitenstil `empty` verwendet, um sowohl Kolumnentitel als auch Seitenzahl auf der leeren linken Seite zu unterdrückt. Die Seite ist damit vollständig leer. Die Arbeitsweise der `\cleardoublepage`-Anweisung ist hingegen von den in Abschnitt 3.1.2 erklärten Layoutoptionen `cleardoublestandard`, `cleardoubleplain` und `cleardoubleempty` abhängig und entspricht je nach Option einer der drei Anweisungen.

Bitte beachten Sie auch, dass die Umschaltung zwischen automatischen und manuellen Kolumnentiteln bei Verwendung des scrpage2-Pakets nicht mehr über den Seitenstil, sondern mit speziellen Anweisungen erfolgt. Die Seitenstile `headings` und `myheadings` sollten zusammen mit diesem Paket nicht verwendet werden (siehe Kapitel 4).

3.3 Die Titelei

Nachdem die Optionen nun bekannt sind, beginnen wir das Dokument, wo es normalerweise beginnt: mit der Titelei. Unter der Titelei versteht man alles, was im weitesten Sinne zum Titel eines Dokuments gehört. Wie in Abschnitt 3.1.2 bereits erwähnt, wird grundsätzlich zwischen Titelseiten und *in-page* Titeln unterschieden. Artikelklassen wie article oder scrartcl haben *in-page* Titel voreingestellt, während bei Klassen wie report, book, scrreprt und scrbook Titelseiten voreingestellt sind. Diese Voreinstellung ist mit den Klassenoptionen `titlepage` und `notitlepage` beeinflussbar.

`titlepage`

Grundsätzlich werden bei den Standardklassen und bei KOMA-Script alle Titelseiten in einer speziellen Umgebung, der `titlepage`-Umgebung, gesetzt. Diese Umgebung startet immer mit einer neuen Seite – im zweiseitigen Layout sogar mit einer neuen rechten Seite – im einspaltigen Modus. Für eine Seite wird der Seitenstil mit `\thispagestyle{empty}` geändert, so dass weder Seitenzahl noch Kolumnentitel ausgegeben werden. Am Ende der Umgebung wird die Seite automatisch beendet. Sollten Sie nicht das automatische Layout der Titelei verwenden können, ist zu empfehlen, eine eigene Titelei mit Hilfe dieser Umgebung zu entwerfen.

Beispiel: Angenommen, Sie wollen eine Titelseite, auf der lediglich oben links möglichst groß und fett das Wort „Me“ steht – kein Autor, kein Datum, nichts weiter. Folgendes Dokument ermöglicht das:

```
\documentclass{scrbook}
\begin{document}
  \begin{titlepage}
    \textbf{\Huge Me}
  \end{titlepage}
\end{document}
```

Einfach? Stimmt.

`\maketitle[`*Seitenzahl*`]`

Während bei den Standardklassen nur maximal eine Titelseite mit den drei Angaben Titel, Autor und Datum existiert, können bei KOMA-Script mit `\maketitle` bis zu sechs Titelseiten gesetzt werden. Im Gegensatz zu den Standardklassen kennt `\maketitle` bei KOMA-Script außerdem noch ein optionales nummerisches Argument. Findet es Verwendung, so wird die Nummer als Seitenzahl der ersten Titelseite benutzt. Diese Seitenzahl wird jedoch nicht ausgegeben, sondern beeinflusst lediglich die Zählweise. Sie sollten hier unbedingt eine ungerade Zahl wählen, da sonst die gesamte Zählung durcheinander gerät. Meiner Auffassung nach gibt es nur zwei sinnvolle Anwendungen für das optionale Argument. Zum einen könnte man dem Schmutztitel die logische Seitenzahl -1 geben, um so die Seitenzählung erst ab der Haupttitelseite mit 1 zu beginnen. Zum anderen könnte man mit einer höhere Seitenzahl beginnen, beispielsweise 3, 5 oder 7, um so weitere Titelseiten zu berücksichtigen, die erst vom Verlag hinzugefügt werden. Wird eine *in-page* Titelei verwendet, wird das optionale Argument ignorriert. Dafür kann der Seitenstil einer solchen Titelei durch Umdefinierung des Makros `\titlepagestyle` verändert werden. Siehe hierzu Abschnitt 3.2.

Die folgenden Anweisungen führen nicht unmittelbar zum Setzen der Titelei. Das Setzen der Titelei erfolgt immer mit `\maketitle`. Mit den nachfolgend erklärten Anweisungen werden lediglich die Inhalte der Titelei festgelegt. Sie müssen daher auch unbedingt vor `\maketitle` verwendet werden. Es ist jedoch nicht notwendig und bei Verwendung des babel-Pakets (siehe [Bra01]) auch nicht empfehlenswert, diese Anweisungen in der Dokumentpräambel vor `\begin{document}` zu verwenden. Beispieldokumente finden Sie am Ende des Abschnitts.

`\extratitle{`*Schmutztitel*`}`

Früher war der Buchblock oftmals nicht durch einen Buchdeckel vor Verschmutzung geschützt. Diese Aufgabe übernahm dann die erste Seite des Buches, die meist einen Kurztitel, eben den *Schmutztitel* trug. Auch heute noch wird diese Extraseite vor dem eigentlichen Haupttitel gerne verwendet und enthält dann Verlagsangaben, Buchreihennummer und ähnliche Angaben.

Bei KOMA-Script ist es möglich, vor der eigentlichen Titelseite eine weitere Seite zu setzen. Als *Schmutztitel* kann dabei beliebiger Text – auch mehrere Absätze – gesetzt werden. Der Inhalt von *Schmutztitel* wird von KOMA-Script ohne zusätzliche Beeinflussung der Formatierung ausgegeben. Dadurch ist dessen Gestaltung völlig dem Anwender überlassen. Die Rückseite des Schmutztitels bleibt leer. Der Schmutztitel ergibt auch dann eine eigene Titelseite, wenn mit *in-page* Titeln gearbeitet wird. Die Ausgabe des mit `\extratitle` definierten Schmutztitels erfolgt als Bestandteil der Titelei mit `\maketitle`.

Beispiel: Kommen wir auf das Beispiel von oben zurück und gehen davon aus, dass das spartanische „Me“ nur den Schmutztitel darstellt. Nach dem Schmutztitel soll noch der Haupttitel folgen. Dann kann wie folgt verfahren werden:

```
\documentclass{scrbook}
\begin{document}
  \extratitle{\textbf{\Huge Me}}
  \title{It's me}
  \maketitle
\end{document}
```

Sie können den Schmutztitel aber auch horizontal zentriert und etwas tiefer setzen:

```
\documentclass{scrbook}
\begin{document}
  \extratitle{\vspace*{4\baselineskip}
    \begin{center}\textbf{\Huge Me}\end{center}}
  \title{It's me}
  \maketitle
\end{document}
```

Die Anweisung `\title` ist grundsätzlich notwendig, damit die Beispiele fehlerfrei sind. Sie wird nachfolgend erklärt.

`\titlehead{`*Titelkopf*`}`
`\subject{`*Typisierung*`}`
`\title{`*Titel*`}`
`\author{`*Autor*`}`
`\date{`*Datum*`}`
`\publisher{`*Herausgeber*`}`
`\and`
`\thanks{`*Fußnote*`}`

Für den Inhalt der Haupttitelseite stehen sechs Elemente zur Verfügung. Der *Titelkopf* wird mit der Anweisung `\titlehead` definiert. Er wird über die gesamte Textbreite in normalem Blocksatz am Anfang der Seite ausgegeben. Er kann vom Anwender frei gestaltet werden.

Die *Typisierung* wird unmittelbar über dem *Titel* ausgegeben. Dabei wird eine gegenüber der Grundschrift leicht vergößerte Schrift verwendet. Der *Titel* wird in einer sehr großen Schrift ausgegeben, dabei findet außerdem `\sectfont` Anwendung (siehe Abschnitt 3.5.2).

Unter dem *Titel* folgt der *Autor*. Es können auch mehrere Autoren innerhalb des Arguments von `\author` angegeben werden. Diese sind dann mit `\and` voneinander zu trennen.

Unter dem Autor oder den Autoren folgt das Datum. Dabei ist das aktuelle Datum, `\today`, voreingestellt. Es kann jedoch mit `\date` eine beliebige Angabe – auch ein leere – erreicht werden.

Als letztes folgt schließlich der *Herausgeber*. Selbstverständlich kann diese Anweisung auch für andere Angaben geringer Wichtigkeit verwendet werden. Notfalls kann durch Verwendung einer `\parbox` über die gesamte Seitenbreite auch erreicht werden, dass diese Angabe nicht zentriert, sondern im Blocksatz gesetzt wird. Sie ist dann als Äquivalent zum Titelkopf zu betrachten. Dabei ist jedoch zu beachten, dass sie oberhalb von eventuell vorhandenen Fußnoten ausgegeben wird.

Element	Anweisung	Schrift	Ausrichtung
Seitenkopf	`\titlehead`	`\normalsize`	Blocksatz
Typisierung	`\subject`	`\Large`	zentriert
Titel	`\title`	`\sectfont\huge`	zentriert
Autoren	`\author`	`\Large`	zentriert
Datum	`\date`	`\Large`	zentriert
Herausgeber	`\publisher`	`\Large`	zentriert

Tabelle 3.3: Schriftgröße und horizontale Ausrichtung der Elemente der Haupttitelseite in der Reihenfolge ihrer vertikalen Position von oben nach unten bei Verwendung von `\maketitle`

Fußnoten werden auf der Titelseite nicht mit `\footnote`, sondern mit der Anweisung `\thanks` erzeugt. Sie dienen in der Regel für Anmerkungen bei den Autoren. Als Fußnotenzeichen werden dabei Symbole statt Zahlen verwendet.

Bis auf den *Titelkopf* und eventuelle Fußnoten werden alle Ausgaben horinzontal zentriert. Diese Angaben sind nocheinmal kurz zusammengefasst in Tabelle 3.3 zu finden.

Beispiel: Nehmen wir nun einmal an, Sie schrieben eine Diplomarbeit. Dabei sei vorgegeben, dass die Titelseite oben linksbündig das Institut einschließlich Adresse und rechtsbündig das Semester wiedergibt. Wie üblich ist ein Titel einschließlich Autor und Abgabedatum zu setzen. Außerdem soll der Betreuer angegeben und zu erkennen sein, dass es sich um eine Diplomarbeit handelt. Sie könnten das wie folgt erreichen:

```
\documentclass{scrbook}
\usepackage{ngerman}
\begin{document}
\titlehead{{\Large Universit"at Schlauenheim
    \hfill SS~2001\\}
  Institut f"ur Raumkr"ummung\\
  Hochschulstra"se~12\\
  34567 Schlauenheim}
\subject{Diplomarbeit}
\title{Digitale Raumsimulation mit dem DSP\,56004}
\author{cand. stup. Uli Ungenau}
\date{30. Februar 2001}
\publishers{Betreut durch Prof. Dr. rer. stup. Naseweis}
\maketitle
\end{document}
```

Ein häufiges Missverständnis betrifft die Bedeutung der Haupttitelseite. Irrtümlich wird oft angenommen, es handle sich dabei um den Buchumschlag oder Buchdeckel. Daher wird häufig erwartet, dass die Titelseite nicht den Randvorgaben für doppelseitige Satzspiegel gehorcht, sondern rechts und links gleich große Ränder besitzt. Nimmt man jedoch einmal ein Buch zur Hand und klappt es auf, trifft man sehr schnell auf mindestens eine Titelseite unter dem Buchdeckel innerhalb des sogenannten Buchblocks. Genau diese Titelseiten werden mit `\maketitle` gesetzt. Wie beim Schmutztitel handelt es sich also auch bei der Haupttitelseite um eine Seite innerhalb des Buchblocks, die deshalb dem Satzspiegel des gesamten Dokuments gehorcht. Überhaupt ist ein Buchdeckel, das *Cover* etwas, was man in einem getrennten Dokument erstellt. Schließlich hat er oft eine sehr individuelle Gestalt. Es spricht auch nichts dagegen, hierfür ein Grafik- oder DTP-Programm zu Hilfe zu nehmen. Ein getrenntes Dokument sollte auch deshalb verwendet werden, weil es später auf ein anderes Druckmedium, etwa Karton, und möglicherweise mit einem anderen Drucker ausgegeben werden soll.

`\uppertitleback{`*Titelrückseitenkopf*`}`
`\lowertitleback{`*Titelrückseitenfuß*`}`

Im doppelseitigen Druck bleibt bei den Standardklassen die Rückseite des Blatts mit der Titelseite leer. Bei KOMA-Script lässt sich die Rückseite der Haupttitelseite hingegen für weitere Angaben nutzen. Dabei wird zwischen genau zwei Elementen unterschieden, die der Anwender frei gestalten kann: dem *Titelrückseitenkopf* und dem *Titelrückseitenfuß*. Dabei kann der Kopf bis zum Fuß reichen und umgekehrt. Nimmt man diese Anleitung als Beispiel, so wurde der Haftungsausschluss mit Hilfe von `\uppertitleback` gesetzt.

`\dedication{`*Widmung*`}`

KOMA-Script bietet eine eigene Widmungsseite. Diese Widmung wird zentriert und in etwas größerer Schrift gesetzt. Die Rückseite ist wie bei der Seite mit dem Schmutztitel grundsätzlich leer. Die Widmungsseite wird zusammen mit der restlichen Titelei mit `\maketitle` ausgegeben und muss daher vor dieser Anweisung definiert sein.

Beispiel: Nehmen wir dieses Mal an, dass Sie einen Gedichtband schreiben, den Sie Ihrer Frau widmen wollen. Das könnte wie folgt aussehen:

```
\documentclass{scrbook}
\usepackage{ngerman}
\begin{document}
\extratitle{\textbf{\Huge In Liebe}}
\title{In Liebe}
\author{Prinz Eisenherz}
\date{1412}
```

```
\lowertitleback{Dieser Gedichtband wurde mit Hilfe von
  {\KOMAScript} und {\LaTeX} gesetzt.}
\uppertitleback{Selbstverlach}
\dedication{Meinem Schnuckelchen\\
  in ewiger Liebe.}
\maketitle
\end{document}
```

`abstract`

scrartcl, scrreprt

Insbesondere bei Artikeln, seltener bei Berichten findet man unmittelbar unter der Titelei und noch vor dem Inhaltsverzeichnis eine Zusammenfassung. Diese wird daher oftmals als Bestandteil der Titelei betrachtet. Einige LaTeX-Klassen bieten eine spezielle Umgebung für diese Zusammenfassung, die `abstract`-Umgebung. Diese wird unmittelbar ausgegeben, ist also nicht Bestandteil der mit `\maketitle` gesetzten Titelei. Bitte beachten Sie unbedingt, dass es sich bei `abstract` um eine Umgebung und nicht um eine Anweisung handelt. Ob die Zusammenfassung mit einer Überschrift versehen wird oder nicht, wird über die Optionen `abstracton` und `abstractoff` gesteuert (siehe Abschnitt 3.1.5).

Bei Büchern (scrbook) ist die Zusammenfassung häufig Bestandteil der Einleitung oder eines gesonderten Kapitels am Ende des Dokuments. Daher gibt es hier keine `abstract`-Umgebung. Bei Verwendung der Klasse scrreprt ist es sicher eine Überlegung wert, ob man nicht genauso verfahren sollte.

3.4 Das Inhaltsverzeichnis

`\tableofcontents`
`\contentsname`

Auf die Titelei folgt normalerweise das Inhaltsverzeichnis. Die Ausgabe des Inhaltsverzeichnisses erreicht man mit der Anweisung `\tableofcontents`. Um ein korrektes Inhaltsverzeichnis zu erhalten, sind nach jeder Änderung mindestens zwei LaTeX-Läufe notwendig. Mit der Option `liststotoc` kann erreicht werden, dass das Abbildungs- und das Tabellenverzeichnis im Inhaltsverzeichnis aufgeführt werden. Mit `idxtotoc` existiert auch eine entsprechende Option für den Index. Im klassischen Buchdruck ist dies eher unüblich. Das Literaturverzeichnis findet man etwas häufiger im Inhaltsverzeichnis aufgeführt. Auch hierfür gibt es mit `bibtotoc` und `bibtotocnumbered` Optionen. Diese Optionen sind in Abschnitt 3.1.4 näher erläutert.

Das Inhaltsverzeichnis wird als nicht nummeriertes Kapitel gesetzt und unterliegt damit den Seiteneffekten der `\chapter*`-Anweisung, die in Abschnitt 3.5.2 genannt sind. Allerdings werden bei Verwendung automatischer Kolumnentitel diese sowohl für linke als auch rechte Seiten korrekt mit der Überschrift des Inhaltsverzeichnisses belegt. Der Text der Überschrift ist im Makro `\contentsname` abgelegt.

Für den Aufbau des Inhaltsverzeichnisses gibt es nur eine Variante. Dabei werden die Gliederungsebenen so eingerückt, dass die Gliederungsnummer jeweils linksbündig mit dem Text der nächst höheren Ebene abschließt. Der Platz für die Gliederungsnummers ist dadurch jedoch limitiert und reicht für etwas mehr als 1,5 Stellen je Gliederungsebene. Sollte dies zu einem Problem werden, findet sich in [RH01] Abhilfe.

Der Eintrag für die oberste Gliederungsebene unter `\part`, also `\chapter` bei scrbook und scrreprt beziehungsweise `\section` bei scrartcl wird nicht eingerückt. Dafür findet auf ihn `\sectfont` Anwendung und es befinden sich zwischen dem Text der Gliederungsebene und der Seitenzahl keine Pünktchen. Die typografischen Gründe dafür liegen in der normalerweise anderen Schriftart, `\sectfont`, sowie der erwünschten Hervorhebung. Das Inhaltsverzeichnis dieser Anleitung kann als beispielhafte Darstellung betrachtet werden.

`tocdepth`

Normalerweise werden bei den Klassen scrbook und scrreprt die Gliederungsebenen `\part` bis `\subsection` und bei der Klasse scrartcl die Ebenen `\part` bis `\subsubsection` in das Inhaltsverzeichnis aufgenommen. Gesteuert wird dies über den Zähler `tocdepth`. Dabei steht der Wert -1 für `\part`, 0 für `\chapter` und so weiter. Durch Setzen oder Erhöhen oder Verringern des Zählers kann bestimmt werden, bis zu welcher Gliederungsebene Einträge in das Inhaltsverzeichnis erfolgen sollen. Dies ist übrigens bei den Standardklassen ganz genauso.

Bei Verwendung des scrpage2-Pakets (siehe Kapitel 4) muss man sich nicht die nummerischen Werte der einzelnen Gliederungsebenen merken. Dann stehen dafür die Makros `\chapterlevel`, `\sectionlevel` und so weiter bis hinunter zu `\subparagraphlevel` zur Verfügung.

Beispiel: Angenommen, Sie setzen einen Artikel, bei dem die Gliederungsebene `\subsubsection` verwendet wird. Gehen wir weiter davon aus, dass Sie diese Gliederungsebene aber nicht im Inhaltsverzeichnis haben wollen. Dann könnte die Präambel Ihres Dokuments wie folgt aussehen:

```
\documentclass{scrartcl}
\setcounter{tocdepth}{2}
```

Sie setzen den Zähler `tocdepth` also auf 2, weil Sie wissen, dass dies der Wert für `\subsection` ist. Wissen Sie stattdessen nur, dass normalerweise bei scrartcl Einträge in das Inhaltsverzeichnis bis zur Ebene `\subsubsection` erfolgen, können Sie auch einfach vom voreingestellten Wert des Zählers `tocdepth` eins abziehen:

```
\documentclass{scrartcl}
\addtocounter{tocdepth}{-1}
```

Wieviel Sie von `tocdepth` subtrahieren oder dazu addieren müssen, können Sie natürlich auch einfach nach einem ersten LaTeX-Lauf im Inhaltsverzeichnis abzählen.

3.5 Der Haupttext

3.5.1 Abgrenzung

```
\frontmatter
\mainmatter
\backmatter
```

scrbook Bevor wir tatsächlich zum Haupttext kommen, sind noch drei Anweisungen kurz anzusprechen, die es bei der Standardklasse book und der KOMA-Script-Klasse scrbook gibt. Sie können bei einem Buch zur Abgrenzung des *Vorspanns*, des *Hauptteil* und des *Nachspanns* verwendet werden.

Mit `\frontmatter` wird der Vorspann eingeleitet. Im Vorspann werden die nummerierten Seiten mit römischen Seitenzahlen versehen. Kapitelüberschriften sind im Vorspann nicht nummeriert. Abschnittsüberschriften wären jedoch nummeriert, gingen von Kapitelnummer 0 aus und wären außerdem über Kapitelgrenzen hinweg durchgehend nummeriert. Dies spielt jedoch keine Rolle, da der Vorspann allenfalls für die Titelei, das Inhalts-, Abbildungs- und Tabellenverzeichnis und ein Vorwort verwendet wird. Das Vorwort kann also als normales Kapitel gesetzt werden. Ein Vorwort sollte niemals in Abschnitte unterteilt, sondern möglichst kurz gefasst werden. Im Vorwort wird also keine tiefere Gliederungsebene als Kapitel benötigt.

Mit `\mainmatter` wird der Hauptteil eingeleitet. Existiert kein Vorspann, so kann diese Anweisung auch entfallen. Im Hauptteil sind arabische Seitenzahlen voreingestellt. Die Seitenzählung beginnt im Hauptteil wieder mit 1.

Mit `\backmatter` wird der Nachspann eingeleitet. Was zum Nachspann gehört, ist unterschiedlich. Teilweise wird im Nachspann nur das Literaturverzeichnis gesetzt, teilweise nur der Index. Manchmal erscheint der gesamte Anhang im Nachspann. Der Nachspann gleicht im übrigen dem Vorspann.

