COMPANION WEBSITE
CWS
www.pearson-studium.de

Helmut Kopka

LaTeX

Band 2: Ergänzungen

3., überarbeitete Auflage

st
scientific tools

ADDISON-WESLEY

Pearson Studium

Helmut Kopka

LaTeX

Band 2: Ergänzungen

3., überarbeitete Auflage

Pearson Studium

ein Imprint der Pearson Education Deutschland GmbH

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http:/ / dnb.ddb.de> abrufbar.

978-3-8273-7039-6

10 9 8 7 6 5 4 3 2

05 04 03

ein Imprint der Pearson Education Deutschland GmbH
Martin-Kollar-Straße 10–12, D-81829 München/Germany

www.pearson-studium.de
Lektorat: Irmgard Wagner, Planegg, Irmgard.Wagner@munich.netsurf.de
Korrektorat: Petra Kienle, Fürstenfeldbruck
Einbandgestaltung: dyadesign, Düsseldorf
Satz: Helmut Kopka
Druck und Verarbeitung: Bercker, Kevelaer
Printed in Germany

Vorwort

Der vorliegende Band 2 der LaTeX-Buchserie erhält ab der überarbeiteten 2. Auflage seine endgültige Form bezüglich der Stoffgliederung für die drei Bände und entspricht in seinem Inhalt den Ankündigungen aus Band 1. Das bisherige Kapitel 1 mit den Hinweisen zu LaTeX 2_ε wurde von diesen Hinweisen befreit, da sie nun Bestandteil der Einführung und damit von Band 1 ab dessen 2. Auflage geworden sind.

Das Kapitel 1 wurde deshalb neu konzipiert und erweitert. Es stellt nun, bis auf die PostScript-Schriften, alle LaTeX-Erweiterungen vor, die von den Autoren des LaTeX 3-Projekts stammen. Es beginnt mit den Hinweisen zur Beschaffung und Installation der Standarderweiterungen, gefolgt von der Nutzungsbeschreibung dieser Erweiterungen. Hierzu zählen erweiterte Tabellenumgebungen für ein- und mehrseitige Tabellen, Umgebungen für mehrspaltige Formatierungen mit wechselseitiger Höhenbalancierung, die gleichzeitig einen Wechsel der Spaltenzahl innerhalb einzelner Seiten erlauben, Erweiterungen der Aufzählungsumgebungen sowie für Regelsätze und Querverweise.

Die bisher genannten Erweiterungen stellen die LaTeX-Standarderweiterungen im engeren Sinne dar. Zusätzlich wird das Babel-System vorgestellt, das die Makropakete und Definitionsfiles zur LaTeX-Bearbeitung von Texten für nahezu alle auf der lateinischen, griechischen oder kyrillischen Schrift aufbauenden Sprachen enthält, wobei die jeweilige Sprache mit einem sog. Sprachschalter lokal oder global aktiviert werden kann. Das Kapitel 1 schließt ab mit der Vorstellung von $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX, mit dem die bereits sehr hohe Formatierungsleistung von LaTeX für den mathematischen Formelsatz nochmals erweitert wird.

Die Kapitel 2 und 5 stellen die Nutzung und Einbindung von TeX-Zusatz- und Sonderschriften sowie der breiten Palette der PostScript-Schriften mit LaTeX vor. Ein neues Zeichensatzauswahlverfahren, das bereits mit LaTeX 2.09 als Zusatz genutzt werden konnte, ist nunmehr Standardbestandteil von LaTeX. Es erleichtert den Zugang zu weiteren Schriften und deren Programmverwaltung. Die vorgestellten Schriften könnten auch mit LaTeX 2.09 verknüpft werden. Dies verlangt jedoch erheblich umfangreichere Anwenderanpassungen, falls sie nicht durch beigestellte Stilfiles unterstützt werden.

Kapitel 2 mit den TeX-Zusatz- und -Sonderschriften sollte ursprünglich auch Hinweise zur Verwendung von Zeichensätzen für den Notensatz enthalten. Diese gediehen dann zu dem eigenen Kapitel 4, das das Ergänzungspaket MusiXTeX vorstellt, mit dem komplexe polyphone Notensätze durch LaTeX erstellt werden können. Wenn man bedenkt, dass trotz der Erfindung der Druckkunst vor mehr als 500 Jahren, die Erstellung der Druckvorlagen für qualitativ hochwertigen Notensatz bis in die jüngste Zeit ein manueller Arbeitsvorgang war, so ist die jetzt möglich gewordene Automatisierung eine bewundernswerte Leistung.