3.5.2 Gliederung

```
\part[Kurzform]{Überschrift}
\chapter[Kurzform]{Überschrift}
\section[Kurzform]{Überschrift}
\subsection[Kurzform]{Überschrift}
\subsubsection[Kurzform]{Überschrift}
\paragraph[Kurzform]{Überschrift}
\subparagraph[Kurzform]{Überschrift}
```

Die Standardgliederungsbefehle funktionieren bei KOMA-Script beinahe genau wie bei den Standardklassen. So kann ganz normal über ein optionales Argument ein abweichender Text für den Kolumnentitel und das Inhaltsverzeichnis vorgegeben werden. `\chapter` existiert nur bei Buch- und Berichtklassen, also bei book, scrbook, report und scrreport, nicht jedoch bei den Artikelklassen article und scrartcl. `\chapter` unterscheidet sich bei KOMA-Script außerdem gravierend von der Version der Standardklassen. Bei den Standardklassen wird die Kapitelnummer mit dem Präfixzusatz „Kapitel" beziehungsweise dem Kapitelnamen in der gewählten Sprache in einer Zeile vor dem eigentlichen Text der Überschrift ausgegeben. Diese sehr mächtige Form wird bei KOMA-Script durch eine einfache Nummer vor dem Text abgelöst. scrartcl

Bitte beachten Sie, dass `\part` und `\chapter` den Seitenstil für eine Seite umschaltet. Der jeweilige Seitenstil ist bei KOMA-Script in den Makros `\partpagestyle` und `\chapterpagestyle` abgelegt (siehe Abschnitt 3.2).

```
\part*{Überschrift}
\chapter*{Überschrift}
\section*{Überschrift}
\subsection*{Überschrift}
\subsubsection*{Überschrift}
\paragraph*{Überschrift}
\subparagraph*{Überschrift}
```

Ebenso existieren die Sternvarianten der Gliederungsbefehle, bei denen keine Nummerierung erfolgt, kein Kolumnentitel gesetzt wird und kein Eintrag ins Inhaltsverzeichnis statt findet. Der Verzicht auf den Kolumnentitel hat übrigens einen oftmals unerwünschten Effekt. Geht beispielsweise ein mit `\chapter*` gesetztes Kapitel über mehrere Seiten, so taucht plötzlich der Kolumnentitel des letzen Kapitels wieder auf. KOMA-Script bietet dafür aber eine Lösung, die im Folgenden beschrieben wird. `\chapter*` existiert selbstverständlich nur bei Buch- und Berichtklassen, also bei book, scrbook, report und scrreport, nicht jedoch bei den Artikelklassen article und scrartcl. scrartcl

Bitte beachten Sie, dass `\part*` und `\chapter*` den Seitenstil für eine Seite umschaltet. Der jeweilige Seitenstil ist bei KOMA-Script in den Makros `\partpagestyle` und `\chapterpagestyle` abgelegt (siehe Abschnitt 3.2).

```
\addpart[Kurzform]{Überschrift}
\addpart*{Überschrift}
\addchap[Kurzform]{Überschrift}
\addchap*{Überschrift}
\addsec[Kurzform]{Überschrift}
\addsec*{Überschrift}
```

KOMA-Script bietet über die Gliederungsbefehle der Standardklassen hinaus die Anweisungen `\addchap` und `\addsec`. Diese ähneln bis auf die fehlende Nummerierung sehr den Standardanweisungen `\chapter` und `\section`. Sie erzeugen also sowohl einen automatischen Kolumnentitel als auch einen Eintrag ins Inhaltsverzeichnis. Die Sternvarianten `\addchap*` und `\addsec*` gleichen hingegen den Standardanweisungen `\chapter*` und `\section*` mit einem winzigen aber wichtigen Unterschied: Die Kolumnentitel werden gelöscht. Dadurch wird der oben erwähnte Effekt veralteter Kolumnentitel ausgeschlossen. Stattdessen bleibt der Kolumnentitel auf Folgeseiten leer. `\addchap` und `\addchap*` existieren selbstverständlich nur bei Buch- und Berichtklassen, also bei book, scrbook, report und scrreport, nicht jedoch bei den Artikelklassen article und scrartcl.

scrartcl

Die Anweisung `\addpart` erstellt entsprechend einen nicht nummerierten Dokumentteil mit einem Eintrag ins Inhaltsverzeichnis. Da bereits `\part` und `\part*` den Kolumnentitel löschen, ergibt sich hier nicht das oben genannte Problem mit veralteten Kolumnentiteln. Die Sternvariante `\addpart*` ist daher identisch mit der Sternvariante `\part*` und wurde nur aus Konsistenzgründen definiert.

Bitte beachten Sie, dass `\addpart` und `\addchap` und deren Sternvarianten den Seitenstil für eine Seite umschaltet. Der jeweilige Seitenstil ist in den Makros `\partpagestyle` und `\chapterpagestyle` abgelegt (siehe Abschnitt 3.2).

```
\minisec{Überschrift}
```

Manchmal ist eine Art Überschrift wünschenswert, die zwar hervorgehoben wird, ansonsten aber eng mit dem nachfolgenden Text zusammenhängt. Eine solche Überschrift soll dann ohne große Abstände gesetzt werden.

Der Befehl `\minisec` bewirkt genau eine derartige Überschrift. Diese Überschrift ist keiner Gliederungsebene zugeordnet. Eine solche *Mini-section* wird nicht in das Inhaltsverzeichnis aufgenommen und erhält auch keine Nummerierung.

Beispiel: Sie haben einen Bausatz für eine Mausefalle entwickelt und wollen diesen getrennt nach den benötigen Materialien und der Anleitung für die Montage beschreiben. Das könnte so gemacht werden:

```
\minisec{Bauteile}

\begin{flushleft}
  1 Brett ($100\times 50 \times 12$)\\
  1 Bierflaschenschnappverschluss\\
  1 Kugelschreiberfeder\\
  1 Reißzwecke\\
  2 Schrauben\\
  1 Hammer\\
  1 Messer
\end{flushleft}

\minisec{Montage}

Zunächst suche man das Mauseloch. Dann lege man die
Reißzwecke innen unmittelbar hinter das Loch, damit bei
den folgenden Aktionen die Maus nicht entschlüpfen kann.
Anschließend klopfe man mit dem Hammer den
Bierflaschenschnappverschluss in das Mauseloch. Sollte
der Verschluss nicht groß genug sein, um das Loch
vollständig und dauerhaft zu verschließen, nehme man
stattdessen das Brett und schraube es unter Zuhilfenahme
der beiden Schrauben und des Messers vor das Loch. Statt
des Messers kann selbstverständlich auch ein
Schraubendreher verwendet werden.
```

Das ganze sieht anschließend so aus:

Bauteile

1 Brett ($100 \times 50 \times 12$)
1 Bierflaschenschnappverschluss
1 Kugelschreiberfeder
1 Reißzwecke
2 Schrauben
1 Hammer
1 Messer

Montage

Zunächst suche man das Mauseloch. Dann lege man die Reißzwecke innen unmittelbar hinter das Loch, damit bei den folgenden Aktionen die Maus nicht entschlüpfen kann. Anschließend klopfe man mit dem Hammer den Bierflaschenschnappverschluss in das Mauseloch. Sollte der Verschluss nicht groß genug sein, um das Loch vollständig und dauerhaft zu verschließen, nehme man stattdessen das Brett und schraube es unter Zuhilfenahme der beiden Schrauben und des Messers vor das Loch. Statt des Messers kann selbstverständlich auch ein Schraubendreher verwendet werden.

`\sectfont`

Bei den Standardklassen wird für Überschriften eine fest eingestellte Schriftart verwendet (fette Variante der Standardschrift: `\bfseries`). Bei KOMA-Script ist die Schriftart, die für die Überschriften verwendet wird, im Makro `\sectfont` abgelegt. Voreingestellt ist die fette Variante der serifenlosen Schrift: `\sffamily\bfseries`. Die in `\sectfont` abgelegte Schriftart wird für alle Überschriften, für Teile des Titels und die oberste Gliederungsebene im Inhaltsverzeichnis verwendet. Durch Umdefinieren des Makros kann eine andere Schriftart vorgegeben werden.

Beispiel: Sie wollen, dass die Überschriften wie bei den Standardklassen lediglich fett aber nach wie vor in der serifenbehafteten Standardschrift gesetzt werden. Also definieren Sie `\sectfont` wie folgt um:

```
\renewcommand*{\sectfont}{\bfseries}
```

Unterschiedliche Schriften für unterschiedliche Gliederungsebenen sind mit KOMA-Script-Mitteln nicht möglich. Dies hat typografische Gründe. Eine Regel der Typografie besagt, dass man möglichst wenig Schriften miteinander mischen soll. Serifenlose für die Überschriften scheinen bereits ein Verstoß gegen diese Regel zu sein. Allerdings muss man wissen, dass fette, große, serifenbehaftete Buchstaben oft viel zu mächtig für eine Überschrift sind. Man müsste dann strenggenommen zumindest auf eine normale statt eine fette oder halbfette Schrift ausweichen. In tiefen Gliederungsebenen kann das aber wieder zu schwach sein. Andererseits haben Serifenlose in Überchriften eine sehr angenehme Wirkung und fast nur für Überschriften eine Berechtigung. Daher wurde diese Voreinstellung für KOMA-Script mit guten Grund gewählt. Größere Vielfalt sollte aber vermieden werden. Schriftenmischung ist etwas für Profis. Aus den genannten Gründen sollten Sie bei Verwendung anderer als der Standard-TEX-Fonts – egal ob CM- oder EC-Fonts – bezüglich der Verträglichkeit der serifenlosen und serifenbehafteten Schrift einen Experten zu Rate ziehen oder `\sectfont` vorsichtshalber wie in obigem Beispiel umdefinieren. Die häufig anzutreffenden Kombinationen Times mit Helvetica oder Palatino mit Helvetica werden vom Autor als unverträglich betrachtet.

```
\raggedsection
```

Bei den Standardklassen werden die Überschriften ganz normal im Blocksatz ausgegeben. Dadurch können in den Überschriften Trennungen auftreten und mehrzeilige Überschriften werden auf Textbreite gedehnt. Dieses Vorgehen ist in der Typografie eher unüblich. KOMA-Script setzt Überschriften daher in linksbündigem Flattersatz mit hängendem Einzug. Verantwortlich ist dafür die Anweisung `\raggedsection`, die vordefiniert ist als:

```
\newcommand*{\raggedsection}{\raggedright}
```

Diese Anweisung kann mit `\renewcommand` umdefiniert werden.

Beispiel: Sie wollen auch für Überschriften Blocksatz. Dazu schreiben Sie in die Präambel Ihres Dokuments:

```
\renewcommand*{\raggedsection}{}
```

oder kürzer:

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

Sie erreichen somit eine ähnliche Formatierung der Überschriften wie bei den Standardklassen. Noch ähnlicher wird es, wenn sie diese Änderung mit der oben vorgestellten Änderung für `\sectfont` kombinieren.

```
\partformat
\chapterformat
\othersectionlevelsformat{Gliederungsname}
\autodot
```

Bekanntlich gibt es bei LATEX zu jedem Zähler eine Anweisung `\the`*Zählername*, mit welcher der Zähler ausgegeben werden kann. Die Darstellung der Zähler für die einzelnen Gliederungsebenen setzt sich dabei je nach Klasse ab `\section` (book, scrbook, report, scrreprt) oder ab `\subsection` (article, scrartcl) aus der Darstellung des Zählers für die übergeordnete Ebene, gefolgt von einem Punkt und der arabischen Zahl des *Zählername*ns der jeweiligen Ebene zusammen.

KOMA-Script hat der Ausgabe der Gliederungsnummern eine weitere logische Ebene zugefügt. Die Zähler werden für die jeweilige Überschrift nicht einfach nur ausgegeben. Sie werden mit Hilfe der Anweisungen `\partformat`, scrbook, scrreprt `\chapterformat` und `\othersectionlevelsformat` formatiert. Die Anweisung `\chapterformat` existiert selbstverständlich nicht in der Klasse scrartcl.

Wie bereits in Abschnitt 3.1.5 erläutert wurde, müssen nach [DUD96] die Gliederungsnummern je nach Gliederung mit einem nachfolgenden Punkt versehen werden oder dieser hat zu entfallen. Die Anweisung `\autodot` ist bei KOMA-Script für die Einhaltung dieser Regel verantwortlich. Auf den Punkt folgt bei allen Gliederungsebenen außer `\part` noch ein `\enskip`. Dies entspricht einem Leerraum von 0,5 em.

Die Anweisung `\othersectionlevelsformat` erwartet als Parameter den Namen der Gliederungsebene, also „*section*", „*subsection*" ... In der Voreinstellung gibt es also nur für die Ebenen `\part` und `\chapter` eigene Formatieranweisungen, während alle anderen Giederungsebenen mit Hilfe einer einzigen Formatieranweisung abgedeckt werden. Dies ist allein historisch begründet. Als Werner Lemberg für sein CJK-Paket eine entsprechende Erweiterung von KOMA-Script angeregt hat, wurde nur diese Unterscheidung benötigt.

Die Formatierungsanweisungen können mit Hilfe von `\renewcommand` umdefiniert werden, um sie eigenen Anforderungen anzupassen. Nachfolgend finden Sie die Originaldefinitionen aus den KOMA-Script-Klassen:

```
\newcommand*{\partformat}{\partname~\thepart\autodot}
\newcommand*{\chapterformat}{\thechapter\autodot\enskip}
\newcommand*{\othersectionlevelsformat}[1]{%
  \csname the#1\endcsname\autodot\enskip}
```

Beispiel: Angenommen, Sie wollen, dass bei `\part` das Wort „Teil" vor der Nummer nicht ausgegeben wird. Dann können Sie beispielsweise folgende Anweisung in die Präambel Ihres Dokuments schreiben:

```
\renewcommand*{\partformat}{\thepart\autodot}
```

Genaugenommen könnten Sie an dieser Stelle auch auf `\autodot` verzichten und stattdessen einen festen Punkt setzen. Da `\part` mit römischen Zahlen nummeriert wird, muss der Punkt laut [DUD96] folgen. Allerdings bringen Sie sich dann um die Möglichkeit, eine der Optionen `pointednumbers` und `pointlessnumbers` einzusetzen und so von der Regel abzuweichen. Näheres zu den Klassenoptionen siehe Abschnitt 3.1.5).

```
\chapappifprefix[Zusatztext]
\chapapp
```

Diese beiden Anweisungen werden nicht nur intern von KOMA-Script verwendet, sondern stehen auch dem Anwender zur Verfügung. Nachfolgend werden sie beispielsweise für die Umdefinierung anderer Anweisungen verwendet. `\chapappifprefix` setzt bei Verwendung der Layoutoption `chapterprefix` im Hauptteil des Dokuments das Wort „Kapitel" in der aktuellen Sprache gefolgt vom `Zusatztext`. Im Anhang wird stattdessen das Wort „Anhang" in der aktuellen Sprache, ebenfalls gefolgt vom `Zuatztext`, ausgegeben. Bei der Einstellung `nochapterprefix` wird hingegen nichts ausgegeben. scrbook, scrreprt

Die Anweisung `\chapapp` setzt immer das Wort „Kapitel" beziehungsweise „Anhang". Dabei spielen die Optionen `chapterprefix` und `nochapterprefix` keine Rolle.

Da es Kapitel nur bei den Klassen scrbook und scrreprt gibt, existieren die beiden Anweisungen auch nur bei diesen Klassen.

```
\chaptermark{Kolumnentitel}
\sectionmark{Kolumnentitel}
\subsectionmark{Kolumnentitel}
\chaptermarkformat
\sectionmarkformat
\subsectionmarkformat
```

Wie bereits in Abschnitt 3.2 erwähnt, arbeitet der Seitenstil `headings` mit automatischen Kolumnentiteln. Dazu werden die Anweisungen `\chaptermark` und `\sectionmark` beziehungsweise `\sectionmark` und `\subsectionmark` entsprechend definiert. Jeder Gliederungsbefehl (`\chapter`, `\section`, `\subsection` ...) führt automatisch eine entsprechende `\...mark`-Anweisung aus. Der übergebene Parameter beinhaltet dabei den Text der Gliederungsüberschrift. Die Gliederungsnummer wird automatisch in der `\dotsmark` -Anweisung hinzugefügt. Die Formatierung erfolgt dabei je nach Gliederungsebene mit `\chaptermarkformat`, `\sectionmarkformat` oder `\subsectionmarkformat`. Selbstverständlich gibt es bei scrartcl weder `\chaptermark` noch `\chaptermarkformat`. Dafür existieren `\subsectionmark` und `\subsectionmarkformat` nur bei scrartcl. Dies ändert sich allerdings bei Verwendung des scrpage2-Pakets (siehe Kapitel 4). scrbook, scrreprt scrartcl

So wie mit `\chaptermark` und `\othersectionlevelsmark` die Nummern der Gliederungsüberschriften formatiert ausgegeben werden, werden mit den Anweisungen `\chaptermarkformat` (nicht scrartcl), `\sectionmarkformat` und `\subsectionmarkformat` (nur scrartcl) die Nummern der Gliederungsebenen in den automatischen Kolumnentiteln formatiert ausgegeben. Sie können mit `\renewcommand` eigenen Anforderungen angepasst werden. Die Originaldefinitionen aus den KOMA-Script-Klassen sind:

```
\newcommand*{\chaptermarkformat}{%
```

```
  \chapappifprefix[\ ]\thechapter\autodot\enskip}
\newcommand*{\sectionmarkformat}{\thesection\autodot\enskip}
\newcommand*{\subsectionmarkformat}{%
  \thesubsection\autodot\enskip}
```

Beispiel: Angenommern Sie wollen, dass der Kapitelnummer in den Kolumnentiteln das Wort „Kapitel“ vorangestellt wird. Dann setzen Sie beispielsweise folgenden Definition in die Präambel Ihres Dokuments:

```
\renewcommand*{\chaptermarkformat}{%
  \chapapp~\thechapter\autodot\enskip}
```

Wie Sie sehen, finden hier die Anweisungen `\chapapp` und `\chapappifprefix` Verwendung, die weiter oben erklärt wurden.

```
\setpartpreamble{Preamble}
\setchapterpreamble{Preamble}
```

scrbook, scrreprt

Teile und Kapitel können bei KOMA-Script mit einer Präambel versehen werden. Dies ist insbesondere im zweispaltigen Layout mit der Klassenoption `twocolumn` nützlich, da die Präambel zusammen mit der Überschrift einspaltig gesetzt wird. Die Präambel kann auch mehrere Absätze beinhalten. Die Präambel muss vor der jeweiligen `\part`- oder `\addpart`- bzw. `\chapter`- oder `\addchap`-Anweisung definiert werden.

Beispiel: Sie schreiben einen Bericht über den Zustand einer Firma. Dabei organisieren Sie den Bericht so, dass jeder Abteilung ein eigener Teilbericht spendiert wird. Jedem dieser Teile soll außerdem eine Zusammenfassung vorangestellt werden. Diese Zusammenfassung soll auf der Titelseite jedes Teils stehen. Das ist wie folgt möglich:

```
\setpartpreamble{%
  \begin{abstract}
    Dies ist ein Blindtext, dessen Inhalt Sie einfach
    unbeachtet lassen sollten. Er dient lediglich zur
    Demonstration der Möglichkeiten von \KOMAScript.
  \end{abstract}
}
\part{Abteilung Grünschnitt}
```

Je nach Einstellung der Optionen für die Überschriftengröße (siehe Abschnitt 3.1.3) und der Optionen für die Form der `abstract`-Umgebung (siehe Abschnitt 3.1.5), sieht das Ergebnis ungefähr wie folgt aus:

Teil III.

Abteilung Grünschnitt

Zusammenfassung

Dies ist ein Blindtext, dessen Inhalt Sie einfach unbeachtet lassen sollten. Er dient lediglich zur Demonstration der Möglichkeiten von KOMA-Script.

Bitte beachten Sie, dass Sie für die Abstände der Prämbel zur Teil-Überschrift bzw. zum Kapiteltext selbst verantwortlich sind. Bitte beachten sie auch, dass die `abstract`-Umgebung bei der Klasse scrbook nicht existiert (siehe Abschnitt 3.3).

3.5.3 Fußnoten

Eine nicht auf den Haupttext beschränkte Eigenschaft eines Dokuments ist das Aussehen der Fußnoten. Da Fußnoten aber hauptsächlich im Haupttext auftreten, werden sie in diesem Abschnitt aufgeführt.

```
\footnote[Nummer]{Text}
\footnotemark[Nummer]
\footnotetext[Nummer]{Text}
```

Fußnoten werden bei KOMA-Script genau wie bei den Standardklassen mit der Anweisung `\footnote` oder den paarweise zu verwendenden Anweisungen `\footnotemark` und `\footnotetext` erzeugt. Genau wie bei den Standardklassen ist es möglich, dass innerhalb einer Fußnote ein Seitenumbruch erfolgt. Dies geschieht in der Regel dann, wenn die zugehörige Fußnotenmarkierung so weit unten auf der Seite gesetzt wird, dass keine andere Wahl bleibt, als die Fußnote auf die nächste Seite zu umbrechen.

```
\deffootnote[Markenbreite]{Ansatzeinzug}{Einzug}{Markendefinition}
\deffootnotemark{Markendefinition}
\thefootnotemark
\textsuperscript{Text}
```

KOMA-Script setzt die Fußnoten etwas anders als die Standardklassen. Die Fußnotenmarkierung im Text erfolgt wie bei den Standardklassen durch kleine hochgestellte Zahlen. Genauso werden die Markierungen auch in der Fußnote selbst wiedergegeben. Sie werden dabei rechtsbündig in einem Feld der *Markenbreite* 1 em gesetzt. Die erste Zeile des Fußnotentextes wird gegenüber dem linken Rand mit einem *Einzug* von ebenfalls 1 em gesetzt. Das bedeutet, dass er direkt an die Fußnotenmarke stößt. Ist der Text der Fußnote mehrzeilig, so wird er ab der zweiten Zeile mit dem *Absatzeinzug* 1,5 em versehen. All diese Angaben einschließlich der {*Markendefinition*} können mit der Anweisung `\deffootnote` verändert werden. Die Voreinstellung entspricht dabei:

```
\deffootnote[1em]{1.5em}{1em}
  {\textsuperscript{\normalfont\thefootnotemark}}
```

Dabei wird mit Hilfe von `\textsuperscript` sowohl die Hochstellung als auch die Wahl einer kleineren Schrift erreicht. `\thefootnotemark` ist die aktuelle Fußnotenmarke ohne jegliche Formatierung.

Wird das optionale Argument *Markenbreite* weggelassen, so wird sie automatisch auf den Wert des *Einzugs* der ersten Zeile gesetzt. Die Voreinstellung ist also ebenfalls identisch mit:

```
\deffootnote{1.5em}{1em}
  {\textsuperscript{\normalfont\thefootnotemark}}
```

Die Fußnotenmarkierung im Text wird getrennt davon mit der Anweisung `\deffootnotemark` definiert. Voreingestellt ist hier:

```
\deffootnotemark{%
  \textsuperscript{\normalfont\thefootnotemark}}
```

Die Markierungen im Text und in der Fußnote selbst sind also identisch.

Beispiel: Relativ häufig wird gewünscht, dass die Markierung in der Fußnote selbst weder hochgestellt noch kleiner gesetzt wird. Dabei soll sie aber nicht direkt am Text kleben, sondern geringfügig davor stehen. Dies kann zum einen wie folgt erreicht werden:

```
\deffootnote{1.5em}{1em}{\thefootnotemark\ }
```

Die Fußnotenmarkierung wird also rechtsbündig, gefolgt von einem Leerzeichen in eine Box der Breite 1 em gesetzt. Die erste Zeile des Fußnotentextes wird gegenüber dem linken Rand ebenfalls um 1 em eingezogen.

Alternativ kann aber auch festgelegt werden, dass der Bereich der Marke gegenüber dem Einzug der ersten Zeile kleiner gewählt wird:

```
\deffootnote[.75em]{1.5em}{1.25em}{\thefootnotemark}
```

Auch so entsteht ein Abstand zwischen Fußnotenzahl und der ersten Zeile.

Häufig wird auch gewünscht, dass die weiteren Zeilen der Fußnote nicht stärker eingezogen werden, als der Text der ersten Zeile. Auch das ist gut zu begründen, sind doch die einzelnen Fußnoten bereits durch die Markierung deutlich voneinander getrennt. Eine mögliche Definition wäre dann:

```
\deffootnote[1em]{1.5em}{1.5em}{\thefootnotemark}
```

Wie die Beispiele zeigen, ermöglich KOMA-Script mit dieser einfachen Schnittstelle eine große Vielfalt unterschiedlicher Fußnotenformatierungen.

3.5.4 Listen

LaTeX und die Standardklassen bieten verschiedene Umgebungen für Listen. All diese Umgebungen bietet KOMA-Script selbstverständlich auch, teilweise jedoch mit leichten Abwandlungen oder Erweiterungen. Grundsätzlich gilt, dass Listen – auch unterschiedlicher Art – bis zu einer Tiefe von vier Listen geschachtelt werden können. Eine tiefere Schachtelung wäre auch aus typografischen Gründen kaum sinnvoll, da genaugenommen schon mehr als drei Ebenen nicht mehr überblickt werden können. Ich empfehle in solchen Fällen, die eine große Liste in mehrere kleinere Listen aufzuteilen.

```
itemize
\item
\labelitemi
\labelitemii
\labelitemiii
\labelitemiv
```

Die einfachste Form einer Liste ist die Stichpunkt- oder `itemize`-Liste. Die Benutzer einer unbeliebten Textverarbeitung nennen diese Form der Liste auch gerne *Bulletpoints*. Vermutlich können sie sich nicht vorstellen, dass je nach

Ebene auch ein anderes Zeichen als ein fetter Punkt zur Einleitung eines Stichpunkts verwendet werden kann. Bei KOMA-Script werden je nach Ebene folgende Zeichen verwendet: „•“, „–“, „∗“ und „·“. Die Definition der Zeichen für die einzelnen Ebenen sind in den Makros `\labelitemi`, `\labelitemii`, `\labelitemiii` und `\labelitemiv` abgelegt. Sie können mit `\renewcommand` umdefiniert werden. Die einzelnen Stichpunkte werden mit `\item` eingeleitet.

Beispiel: Sie haben eine einfache Aufzählung, die in mehreren Ebenen geschachtelt ist. Sie schreiben beispielsweise:

```
\minisec{Die Fahrzeuge im Spiel}
\begin{itemize}
  \item Flugzeuge
  \begin{itemize}
    \item Doppeldecker
    \item Jets
    \item Transportmaschinen
    \begin{itemize}
      \item einmotorig
      \begin{itemize}
        \item{düsengetrieben}
        \item{propellergetrieben}
      \end{itemize}
      \item mehrmotorig
    \end{itemize}
    \item Drehflügler
  \end{itemize}
  \item Motorräder
  \begin{itemize}
    \item historisch korrekt
     \item futurisch nicht real
  \end{itemize}
  \item Automobile
  \begin{itemize}
    \item Rennwagen
    \item Personenwagen
    \item Lastwagen
  \end{itemize}
  \item Fahrräder
\end{itemize}
```

Anschließend erhalten Sie:

Die Fahrzeuge im Spiel

- Flugzeuge
  - Doppeldecker
  - Jets
  - Transportmaschinen
    - ∗ einmotorig
      - · düsengetrieben
      - · propellergetrieben
    - ∗ mehrmotorig
  - Drehflügler
- Motorräder
  - historisch korrekt
  - futurisch nicht real
- Automobile
  - Rennwagen
  - Personenwagen
  - Lastwagen
- Fahrräder

```
enumerate
\item
\theenumi
\theenumii
\theenumiii
\theenumiv
\labelenumi
\labelenumii
\labelenumiii
\labelenumiv
```

Die nummerierte Liste ist ebenfalls sehr häufig zu finden und bereits vom LaTeX-Kern vorgesehen. Die Nummerierung erfolgt je nach Ebene in unterschiedlicher Art: mit arabischen Zahlen, mit Kleinbuchstaben, mit kleinen römischen Zahlen und mit Großbuchstaben. Die Art der Nummerierung wird dabei über die Makros `\theenumi` bis `\theenumiv` festgelegt. Das Format der Ausgabe wird von den Makros `\labelenumi` bis `\labelenumiv` bestimmt.

Dabei folgt auf den Wert der zweiten Ebene, der in Kleinbuchstaben ausgegeben wird, eine runde Klammer, während die Werte aller anderen Ebenen von einem Punkt gefolgt werden. Die einzelnen Stichpunkte werden wieder mit `\item` eingeleitet.

Beispiel: Übernehmen wir das Beispiel der `itemize`-Umgebung, wobei wir jede `itemize`-Umgebung durch eine `enumerate`-Umgebung ersetzen. Das Ergebnis wäre dann:

Die Fahrzeuge im Spiel

1. Flugzeuge
   a) Doppeldecker
   b) Jets
   c) Transportmaschinen
      i. einmotorig
         A. düsengetrieben
         B. propellergetrieben
      ii. mehrmotorig
   d) Drehflügler
2. Motorräder
   a) historisch korrekt
   b) futurisch nicht real
3. Automobile
   a) Rennwagen
   b) Personenwagen
   c) Lastwagen
4. Fahrräder

Innerhalb der Aufzählung können ganz normal mit `\label` Marken gesetzt werden, auf die dann mit `\ref` zugegriffen werden kann. So wurde oben hinter den düsengetriebenen, einmotorigen Flugzeugen mit „`\label{bsp:duesen}`“ ein Label gesetzt. Der `\ref`-Wert ist dann „1(c)iA“.

```
description
\item[Stichwort]
\descfont
```

Eine weitere Listenform ist die Stichwortliste. Sie dient in erster Linie der Beschreibung einzelner Stichworte. Das Stichwort selbst wird als optionaler Parameter bei `\item` angegeben. Die Schriftart, die für die Hervorhebung des Stichworts verwendet wird, ist bei KOMA-Script im Makro `\descfont` abgelegt. Die Standarddefinition:

```
\newcommand*{\descfont}{\sffamily\bfseries}
```

kann mit Hilfe von `\renewcommand` geändert werden.

Beispiel: Sie wollen, dass die Stichworte statt serifenlos und fett lediglich fett aber in der Standardschriftart ausgegeben werden. Mit

```
\renewcommand*{\descfont}{\normalfont\bfseries}
```

definieren Sie daher das Schriftmakro entsprechend um.

Ein Beispiel für die Ausgabe einer Stichwortliste ist die Aufzählung der Seitenstile in Abschnitt 3.2. Der Quelltext dazu lautet (stark gekürzt):

```
\begin{description}
  \item[empty] ist der Seitenstil, bei dem Kopf- und
       Fußzeile vollständig leer bleiben.
  \item[plain] ist der Seitenstil, bei dem keinerlei
       Kolumnentitel verwendet wird.
  \item[headings] ist der Seitenstil für automatische
       Kolumnentitel.
  \item[myheadings] ist der Seitenstil für manuelle
       Kolumnentitel.
\end{description}
```

Diese gekürzte Version ergibt:

empty ist der Seitenstil, bei dem Kopf- und Fußzeile vollständig leer bleiben.

plain ist der Seitenstil, bei dem keinerlei Kolumnentitel verwendet wird.

headings ist der Seitenstil für automatische Kolumnentitel.

myheadings ist der Seitenstil für manuelle Kolumnentitel.

`labeling[`*Trennzeichen*`]{`*längstes Muster*`}`
`\item[`*Schlüsselwort*`]`

Eine andere Form der Stichwortliste ist bei KOMA-Script mit der `labeling`-Umgebung gegeben. Im Unterschied zur `description`-Umgebung kann hier ein Muster angegeben werden, dessen Länge die Einrücktiefe bei allen Stichpunkten ergibt. Darüber hinaus kann zwischen Stichpunkt und Beschreibungstext ein optionales *Trennzeichen* festgelegt werden.

Beispiel: Wir schreiben das Beispiel der `description`-Umgebung etwas um:

```
\begin{labeling}[~--]{\descfont myheadings}
  \item[\descfont empty] Seitenstil für leere Seiten ohne
       Kopf und Fuß
  \item[\descfont plain] Seitenstil für Kapitelanfänge ohne
       Kolumnentitel
  \item[\descfont headings] Seitenstil für automatische
       Kolumnentitel
  \item[\descfont myheadings] Seitenstil für manuelle
       Kolumnentitel
\end{labeling}
```

Als Ergebnis erhalten wir dann:

empty	–	Seitenstil für leere Seiten ohne Kopf und Fuß
plain	–	Seitenstil für Kapitelanfänge ohne Kolumnentitel
headings	–	Seitenstil für automatische Kolumnentitel
myheadings	–	Seitenstil für manuelle Kolumnentitel

Wie in diesem Beispiel zu sehen ist, muss eine eventuell geforderte Schriftumschaltung bei dieser Umgebung sowohl im Muster als auch im optionalen Parameter jeder `\item`-Anweisung wiederholt werden.

Gedacht war die Umgebung ursprünglich für Strukturen wie „Voraussetzung, Aussage, Beweis“ oder „Gegeben, Gesucht, Lösung“ wie man sie in Vorlesungsskripten häufiger findet. Inzwischen findet die Umgebung aber ganz unterschiedliche Anwendungen. So wurde die Umgebung für Beispiele in dieser Anleitung mit Hilfe der `labeling`-Umgebung definiert.

`verse`

Die `verse`-Umgebung wird normalerweise nicht als Listenumbegung wahrgenommen, da hier nicht mit `\item` gearbeitet wird. Stattdessen wird wie innerhalb der `flushleft`-Umgebung mit festen Zeilenumbrüchen gearbeitet. Intern handelt es sich jedoch sowohl bei den Standardklassen als auch bei KOMA-Script durchaus um eine Listenumgebung.

Die `verse`-Umgebung findet hauptsächlich für Gedichte Anwendung. Dabei werden die Zeilen links und rechts eingezogen. Einzelne Verse werden mit einen festen Zeilenumbruch, also mit `\\` beendet. Strophen werden ganz normal als Absatz gesetzt, also durch eine Leerzeile getrennt. Häufig findet stattdessen auch `\medskip` oder `\bigskip` Verwendung. Will man verhindern, dass am Ende eines Verses ein Seitenumbruch erfolgt, so verwendet man ganz normal `\\*` an Stelle von `\\`.

Beispiel: Als Beispiel ein kurzes Gedicht von Willhelm Busch:

```
\begin{verse}
  Wenn einer, der mit Mühe kaum\\*
  Gekrochen ist auf einen Baum,\\*
  Schon meint, dass er ein Vogel wär,\\*
```

```
    So irrt sich der.
  \end{verse}
```

Mit dem Resultat:

> Wenn einer, der mit Mühe kaum
> Gekrochen ist auf einen Baum,
> Schon meint, dass er ein Vogel wär,
> So irrt sich der.

Bei einem sehr langen Vers kann mit `\\*` allerdings nicht verhindert werden, dass innerhalb des Verses ein Seitenumbruch erfolgt. Dies wäre beispielsweise hier möglich:

> Der Philosoph wie der Hausbesitzer hat immer Reparaturen.
>
> Wer dir sagt, er hätte noch nie gelogen, dem traue nicht, mein Sohn.

Hier noch zwei Sprüche, die man immer bedenken sollte, wenn man mit scheinbar seltsamen Fragen zu LaTeX oder den dazugehörigen Antworten konfrontiert ist:

> Wir mögen's keinem gerne gönnen,
> Dass er was kann, was wir nicht können.
>
> Wie klein ist das, was einer ist,
> Wenn man's mit seinem Dünkel misst.

Hier fand übrigens `\bigskip` Anwendung, um die beiden Sprüche voneinander zu trennen.

`quote`
`quotation`

Diese beiden Umgebungen sind intern ebenfalls Listenumgebungen und sowohl bei den Standardklassen als auch bei KOMA-Script zu finden. Beide

Umgebungen setzen Blocksatz, der sowohl rechts als auch links eingezogen ist. Verwendet werden die Umgebungen häufig, um längere Zitate abzusetzen. Der Unterschied zwischen beiden liegt in der Art und Weise, wie Absätze abgesetzt werden. Während bei `quote` Absätze durch vertikalen Abstand gekennzeichnet werden, wird bei `quotation` mit horizontalem Einzug der ersten Zeile eines Absatzes gearbeitet. Dies gilt auch für den ersten Absatz einer `quotation`-Umgebung. Wollen Sie dort den Einzug verhindern, müssen Sie die `\noindent`-Anweisung voranstellen.

Beispiel: Sie wollen eine kleine Anekdote hervorheben. Also schreiben Sie folgende `quotation`-Umgebung:

```
Ein kleines Beispiel für eine Anekdote:
\begin{quotation}
  Es klingelt an der Tür eines Pfarrhauses in Stuttgart.
  Als die Haushälterin öffnet, steht ein unrasierter Mann
  in einem reichlich schäbigen Mantel vor der Tür und hält
  seine Strickmütze in der Hand.

  "‘Gute Frau, ich habe seit drei Tagen nichts gegessen"’,
  verkündert der Mann in bestem Hochdeutsch aber recht
  gequältem Ton, der bestens zu seinem Gesichtsausdruck
  passt.

  Die Frau schüttelt mitleidig den Kopf.

  "‘Guda Moh, Sie missat sich halt zwinga!"’
\end{quotation}
```

Das Ergebnis ist dann:

Ein kleines Beispiel für eine Anekdote:

> Es klingelt an der Tür eines Pfarrhauses in Stuttgart. Als die Haushälterin öffnet, steht ein unrasierter Mann in einem reichlich schäbigen Mantel vor der Tür und hält seine Strickmütze in der Hand.
>
> „Gute Frau, ich habe seit drei Tagen nichts gegessen“, verkündert der Mann in bestem Hochdeutsch aber recht gequältem Ton, der bestens zu seinem Gesichtsausdruck passt.

Die Frau schüttelt mitleidig den Kopf.
„Guda Moh, Sie missat sich halt zwinga!“

Wenn Sie stattdessen eine `quote`-Umgebung verwenden, sieht das ganze so aus:

Ein kleines Beispiel für eine Anekdote:

> Es klingelt an der Tür eines Pfarrhauses in Stuttgart. Als die Haushälterin öffnet, steht ein unrasierter Mann in einem reichlich schäbigen Mantel vor der Tür und hält seine Strickmütze in der Hand.
>
> „Gute Frau, ich habe seit drei Tagen nichts gegessen“, verkündert der Mann in bestem Hochdeutsch aber recht gequältem Ton, der bestens zu seinem Gesichtsausdruck passt.
>
> Die Frau schüttelt mitleidig den Kopf.
>
> „Guda Moh, Sie missat sich halt zwinga!“

Erlauben Sie mir noch eine abschließende Bemerkung zu den Listenumgebungen. Im Internet und im Support wird häufig danach gefragt, warum nach einer Listenumgebung ein Einzug wie bei einem Absatz erfolge. Tatsächlich ist dies gar nicht der Fall, sondern der Effekt resultiert daraus, dass der Anwender einen Absatz verlangt. Bei LaTeX werden Leerzeilen als Absatz interpretiert. Dies gilt auch vor und nach Listenumgebungen. Soll eine Listenumgebung also innerhalb eines Absatzes gesetzt werden, so ist weder davor noch danach eine Leerzeile zu setzen. Um die Umgebung trotzdem in der LaTeX-Datei besser vom Rest abzusetzen, kann man davor und dahinter eine leere Kommentarzeile setzen, also eine Zeile, die direkt mit einem Prozentzeichen beginnt und nichts weiter enthält.

3.5.5 Randnotizen

`\marginpar[`*Randnotiz links*`]{`*Randnotiz*`}`
`\marginline{`*Randnotiz*`}`

Randnotizen werden normalerweise bei LaTeX mit der Anweisung `\marginpar` gesetzt. Sie werden dabei im äußeren Rand gesetzt. Bei einseitigen Dokumenten wird der rechte Rand verwendet. Zwar kann bei `\marginpar` optional eine abweichende Randnotiz angegeben werden, falls die Randnotiz im linken Rand landet, jedoch werden Randnotizen immer im Blocksatz ausgegeben. Die Erfahrung zeigt, dass bei Randnotizen statt dem Blocksatz oft je nach Rand linksbündiger oder rechtsbündiger Flattersatz zu bevorzugen ist. KOMA-Script bietet hierfür die Anweisung `\marginline`.

Beispiel: In der Einleitung ist an einer Stelle die Klassenangabe scrartcl im Rand zu finden. Diese kann mit:

```
\marginline{\texttt{scrartcl}}
```

erreicht werden.[2]

Statt `\marginline` wäre auch die Verwendung von `\marginpar` möglich gewesen. Tatsächlich wird bei obiger Anweisung intern nichts anders gemacht als:

```
\marginpar[\raggedleft\texttt{scrartcl}]
  {\raggedright\texttt{scrartcl}}
```

Letztlich ist `\marginline` also nur eine abkürzende Schreibweise.

Leider funktioniert `\marginpar` im doppelseitigen Layout nicht immer ganz korrekt. Die Entscheidung, ob eine Randnotiz links oder rechts landet, wird bereits bei der Auswertung von `\marginpar` getroffen. Verschiebt nun die Ausgaberoutine eine Randnotiz über einen Seitenumbruch auf die nächste Seite, so ist die Formatierung nicht mehr korrekt. Dieses Verhalten ist tief in LaTeX verankert und wurde vom LaTeX3-Team deshalb als Feature deklariert. `\marginline` ändert nichts an diesem unerwünschten Verhalten.

3.5.6 Tabellen und Abbildungen

LaTeX bietet mit den Fließumgebungen einen sehr leistungsfähigen und komfortablen Mechanismus für die automatische Anordnung von Abbildungen und Tabellen. Genaugenommen sollte statt von „Tabellen" von „Tafeln" die Rede sein. Dies wäre auch zur Unterscheidung der Umgebungen `table` und `tabular` von Vorteil. Es hat sich im Deutschen aber für beides die Bezeichnung „Tabelle" eingebürgert. Das kommt vermutlich daher, dass man in `table`-Umgebungen üblicherweise `tabular`-Umgebungen setzt.

Häufig werden die Fließumgebungen von Anfängern nicht richtig verstanden. So wird oft die Forderung aufgestellt, eine Tabelle oder Abbildung genau an einer bestimmten Position im Text zu setzen. Dies ist jedoch nicht erforderlich, da auf Fließumgebungen im Text über eine Referenz verwiesen wird. Es ist auch nicht sinnvoll, da eine solches

[2] Tatsächlich wurde nicht `\texttt`, sondern eine semantische Auszeichnung verwendet. Um nicht unnötig zu verwirren, wurde diese im Beispiel ersetzt.

Objekt an einer Stelle nur dann gesetzt werden kann, wenn auf der Seite noch genügend Platz für das Objekt vorhanden ist. Ist dies nicht der Fall, müsste das Objekt auf die nächste Seite umbrochen werden und auf der aktuellen Seite würde ein möglicherweise sehr großer leerer Raum bleiben.

Häufig findet sich in einem Dokument auch bei jedem Fließobjekt das gleiche optionale Argument zur Platzierung des Objekts. Auch dies ist nicht sinnvoll. In solchen Fällen sollte man besser den Standardwert global ändern. Näheres dazu ist [RH01] zu entnehmen.

Ein wichtiger Hinweis sei diesem Abschnitt noch vorangestellt: Die meisten Mechanismen, die hier vorgestellt werden und über die Fähigkeiten der Standardklassen hinaus gehen, funktionieren nicht mehr, wenn Sie ein Paket verwenden, das in die Generierung von Tabellen- und Abbildungstiteln eingreift und deren Aussehen verändert. Dies sollte selbstverständlich sein, wird aber leider häufig nicht bedacht.

```
\caption[Verzeichniseintrag]{Titel}
\captionbelow[Verzeichniseintrag]{Titel}
\captionabove[Verzeichniseintrag]{Titel}
```

Tabellen und Abbildungen werden bei den Standardklassen mit Hilfe der Anweisung `\caption` mit einem *Titel* in Form einer Unterschrift versehen. Bei Abbildungen ist dies grundsätzlich korrekt. Bei Tabellen wird gestritten, ob der *Titel* als Überschrift über oder konsistent mit der Bildunterschrift unter die Tabelle gehört. Daher bietet KOMA-Script im Gegensatz zu den Standardklassen die Anweisungen `\captionbelow` für *Titel* in Form von Unterschriften und `\captionabove` für *Titel* in Form von Überschriften. `\caption` verhält sich bei Abbildungen immer wie `\captionbelow`. Bei Tabellen lässt sich das Verhalten von `\caption` hingegen mit den Optionen `tablecaptionabove` und `tablecaptionbelow` steuern (siehe Abschnitt 3.1.5). Aus Gründen der Kompatibilität ist voreingestellt, dass sich `\caption` auch bei Tabellen wie `\captionbelow` verhält.