Das Kapitel 2 schloss bei den ersten beiden Auflagen dieses Buches mit der Vorstellung von Zeichensätzen zur Schachdokumentation und Makrosätzen zu deren Eingabenotation ab.

Mit der nun vorliegenden überarbeiteten 3. Auflage leitet diese Vorstellung nun als Abschnitt 3.1 das neue Kapitel 3 ein, das in weiteren Abschnitten die Zeichensätze und Eingabemakros für weitere Spieledokumentationen wie Backgammon, Go und Kartenspielen sowie zur Erstellung von Ausgabediagrammen von Kreuzworträtseln vorstellt.

Die bisherige Beschreibung von MusicTEX zur Erstellung von Musiknotensätzen mittels LATEX wurde mit der jetzigen 3. Auflage durch die Vorstellung des verbesserten und leistungsfähigeren Makropakets MusiXTEX und seiner speziellen Noten-Zeichensätzen ersetzt. Dieses Nachfolgepaket für den Musiknotensatz besticht zusätzlich durch seine schlüssigere Eingabenotationen und der Verwendung ausschließlicher englischer Befehlsnamen gegenüber der Mischung von Befehlsnamen aus englischen und französischen Begriffen.

Das Kapitel 5 über die PostScript-Schriften und die Einbindung von PostScript-Grafiken leitet über zu weiteren Kombinationen von Text und Grafik, für die TEX zunächst nicht vorgesehen war. Kapitel 6 stellt das Programm bm2fonts vor, das Grafikfiles aus verschiedensten Quellen für die TEX-Bearbeitung aufbereitet, die anschließend zwanglos mit dem Text verknüpft werden und zusammen mit dem Text auf jedem Drucker, der diesen Text ausgeben kann, dort ebenfalls erscheinen.

Kapitel 7 stellt als weiteres Grafikwerkzeug PICTEX vor, das komplexe zweidimensionale Digramme mit TEX-spezifischen Mitteln erstellt, bearbeitet und mit dem umgebenden Text verknüpft. Die erforderlichen Eingabebefehle werden dem LATEX-Anwender schnell vertraut werden, da sie formal und strukturell den bisherigen LATEX-Befehlen verwandt erscheinen.

Nach der breiten Palette von Schriften und Grafiken erschien es mir konsequent, den Band 2 mit einer Kurzvorstellung von METAFONT abzuschließen. Die Installation neuer Zeichensätze oder deren Umstellung auf einen neuen Druckertyp wird fast immer den Aufruf von METAFONT verlangen, mit dem die Zeichensatz-Quellenfiles auf die beim Anwender verfügbare Rechner- und Druckereinrichtung zielgerichtet zugeschnitten werden.

Die 3. überarbeitete und erweiterte Neuauflage vom April 2002 enthält als Buchbeilage die CD-ROM „TEX Live 6b“ mit der Bereitstellung aller offiziellen TEX-Programme für eine Vielzahl von Rechnern und Betriebssystemen sowie nahezu aller in diesem Buch vorgestellten Makropaketen. Eine kurze Inhalts- und Nutzungsbeschreibung dieser CD erfolgt in Abschnitt 1.7.3.

Helmut Kopka, März 1997 und April 2002

Inhaltsverzeichnis

Kapitel 1

LaTeX-Weiterentwicklungen

1.1 Vorbemerkungen

Mit Bereitstellung der LaTeX-Version 2.09 hat LESLIE LAMPORT, der Autor des Originalprogramms von LaTeX, seine Tätigkeit an LaTeX-Fortentwicklungen definitiv eingestellt. Die Versionsnummer 2.09 war nahezu 8 Jahre aktuell, was aber nicht bedeutet, dass LaTeX-Pakete mit dieser Kennzeichnung identisch sind. Unter der gleichen Versionsnummer erschienen, durch ihre Erstellungsdaten gekennzeichnet, unterschiedliche Fassungen. So wurde z. B. LaTeX, das von seinem Autor primär für die Bearbeitung englischer Texte eingerichtet wurde, internationalisiert. Trägt ein LaTeX-Paket 2.09 ein Erstellungsdatum ab dem 1. Dezember 1991, so handelt es sich um die internationale Version, die zusammen mit sprachspezifischen Dokumentstiloptionen, wie z. B. `german`, dann automatisch erscheinende Begriffe wie **Chapter**, **Contents** u. a. in deutsch als **Kapitel**, **Inhaltsverzeichnis** usw. ausgibt.