Beispiel: Sie wollen mit Tabellenüberschriften statt mit Tabellenunterschriften arbeiten, weil Sie teilweise Tabellen haben, die über mehr als eine Seite gehen. Mit den Standardklassen bliebe Ihnen nur die Möglichkeit:

```
\begin{table}
  \caption{Dies ist nur eine Beispieltabelle}
  \begin{tabular}{llll}
    Dies & ist & ein & Beispiel.\\\hline
    Bitte & lassen & Sie & den \\
    Inhalt & dieser & Tabelle & unbeachtet.
  \end{tabular}
```

```
\end{table}
```

Damit hätten Sie das unschöne Ergebnis:

Tabelle 30.2: Dies ist nur eine Beispieltabelle

Dies	ist	ein	Beispiel.
Bitte	lassen	Sie	den
Inhalt	dieser	Tabelle	unbeachtet.

Bei KOMA-Script schreiben Sie hingegen:

```
\begin{table}
  \captionabove{Dies ist nur eine Beispieltabelle}
  \begin{tabular}{llll}
    Dies & ist & ein & Beispiel.\\\hline
    Bitte & lassen & Sie & den \\
    Inhalt & dieser & Tabelle & unbeachtet.
  \end{tabular}
\end{table}
```

Sie erhalten dann das gewünschte Ergebnis:

Tabelle 30.2: Dies ist nur eine Beispieltabelle

Dies	ist	ein	Beispiel.
Bitte	lassen	Sie	den
Inhalt	dieser	Tabelle	unbeachtet.

Da Sie konsistent alle Tabellen mit Überschriften versehen, können Sie stattdessen natürlich auch die Option `tablecaptionabove` setzen (siehe Abschnitt 3.1.5). Dann genügt es, wenn Sie wie bei den Standardklassen `\caption` verwenden. Sie erhalten trotzdem das Ergebnis von `\captionabove`.

Einige werden nun einwenden, dass man das gleiche auch mit dem topcapt-Paket und der dort definiert Anweisung `\topcaption` erreichen kann (siehe [Fai99]). Dies ist jedoch nicht der Fall. `\topcaption` bleibt von Paketen, die direkt das `\caption`-Makro umdefinieren unberücksichtigt. Ein Beispiel dafür ist das hyperref-Paket (siehe [Rat01]).

Demgegenüber sind `\captionabove` und `\captionbelow` so implementiert, dass sich die Änderungen auch auf diese beiden Anweisungen auswirken.

Bei Verwendung des longtable-Pakets wird dafür gesorgt, dass auch die Tabellenüberschriften, die innerhalb einer `longtable`-Umgebung gesetzt werden, in Aussehen und Form denen einer normalen `table`-Umgebung entsprechen. Es gelten damit auch dieselben Einstellmöglichkeiten. Bitte beachten Sie, dass beim longtable-Paket die maximale Breite einer Tabellenüberschrift begrenzt werden kann und per Voreinstellung auf 4 in begrenzt ist. Siehe dazu [Car98]. Ist das Paket caption2 (siehe [Som95]) geladen, so wird werden die Tabellenüberschriften von jenem Paket behandelt. Siehe außerdem die Erklärung zur Option `origlongtable` in Abschnitt 3.1.5.

Bitte beachten Sie, dass sich `\captionabove` und `\captionbelow` innerhalb einer `float`-Umgebung, die mit Hilfe des float-Pakets definiert wurde, genau wie in [Lin01] für die `\caption`-Anweisung beschrieben verhalten. In diesem Fall kontrolliert allein der float-Stil, ob es sich um eine Überschrift oder um eine Unterschrift handelt.

```
komaabove
komabelow
```

float Bei Verwendung des float-Pakets wird das Aussehen der damit definierten Fließumgebungen allein vom *float*-Stil bestimmt. Dies schließt auch die Frage ein, ob mit Überschriften oder Unterschriften gearbeitet wird. Im float-Paket gibt es keinen vordefinierten Stil, der im Aussehen dem von KOMA-Script entspricht und dieselben Einstellmöglichkeiten (siehe unten) bietet. KOMA-Script definiert deshalb zusätzlich die beiden Stile `komaabove` und `komabelow`. Diese können bei Verwendung des float-Pakets wie die dort definierten Stile `plain`, `boxed` oder `ruled` aktiviert werden. Siehe dazu [Lin01]. Beim Stil `komaabove` werden `\caption`, `\captionabove` und `\captionbelow` als Überschrift gesetzt, beim Stil `komabelow` als Unterchrift.

```
\capfont
\caplabelfont
\captionformat
```

Bei KOMA-Script gibt es verschiedene Eingriffsmöglichkeiten, um die Formatierung der Beschreibung zu ändern. Die Schriftart für die Beschreibung ist im Makro `\capfont` abgelegt. Die Originaldefinition dafür lautet:

```
\newcommand*{\capfont}{\normalfont}
```

Es wird also mit der Standardschriftart gearbeitet. Diese Voreinstellung kann mit Hilfe von `\renewcommand` umdefiniert werden. Dabei sind auch Größenangaben möglich.

Abweichend von der generellen Schriftart der Beschreibung kann das Label – „Abbildung“ oder „Tabelle“ gefolgt von der Nummer und einem Trennzeichen – in einer anderen Schriftart gesetzt werden. Diese ist im Makro `\caplabelfont` als:

```
\newcommand*{\caplabelfont}{\normalfont}
```

vordefiniert. Auch dieses Makro kann mit `\renewcommand` umdefiniert werden.

Das oder die Trennzeichen zwischen dem Label und dem eigentlichen Beschreibungstext ist im Makro `\captionformat` abgelegt. Abweichend von allen anderen `\...`format-Anweisungen ist hier also nicht der Zähler enthalten, sondern nur die auf den Zähler folgenden Angaben. Die Originaldefinition lautet:

```
\newcommand*{\captionformat}{:\ }
```

Auch diese kann mit `\renewcommand` geändert werden.

Beispiel: Sie wollen, dass Tabellen- und Abbildungsbeschreibungen in einer kleineren Schriftart gesetzt werden. Also definieren Sie beispielsweise in der Präambel Ihres Dokuments:

```
\renewcommand*{\capfont}{\normalfont\small}
```

Außerdem hätten Sie gerne, dass das Label serifenlos und fett gedruckt wird. Sie definieren also weiter:

```
\renewcommand*{\caplabelfont}{\sffamily\bfseries}
```

Aus mir unerfindlichen Gründen wollen Sie außerdem, dass als Trennzeichen kein Doppelpunkt gefolgt von einem Leerzeichen, sondern ein Gedankenstrich einschließlich der notwendigen Leerzeichen gesetzt wird. Daher definieren Sie zusätzlich noch:

```
\renewcommand*{\captionformat}{~--~}
```

Damit sind Sie dann wunschlos glücklich.

```
\figureformat
\tableformat
```

Es wurde schon darauf hingewiesen, dass `\captionformat` keine Formatierung für das Label selbst enthält. Dieses sollte nun keineswegs über Umdefinierung der Anweisungen für die Zählerausgabe, `\thefigure` oder `\thetable`, verändert werden. Eine solche Umdefinierung hätte nämlich auch Auswirkungen auf die Ausgabe von `\ref` oder der Verzeichnisse. Stattdessen bietet KOMA-Script auch hier zwei `\...format`-Anweisungen. Diese sind wie folgt vordefiniert:

```
\newcommand*{\figureformat}{\figurename~\thefigure\autodot}
\newcommand*{\tableformat}{\tablename~\thetable\autodot}
```

Sie können ebenfalls mit `\renewcommand` eigenen Anforderungen angepasst werden.

Beispiel: Hin und wieder wird gewünscht, dass die Beschreibungstexte ganz ohne Label und natürlich auch ohne Trennzeichen ausgegeben werden. Bei KOMA-Script genügen folgende Definitionen, um dies zu erreichen:

```
\renewcommand*{\figureformat}{}
\renewcommand*{\tableformat}{}
\renewcommand*{\captionformat}{}
```

```
\setcapindent{Einzug}
\setcapindent*{XEinzug}
\setcaphanging
```

Wie bereits erwähnt wurde, werden in den Standardklassen die Beschreibungen nicht hängend gesetzt. Das heißt: In mehrzeiligen Beschreibungen beginnt die zweite Zeile direkt unter dem Labeltext. Es gibt bei den Standardklassen auch keinen Mechanismus, dies direkt zu beeinflussen. Bei KOMA-Script werden hingegen alle Zeilen ab der zweiten so weit eingerückt, dass diese nicht mehr unter dem Label, „Abbildung …:“ oder „Tabelle …:“, sondern unter dem eigentlichen Text der ersten Zeile beginnen.

Dieses Verhalten, das der Verwendung der Anweisung `\setcaphanging` entspricht, kann bei KOMA-Script jederzeit durch Verwendung der Anweisung `\setcapindent` oder `\setcapindent*` geändert werden. Dabei gibt der Parameter *Einzug* an, wie weit ab der zweiten Zeile eingerückt werden soll.

Soll nach dem Label und vor dem Beschreibungstext noch ein Zeilenumbruch erfolgen, so definieren Sie die Einrücktiefe *XEinzug* der Beschreibung stattdessen mit der Sternvariante der Anweisung: `\setcapindent*`.

Mit einem negativen *Einzug* erreicht man hingegen, dass vor der Beschreibung ebenfalls ein Umbruch erfolgt und nur die erste Zeile der Beschreibung, nicht jedoch die folgenden, um −*Einzug* eingerückt werden.

Beispiel: Die Beispiele entnehmen Sie bitte den Abbildungen 3.1 bis 3.4. Dabei zeigt sich auch, dass bei geringer Spaltenbreite der komplett hängende Einzug unvorteilhaft ist. Der Quelltext der zweiten Abbildung sei hier mit abgewandelter Unterschrift beispielhaft wiedergegeben:

```
\begin{figure}
  \setcapindent{1em}
  \fbox{\parbox{.95\linewidth}{\centering{\KOMAScript}}}
```

KOMA-Script

Abbildung 3.1: Entspricht der Standardeinstellung, also wie bei Verwendung von `\setcaphanging`

KOMA-Script

Abbildung 3.2: Mit teilweise hängendem Einzug ab der zweiten Zeile durch Verwendung von `\setcapindent{1em}`

KOMA-Script

Abbildung 3.3:
Mit hängendem Einzug ab der zweiten Zeile und Umbruch vor der Beschreibung durch Verwendung von `\setcapindent*{1em}`

KOMA-Script

Abbildung 3.4:
Mit Einzug lediglich in der zweiten Zeile und einem Umbruch vor der Beschreibung durch Verwendung von `\setcapindent{-1em}`

```
  \caption{Beispiel mit teilweise hängendem Einzug ab der
    zweiten Zeile}
\end{figure}
```

Wie zu sehen ist, kann die Formatierung also auch lokal innerhalb der `figure`-Umgebung geändert werden.

3.5.7 Textauszeichnung

LaTeX bietet verschiedene Möglichkeiten der Textauszeichnung. Streng genommen stellt eine Überschrift ebenfalls eine Auszeichnung dar. In diesem Abschnitt beschäftigen wir uns aber ausschließlich mit unmittelbaren Auszeichnungen, also solchen, die keine zusätzliche Bedeutung in sich besitzt, sondern die für verschiedene Zwecke verwendet werden können. Näheres zu den normalerweise definierten Möglichkeiten sind [SKPH99], [Tea99a] und [Tea00] zu entnehmen.

`\textsubscript{`*Text*`}`

In Abschnitt 3.5.3 wurde bereits die Anweisung `\textsuperscript` vorgestellt, die Bestandteil des LaTeX-Kerns ist. Leider bietet LaTeX selbst keine entsprechende Anweisung, um Text tief statt hoch zu stellen. KOMA-Script definiert dafür `\textsubscript`.

Beispiel: Sie schreiben einen Text über den menschlichen Stoffwechsel. Darin kommen hin und wieder einfache chemische Strukturformeln vor. Dabei sind einzelne Ziffern tief zu stellen. Im Sinne des logischen Markup definieren Sie zunächst in der Dokumentpräambel oder einem eigenen Paket:

```
\newcommand*{\Molek}[2]{#1\textsubscript{#2}}
```

Damit schreiben Sie dann:

```
Die Zelle bezieht ihr Energie unter anderem aus der
Reaktion von \Molek C6\Molek H{12}\Molek O6 und \Molek O2
zu \Molek H2\Molek O{} und \Molek C{}\Molek O2.
Arsen (\Molek{As}{}) wirkt sich allerdings auf den
Stoffwechsel sehr nachteilig aus.
```

Das Ergebnis sieht daraufhin so aus:

> Die Zelle bezieht ihr Energie unter anderem aus der Reaktion von $C_6H_{12}O_6$ und O_2 zu H_2O und CO_2. Arsen (As) wirkt sich allerdings auf den Stoffwechsel sehr nachteilig aus.

Etwas später entscheiden Sie, dass Strukturformeln grundsätzlich serifenlos geschrieben werden sollen. Nun zeigt sich, wie gut die Entscheidung für konsequentes logische Markup war. Sie müssen nur die `\Molek`-Anweisung umdefinieren:

```
\newcommand*{\Molek}[2]{\textsf{#1\textsubscript{#2}}}
```

Schon ändert sich die Ausgabe im gesamten Dokument:

> Die Zelle bezieht ihr Energie unter anderem aus der Reaktion von $\mathsf{C_6H_{12}O_6}$ und $\mathsf{O_2}$ zu $\mathsf{H_2O}$ und $\mathsf{CO_2}$. Arsen (As) wirkt sich allerdings auf den Stoffwechsel sehr nachteilig aus.

Im Beispiel wird die Schreibweise „`\Molek C6`“ verwendet. Dabei wird Nutzen aus der Tatsache gezogen, dass Argumente, die nur aus einem Zeichen bestehen, nicht geklammert werden müssen. Damit ist „`\Molek C6`“ gleichbedeutend mit „`\Molek{C}{6}`“. Bekannt ist dieser Umstand hauptsächlich von Indizes und Potenzen in mathematischen Umgebungen, etwa „`$x^2$`“ statt „`$x^{2}$`“ für „x^2“.

3.6 Schlussteil

Im Schlussteil eines Dokuments finden sich üblicherweise die Anhänge, das Literaturverzeichnis und gegebenenfalls ein Stichwortverzeichnis.

`\appendix`

Der Anhang wird in den Standardklassen und den KOMA-Script-Klassen mit der Anweisung `\appendix` eingeleitet. Diese Anweisung schaltet unter anderem die Kapitelnummerierung auf Großbuchstaben um und sorgt gleichzeitig dafür, dass die Regeln für die Nummerierung der Gliederungsebenen nach [DUD96] eingehalten werden. Diese Regeln sind in der Beschreibung der Klassenoptionen `pointednumbers` und `pointlessnumbers` in Abschnitt 3.1.5 näher erläutert.

Bitte beachten Sie, dass es sich bei `\appendix` um eine Anweisung und *nicht* um eine Umgebung handelt! Die Anweisung erwartet auch nicht etwa ein Argument. Die Kapitel beziehungsweise Abschnitte des Anhangs werden ganz normal mit `\chapter` und `\section` gesetzt.

`\appendixmore`

Bei den KOMA-Script-Klassen gibt es innerhalb der Anweisung `\appendix` eine Besonderheit. Ist nämlich die Anweisung `\appendixmore` definiert, so wird sie von der `\appendix`-Anweisung ebenfalls ausgeführt. Intern wird dies von den KOMA-Script-Klassen scrbook und scrreprt für die Realisierung der Layoutoptionen `appendixprefix` und `noappendixprefix` genutzt (siehe Abschnitt 3.1.2). Dies sollten Sie unbedingt beachten, falls Sie selbst das Makro `\appendixmore` definieren oder umdefinieren wollen. Ist eine dieser beiden Optionen gesetzt, so erhalten Sie bei `\newcommand{\appendixmore}{...}` eine Fehlermeldung. Dadurch soll verhindert werden, dass Sie die Optionen außer Kraft setzen, ohne es zu merken.

Beispiel: Sie wollen nicht, dass bei Verwendung der Klasse scrbook oder scrreprt im Hauptteil die Kapitel mit einer Präfixzeile versehen werden (siehe Layoutoptionen `chapterprefix` und `nochapterprefix` in Abschnitt 3.1.2). Damit die Konsitenz gewahrt bleibt, wollen Sie auch nicht, dass eine solche Zeile im Anhang verwendet wird. Stattdessen sollen in den Anhängen direkt vor dem Kapitelbuchstaben das Wort „Anhang“ in der jeweiligen Sprache stehen. Dies soll auch für die Kolumnentitel gelten. Also verwenden Sie weder die Layoutoption `appendixprefix` noch `noappendixprefix`, sondern definieren in der Dokumentpräambel:

```
\newcommand*{\appendixmore}{%
```

```
    \renewcommand*{\chapterformat}{%
      \appendixname~\thechapter\autodot\enskip}
    \renewcommand*{\chaptermarkformat}{%
      \appendixname~\thechapter\autodot\enskip}
  }
```

Sollten Sie später dann doch noch entscheiden, dass Sie die Option `appendixprefix` verwenden wollen, so erhalten Sie aufgrund der dann bereits definierten Anweisung `\appendixmore` eine Fehlermeldung. Damit wird verhindert, dass obige Definition unbemerkt die Einstellungen überschreibt, die Sie per Option getroffen haben.

Wenn Sie ein vergleichbares Verhalten des Anhangs für die Klasse scrartcl erreichen wollen, so ist dies ebenfalls möglich. Dazu schreiben Sie in die Präambel Ihres Dokuments:

```
  \newcommand*{\appendixmore}{%
    \renewcommand*{\othersectionlevelsformat}[1]{%
      \ifthenelse{\equal{#1}{section}}{\appendixname~}{}%
      \csname the#1\endcsname\autodot\enskip}
    \renewcommand*{\sectionmarkformat}{%
      \appendixname~\thesection\autodot\enskip}
  }
```

Sie benötigen dafür außerdem das ifthen-Paket (siehe [Car99a]).

Die Erklärung zu den umdefinierten Anweisungen finden Sie in Abschnitt 3.5.2.

`\setbibpreamble{`*Präambel*`}`

Mit der Anweisung `\setbibpreamble` kann eine Präambel für das Literaturverzeichnis gesetzt werden. Bedingung dafür ist, dass die Präambel vor der Anweisung zum Setzen des Literaturverzeichnisses gesetzt wird. Dies muss nicht unmittelbar davor sein. Es kann also beispielsweise am Anfang des Dokuments erfolgen. Ebenso wie die Klassenoptionen `bibtotoc` und `bibtotocnumbered` kann die Anweisung aber nur erfolgreich sein, wenn nicht ein Paket geladen wird, das dies durch Umdefinierung der `thebibliography`-Umgebung verhindert. Obwohl das natbib-Paket unautorisiert interne Makros von KOMA-Script verwendet, konnte erreicht werden, dass `\setbibpreamble` auch mit der aktuellen Version von natbib funktioniert (siehe [Dal99]).

Beispiel: Sie wollen darauf hinweisen, dass das Literaturverzeichnis nicht in der Reihenfolge der Zitierung im Dokument, sondern alphabetisch sortiert ist. Daher setzen Sie folgende Anweisung:

```
\setbibpreamble{Die Literaturangaben sind alphabetisch nach
  den Namen der Autoren sortiert. Bei mehreren Autoren wird
  nach dem ersten Autor sortiert.\par\bigskip}
```

Die Anweisung `\bigskip` sorgt dafür, dass zwischen der Präambel und der ersten Literaturangabe ein großer Zwischenraum gesetzt wird.

Eine andere Möglichkeit, die Präambel des Literaturverzeichnisses nutzbringend einzusetzen, ergäbe sich, wenn Sie beispielsweise die Literaturangaben mit linksbündigem Flattersatz setzen wollten. Dann setzen Sie als Präambel einfach:

```
\setbibpreamble{\raggedright}
```

Das Ergebnis sehen Sie im Literaturverzeichnis dieser Anleitung.

`\setindexpreamble`

Analog zur Präambel des Literaturverzeichnisses können Sie auch das Stichwortverzeichnis mit einer Präambel versehen. Dies findet häufig dann Anwendung, wenn es mehr als einen Index gibt oder im Index unterschiedliche Arten der Referenzierung durch unterschiedliche Hervorhebung der Seitenzahlen markiert werden.

Beispiel: Sie haben ein Dokument, in dem Begriffe sowohl definiert als auch verwendet werden. Die Seitenzahlen der Begriffsdefinitionen sind fett dargestellt. Natürlich möchten Sie gerne auf diesen Umstand hinweisen. Also setzen Sie eine entsprechende Präambel für den Index:

```
\setindexpreamble{Alle \textbf{fett} gedruckten
  Seitenzahlen sind Referenzen auf die Definition des
  jeweiligen Begriffs. Demgegenüber geben normal gedruckte
  Seitenzahlen die Seiten der Verwendung des jeweiligen
  Begriffs wieder.\par\bigskip}
```

Bitte beachten Sie, dass für die erste Seite des Index der Seitenstil umgeschaltet wird. Welcher Seitenstil hierbei Verwendung findet, ist im Makro `\indexpagestyle` abgelegt (siehe Abschnitt 3.2).

Für die Erstellung, Sortierung und Ausgabe des Stichwortverzeichnisses sind die üblichen Standard-LaTeX-Pakete und Zusatzprogramme zuständig. Von KOMA-Script werden genau wie von den Standardklassen lediglich die grundlegenden Makros und Umgebungen dafür zur Verfügung gestellt.

3.7 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- Frank Neukam
- **Markus Kohm** <Markus.Kohm@gmx.de>
- Axel Sommerfeldt

4 Kopf- und Fußzeilen mit scrpage2

Wie bereits in Kapitel 2 zur Bestimmung des Satzspiegels hingewiesen wurde, stellt KOMA-Script ein Werkzeug zur Verfügung, um die im Layout definierten Freiräume für Kopf- und Fußzeilen auch mit Inhalt zu füllen.

Dazu dient in KOMA-Script das Paket scrpage2, das eine stark erweiterte Variante des originalen scrpage darstellt. Verglichen mit dem häufig anzutreffenden Paket fancyhdr[Oos00] ist scrpage2 extrem flexibel in seiner Gestalt und in seiner Benutzung. Weiterhin ist es durch seine Nähe zu den KOMA-Script-Klassen dort gut integriert und somit eine perfekte Ergänzung der Standardfunktionalität.

Für scrpage2 und scrpage gilt, dass sie nicht an die KOMA-Script-Klassen gebunden sind. Sie können mit jeder anderen Dokumentenklasse, zum Beispiel den Standardklassen eingesetzt werden. Für neue Dokumente sollte unbedingt das neue Paket scrpage2 benutzt werden – die alte Version steht aus Gründen der Kompatibilität jedoch weiterhin zur Verfügung. Dieses Kapitel fokussiert auf scrpage2. Befehle, die in gleicher Form bei scrpage vorkommen, werden als solche markiert (siehe Rand). Treten in der neuen Variante Abweichungen *scrpage*
auf, wird im Text an markierten Stellen darauf hingewiesen. *! scrpage*

Um die nachfolgende Beschreibung zu verstehen, muss noch einiges zu LaTeX gesagt werden. Grundlegend definiert der LaTeX-Kern die Standardseitenstile `empty`, welcher eine völlig undekorierte Seite erzeugt und `plain`, welcher meist nur die Seitenzahl enthält. Weiterhin ist in vielen Klassen der Stil `headings` zu finden, welcher eine komplexe Seitendekoration erzeugt, wobei es hier noch eine Untervariante, die *my*-Variante, gibt. Dieser Seitenstil `myheadings` ist wie `headings` gestaltet, jedoch müssen die sonst automatisch aktualisierten Kolumnentitel manuell gesetzt werden. Ausführlicher wird das im Abschnitt 3.2 beschrieben. Weiterhin ist zu beachten, dass einige LaTeX-Befehle temporär, dass heißt für die aktuelle Seite, auf den Seitenstil `plain` wechseln, auch wenn der Autor einen anderen Stil zum Beispiel `headings` eingestellt hat. Das bedeutet, dass zu einem `headings` Stil auch ein passender `plain` Stil vorhanden sein sollte.

Das Paket scrpage2 definiert dazu einen `headings`-Stil namens `scrheadings` und einen korrespondieren `plain`-Seitenstil, der einfach `scrplain` heißt. Letzteren direkt zu aktivieren ist eigentlich nicht notwendig, da dies automatisch durch `scrheadings` geschieht. Eine Ausnahme gibt es nur, wenn `scrheadings` nicht genutzt wird, man aber eigenen Stile mit `scrplain` kombinieren will. Dazu muss vor dem eigenen Stil `scrplain` mindestens einmal, beispielsweise mit `\pagestyle{`*scrplain*`}\pagestyle{`*eigenerStil*`}`, aktiviert werden.

4.1 Grundlegende Funktionen

4.1.1 Vordefinierte Seitenstile

```
scrheadings
scrplain
```

Das Paket scrpage2 liefert einen eigenen Seitenstil namens `scrheadings`. Dieser Seitenstil kann mittels `\pagestyle{scrheadings}` aktiviert werden. Wird dieser Seitenstil benutzt, dann ist gleichzeitig ein dazu passender `plain`-Stil verfügbar. Passend bedeutet, dass auch der `plain`-Stil auf in Abschnitt 4.1.3 vorgestellten Befehle, die beispielsweise die Kopfbreite ändern, reagiert.

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

Diese Seitenstile sind so definiert, dass sowohl im Kopf als auch im Fuß drei Felder vorhanden sind, deren Inhalt modifiziert werden kann. Die Befehle zur Modifikation sind Abbildung 4.1 verdeutlicht. Die in der Mitte dargestellten Befehle modifizieren sowohl die Felder der linken als auch der rechten Seite.

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

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

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

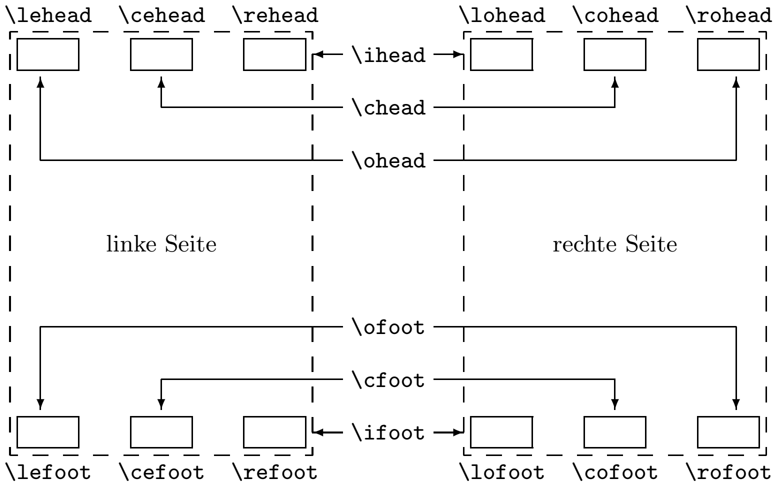


Abbildung 4.1: Befehle zur Manipulation der Seitenstile `scrheadings` und `scrplain` in ihrer Zuordnung zu den manipulierten Seitenelementen

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

Das `\cfoot{}` ist nur dann notwendig, wenn eine möglicherweise im Fuß vorhandene Seitenzahl entfernt werden muss.

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

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

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

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

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

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

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

Wird nun der Stil `plain` genutzt, beispielsweise weil `\chapter` darauf umschaltet, nachdem es eine neue Seite begonnen hat, dann steht die Seitenzahl zentriert im Seitenfuß.

```
\clearscrheadings
\clearscrpain
\clearscrheadfoot
```

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

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

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

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

```
\ihead[]{}
\chead[]{}
```

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

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

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

`\leftmark`
`\rightmark`

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

`\headmark`

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

`\pagemark`

Dieser Befehl ermöglicht den Zugriff auf die Seitenzahl. Im Abschnitt 4.1.3 *scrpage*
wird der Befehl `\pnumfont` zur Formatierung der Seitenzahl vorgestellt, den `\pagemark` automatisch berücksichtigt.

`useheadings`

Das Paket scrpage2 ist in erster Linie dafür gedacht, dass die bereitgestellten Stile benutzt werden bzw. eigene Stile definiert werden. Jedoch kann es notwendig sein, auch auf einen von der Dokumentenklasse zur Verfügung gestellten Stil zurückzuschalten. Naheliegend wäre es, mit `\pagestyle{headings}` dies vorzunehmen, hat aber den Nachteil, dass die nachfolgend besprochenen Befehle `\automark` und `\manualmark` nicht wie erwartet funktionieren. Aus diesem Grund sollte auf die originalen Stile mit `\pagestyle{useheadings}` umgeschalten werden.

Wichtig ist in diesem Zusammenhang, dass scrpage2 diesen Stil automatisch beim Laden des Pakets aktiviert, wenn eine Dokumentenklasse verwendet wird, die eine Gliederungsebene *chapter* besitzt.

4.1.2 Manuelle und automatische Kolumnentitel

Gewöhnlich gibt es zu einem `headings`-Stil eine *my*-Variante. Ist ein solcher Stil aktiv, dann werden die Kolumnentitel nicht mehr automatisch aktualisiert. Bei scrpage2 wird ein anderer Weg beschritten. Ob die Kolumnentitel lebend sind oder nicht, bestimmen die Befehle `\automark` und `\manualmark`.

`\manualmark`

Wie der Name bereits verdeutlicht, schaltet `\manualmark` die Aktualisierung der Kolumnentitel aus. Es bleibt somit dem Nutzer überlassen, für eine Aktualisierung bzw. für den Inhalt der Kolumnentitel zu sorgen. Dazu stehen die Befehle `\markboth` und `\markright` bereit.

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

Das Makro `\automark` hingegen aktiviert die automatische Aktualisierung. Für die beiden Parameter sind die Bezeichnungen der Gliederungsebenen einzusetzen, deren Titel an entsprechender Stelle erscheinen soll. Gültige Werte für die Parameter sind: `chapter`, `section`, `subsection`, `subsubsection`, `paragraph` und `subparagraph`. Das optionale Argument *rechte Seite* ist verständlicherweise nur für zweiseitigen Satz gedacht. Im einseitigen Satz sollten Sie normalerweise darauf verzichten. Mit Hilfe der Option `autooneside` können Sie auch einstellen, dass das optionale Argument im einseitigen Satz automatisch ignoriert wird (siehe Abschnitt 4.1.4).

Beispiel: Angenommen es wird mit einer *book*-Klasse gearbeitet, dessen höchste Gliederungsebene *chapter* ist, dann stellt nach einem vorhergehenden `\manualmark` der Befehl

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

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

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

Wie sinnvoll letzteres ist, muss jeder für sich entscheiden.

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

4.1.3 Formatierung der Kopf- und Fußzeilen

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

```
\headfont
\pnumfont
```

Die Schriftformatierung übernimmt der Befehl `\headfont` für den Seitenkopf und -fuß und `\pnumfont` für die Seitenzahl. *scrpage*

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

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

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

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

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

`paper` – die Breite des Papiers

`page` – die Breite der Seite

`text` – die Breite des Textkörpers

`textwithmarginpar` – die Breite des Textkörpers inklusive dem Seitenrand `head` – die aktuelle Breite des Seitenkopfes

`foot` – die aktuelle Breite des Seitenfußes

Der Unterschied zwischen `paper` und `page` besteht darin, dass `page` die Breite des Papiers abzüglich der Bindekorrektur ist, falls das typearea-Paket verwendet wird (siehe Kapitel 2). Ohne Verwendung von typearea sind `paper` und `page` identisch.

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

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

und sieht dann auf einer rechten Seite folgendermaßen aus.

KOMA-Script 3

Dieser Blindtext wird gerade von 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt, der sich vom Sehnerv in den hinteren Teil. . .

Netzhaut (*Retina*)

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

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

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

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

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

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

```
\setheadwidth{textwithmarginpar}
```

und sieht dann auf einer rechten Seite folgendermaßen aus.

KOMA-Script 3

Dieser Blindtext wird gerade von 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt, der sich vom Sehnerv in den hinteren Teil...

Netzhaut (*Retina*)

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

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

Entsprechend den Größenparametern für die Kopf- und Fußzeile gibt es auch Befehle, die die Dimensionen der Linien im Kopf und Fuß modifizieren können.

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

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

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

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

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

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

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

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

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

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

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

KOMA-Script 3

Dieser Blindtext wird gerade von 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt, der sich vom Sehnerv in den hinteren Teil...

Netzhaut (*Retina*)

Die automatische Anpassung an die Kopf- und Fußbreiten illustriert folgendes Beispiel:

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

Dieser Blindtext wird gerade von 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt,

Netzhaut (*Retina*)

KOMA-Script 3

Nun mag nicht jedem die Ausrichtung der Linie über der Fußzeile gefallen, sondern es wird erwartet, dass sie linksbündig zum Text ist. Diese Einstellung kann nur global in Form einer Paketoption erfolgen und wird im folgenden Abschnitt 4.1.4 mit anderen Optionen beschrieben.

4.1.4 Optionen beim Laden des Paketes

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

Diese Optionen, die beim Laden des Paketes angegeben werden können, bestimmen ob der Seitenkopf bzw. der Seitenfuß für die Satzspiegel zum Textkörper gezählt werden. Die durch die Verwendung der Parameter notwendigen Einstellungen werden vom Paket typearea (siehe 2.4) vorgenommen, wenn dieses Paket nach scrpage2 geladen wird. Wichtig ist hier, dass bei Verwendung einer KOMA-Script-Klasse diese Optionen bei der Dokumentenklasse und nicht bei scrpage2 angegeben werden müssen, um eine Wirkung zu erzielen.

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

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

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

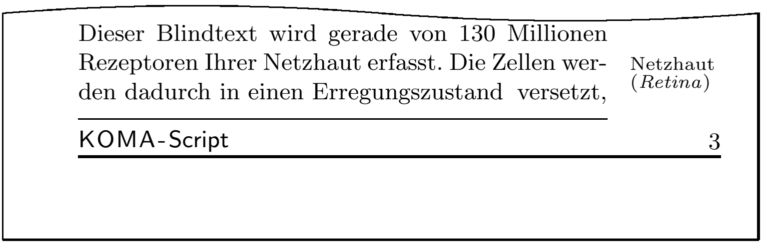
Dieser Automatismus unterliegt den gleichen Bedingungen, wie bei den Optionen im vorhergehenden Abschnitt. Das bedeutet, dass das Paket typearea nach scrpage2 geladen werden muss, beziehungsweise, dass bei Verwendung einer KOMA-Script-Klasse, die Optionen `headinclude` und `footinclude` explizit bei `\documentclass` gesetzt werden müssen, um Kopf- bzw. Fußzeile in den Texblock zu übernehmen.

`ilines`
`clines`
`olines`

Bei der Festlegung der Linienlängen kann es vorkommen, dass die Linie zwar die gewünschte Länge aber nicht die erwünschte Ausrichtung hat, da sie im Kopf- bzw. Fußbereich zentriert wird. Mit den hier vorgestellten Paketoptionen kann global für alle mit scrpage2 definierten Seitenstile diese Vorgabe modifiziert werden. Dabei setzt `ilines` die Ausrichtung so, dass die Linien an den inneren Rand vorschoben werden. Die Option `clines` verhält sich wie die Standardeinstellung und `olines` richtet am äußeren Rand aus.

Beispiel: Hier gilt es das Beispiel zu `\setfootsepline` auf Seite 100 mit dem folgenden zu vergleichen, um die Wirkung der Option `ilines` zu sehen.

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



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

`automark`
`manualmark`

Diese Optionen setzen gleich zu Beginn des Dokuments die Einstellung, ob eine automatische Aktualisierung der Kolumnentitel erfolgt. Die Option `automark` schaltet die automatische Aktualisierung ein, `manualmark` deaktiviert sie.

`autooneside`

Diese Option sorgt dafür, dass das optionale Argument von `\automark` im einseitigen Satz automatisch ignoriert wird. Siehe hierzu auch die Erläuterung zum Befehl `\automark` in Abschnitt 4.1.2.

`komastyle`
`standardstyle`

Diese Optionen bestimmen, wie der vordefinierte Seitenstil `scrheadings` gestaltet ist. Bei `komastyle` wird eine Definition vorgenommen, wie sie den KOMA-Script-Klassen entspricht. Bei den KOMA-Script-Klassen ist dies Voreinstellung und kann somit auch für andere Klassen gesetzt werde.

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

`markuppercase`
`markusedcase`

Für die Funktionalität von `\automark` modifiziert scrpage2 interne Befehle, die die Gliederungsbefehle benutzen, um die lebenden Kolumnentitel zu setzen. Da einige Klassen, im Gegensatz zu den KOMA-Script-Klassen, die Kolumnentitel in Großbuchstaben schreiben, muss scrpage2 wissen, wie die genutzte Dokumentenklasse die lebenden Kolumnentitel darstellt.

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

`nouppercase`

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

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

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

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

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

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

4.2 Seitenstile selbst gestalten

4.2.1 Die Anwenderschnittstelle

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

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

scrpage Die einzelnen Felder haben folgende Bedeutung:

Name – die Bezeichnung des Seitenstils, um ihn mit `\pagestyle{`*Name*`}` zu aktivieren

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

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

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

KM – Inhalt des Feldes im Kopf zentriert

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

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

FM – Inhalt des Feldes im Fuß zentriert

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

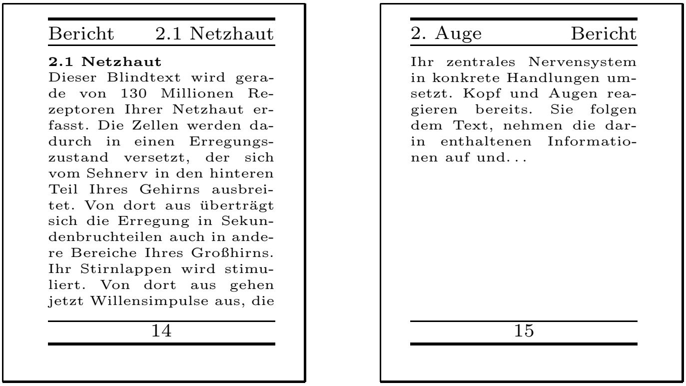
Der Befehl `\deftripstyle` stellt sicherlich die einfachste Möglichkeit dar, Seitenstile zu definieren. Leider sind damit auch Einschränkungen verbunden, da in einem Seitenbereich mit einem durch `\deftripstyle` deklarierten Seitenstil keine Änderung der Kopf- ind Fußlinien erfolgen kann.

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

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

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

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



4.2.2 Die Expertenschnittstelle

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

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

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

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

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

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

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

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

Name – die Bezeichnung des Seitenstils für `\pagestyle{`*Name*`}`

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

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

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

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

OLD – obere Liniendicke

GS – Definition für die *gerade* Seite

US – Definition für die *ungerade* Seite

ES – Definition für *einseitiges* Layout

ULL – untere Linienlänge (Kopf = Trennlinie, Fuß = außen)

ULD – untere Liniendicke

Werden die optionalen Linienargumente nicht gesetzt, dann bleibt das Verhalten weiterhin durch die im Abschnitt 4.1.3 vorgestellten Linienbefehle konfigurierbar. In der alten Version scrpage sind die Linienargumente obligatorisch. *! scrpage*

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

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

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

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

Dies und die Verwendung der Expertenschnittstelle in Zusammenhang mit anderen Befehlen von scrpage2 nun als abschließendes Beispiel.

Beispiel: Das Paket wird geladen:

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

Zwei Seitenstile werden definiert:

```
\defpagestyle{ohneLinien}{%
  {Beispiel\hfill\headmark}
  {\headmark\hfill Ohne Linien}
  {\rlap{Beispiel}\hfill\headmark\hfill%
   \llap{Ohne Linien}}
}{%
  {\pagemark\hfill}
  {\hfill\pagemark}
  {\hfill\pagemark\hfill}
}

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

\begin{document}
\pagestyle{scrheadings}\chead[\headmark]{}
\chapter{Das Bein}
\section{Der Fuß}
Der Fuß ist fest am unteren Ende vom Bein befestigt.
Man könnte natürlich auch von angewachsen sprechen.
```

Hier wird erst auf den Stil `scrheadings` geschaltet. Diese erste Seite ist eine Kapitelanfangsseite, der Befehl `\chapter` schaltet auf den `plain`-Stil. Da auch bei Kapitelanfangsseiten im Kopf die Kapitelbeschriftung erscheinen soll (eigentlich macht man sowas nicht), wird das optionale Argument von `\chead` genutzt.

1. Das Bein

1. Das Bein

1.1 Der Fuß

Der Fuß ist fest am unteren Ende vom Bein...

```
\manualmark
\markboth{2.1 Netzhaut}{2 Auge}
Dieser Blindtext wird gerade von 130 Millionen
Rezeptoren ...
```

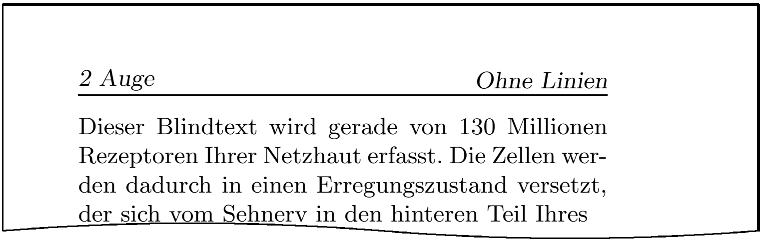
Auf der folgenden geraden Seite werden die lebenden Kolumnentitel deaktiviert und müssen somit mittels `\markboth` aktualisiert werden. Da beim Laden des Paketes die Option `headsepline` genutzt und auf dieser Seite der Stil `scrheadings` nun wie gewünscht aktiv ist, erscheint hier jetzt eine Linie zwischen Kopf und Text.

2.1 Netzhaut

Dieser Blindtext wird gerade von 130 Millionen Rezeptoren Ihrer Netzhaut erfasst. Die Zellen werden dadurch in einen Erregungszustand versetzt, der sich vom Sehnerv in den hinteren Teil Ihres

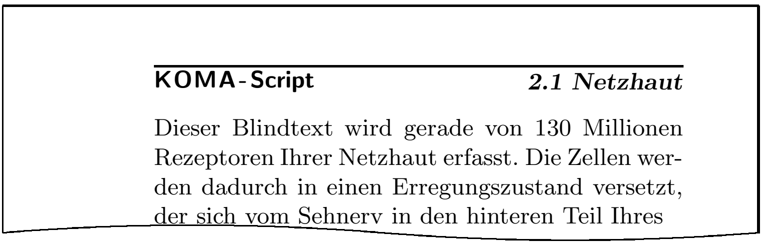
```
\pagestyle{ohneLinien}
Dieser Blindtext wird gerade von 130 Millionen
Rezeptoren ...
```

Auf der folgenden ungeraden Seite wird der Stil `ohneLinien` aktiviert. Bei diesem wurde, wie oben zu sehen, auf Angaben zu Liniendicken und -längen verzichtet. Es erscheint trotzdem die Kopf-Trennlinie, da die Einstellungen durch `\setheadtopline` und verwandte Befehle einschließlich den Paketoptionen auch Gültigkeit haben, wenn bei der Stildefinition mit `\defpagestyle` keine Linien-Argumente angegeben wurden.




```
\pagestyle{mitLinien}
\renewcommand{\headfont}{\itshape\bfseries}
Dieser Blindtext wird gerade von 130 Millionen
Rezeptoren ...
\end{document}
```

Nach dem Wechsel auf die nächsten Seite wird der Stil `mitLinien` aktiviert. Da die Kopf-Trennlinie mit einer Länge von 0 pt definiert wurde, erscheint diese Linie nun nicht.



4.2.3 Seitenstile verwalten

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

4.3 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- Markus Kohm <Markus.Kohm@gmx.de>
- **Jens-Uwe Morawski**

5 Wochentag und Uhrzeit mit scrdate und scrtime

Zu KOMA-Script gehören auch zwei Pakete, um den Umgang mit Datum und Zeit über die beiden Standardbefehle `\today` und `\date` hinaus zu erweitern. Ebenso wie die anderen Pakete aus KOMA-Script, können diese Pakete auch mit den Standardklassen verwendet werden.