Die lange Lebensdauer der Version 2.09 beweist die Stabilität von LaTeX als Standard-Zugangspaket zu TeX. Die unterschiedlichen Varianten mit verschiedenen Erstellungsdaten haben den LaTeX-Kern, von Fehlerkorrekturen abgesehen, nie geändert. Die wesentlichen Änderungen erfolgten in den zusätzlichen Stilfiles. Andere Ergänzungen erforderten dagegen die Erstellung neuer oder weiterer Formatfiles. So war es bis zur Bereitstellung von TeX 3.0 stets erforderlich, bei mehrsprachigen Anwendungen je ein LaTeX-Formatfile (genauer, ein Formatfile aus `lplain.tex`) jeweils mit den sprachspezifischen Trennmusterfiles zu erstellen. Ab TeX 3.0 können im Prinzip bis zu 256 verschiedene Trennmusterfiles in einem Formatfile zusammengefasst werden, wobei das jeweils zu aktivierende Trennmuster mit der Zuweisung `\language=`*sprache_no* oder mit einem sprachspezifischen Ergänzungspaket (Stilfile) über einen Sprachauswahlbefehl, wie `\selectlanguage{`*sprache*`}`, eingestellt wird.

Auch das seit längerem verbreitete neue Zeichensatzauswahlverfahren NFSS (New Font Selection Scheme) von FRANK MITTELBACH und RAINER M. SCHÖPF verlangt die Erzeugung eines weiteren oder ersetzenden Formatfiles. Die Verwendung verschiedener Formatfiles ist unproblematisch, solange die Bearbeitung nur lokal, also beim Anwender, der diese Formatfiles erzeugte, erfolgt. Sollen dagegen LaTeX-Textfiles im Originalzustand per E-Mail oder als Diskettenkopien versandt und an anderer Stelle mit LaTeX bearbeitet werden, so kann dies wegen eventuell verschiedener Formatfiles auf Kompatibilitätsprobleme stoßen.

Damit kann auch eine Grundeigenschaft von LaTeX verloren gehen: LaTeX sollte auf allen Rechnern und unter allen Betriebssystemen identische Ergebnisse liefern. Auf Initiative von FRANK MITTELBACH und RAINER SCHÖPF entstand die internationale Planungs- und Entwicklungsgruppe für das „LaTeX 3-Projekt“, an dem sich auch LESLIE LAMPORT beratend beteiligt. Zu den Zielen des LaTeX 3-Projekts gehört es, einen LaTeX-Kern zu entwickeln, der für alle zukünftigen Ergänzungen, wie immer sie auch geartet sein mögen, nur ein Formatfile vorhält. Das setzt voraus, dass die Schnittstelle zwischen dem eigentlichen LaTeX-Kern und den Ergänzungen klar, eindeutig und umfassend zu definieren und programmtechnisch zu realisieren ist.

Eine weitere Zielsetzung des LaTeX 3-Projekts ist die weitgehende Ersetzung der Erklärungsbefehle (engl. *declaration,* [5a, Abschn. 2.3]) durch Befehle mit Argumenten. Erklärungen sind Befehle (Erklärungsbefehle), nach deren Aufruf bestimmte Bearbeitungseigenschaften lokal bis zum Ende der laufenden Umgebung oder bis zum nächsten umstellenden Erklärungsbefehl wirken. Damit wird dann z. B. statt der bisherigen Angabe `{\em Hervorhebung}` die sinnvollere Angabe `\emph{Hervorhebung}` möglich. Die Bereitstellung weiterer oder leistungsfähigerer Befehle wird ganz besonders die Zeichensatzauswahl betreffen. Eine modifizierte Form des Zeichensatzauswahlverfahren NFSS der Version 2 wird integraler Bestandteil künftiger LaTeX-Versionen werden.