5.1 Der aktuelle Wochentag mit dem scrdate Paket

`\todaysname`

Bekanntlich erhält man mit `\today` das aktuelle Datum in der landestypischen Schreibweise. scrdate bietet mit `\todaysname` eine Anweisung, um den aktuellen Wochentag zu erhalten.

Beispiel: Sie wollen in Ihrem Dokument ausgeben, an was für einem Tag es mit LaTeX in eine dvi-Datei übersetzt wurde. Sie schreiben dazu:

```
Dieses Dokument wurde an einem {\todaysname} übersetzt.
```

und erhalten beispielsweise:

> Dieses Dokument wurde an einem Samstag übersetzt.

`\nameday{`*Name*`}`

So wie mit `\date` die Ausgabe von `\today` direkt geändert werden kann, setzt `\dayname` die Ausgabe von `\todaysname` auf den Wert *Name*.

Beispiel: Sie setzen mit `\date` das aktuelle Datum auf einen festen Wert. Für die Ausgabe des zugehören Wochentags interessiert es nur, dass dieser Tag ein Werktag war. Daher schreiben sie:

```
\nameday{Werktag}
```

und erhalten so mit dem Satz auf dem vorherigen Beispiel:

> Dieses Dokument wurde an einem Werktag übersetzt.

Das scrdate Paket beherrscht derzeit die Sprachen Englisch (english and USenglish), Deutsch (german und ngerman), Französisch (french), Italienisch (italian) und Spanisch (spanish), kann aber auch für andere Sprachen konfiguriert werden. Näheres dazu entnehme man `scrdate.dtx`.

Bei der aktuellen Version ist es egal, ob scrdate vor oder nach german, babel oder ähnlichen Paketen geladen wird, in jedem Falle wird die korrekte Sprache gewählt.

Etwas genauer ausgedrückt: Solange die Spachauswahl in einer zu babel bzw. german kompatiblen Form erfolgt und die Sprache scrdate bekannt ist, wird die Sprache korrekt gewählt. Ist dies nicht der Fall, werden (US)englische Ausdrücke verwendet. Wie man scrdate neue Sprachen beibringen kann, ist `scrdate.dtx` zu entnehmen.

5.2 Die aktuelle Zeit mit dem scrtime Paket

`\thistime[`*Trennung*`]`
`\thistime*[`*Trennung*`]`

`\thistime` liefert die aktuelle Zeit. Als Trennbuchstabe zwischen den Werten Stunden, Minuten und Sekunden wird das optionale Argument *Trennung* verwendet. Die Voreinstellung ist hierbei das Zeichen „`:`“.

`\thistime*` funktioniert fast genau wie `\thistime`. Der einzige Unterschied besteht darin, daß im Gegensatz zu `\thistime` bei `\thistime*` die Minutenangaben bei Werten kleiner 10 nicht durch eine vorangestellte Null auf zwei Stellen erweitert wird.

Beispiel: Die Zeile

```
Ihr Zug geht um \thistime\ Uhr.
```

liefert als Ergebnisse beispielsweise eine Zeile wie:

> Ihr Zug geht um 16:49 Uhr.

oder:

> Ihr Zug geht um 23:09 Uhr.

Demgegenüber liefert die Zeile:

```
Beim nächsten Ton ist es \thistime*[\ Uhr,\ ] Minuten und
42 Sekunden.
```

als mögliches Ergebniss etwas wie:

> Beim nächsten Ton ist es 16 Uhr, 49 Minuten und 42 Sekunden.

oder:

> Beim nächsten Ton ist es 23 Uhr, 9 Minuten und 42 Sekunden.

`\settime{`*Wert*`}`

`\settime` setzt die Ausgabe von `\thistime` und `\thistime*` auf einen festen *Wert*.[1] Anschließend wird das optionale Argument von `\thistime` bzw. `\thistime*` ignoriert, da ja die komplette Zeichenkette, die `\thistime` bzw. `\thistime*` nun liefert, hiermit explizit festgelegt wurde.

`12h`
`24h`

Mit den Optionen `12h` und `24h` kann ausgewählt werden, ob die Zeit bei `\thistime` und `\thistime*` im 12-Stunden- oder 24-Stunden-Format ausgegeben werden soll. Voreingestellt ist `24h`.[2] Die Option verliert bei einem Aufruf von `\settime` ebenfalls ihre Gültigkeit.

5.3 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- Frank Neukam
- Axel Sommerfeldt
- Markus Kohm <Markus.Kohm@gmx.de>

Die Fußnoten stammen von Axel Sommerfeldt.

[1] Allerdings darf man nicht erwarten, daß nun die Zeit stillsteht!
[2] Leider beherrscht das scrtime Paket noch nicht die Sternzeit nach StarTrek, ein echter Mangel!

6 Briefe schreiben mit scrlettr

6.1 Überblick

Die Dokumentenklasse scrlettr ist eine erweiterte und an europäische Verhältnisse angepasste Version der originalen LaTeX-Briefklasse letter. Ursprünglich wurde sie von Axel Kielhorn entwickelt, erfuhr aber durch Markus Kohm einige Veränderungen.

Hervorzuhebende Eigenschaften von scrlettr gegenüber letter sind die Anpassung an das A4 Papierformat, die erweiterte Sprachunterstützung und ein umfangreicherer Befehlssatz, mit dem auch komplexere Wünsche umsetzbar sind.

Ein Beispiel

Bevor alle Befehle der Klasse scrlettr vorgestellt werden, soll mit Hilfe eines Minimalbeispiels ein erster Überblick über Aufbau und Funktion eines Briefes gegeben werden.

Beispiel: Ein mit nur den nötigsten Befehlen erstellter scrlettr-Brief sieht beispielsweise so aus.

```
\documentclass[10pt]{scrlettr}
\usepackage{ngerman}
\name{\KOMAScript{}-Gruppe}
\address{Klassengasse 1\\12345 \LaTeX{}hausen}
\signature{Euer \KOMAScript{}-Team}
\begin{document}
  \begin{letter}{Die \KOMAScript{}-Nutzer\\
                 Irgendwo\\weltweit}

    \opening{Liebe \KOMAScript{}-Nutzer,}
    das \KOMAScript{}-Team möchte Euch mit ein paar
    Informationen ...

    \closing{Viel Spaß}
  \end{letter}
\end{document}
```

Wie zu sehen ist, werden Informationen, die unabhängig vom einzelnen Brief sind, getrennt definiert, wie beispielsweise der Absender mittels `\name`. Die einzelnen briefspezifische Daten werden innerhalb der `letter`-Umgebung angegeben. Natürlich können durch mehrmaliges Nutzen der `letter`-Umgebung mehrere Briefe in einem Dokument erzeugt werden.

Hierbei ist jedoch zu beachten, dass TEX Zähler grundsätzlich *global* verwaltet. Es ist also notwendig, alle Zähler vor einer neuen `letter`-Umgebung zurückzusetzen. Einzige Ausnahme ist der Seitenzähler. Dieser wird bei jedem Aufruf von `\begin{letter}` wieder auf 1 zurückgesetzt.

6.2 Briefübergreifende Befehle

`\name{`*Absendername*`}`
`\address{`*Adresse des Absenders*`}`
`\signature{`*Unterschrift*`}`

Der Befehl `\name` nimmt den Namen des Absenders auf und gibt diesen im voreingestellten Seitenstil für die erste und die folgenden Seiten aus. Weiterhin wird dieser Text als Unterschrift gesetzt, wenn `\signature` nicht angegeben wurde, da dieser Befehl optional ist, und somit nicht immer angegeben werden muss. Die Absenderadresse wird mit dem `\address`-Befehl gesetzt.

`\backaddress{`*Absender*`}`
`\specialmail{`*Versandart*`}`
`\addrfieldon`
`\addrfieldoff`

Der Befehl `\backaddress` erzeugt über dem Adressfeld des Empfängers einen einzeiligen Eintrag, der auch in Briefumschlägen mit Sichtfenster zu sehen ist. Es bietet sich somit als Angabe des Absenders an. Versandhinweise, wie beispielsweise *Einschreiben* können mit `\specialmail` gesetzt werden.

Mit dem Befehl `\addrfieldoff` werden weder das Adressfeld noch das `locfield` gesetzt. Alle Angaben über Empfänger- und Rückadresse, die Versandart und die Ergänzungen aus `\location` werden ignoriert und *nicht* gesetzt. Da `\addrfieldon` und `\addrfieldoff` als Befehle implementiert sind, ist es möglich, sie für verschiedene Briefe eines Dokuments je nach Bedarf anzuwenden. Voreingestellt ist `\addrfieldon`.

`\location{`*zusätzliche Adresstext*`}`
`\place{`*Ort*`}`
`\date{`*Datum*`}`

Der Befehl `\location` schreibt sein Argument in ein Textfeld rechts neben dem Adressfeld. Es kann beispielsweise Daten einer Abteilung oder die Zweigstelle eine Firma aufnehmen. Mit dem Befehl `\place` wird der Ort des Absenders eingestellt. Der Befehl `\date` ist nur wichtig, wenn der Brief länger in Quellform gespeichert werden soll und man nicht möchte, dass die Datumsinformation des Originalbriefes verloren geht. In normalen Anwendungsfällen wird das Datum aus dem Systemdatum beim LaTeX-Lauf ermittelt.

Beispiel: Sie möchten einen monatlichen Rundbrief an die Mitglieder eines Vereins versenden. Hierbei spielt das genaue Datum keine große Rolle. Ein Vermerk in der Form „Vereinssitz im März 2001“ erreichen Sie durch folgende Definitionen.

```
\place{Vereinssitz}
\date{im März 2001}
```

```
wlocfield
slocfield
```

Die Breite des Textfeldes mit dem `\location`-Eintrag ist standardmäßig die Hälfte des freien Raums neben dem Adressfeld. Dies entspricht der Einstellung der Option `slocfield`. Mit der Klassenoptionen `wlocfield` stehen dem Textfeld zwei Drittel der freien Textbreite neben dem Adressfeld zur Verfügung.

6.3 Briefspezifische Befehle

```
letter
\title{Überschrift}
\subject{Kurzinhalt oder Betreff}
\subjecton
\subjectoff
\opening{Anrede}
```

Das zentrale Element ist die `letter`-Umgebung. Ihr obligatorisches Argument nimmt die Anschrift des Empfängers des Briefes auf. Notwendige Zeilenumbrüche müssen selbständig eingefügt werden. Dabei muss beachtet werden, dass das Textfeld nur eine bestimmte Breite zulässt, und dann die Zeile automatische umbrochen wird.

Beispiel: Sie möchten einen Brief an die Deutsche Anwendervereinigung TEX e. V. schreiben. Dieser Brief müßte durch folgende Zeilen eingeleitet werden:

```
\begin{letter}{DANTE, Deutschsprachige
               Anwendervereingung TeX e.\,V.\\
               Postfach 101840\\
               69008 Heidelberg}
```

Gewöhnlich möchte man dem Empfänger eines Briefes schnell das Anliegen des Briefes mitteilen. Dazu gibt es zwei Möglichkeiten. Mit dem `\title` kann einem Brief eine Überschrift vorangestellt werden. Dies erzeugt eine zentrierte in `\LARGE` gesetzte Überschrift. Für gewöhnlich soll der Hinweis auf den Inhalt des Briefes nicht so großspurig ausfallen. Vielmehr wird eine schlichte Betreffzeile mit einer kurzen Zusammenfassung des Schreibens erwartet. Dies geschieht mit dem Makro `\subject`.

Beispiel: Anlässlich eines Firmenjubiläums möchten Sie ein Preisauschreiben für alle Kunden ausrichten. Damit Ihre Kunden sofort sehen, dass es sich bei Ihrem Brief nicht um eine Rechnung handelt, möchten Sie deutlich auf den Inhalt Ihres Briefes aufmerksam machen.

```
\title{Preisausschreiben}
```

Eine schlichte Betreffzeile erzeugen Sie dagegen mit folgender Zeile.

```
\subject{Lagerverkauf}
```

Darüber hinaus kann mit dem Befehl `\subjecton` vor der Betreffzeile noch den Eintrag *Betr.:* gesetzt werden. Voreingestellt ist `\subjectoff`, so dass nur die bloße Betreffzeile gesetzt wird.

Der Brief beginnt mit dem `\opening`-Befehl. Erst nach diesem Befehl werden die Angaben im Referenzfeld, Betreff und Empfänger gesetzt. Diesem Befehl wird die Anrede des Briefpartners übergeben.

`\closing{`*Grußformel*`}`
`\ps{`*Postskriptum*`}`
`\cc{`*Verteiler*`}`
`\ccnameseparator`
`\ccname`
`\encl{`*Anlagen*`}`
`\enclnameseparator`
`\enclname`

Nachdem der Brief geschrieben wurde, wird mit `\closing` das Briefende eingeleitet. Diesem Befehl wird die Grußformel (zum Beispiel „Mit freundlichen Grüßen“) übergeben.

Manchmal ist es auch erwünscht, nach dem eigentlichen Brief noch Informationen unterzubringen, die nicht dem engsten Briefanliegen entsprechen aber dennoch von Interesse für den Empfänger sein können. Diese Informationen werden dem Befehl `\ps` übergeben.

Beispiel: Sie möchten Ihre Kunden darauf hinweisen, dass Ihr Geschäft im August Betriebsurlaub macht und deshalb geschlossen bleibt. Da diese Information nicht in unmittelbaren Zusammenhang mit einer Rechnung steht, möchten Sie dies als Postscript schreiben.

```
\closing{Mit freundlichen Grüßen,}
\ps{Bitte beachten Sie, dass wir vom 01.08. bis 31.08.
Betriebsurlaub machen. Unser Geschäft bleibt in dieser
Zeit geschlossen. Wir sind ab dem 01.09. in alter
Frische wieder für Sie da.}
```

Häufig liegen geschäftlichen Briefen Anlagen bei, beispielsweise Vertragsentwürfe, Rechnungskopien oder ähnliches. Darauf kann der Adressat mit dem Befehl `\encl` aufmerksam gemacht werden. Gehen Kopien eines Briefes an die Empfänger eines Verteilers, so kann auch dies im Abschluss des Briefes bekannt gegeben werden. Dazu dient der Befehl `\cc`.

Beispiel: Die Währungsumstellung auf den Euro stellt Ihr kleines Unternehmen vor große Schwierigkeiten. Um Zahlungsausfälle zu vermeiden, ziehen Sie alle Verbindlichkeiten für den Monat Dezember vor und möchten diese bereits im November auszahlen. Außerdem sollen alle Fälligkeiten im neuen Jahr erst ab der dritten Januarwoche beglichen werden. Diesen Beschluss der Geschäftsleitung möchten Sie als Kopie an die Buchhaltung und den Betriebsrat schicken.

```
\encl{Beschluss über die Zahlungsmodalitäten beim
Übergang auf den Euro}
\cc{Buchhaltung\\Betriebsrat}
```

Der Aufruf `\cc` setzt vor dem übergebenen Argument noch den `\ccname` „Kopie an“ und den Trenner `\ccnameseparator` „: “ (Doppelpunkt gefolgt von einem Leerzeichen). Ähnlich funktioniert der Befehl `\encl`. Der `\enclname` lautet „Anlagen“ – der Trenner `\enclnameseparator` ist genauso definiert, wie der für Verteiler.

Beispiel: Da Sie im obigen Beispiel nur eine Anlage verschicken, ist es besser, den Singular zu verwenden. Sie möchten, dass der Verteiler auch als solcher benannt wird und außerdem halten Sie den Doppelpunkt für überflüssig. Alle Ihre Wünsche können erfüllt werden:

```
\renewcommand*{\ccnameseparator}{\ }
\renewcommand*{\enclnameseparator}{\ccnameseparator}
\renewcommand*{\ccname}{Verteiler}
\renewcommand*{\enclname}{Anlage}
```

6.3.1 Das Referenzfeld

`\yourref{`*Ihr Zeichen*`}`
`\yourmail{`*Ihr Schreiben vom*`}`
`\myref{`*Unser Zeichen*`}`
`\customer{`*Kundennummer*`}`
`\invoice{`*Rechnungsnummer*`}`

In Geschäftsbriefen werden häufig Informationen wie Aktenzeichen, Rechnungs- oder Kundennummer oder ein Hinweis auf das Schreiben, das beantwortet wird, benötigt. Um diese Anforderungen realisieren zu können, sind in der scrletter-Klasse einige Makros implementiert.

Beispiel: Sie möchten einen Brief Ihres Geschäftspartners Maier beantworten, den dieser am 14. August 2000 geschrieben hat. Ihr Zeichen ist *xyz* Herr Maier hat die Kundennummer *maier007* und beschwerte sich in seinem Brief, dass in der Rechnung mit der Nummer *197200/01* kein Mehrwertsteueranteil angegeben war. Das Zeichen von Herrn Maier lautet *maier*. Sie schreiben also folgenden Brief:

```
\documentclass[10pt,a4paper]{scrlettr}
\usepackage{ngerman}
\name{Firma xyz}
\address{Industriegasse 12\\23987 Stahlhausen}
\signature{Herr Schmidt\\ Reklamationen}
  \begin{document}
    \begin{letter}{Herr Maier\\Wiesenweg 37\\ Blumental}
    \yourref{maier}
    \yourmail{14.08.2000}
    \myref{xyz}
    \customer{maier007}
    \invoice{197200/01}
    \opening{Sehr geehrter Herr Maier,}
    vielen Dank für Ihr Schreiben vom 14. August.
    Wir bedauern unseren Fehler und senden Ihnen
    anbei eine korrigierte Rechnung.
    \closing{Mit freundlichen Grüßen}
    \end{letter}
  \end{document}
```

Sie sehen, dass Ihre Angaben im Referenzfeld zwischen der Empfängeradresse und dem eigentlichen Brieftext gesetzt werden.

```
\refitemi{Eigenes Feld 1}
\refitemii{Eigenes Feld 2}
\refitemiii{Eigenes Feld 3}
\refitemnamei{Bezeichnung des eigenen Feldes 1}
\refitemnameii{Bezeichnung des eigenen Feldes 2}
\refitemnameiii{Bezeichnung des eigenen Feldes 3}
```

Neben den bereits vordefinierten Makros stehen noch bis zu drei frei definierbare Makros zur Verfügung, um das Referenzfeld verschiedenen Anforderungen gemäß anpassen zu können.

Beispiel: Angenommen, Sie stehen mit Herrn Maier in Verhandlungen über die Abnahme von 10.000 Stück Ihres schlimmsten Ladenhüters. Dann benötigen Sie natürlich keine Rechnungsnummer. Statt dessen möchten Sie im Referenzfeld ein Aktenzeichen vermerken. Dazu definieren zu mit `\refitemnamei` den ersten frei wählbaren Referenzeintrag als *Aktenzeichen*. Diesem Eintrag können Sie dann das Aktenzeichen *123/01mai* zuweisen.

```
\refitemnamei{Aktenzeichen}
\refitemi{123/01mai}
```

6.4 Seitenstile

```
\firsthead{Kopfdefinition}
\firstfoot{Fußdefinition}
\nexthead{Kopfdefinition}
\nextfoot{Fußdefinition}
```

Die scrlettr-Klasse ermöglicht es, den Seitenstil eines Dokuments an die eigenen Bedürfnisse anzupassen. Dazu kann getrennt für die erste und alle folgenden Seiten der Fuß und der Kopf frei definiert werden. Diese Definition muss *vor* dem Aufruf von `\pagestyle{...}` erfolgen.

Beispiel: Sie möchten in der Fußzeile der ersten Seite eines Briefes die Bankverbindung notieren. In den Fußzeilen der folgenden Seiten möchten Sie auf Ihren verantwortungsbewussten Umgang mit der Umwelt hinweisen.

```
\firstfoot{Bankverbindung:\hfill $\bullet$\hfill Deutsche
Bank AG\hfill $\bullet$\hfill BLZ: 999\,720\,00\hfill
$\bullet$\hfill Konto: 123\,456\,890}
\nextfoot{\centerline{Dieses Schreiben wird
ausschließlich auf chlorfrei gebleichten Papier gedruckt.}}
\pagestyle{firstpage}
```

Der Aufruf des Steitenstils `\pagestyle{firstpage}` sorgt dafür, dass auf allen Seiten Kopf und Fuß wie auf der ersten Seite gestaltet werden.

Voreingestellt ist der Stil `plain`. Wird der Stil `headings` ohne eigene Definition der Kopf- und Fußzeilen verwendet, so sind die Fußzeilen grundsätzlich leer. Die Kopfzeile der ersten Seite enthält zentriert den Absendernamen eine Trennlinie und die Absenderadresse. Der Kopf der folgenden Seiten besteht aus den linksbündig gesetzten Absendernamen in der ersten Zeile. In einer zweiten Zeile steht der Empfängername, das Datum und die Seitenzahl. Wird der Stil `empty` gewählt, so bleiben Kopf- und Fußzeile auf allen Seiten leer.

Auf diese Weise können sehr individulle Briefbögen erstellt werden.

```
twoside
```

Zweiseitig gedruckte Briefe werden durch die Angabe der Option `twoside` unterstützt. Im Gegensatz zu den übrigen KOMA-Script-Klassen ändert sich hier der Satzspiegel nicht, sondern es wird lediglich sichergestellt, dass ein neuer Brief immer auf einer ungeraden *Druck*seite beginnt. Darüber hinaus wird eine Warnung ausgegeben, um darauf hinzuweisen, dass es sich nicht wirklich um ein zweiseitiges Layout handelt.

```
\foldmarkson
\foldmarksoff
```

Die Faltmarken können mit dem Befehl `\foldmarkson` eingeschaltet und mit dem Befehl `\foldmarksoff` ausgestellt werden. Dies ist für jeden Brief eines Dokuments getrennt möglich. Voreingestellt ist `\foldsmarkson`. Diese Faltmarken werden von `\opening` gesetzt. Die Schalter und evtl. Änderungen der Maße (vgl. Abschnitt 6.7) müssen also vor diesem Makro aufgerufen werden.