Bei der Fort- und Weiterentwicklung großer Programmsysteme oder Programmiersprachen hat der Gesichtspunkt der Kompatibilität zu früheren Programmversionen stets einen sehr hohen Stellenwert. Genau genommen ist es die Abwärtskompatibilität, deren Erhaltung stets gefordert wird. Diese erlaubt für Fortentwicklungen zwar neue und weitere Sprachelemente, durch die jedoch die Bearbeitung bestehender Programme oder Texte unverändert möglich sein muss. Neue Texte und Programme, die von den neuen oder erweiterten Sprachelementen Gebrauch machen, können mit den älteren Programmversionen im Allgemeinen nicht oder nicht im erweiterten Sinne bearbeitet werden.

Die Kompatibilitätsforderung steht einer Fortentwicklung oft sehr störend im Weg, bis hin zu der Folge, dass wünschenswerte Erweiterungen unterbleiben müssen, wenn sie die Kompatibilität aufheben. Um diese Schranke für Weiterentwicklungen zu durchbrechen, wird oft ein Kompromiss akzeptiert, bei dem der neuen Version eine oder einige Optionen zugefügt werden, nach deren Aktivierung frühere Programme oder Texte in herkömmlicher Weise bearbeitet werden können.

Viele LaTeX-Anwender würden vermutlich eine neue LaTeX-Version ablehnen, wenn ihre alten Texte damit nicht mehr bearbeitet werden könnten. Sie würden eine solche Version aber freudig begrüßen, wenn diese einerseits weitere leistungsfähige Möglichkeiten eröffnet und andererseits die Bearbeitung bestehender Dokumente sicherstellt. Die Erhaltung der Abwärtskompatibilität ist deshalb auch für das LaTeX 3-Projekt zu fordern.

Der Arbeitskreis für das LaTeX 3-Projekt hatte Ende 1993 eine neue LaTeX-Version mit der Bezeichnung LaTeX 2_ε als Testversion auf den internationalen TeX-Servern bereitgestellt. Nach einer Erprobungsphase und einigen Korrekturen und Verbesserungen ist dies ab Juni 1994 die offizielle LaTeX-Version, die auf den TeX-Fileservern unter dem Verzeichnisnamen `latex` geführt wird. Die bisherige LaTeX-Version wird nunmehr explizit mit LaTeX 2.09 gekennzeichnet, die entsprechend auf den Filesevern nun unter `latex209` bis auf weiteres vorgehalten wird.

Die LaTeX 2_ε-Version erfüllt bereits wesentliche Zielvorgaben für das LaTeX 3-Projekt. Der zugehörige LaTeX-Kern erlaubt weitgehend beliebige Ergänzungen, so dass alle solche Ergänzungen auf ein einheitliches LaTeX-Formatfile zurückgreifen. Ebenso sind viele der

früheren Erklärungen aus LaTeX 2.09 durch argumentbehaftete Befehle ergänzt worden. Das frühere Zeichensatzauswahlverfahren NFSS von FRANK MITTELBACH und RAINER SCHÖPF ist, basierend auf dessen Version 2, nunmehr integraler Bestandteil von LaTeX. Die geforderte Abwärtskompatibilität zu LaTeX 2.09 ist sichergestellt.

Der Einführungsband 1 dieser Buchserie stellt deshalb ab der 2. Auflage vom November 1995 die Eigenschaften von LaTeX 2_ε als Standardeigenschaften von LaTeX vor. Ab der dortigen 3. Auflage vom April 2000 entfallen auch die bisherigen zusätzlichen Vorstellungen von LaTeX 2.09-Eigenschaften, da diese inzwischen als obsolet gelten und LaTeX-Neueinsteiger nur verwirren würden.

1.2 LaTeX-Ergänzungen und deren Installation

Das LaTeX-Gesamtpaket ist auf den offiziellen TeX-Fileservern in die drei Eingangsverzeichnisse `.../base`, `.../required` (bis März 1999 unter dem Namen `.../packages`) und `.../contrib` untergliedert. Diese Eingangsverzeichnisse findet man auf den TeX-Fileservern unter dem mit ... symbolisierten Pfadnamen unter `/tex-archive/macros/latex`, wobei der Eingangspfadname `/tex-archive` je nach Serverquelle eventuell einen anderen Namen trägt, aus dem das Wurzelverzeichnis für das gesamte TeX-Archiv jedoch ebenfalls erkennbar wird.

Die Eingangsverzeichnisse `./base`, `./required` und `./contrib` mit den Quellenfiles für das LaTeX-Gesamtpaket werden unter diesen Namen vermutlich auch bei jeder lokalen LaTeX-Installation auftreten, auch wenn einige der lokalen Unterverzeichnisse dann meistens nur einen Bruchteil der Files enthalten, die die offiziellen TeX-Fileserver unter diesen Verzeichnissen anbieten.

1.2.1 Das TeX-Filesystem

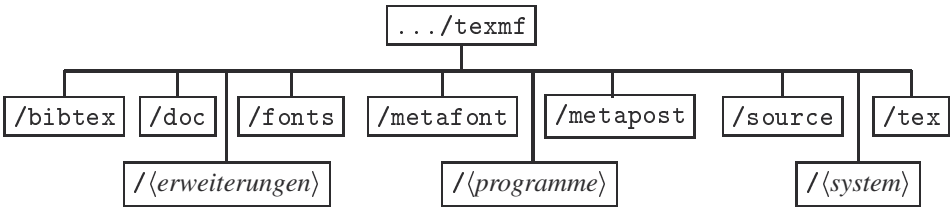
Die ausführbaren TeX-Programme erwarten die angeforderten Files unter bestimmten Pfad- und Verzeichnisnamen, die das so genannte TeX-Filesystem bilden. Dieses wird bei der Kompilierung der Quellenprogramme vorbestimmt und kann bei den meisten Betriebssystemen mit Umgebungsvariablen abgeändert werden. Soweit die ausführbaren Programme beim Anwender durch eigene Kompilierung erstellt werden, kann das geforderte Filesystem nach den Vorstellungen des Anwenders durch Editieren der zugehörigen Makefiles vorgenommen werden. Dies ist z. B. bei der Installation eines TeX-Systems unter UNIX der Fall.

Werden die ausführbaren Programme als fertige binäre Files geliefert, wie das bei den meisten PC-TeX-Systemen der Fall ist, dann muss das eingebaute Filesystem übernommen oder mit dem Setzen expliziter Umgebungsvariablen an die Wünsche des Anwenders angepasst werden. Dabei hängt es von der Lieferquelle ab, welche Files mit welchen Umgebungsvariablen zugeordnet werden können. Dies führt meistens zu unterschiedlichen TeX-Filesystemen für die verschiedenen Herkunftsquellen.

Die internationale TeX-Users-Group (TUG) hat deshalb einen Arbeitskreis eingerichtet, der einen Vorschlag für ein einheitliches TeX-Filesystem erarbeiten soll. Ein solcher wurde Mitte Juni 1995 unter dem Kürzel TDS (TeX Directory Structure, Vers. 0.98) vorgestellt und seitdem als Folge vielfältiger Diskussionen und Anregungen mehrfach verbessert. Die derzeit (März 2002) aktuelle Version ist immer noch 0.9995 aus 1998. Eine neuere Probeversion 0.9996 vom 21. April 1999 wurde offiziell bisher noch als endgültig akzeptiert.

Nach diesem Vorschlag soll das Ausgangsverzeichnis für das TeX-Gesamtsystem den Namen `.../texmf` tragen. Die Einbindung dieses TeX-Ausgangsverzeichnisses in das gesamte Rechner-Filesystem bleibt dem Systemverwalter überlassen. Unter UNIX ist es traditionell `/usr/local/lib/texmf` und unter DOS sowie WINDOWS*xx* vermutlich ein Hauptverzeichnis eines Laufwerks, z. B. `D:\texmf`.

Die erste Ebene für das TeX-Eingabeverzeichnis `.../texmf` soll dann mindestens aus den Unterverzeichnissen



bestehen. Die Namen sind teilweise selbsterklärend, zumindest die in Schreibmaschinenschrift angegebenen. Die meisten dieser Unterverzeichnisse sind ihrerseits untergliedert. So enthält das Unterverzeichnis `./tex` alle TeX-Makro- und -Definitionspakete, die zum Ablauf eines TeX-Auftrags erforderlich sind. Es sollte mindestens in `./plain`, `./latex` und `./generic` untergliedert sein, evtl. ergänzt durch weitere Makropakete wie `./amstex`, `./musixtex` und Ähnliche, wenn diese beim Anwender genutzt werden. Die Trennung der Pfadnamensbestandteile durch den Schrägstrich `/` entspricht der UNIX-Notation. Die Umsetzung auf den äquivalenten Rückstrich `\` für DOS oder die Syntaxübernahme der Filenamen für VMS sollte ohne Erläuterung leicht möglich sein.