6.5 Unterstützung verschiedener Sprachen

6.5.1 Sprachauswahl und -umschaltung

Die scrlettr-Klasse unterstütz viele Sprachen. Dazu zählen neben Deutsch auch Österreichisch, Englisch (britisch und amerikanisch), Französich, Italienisch und Spanisch. Zwischen den Sprachen wird bei Verwendung des babel-Pakets mit dem Befehl `\selectlanguage{`*Sprachauswahl*`}` gewechselt.

```
\captionsenglish
\captionsUSenglish
\captionsamerican
\captionsbritish
\captionsUKenglish
\captionsgerman
\captionsaustrian
\captionsfrench
\captionsitalian
\captionsspanish
```

Wird die Sprache eines Briefes gewechselt, so ändern sich automatisch die Eintragungen der automatisch gesetzten „Caption"-Texte wie *Betreff*, *Seite* oder *Anlagen*. Sollte das verwendete Sprachumschaltpaket diese Texte nicht automatisch verwalten, so können die entsprechenden Befehle notfalls auch dirket verwendet werden.

```
\dateenglish
\dateUSenglish
\dateamerican
\datebritish
\dateUKenglish
\dategerman
\dateaustrian
\datefrench
\dateitalian
\datespanish
```

Je nach verwendeter Sprache werden auch die Datumsangaben in unterschiedlicher Form umgesetzt. Die genauen Angaben können der Tabelle 6.1 entnommen werden.

```
orgdate
scrdate
```

Sollen die Datumseinstellungen des babel- oder ngerman-Pakets oder eines eigenen Sprachumschaltpakets benutzt werden, so kann dies durch die Klassenoption `orgdate` erreicht werden. Voreingestellt ist die Verwendung der scrlettr-eigenen Definition (`scrdate`).

`\dateenglish`	1/12/1993
`\dateUSenglish`	12/1/1993
`\dateamerican`	12/1/1993
`\datebritish`	1/12/1993
`\dateUKenglish`	1/12/1993
`\dategerman`	1. 12. 1993
`\dateaustrian`	1. 12. 1993
`\datefrench`	1. 12. 1993
`\dateitalian`	1. 12. 1993
`\datespanish`	1. 12. 1993

Tabelle 6.1: Sprachabhängige Ausgabeformate für Datum

6.5.2 Sprachabhängige Variablen

```
\yourrefname
\yourmailname
\myrefname
\customername
\invoicename
\subjectname
\ccname
\enclname
\headtoname
\datename
\pagename
```

Die aufgeführten Befehle enthalten die jeweils sprachtypischen *Captiontexte*. Diese können für die Realisierung einer weiteren Sprache oder aber auch zur eigenen freien Gestaltung angepasst werden. Dazu benutzt man den Befehl `\renewcommand`.

Beispiel: Möchten Sie statt des Eintrags „Ihr Schreiben vom“ lieber „Ihre Nachricht vom“ im Referenzfeld stehen haben, müssen Sie den Befehl `\yourmailname` wie folgt umdefinieren.

```
\renewcommand*{\yourmailname}{Ihre Nachricht vom}
```

Auf diese Weise können Sie natürlich auch alle Variablen den Vorgaben einer anderen Sprache anpassen.

Es ist darauf zu achten, dass die Variablen erst *nach* `\begin{document}` definiert werden. Der Aufruf `\renewcommand*{...}` muss daher zwingend nach

`\begin{document}` oder mit Hilfe von `\AtBeginDocument` (siehe [Tea99b]) erfolgen.

6.6 Adressdateien

`\adrentry{`*Name*`}{`*Vorname*`}{`*Adresse*`}{`*Telefon*`}{`*F1*`}{`*F2*`}{`*Kommentar*`}{`*Kürzel*`}`

Mit der scrlettr-Klasse können auch Adressdateien ausgewertet werden. Dies ist beispielsweise für Serienbriefe sehr nützlich (siehe Abschnitt 6.6.1). Eine Adressdatei muss die Endung `.adr` haben und besteht aus einer Reihe von `\adrentry`-Einträgen. Ein solcher Eintrag besteht aus acht Elementen und kann beispielsweise wie folgt aussehen:

```
\adrentry{Maier}
         {Herbert}
         {\Wiesenweg 37\\ 09091 Blumental}
         {0\,23\,34 / 91\,12\,74}
         {Bauunternehmer}
         {}
         {kauft alles}
         {MAIER}
```

Die Elemente fünf und sechs, `F1` und `F2`, können frei bestimmt werden. Denkbar wären neben Hinweisen auf das Geschlecht oder akademische Grade auch der Geburtstag oder das Eintrittsdatum in einen Verein. Um das Überschreiben von TEX- oder LATEX-Befehlen zu vermeiden, ist es empfehlenswert, für *Kürzel* ausschließlich Großbuchstaben zu verwenden.

Beispiel: Herr Maier gehört zu Ihren engeren Geschäftspartnern. Da Sie eine rege Korrespondenz mit ihm pflegen, ist es Ihnen auf Dauer zu müßig, jedesmal alle Empfängerdaten aufs Neue einzugeben. scrlettr nimmt Ihnen diese Arbeit ab. Angenommen, Sie haben Ihre Kundenkontakte in der Datei `partner.adr` gespeichert und Sie möchten Herrn Maier einen Brief schreiben, dann sparen Sie sich viel Tipparbeit, wenn Sie folgendes eingeben:

```
\input{partner.adr}
\begin{letter}{\MAIER}
  Der Brief ...
\end{letter}
```

Achten Sie bitte darauf, dass Ihr TEX-System auch auf die `.adr`-Dateien zugreifen kann, da sonst eine Fehlermeldung von `\input` verursacht wird. Entweder Sie legen die Brief- und Adressdateien im selben Verzeichnis an, oder Sie binden ein Adressverzeichnis fest in Ihr TEX-System ein.

6.6.1 Serienbriefe mit der scrlettr-Klasse

Neben dem vereinfachten Zugriff auf Kundendaten können die `.adr`-Dateien auch für Serienbriefe genutzt werden. So ist es ohne die komplizierte Anbindung an Datenbanksysteme möglich, solche Massenpostsendungen zu erstellen.

Beispiel: Sie wollen einen Serienbrief an alle Mitglieder Ihres Anglervereins schicken, um zur nächsten Mitgliederversammlung einzuladen.

```
\documentclass{scrlettr}
\usepackage{ngerman}
  \begin{document}
  \def\adrentry#1#2#3#4#5#6#7#8{
    \begin{letter}{#2 #1\\#3}
      \opening{Liebe Vereinsmitglieder,}
      unsere nächste Mitgliederversammlung
      findet am Montag,
      dem 13.\, August 2001, statt.

      Folgende Punkte müssen besprochen werden...
      \closing{Petri Heil,}
    \end{letter}
  }
  \input{mitglieder.adr}
  \end{document}
```

Natürlich kann der Briefinhalt auch von den Adressatenmerkmalen abhängig gemacht werden. Als Bedingungsfelder können die frei bestimmbaren Elemente fünf oder sechs eines `\adrentry`-Eintrages genutzt werden.

Beispiel: Angenommen, Sie verwenden das Element fünf, um das Geschlecht eines Vereinmitgliedes zu hinterlegen (`m/w`) und das sechste Element weist auf eine Rückstand der Mitgliedsbeiträge hin. Wollen Sie nun alle säumigen Mitglieder anschreiben und persönlich anreden, so hilft Ihnen folgendes Beispiel weiter:

```
\def\adrentry#1#2#3#4#5#6#7#8{
  \ifcase #6
```

```
  % #6 > 0
  % hier werden die säumigen Mitglieder herausgefiltert
  \else
    \begin{letter}{#2 #1\\#3}
      \if #5m \opening{Lieber #2,} \fi
      \if #5w \opening{Liebe #2,} \fi

      Leider mussten wir feststellen, dass du mit der Zah-
      lung deiner Mitgliedsbeiträge im Rückstand bist.

      Wir möchten Dich bitten, den offenen Betrag von #6 DM
      auf das Vereinskonto einzuzahlen.
     \closing{Petri Heil,}
    \end{letter}
   \fi
}
```

Es ist also möglich, den Brieftext auf bestimmte Empfängermerkmale gezielt abzustimmen und so den Eindruck eines persönlichen Schreibens zu erwecken. Die Anwendungsbreite ist lediglich durch die maximale Anzahl von zwei freien `\adrentry`-Elementen begrenzt.

6.6.2 Adressverzeichnisse und Telefonlisten erstellen

`dir.tex`
`phone.tex`
`\adrchar{`*Überschrift*`}`

Letztlich ist es auch möglich, die Informationen einer `.adr`-Datei in Adressverzeichnisse oder Telefonlisten umzuwandeln. Als Beispiel sollen die Dateien `dir.tex` bzw. `phone.tex` vorgestellt werden.

Ein LaTeX-Durchlauf dieser Dateien ist interaktiv und fragt zunächst, ob man die deutsche Sprachanpassung und wenn ja auch die der neuen deutschen Rechtschreibung nutzen möchte. Als positive Antwort gelten die Eingaben `j`, `ja`, `y` oder `yes`. Danach wird man aufgefordert, den Namen der Adressdatei ohne die Dateierweiterung `.adr` einzugeben. Anschließend kann man noch einen Namen für das Adressverzeichnis bzw. die Telefonliste vergeben.

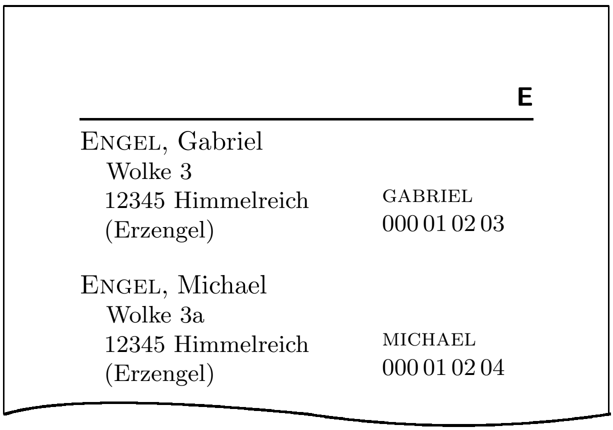
Nach dem LaTeX-Durchlauf befindet sich im aktuellen Verzeichnis eine Datei `dir.dvi` bzw. `phone.dvi`, die das Adressverzeichnis respektive die Telefonliste enthält.

Damit die Listen alphabethisch sortiert ausgegeben werden, muss bereits die Adressdatei sortiert gewesen sein. Es empfiehlt sich dabei, vor jedem neuen Anfangsbuchstaben eine Anweisung `\adrchar` mit dem diesem Buchstaben als Argument einzufügen. scrlettr selbst ignoriert diese Anweisung. Die interaktiven LaTeX-Dokumente `dir.tex` und `phone.tex` definieren die Anweisung jedoch so um, dass eine Ausgabe erfolgt.

Beispiel: Sie haben folgende, winzige Adressdatei:

```
\adrchar{E}
\adrentry{Engel}{Gabriel}
         {Wolke 3\\12345 Himmelreich}
         {000\,01\,02\,03}{}{}{Erzengel}{GABRIEL}
\adrentry{Engel}{Michael}
         {Wolke 3a\\12345 Himmelreich}
         {000\,01\,02\,04}{}{}{Erzengel}{MICHAEL}
\adrchar{K}
\adrentry{Kohm}{Markus}
         {Fichtenstra{\ss}e 63\\68535 Edingen-Neckarhausen}
         {+49~62\,03~1\,??\,??}{}{}{"Uberhaupt kein Engel}
         {KOMA}
```

Diese bearbeiten Sie nun unter Verwendung von `dir.tex`. Seite 3 des Ergebnisses sieht dann etwa so aus:



Dabei wird der Buchstabe in der Kopfzeile von `\adrchar` erzeugt. Siehe dazu die Definition in `dir.tex`. Bei Verwendung des Beispieldokument `phone.tex` wird der mit `\adrchar` gesetzte Buchstabe hingegen nicht ausgegeben.

6.7 Befehls- und Varialblenübersicht

Briefspezifische Befehle, die strukturbeschreibend sind oder eine Ausgabe erzeugen:

`\begin{letter}{`*Adressat*`}`	Markiert den Beginn eines Briefes an *Adressat* und beginnt eine neue Seite
`\end{letter}`	Markiert das Ende eines Briefes
`\opening{`*Anrede*`}`	Setzt alle Teile eines Briefes oberhalb und einschließlich der *Anrede*
`\closing{`*Grußformel*`}`	Setzt *Grußformel* und Unterschrift
`\ps{`*Postscriptum*`}`	Setzt ein *Postscriptum*
`\cc{`*Verteiler*`}`	setzt eine Verteilerliste, deren Einträge durch `\\` zu trennen sind (vgl.`\ccnameseparator` und `\ccname`)
`\encl{`*Anlagen*`}`	Setzt eine Anlagenliste, deren Einträge durch `\\` zu trennen sind (vgl. `\enclnameseparator` und `\enclname`)

Befehle der Adressdateien:
`\adrchar` und `\adrentry` siehe Abschnitt 6.6

Befehle zur Sprachumschaltung:

`\captionsenglish`	Umschaltung auf englische Caption-Texte
`\captionsUSenglish`	Umschaltung auf amerikanische Cation-Texte
`\captionsgerman`	Umschaltung auf deutsche Caption-Texte
`\captionsfrench`	Umschaltung auf französische Caption-Texte
`\captionsitalian`	Umschaltung auf italienische Caption-Texte
`\captionsaustrian`	Umschaltung auf österreichische Caption-Texte
`\captionsspanish`	Umschaltung auf spanische Caption-Texte
`\dateenglish`	Englisches Datum (vgl. Tabelle 6.1)
`\dateUSenglish`	Amerikanisches Datum (vgl. Tabelle 6.1)

`\dategerman`	Deutsches Datum (vgl. Tabelle 6.1)
`\datefrench`	Französisches Datum (vgl. Tabelle 6.1)
`\dateitalian`	Italienisches Datum (vgl. Tabelle 6.1)
`\dateaustrian`	Österreichisches Datum (vgl. Tabelle 6.1)
`\datespanish`	Spanisches Datum (vgl. Tabelle 6.1)

Sprachabhängige Variablen

Diese Varialblen dürfen an jeder Stelle nach dem `\begin{document}`-Befehl aufgerufen werden. Sie können nur mit `\renewcommand` geändert werden. Die untenstehende Aufstellung listet die voreingestellten Einträge für die Sprachen Deutsch, Englisch, Französisch, Italienisch und Spanisch auf. Die amerikanischen Caption-Texte entsprechen den englischen. Deutsche und österreichische Eintragungen sind identisch.

`\yourrefname`	Ihr Zeichen / Your ref. / Vos références / Vs./Rif. / Su ref.
`\yourmailname`	Ihr Schreiben vom / Your letter of / Votre lettre du / Vs. lettera del / Su carta de
`\myrefname`	Unser Zeichen / Our ref. / Nos références / Ns./Rif. / Nuestra ref.
`\customername`	Kundennummer / Customer no. / Numéro de client / Nr. cliente / No. de cliente
`\invoicename`	Rechnungsnummer / Invoice no. / Numéro de facture / Nr. fattura / No. de factura
`\subjectname`	Betr. / Subject / Concernant / Oggetto / Asunto
`\ccname`	Kopie an / cc / Copia á / Per conoscenza / Copias
`\enclname`	Anlagen / encl / Annexes / Allegato / Adjunto
`\headtoname`	An / To / A / A / A
`\datename`	Datum / Date / Date / Data / Fecha
`\pagename`	Seite / Page / Page / Pagina / Página

Briefspezifische Variablen und deren Befehle zur Neu- und Umdefinierung

Die in Klammern stehenden Variablen werden durch Aufruf folgender Makros geändert.

`\name`	Name des Absenders (`\fromname`)
`\branch`	Branche des Absenders (`\frombranch`)
`\signature`	Unterschrift, voreingestellt ist die Übernahme des Wertes von `\name` (`\fromsig`)
`\address`	Absenderadresse (`\fromaddress`)
`\place`	Absenderort (`\fromplace`)
`\location`	weitere Angabe zur Absenderadresse (`\fromlocation`)
`\backaddress`	Absenderadresse im Adressfeld (`\frombackaddress`)
`\telephone`	Telefonnummer des Absenders (`\telephonenum`)
`\yourref`	Referenzfeldeintrag (`\varyourref`)
`\yourmail`	Referenzfeldeintrag (`\varyourmail`
`\myref`	Referenzfeldeintrag (`\varmymail`)
`\customer`	Referenzfeldeintrag (`\varcustomer`)
`\invoice`	Referenzfeldeintrag (`\varinvoice`)
`\refitemi`	Referenzfeldeintrag, frei definierbar (`\varrefitemi`)
`\refitemii`	Referenzfeldeintrag, frei definierbar (`\varrefitemii`)
`\refitemiii`	Referenzfeldeintrag, frei definierbar (`\varrefitemiii`)
`\refitemnamei`	Bezeichnung eines frei definierbaren Referenzfeldeintrags (`\varrefitemnamei`)
`\refitemnameii`	Bezeichnung eines frei definierbaren Referenzfeldeintrags (`\varrefitemnameii`)
`\refitemnameiii`	Bezeichnung eines frei definierbaren Referenzfeldeintrags (`\varrefitemnameiii`)
`\specialmail`	Versandart (`\@specialmail`)
`\title`	Überschrift (`\@title`)
`\subject`	Betreff, sprachabhängig (`\@subject`)

`\firsthead`	Kopfzeilendefinition für erste Seite (`\@firsthead`)
`\firstfoot`	Fußzeilendefinition für erste Seite (`\@firstfoot`)
`\nexthead`	Kopfzeile der folgenden Seiten (`\@nexthead`)
`\nextfoot`	Fußzeile der folgenden Seiten (`\@nextfoot`)

Briefspezifische Längenangaben
Voreingestellte Längen sind in Klammern angegeben.

`\foldskip`	Abstand der Falzmarke vom linken Papierrand (3,5 mm)
`\foldvskipi`	Abstand zwischen der ersten Falzmarke und dem oberen Seitenrand (62 mm)
`\foldvskipii`	Abstand der zweiten Falzmarke von der ersten Falzmarke (45 mm)
`\foldvskipiii`	Abstand der dritten Falzmark von der zweiten Falzmarke (54 mm)
`\addvskip`	Abstand des Adressfensters von der Textbereichsoberkante (7,5 mm)
`\addrindent`	Abstand des Adressfensters vom linken Rand des Textbereiches (0 mm)
`\addrwidth`	Breite des Adressfeldes (70 mm)
`\locwidth`	Breite des „Location"-Feldes [(`\textwidth` − `\addrwidth`)/2 bei Option `slocfield` oder (`\textwidth`−`\addrwidth`)∗2/3 bei Verwendung der Option `wlocfield`]
`\refvskip`	Abstand zwischen dem Referenzfelde und der Adressfeldunterkante (5,5 mm)
`\sigindent`	Abstand der Grußformel und der Unterschrift vom linken Rand des Textbereiches (0 mm)

Befehle zum Setzen interner Abstände
Voreingestellte Längen sind in Klammern angegeben.

`\setpresigskip`	Abstand zwischen der Grußformel und der Unterschrift voreingestellt sind (2`\baselineskip`)

Schalter
An den jeweiligen Schalternamen ist noch ein `on` bzw. `off` anzuhängen.

`\foldmarks`	Faltmarken (Default = on)
`\addrfield`	Adress- und „Location“-Feld (Default = on)
`\subject`	„Betreff: “ vor `\subject` (Default = off)
`\subjectafter`	Betreff nach der Anrede setzen (Default = off)

Klassenoptionen
Die Standartoptionen (`12pt`, `oneside`, `final`, `slocfield`) können durch explizite Optionsangaben überschrieben werden.

`10pt`, `11pt`, `12pt`	Option für die Schriftgröße
`oneside`	einseitiges Layout
`twoside`	pseudo-doppelseitiges Layout
`draft`	Dokumente im Entwurfsstadium setzen
`final`	Dokumente in der Endfassung setzen
`a4paper`	Papiergröße
`wlocfield`	großes „Location“-Feld
`slocfield`	kleines „Location“-Feld
`orgdate`	eigene Datumsanpassungen oder die eines externen Pakets verwenden
`scrdate`	Datumsanpassungen des scrlettr-Pakets verwenden.

6.8 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- Markus Kohm
- **Enrico Kunz** <enricokunz@web.de>
- Jens-Uwe Morawski

7 Adressdateien mit scraddr erschließen

7.1 Überblick

Das Paket scraddr ist eine kleine Beigabe zur scrletter-Klasse. Ziel ist, die Benutzung von Adressdateien zu vereinfachen, und ihre Anwendung flexibler zu gestalten. Im Grunde stellt das Paket nur einen Lademechanismus für Adressdateien bereit, die aus `\adrentry`-Einträgen bestehen, wie sie im vorhergehenden Kapitel beschrieben sind.

`\InputAddressFile{`*Dateiname*`}`

Der Befehl `\InputAddressFile` ist der zentrale Ladebefehl von scraddr. Er erwartet als obligatorisches Argument den Namen der einzulesenden Adressdatei. Wird diese Datei nicht gefunden, wird ein Fehler ausgegeben.

Für jeden Eintrag dieser Adressdatei wird eine Reihe von Makros generiert, die es ermöglichen, auf die Daten der Adressdatei zuzugreifen. Zur Erinnerung nochmals der Aufbau eines Eintrags.

`\adrentry{`*Name*`}{`*Vorname*`}{`*Adresse*`}{`*Telefon*`}{`*F1*`}{`*F2*`}{`*Kommentar*`}{`*Kürzel*`}`

Die Zugriffsbefehle sind mit englischen, den Argumenten entsprechenden, Bezeichnungen versehen.

`\Name{`*Kürzel*`}`
`\FirstName{`*Kürzel*`}`
`\Lastname{`*Kürzel*`}`
`\Address{`*Kürzel*`}`
`\Telephone{`*Kürzel*`}`
`\FreeI{`*Kürzel*`}`
`\FreeII{`*Kürzel*`}`
`\Comment{`*Kürzel*`}`

Der Zugriff erfolg anhand des Kürzels im Argument Nummer 8 des `\adrentry`-Eintrags. Das bedeutet auch, dass das 8. Argument nicht leer sein darf. Um eine sichere Funktionsweise zu garantieren, empfielt es sich, das Kürzel nur als Folge von Ziffern und Buchstaben aufzubauen, wobei jedoch keine Umlaute benutzt werden dürfen.