Das Unterverzeichnis `./source` sammelt alle Quellenfiles, wie sie vor der eigentlichen Installation zunächst aus ihren Lieferquellen zu kopieren sind. Bei beschränkter Plattenspeicherkapazität können sie nach einer erfolgreichen Installation dort eventuell wieder gelöscht werden. Zur systematischen Zuordnung wird man das Unterverzeichnis `./source` weiter untergliedern, für die Quellenfiles eines LaTeX-Gesamtsystems z. B. zunächst in `./source/latex` und hierunter dann in `./base`, `./required` und `./contrib` zur Aufnahme der ausgewählten Quellenfiles aus den gleichnamigen Verzeichnissen der TeX-Fileserver.

Ausführliche Hinweise zur TeX-Verzeichnisstruktur gemäß dem TDS-Vorschlag wurden bereits im Anhang F.1.4 von Band 1 (3. Aufl.) dieser Buchserie gegeben. Bei Bedarf wird auf diese bzw. auf die Originaldokumentation zum TDS-Vorschlag verwiesen. Hier folgen nur einige Erläuterungen zu den Kästchen für die erste Unterverzeichnisebene mit den Angaben in Kursivschrift innerhalb der Winkelklammern.

`./`⟨*erweiterungen*⟩ steht für ein oder mehrere Verzeichnisse in Ergänzung zu `./tex`, das alle TeX-Makro- und Definitionspakete enthält, die zum Ablauf eines Standard-TeX-Auftrags erforderlich sind. Inzwischen gibt es neben dem TeX-Standardpaket TeX-Erweiterungen, wie z. B. das `etex`- (extended TeX), `pdftex`- oder `omega`-Paket. Auf Eigenschaften dieser Erweiterungen gehe ich, mit Ausnahme einiger kurzer Hinweise zu `pdftex`, hier nicht ein.

Die ausführbaren Programme von `pdftex` erzeugen als Bearbeitungsergebnisse keine `.dvi`-Files, sondern `.pdf`-Files, also erweiterte PostScript-Files, die mit den Program-

men aus dem Acrobat-Progammpaket von Adobe ausgegeben oder nachbearbeitet, z. B. gemischt oder verschachtelt und mit weiteren Grafiken angereichert werden können.

Weitere TeX-Erweiterungen werden hier nicht genannt. Wichtig ist nur, dass für solche Erweiterungen jeweils ein eigenes Eingangsverzeichnis eingerichtet wird, das parallel zum Standard-TeX-Eingangsverzeichnis steht. Soweit entsprechende Erweiterungen für METAFONT angeboten werden, gilt dasselbe für entsprechende Parallelverzeichnisse zu `./metafont`.

`./`⟨*programme*⟩ steht für jeweils ein Verzeichnis, mit den Konfigurations- und Definitionsfiles für Zusatzprogramme wie die diversen DVI-Treiber oder das Programm MakeIndex. Beispiele für Verzeichnisnamen von `/`⟨*programme*⟩ sind damit `./dvips` und `./makeindx`.

`./`⟨*system*⟩ enthält system- oder herkunftsspezifische Bestandteile einer TeX-Installation. In der TDS-Dokumentation steht hier die englische Bezeichnung ‘implementation’, dessen Bedeutung in diesem Zusammenhang am ehesten mit ‚Ausführung‘ zu übersetzen ist. In diesem Verzeichnis sind die beim Anwender erstellten oder kopierten Format-, Basis- und Poolfiles abzulegen. Als Beispiel für einen realen Verzeichnisnamen könnte hier `/emtex` stehen, wenn beim Anwender emTeX zur Anwendung kommt. Auf meinem UNIX-System steht für dieses Verzeichnis `/web2c`.

1.2.2 Aufbereitung und Dokumentation der LaTeX-Quellenfiles

Das Eingangsverzeichnis `./base` enthält die Quellenfiles für das LaTeX-Grundsystem sowie dessen Einrichtungswerkzeuge, die zur Einrichtung aller weiteren Ergänzungen ebenfalls benötigt werden. Eigenschaften des LaTeX-Grundsystems wurden vollständig in Band 1 dargestellt und dort durch das Ergänzungspaket `german.sty` für die LaTeX-Bearbeitung deutschsprachiger Texte erweitert. Die Installation des LaTeX-Grundpakets wurde in Anhang F.2.1 von Band 1 beschrieben, so dass hier nur eine kurze Zusammenfassung der Installationswerkzeuge und ihrer Nutzung gegeben wird.