Weiterhin ist zu beachten, dass bei mehrmaligen Auftreten eines Kürzels im `\adrentry`-Eintrag, die Angaben beim letzten Auftreten die gültigen sind.

7.2 Benutzung

Um das Paket benutzen zu können, ist wie gesagt, eine gültige Adressdatei zu erstellen. Diese, hier *lotr.adr* genannt, könnte beispielsweise folgendermaßen aussehen.

```
\adrentry{Beutlin}{Frodo}%
          {Der Bühl\\ Beutelsend/Hobbingen im Auenland}{}%
          {Bilbo Beutlin}{Rauchen von Pfeifenkraut}%
          {der Ringträger}{FRODO}
\adrentry{Gamdschie}{Samweis}%
          {Beutelhaldenweg 3\\Hobbingen im Auenland}{}%
          {Rosie Kattun}{Knullen}%
          {des Ringträgers treuester Gefährte}{SAM}
\adrentry{Bombadil}{Tom}%
          {Im Alten Wald}{}%
          {Goldbeere}{trällern von Nonsensliedern}%
          {Meister von Wald, Wasser und Berg}{TOM}
```

Das vierte Argument, die Telephonnummer, wurde hier leer gelassen. Erstens macht es in dem Zusammenhang keinen Sinn, und zweitens sollte dies ja auch möglich sein.

Mit dem oben beschriebenen Ladebefehl lesen wir die Adressdatei in unser Briefdokument ein:

```
\InputAddressFile{lotr.adr}
```

Mit Hilfe der vorgestellten Makros können wir dann einen Brief an den alten Tom Bombadil schreiben, in dem wir ihn fragen, ob er sich noch an zwei Gefährten aus alter Zeit erinnern kann.

```
\begin{letter}{\Name{TOM}\\\Address{TOM}}
   \opening{Lieber \FirstName{TOM} \LastName{TOM},}

    oder \Comment{TOM}, wie Dich Deine geliebte
    \FreeI{TOM} nennt.
    Kannst Du Dich noch an einen Herrn \LastName{FRODO},
    genauer gesagt \Name{FRODO}, denn es gab ja auch noch
    den Herrn \FreeI{FRODO}, erinnern.
```

```
    Er war \Comment{FRODO} im dritten Zeitalter.
    Begleitet wurde er von \Name{SAM}, \Comment{SAM}.

    Beider Vorlieben waren sehr weltlich.
    Der \FirstName{FRODO} genoß das \FreeII{FRODO}, sein
    Gefährte schätzte eine gute Mahlzeit mit \FreeII{SAM}.

    Weißt du noch? Mithrandir hat Dir bestimmt viel
    von ihnen erzählt.
  \closing{\glqq He, \Name{TOM}, komm zu unsrer Freude\grqq}
\end{letter}
```

Die Zusammensetzung aus `\FirstName{`*Kürzel*`}` `\LastName{`*Kürzel*`}` kann direkt mittels `\Name{`*Kürzel*`}` erhalten werden.

Das fünfte und sechste Argument von `\adrentry` steht zur freien Verfügung. Mit den Makros `\FreeI` und `\FreeII` kann auf diese Inhalte zugegriffen werden. Im vorliegenden Fall wurde das fünfte Argument für die Person benutzt, die der Person des `\adrentry`-Eintrages am nahsten steht. Das sechste Argument enthält im Beispiel die besondere Vorliebe der jeweiligen Person.

7.3 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- **Jens-Uwe Morawski**

8 Adressdateien aus Adressdatenbanken Dank addrconv

8.1 Adressdatenbanken

Wenn Sie Adressdateien zum Briefschreiben mit scrlettr verwenden, wie es im Abschnitt 6.6 beschrieben wird, müssen Sie sich selbst um die Ordnung in einer solchen Datei kümmern. Falls Sie nur gelegentlich und mit einem leicht überschaubaren Adressatenkreis per Brief korrespondieren, werden Sie mit den bisher gezeigten Möglichkeiten meist auch zufrieden sein. Wenn jedoch die Menge der Adressen zunimmt und auch der Umfang der Adressinformationen über die bloße Postanschrift hinauswächst, beginnt die Verwaltung der Adressen zu einem eigenen Problem zu werden. Dieses Problem besteht zunächst ganz unabhängig von KOMA-Script, und so war auch die Lösung, die Gerd Neugebauer 1994 vorstellte, nicht auf KOMA-Script und dessen Vorgänger, sondern auf BibTeX bezogen. Sein Bibliographie-Stil `address.bst` in Verbindung mit einem speziell definierten Eintragstypen für BibTeX-Datenbanken und einer `tex`-Datei machte sich den Umstand zunutze, dass BibTeX in der Lage ist, strukturierte Daten zu sortieren und in konfigurierbaren Listen auszugeben. BibTeX kann somit als Hilfsprogramm eingesetzt werden, das für Ordnung in Adressdatenbeständen sorgt.

Damit BibTeX eine Datei bearbeiten kann, muss diese in einem bestimmten Format vorliegen. Normalerweise besitzt eine solche Datei die Dateinamenserweiterung `bib` und enthält bibliographische Daten. Diese Daten werden nach Eintragstypen klassifiziert. Es ist möglich, neue Eintragstypen zu bilden und von BibTeX auswerten zu lassen.[1]

Unter einer *Adressdatenbank* verstehen wir im KOMA-Script-Guide eine BibTeX-konforme Datei. Der Begriff sollte nicht mit der Bezeichnung *Adressdatei* verwechselt werden, der im Abschnitt 6.6 eingeführt wurde. Zwar enthalten beide Dateitypen Adressdaten, doch im Unterschied zu `bib`-Dateien können `adr`-Dateien direkt mit scrlettr benutzt werden.

`@address{...}`

Für Einträge in einer Adressdatenbank gibt es den speziellen Eintragstyp `@address`. Das folgende Beispiel beschreibt das Format eines `@address`-Eintrags in einer `bib`-Datei:

[1] Die für LaTeX standardmäßig definierten Eintragstypen, ihr formaler Aufbau und die Funktionsweise von BibTeX überhaupt können hier nicht beschrieben werden. Für sie sei auf die Originaldokumentation in [Pat88a] und [Pat88b] sowie auf die Beschreibung im LaTeX-Handbuch [Lam95] verwiesen. Unsere Darstellung beschränkt sich auf die Besonderheiten, die für die Benutzung mit scrlettr zu beachten sind.

Beispiel:

```
@address{HMUS,
   name =         {Hans Mustermann},
   title =        {Mag. art.},
   organization = {Verband der Vereine},
   city =         {Heimstatt},
   zip =          01234,
   country =      {Germany},
   street =       {Mauerstra{\ss}e 1},
   phone =        {01234 / 5 67 89},
   fax =          {01234 / 5 67 89},
   mobile =       {0171 / 45 67 89},
   email =        {hm{@}work.com},
   url =          {http://www.work.com},
   note =         {Alles nur Erfindung},
   key =          {HMUS},
   birthday =     {13. August anno muri},
   nbirthday =    {0813}
}
```

Ähnliche Mustereinträge wie diesen finden Sie in der Datei `example.bib`, die KOMA-Script beiliegt. Die Adresseinträge dort sind jedoch weniger umfangreich. Die hier dargestellte ausführliche Form zeigt die Version 1.2 der von Axel Kielhorn für das Paket adrconv erweiterten Definition des Formats der Adressdatenbank. Es wird in `adrguide.tex`, der Anleitung zu adrconv, die KOMA-Script beiliegt, vollständig beschrieben. Hier beschränken wir uns auf diejenigen Felder eines `@address`-Eintrags, die für scrlettr ausgewertet werden können:

name Der Name; BibTeX erwartet als Format: `{Vorname Nachname}` oder `{Nachname, Vorname}` (wird benutzt für Briefe, Adress- und E-Mail-Verzeichnisse)

city Wohnort (wird benutzt für Briefe)

zip Postleitzahl (wird benutzt für Briefe)

street Straße (wird benutzt für Briefe)

phone Telefonnummer (wird benutzt für Briefe)

email E-Mail-Adresse (wird benutzt für E-Mail-Verzeichnisse)

key Das Kürzel (wird benutzt zum Aufrufen einer Adresse für Briefe)

birthday Geburtstagstext (so wie er in einem Geburtstagsverzeichnis gedruckt wird)

nbirthday Geburtstag in nummerischem Format: Monat zweistellig Tag zweistellig (MMDD) (wird benutzt zum Sortieren von Geburtstagsverzeichnissen)

8.2 Adressdatenbankkonverter

BibTeX erzeugt aus `bib`-Dateien (Datenbanken) `bbl`-Dateien. Eine `bbl`-Datei besteht im wesentlichen aus einer sortierten Liste. Welche Elemente einer `bib`-Datei hierfür ausgewertet werden und wie die resultierende `bbl`-Datei im einzelnen aufgebaut ist, wird dabei jeweils durch einen Bibliographie-Stil (eine `bst`-Datei) gesteuert. Die standardmäßig für die Erzeugung von Literaturverzeichnissen mit LaTeX eingesetzten Bibliographie-Stile können freilich weder `@address`-Eintragstypen auswerten noch Adressdateien im `adr`-Format erzeugen. Um eine Adressdatenbank in eine Adressdatei zu konvertieren, wird also ein eigens dafür eingerichteter Bibliographie-Stil benötigt. Axel Kielhorn hat mehrere Bibliographie-Stile entwickelt, die als Konverter von Adressdatenbanken in Adressdateien dienen können. Einige davon werden mit dem KOMA-Script-Paket mitgeliefert:[2]

`addrconv.bst` Erzeugt eine Adressdatei, die sowohl mit scrlettr zum Einfügen von Adressen in Briefe als auch mit den Programmen `dir.tex` und `phone.tex` zur Erzeugung von Adress- und Telefonverzeichnissen verwendet werden kann. Es werden dabei jeweils die ersten vier Felder sowie das achte Feld der Adress-Einträge (*Name*, *Vorname*, *Adresse*, *Telefonnummer* und *Kürzel*) belegt. Die Adress-Einträge in der Datei werden alphabetisch nach den Namen sortiert und die `\adrchar`-Einträge werden am Beginn jeder Buchstabengruppe automatisch eingesetzt. Bitte beachten Sie, dass Sie das Kürzel, mit dem Sie eine Adresse für einen Brief in scrlettr aufrufen wollen, von hier entnehmen müssen.

`birthday.bst` Erzeugt eine Adressdatei, die mittels `dir.tex` als Geburtstagsverzeichnis ausgegeben werden kann. Hierfür werden die Einträge nach Monat und Jahr sortiert.

[2] Weitere und aktuellere finden sich in dem Paket adrconv vom selben Autor [Kie99], das sich jedoch noch in der Entwicklung befindet. adrconv ist zwar funktional auf KOMA-Script abgestimmt, wird jedoch eigenständig gepflegt. Es ist daher geplant, Dateien des Pakets adrconv zukünftig KOMA-Script nicht mehr beizufügen, sondern sie durch einen Verweis auf adrconv [Kie] zu ersetzen.

`email.bst` Erzeugt eine Adressdatei, die durch Bearbeitung mit `dir.tex` ein E-Mail-Verzeichnis ergibt. Sie ist alphabetisch nach Namen sortiert.

8.3 Ablauf der Konvertierung

Damit BibTeX eine Adressdatenbank mit Hilfe eines Bibliographie-Stils in eine Adressdatei konvertieren kann, benötigt es noch Informationen darüber, welche der Einträge aus der `bib`-Datei auf diese Weise bearbeitet werden sollen. Diese Informationen entnimmt BibTeX der `aux`-Datei, die beim TeX-Lauf über eine `tex`-Datei entsteht und die Schlüsselwörter für BibTeX enthält, welche normalerweise durch `\cite`-Befehle in der `tex`-Datei erzeugt werden.

In unserem Fall gibt es keine derartige `tex`-Datei. Stattdessen wird die benötigte `aux`-Datei im Zuge eines interaktiven TeX-Laufs gebildet, bei dem Sie nach dem Namen der Adressdatenbank gefragt werden, die Sie konvertieren möchten. Zu den drei Bibliographie-Stilen existieren dafür interaktive TeX-Programme mit den Namen `addrconv.tex`, `birthday.tex` und `email.tex`, die jeweils die passenden `aux`-Dateien erzeugen können.

Die Konvertierung einer Adressdatenbank in eine Adressdatei läuft daher in drei Schritten ab:

1. Vorbereitung der Konvertierung durch Erzeugen der `aux`-Datei aus einer `bib`-Datei
2. Konvertierung der `bib`-Datei mittels BibTeX
3. Umbenennung der entstandenen `bbl`-Datei in die `adr`-Namensform für Adressdateien

Beispiel: Angenommen, Sie haben einen neuen Eintrag in Ihre Adressdatenbank `adressen.bib` aufgenommen, der so aussehen könnte:

```
@address{DANTE,
name      = {{DANTE~e.\,V.}},
street    = {Postfach 10 18 40},
zip       = {69008},
city      = {Heidelberg},
phone     = {0 62 21 / 2 97 66},
fax       = {0 62 21 / 16 79 06},
email     = {dante{@}dante.de},
url       = {http://www.dante.de},
key       = {DANTE},
```

```
birthday  = {14. April 1989},
nbirthday = {0414}
}
```

Wenn Sie eine Adressdatei für Briefe und ein Adressverzeichnis brauchen, wählen Sie den Konverter `addrconv` und erzeugen die `aux`-Datei. Die Protokolldatei `addrconv.log` zeigt, wie das abgelaufen ist:

```
sh>tex addrconv.tex
This is TeX, Version 3.14159 (Web2C 7.3.2x) (format=tex
2001.8.1)  13 AUG 2001 05:26
**addrconv.tex
(/texmf/tex/latex/koma-script/addrconv.tex
Now you have to typein the name of the BibTeX
addressfile, you want to convert to
script-address-file-format (without extension '.bib'):
Geben Sie nun den Namen der BibTeX-Adressdatei ein, die
Sie in das Script-Adressdateiformat konvertieren wollen
(ohne '.bib'):

addressfile=adressen
\auxfile=\write0
\openout0 = 'adressen.aux'.


After running BibTeX rename file 'adressen.bbl' to
'adressen.adr'!
Nach dem BibTeX-Lauf benennen Sie bitte die Datei
'adressen.bbl' in 'adressen.adr' um!
[1] )
Output written on addrconv.dvi (1 page, 224 bytes).
```

Als zweiten Schritt rufen Sie BibTeX zur Konvertierung auf. Wir zeigen das Protokoll `adressen.blg`:

```
sh>bibtex adressen
This is BibTeX, Version 0.99c (Web2C 7.3.2x)
The top-level auxiliary file: adressen.aux
The style file: addrconv.bst
Database file #1: adressen.bib
```

Zuletzt benennen Sie die Datei um:

```
sh>mv adressen.bbl adressen.adr
```

Die konvertierte Adressdatei hat folgenden Inhalt:

```
\adrchar{K}
\adrentry{Kielhorn}{Axel}
{Stra{\ss}e des 16.~Mai 17 \\
 38118 Braunschweig}{0531 / 89 34 39}{}{}{}{}
\adrentry{Kielhorn}{Ralf}
{Stra{\ss}e des 17.~Juni 17 \\
 38118 Braunschweig}{0531 / 89 34 39}{}{}{}{}
\adrentry{Kohm}{Markus}
{Fichtenstra{\ss}e 63 \\
 68535 Edingen-Neckarhausen}{}{}{}{}{}

\adrchar{M}
\adrentry{Mustermann}{Hans}
{Einbahnstra{\ss}e 1 \\
 01234 Heimstatt}{01234 / 5 67 89}{}{}{}{}

\adrchar{{}
\adrentry{{DANTE~e.\,V.}}{}
{Postfach 10 18 40 \\
 69008 Heidelberg}{0 62 21 / 2 97 66}{}{}{}{DANTE}
```

Der fehlende Vorname führt hier zu einem Fehler im `\adrchar`-Eintrag. Nachdem Sie ihn in:

```
\adrchar{D}
```

verbessert und zusammen mit dem Adress-Eintrag an die richtige Stelle verschoben haben, können Sie einen Brief an DANTE e. V. dann so beginnen:

```
\documentclass{scrlettr}
\usepackage{german}
\begin{document}
\begin{letter}{\DANTE}
...
```

Um Ihre Adressdateien aktuell zu halten, müssen Sie diese drei Schritte jedesmal wiederholen, wenn Sie Änderungen an Ihrer Adressdatenbank vorgenommen haben.

Bitte beachten Sie dabei, dass die drei Programme zum Erzeugen der `aux`-Datei (`addrconv.tex`, `birthday.tex` und `email.tex`) sowohl mit (Plain)TeX als auch mit LaTeX aufgerufen werden können, während die beiden Programme `dir.tex` und `phone.tex`, die Sie nach erfolgreicher Konvertierung auf Ihre Adressdatei anwenden können, um daraus fertige Adress-, E-Mail- bzw. Telephonverzeichnisse zu produzieren, nur mit LaTeX funktionieren.

8.4 Autoren

Die folgenden Autoren waren an diesem Kapitel beteiligt oder haben die Vorlage dafür geliefert.

- Frank Neukam
- Markus Kohm <Markus.Kohm@gmx.de>
- Axel Sommerfeldt
- **Thomas Neumann** <th-neumann@gmx.net>

Literaturverzeichnis

[Bra01] JOHANNES BRAAMS: *Babel, a multilangual package for use with LaTeX's standard document classes*, Februar 2001.
`CTAN:/tex-archive/macros/latex/required/babel`

[Car98] DAVID CARLISE: *The longtable package*, Mai 1998.
`CTAN:/tex-archive/macros/latex/required/tools`

[Car99a] DAVID CARLISLE: *The ifthen package*, September 1999.
`CTAN:/tex-archive/macros/latex/base`

[Car99b] DAVID P. CARLISLE: *Packages in the 'graphics' bundle*, Februar 1999.
`CTAN:/tex-archive/macros/latex/required/graphics`

[Dal99] PATRICK W. DALY: *Natural Sciences Citations an References*, Mai 1999.
`CTAN:/tex-archive/macros/latex/contrib/supported/natbib`

[DUD96] DUDEN: *Die deutsche Rechtschreibung*, DUDENVERLAG, Mannheim, 21. Auflage, 1996.

[Fai99] ROBIN FAIRBAIRNS: *topcapt.sty*, März 1999.
`CTAN:/tex-archive/macros/latex/contrib/misc/topcapt.sty`

[Kie] AXEL KIELHORN: *LaTeX page.*
`http://www-public.tu-bs.de:8080/~i0080108/latex.html`

[Kie99] AXEL KIELHORN: *Address guide*, November 1999.
`CTAN:/tex-archive/macros/latex/contrib/supported/koma-script/adrguide.tex`

[Kil99] JAMES KILFIGER: *extsizes, a non standard LaTeX-package*, November 1999.
`CTAN:/tex-archive/macros/latex/contrib/other/extsizes`

[Lam95] LESLIE LAMPORT: *Das LaTeX-Handbuch*, Addison-Wesley Publishing Company, Bonn, Paris u. a., 1995.

[Lin01] Anselm Lingnau: *An Improved Environment for Floats*, Juli 2001.
`CTAN:/tex-archive/macros/latex/contrib/supported/float`

[Mit00] Frank Mittelbach: *An environment for multicolumn output*, Juli 2000.
`CTAN:/tex-archive/macros/latex/required/tools`

[Oos00] Piet van Oostrum: *Page layout in LaTeX*, Oktober 2000.
`CTAN:`
`/tex-archive/macros/latex/contrib/supported/fancyhdr`

[Pat88a] Oren Patashnik: *BibTeXing*, Februar 1988.
`CTAN:/tex-archive/biblio/bibtex/distribs/doc/btxdoc.tex`

[Pat88b] Oren Patashnik: *Designing BibTeX Styles*, Februar 1988.
`CTAN:/tex-archive/biblio/bibtex/distribs/doc/btxhak.tex`

[Rat01] Sebastian Rathz: *Hypertext marks in LaTeX: the hyperref package*, Februar 2001.
`CTAN:`
`/tex-archive/macros/latex/contrib/supported/hyperref`

[RH01] Bernd Raichle und Thomas Hafner: *DE-TeX-/DANTE-FAQ*, Februar 2001.
`CTAN:/tex-archive/usergrps/dante/de-tex-faq`

[SKPH99] Walther Schmidt, Jörg Knappen et al.: *LaTeX 2ε-Kurzbeschreibung*, April 1999.
`CTAN:/tex-archive/info/lshort/german/l2kurz.dvi`

[Som95] Harald Axel Sommerfeldt: *caption package*, Oktober 1995.
`CTAN:`
`/tex-archive/macros/latex/contrib/supported/caption`

[Tea99a] LaTeX3 Project Team: *LaTeX 2ε for authors*, September 1999.
`CTAN:/tex-archive/macros/latex/base/usrguide.tex`

[Tea99b] LaTeX3 Project Team: *LaTeX 2ε for class and package writers*, September 1999.
`CTAN:/tex-archive/macros/latex/base/clsguide.tex`

[Tea00] LaTeX3 Project Team: *LaTeX 2ε font selection*, Januar 2000.
`CTAN:/tex-archive/macros/latex/base/fntguide.tex`

[Tob00] GEOFFREY TOBIN: *setspace LaTeX package*, Dezember 2000. `CTAN:/tex-archive/macros/latex/supported/setspace`

[Tsc87] JAN TSCHICHOLD: *Ausgewählte Aufsätze über die Gestalt des Buches und der Typographie*, Birkhäuser Verlag, Basel, 2. Auflage, 1987.

[Ume00] HIDEO UMEKI: *The geometry package*, Juni 2000. `CTAN: /tex-archive/macros/latex/contrib/supported/geometry`

[WF00] HANS PETER WILLBERG und FRIEDRICH FORSSMAN: *Erste Hilfe in Typografie*, Verlag Hermann Schmidt, Mainz, 2000.

Index

Kursive Zahlen geben die Seiten der Erklärung zu einem Stichwort wieder. Normal gedruckte Zahlen sind hingegen Seiten mit zusätzlichen Informationen zum jeweiligen Stichwort.

P

Q

R

S

T

U

V

W

Y

Z