Nahezu alle LaTeX-Quellenfiles werden als dokumentierte Makrofiles bereitgestellt, die durch den Namensanhang `.dtx` gekennzeichnet sind. Die dokumentierten Makrofiles enthalten die Makrodefinitionen für die zugehörigen LaTeX-Werkzeuge und sind durch umfangreiche Erläuterungen in Form von LaTeX-Kommentaren angereichert und evtl. auch noch mit bedingungsabhängigen Auswahlstrukturen versehen.

Bei der Installation des LaTeX-Grundsystems entstehen neben den Eingabefiles für den LaTeX-Kern, aus denen durch INITEX-Bearbeitung von `latex.ltx` das LaTeX-Formatfile `latex.fmt` entsteht, alle Klassen- und Klassenoptionsfiles sowie die Ergänzungspakete für das LaTeX-Grundsystem. Außerdem entstehen bei dieser Grundinstallation die Entwicklungswerkzeuge

```
docstrip.tex   gind.ist   ltxdoc.cls
doc.sty        gglo.ist
```

auch wenn dem Anfänger, dem die Grundinstallation mit den beigefügten Installationshilfen ohne Probleme gelingen sollte, dies kaum richtig registriert. Die Entwicklungswerkzeuge `docstrip.tex`, `doc.sty` und `ltxdoc.cls` wurden bei der Grundinstallation neben vielen weiteren Bearbeitungsergebnissen dann in den Verzeichniszweig

`.../texmf/tex/latex/base` verschoben. Die MakeIndex-Stilfiles `gind.ist` und `gglo.ist` wurden bei der Grundinstallation vermutlich nach `.../texmf/makeindx` verschoben.

Das Programm `docstrip.tex` dient dazu, dokumentierten Makrokode von den beigefügten Erläuterungskommentaren zu befreien und den endgültigen Makrokode ggf. bedingungsspezifisch auszugeben oder aus mehreren dokumentierten Makrofiles zusammenzusetzen. Nahezu alle LaTeX-Quellenpakete enthalten sog. Installationsfiles, die durch den Namensanhang `.ins` gekennzeichnet sind.

Ist der Grundname des Installationsfiles identisch mit dem Grundnamen eines dokumentierten Makrofiles, so entsteht durch die LaTeX-Bearbeitung dieses Installationsfiles nur das Bearbeitungsergebnis für dieses dokumentierte Makrofile, das bei den sog. Ergänzungspaketen ein File mit dem gleichen Grundnamen und dem Anhang `.sty` ist. In einigen Fällen entstehen weitere Files, z. B. solche mit dem Anhang `.def`, die bei der Aktivierung des zugehörigen Ergänzungspakets aus dem `.sty`-File dann ihrerseits eingelesen werden.

In einigen Fällen enthalten Gruppen von dokumentierten Makrofiles ein Installationsfile mit dem Grundnamen des zugehörigen Unterverzeichnisses und dem Anhang `.ins`. Die LaTeX-Bearbeitung eines solchen Installationsfiles führt dann in einem Zug zur Bearbeitung aller dokumentierten Makrofiles aus diesem Unterverzeichnis und damit zur Erstellung der endgültigen Werkzeuge aus diesem Unterverzeichnis.

Die LaTeX-Bearbeitung dieser Installationsfiles verläuft nach meinen Erfahrungen nahezu immer problemlos, so dass dem Anwender zur Installation dieser Werkzeuge keine weiteren Kenntnisse über das unterliegende `docstrip`-Programm abverlangt werden. Solche werden erst benötigt, wenn eigene Ergänzungspakete entwickelt werden sollen. Die syntaktischen Vorschriften für dokumentierte Makrofiles und die Bearbeitungseigenschaften von `docstrip.tex` und dessen Optionsmöglichkeiten werden deshalb erst in Band 3 dieser Buchserie vorgestellt.

Hier genügt es zu wissen, dass mit der LaTeX-Bearbeitung von Installationsfiles, also mit den Aufrufen der Form

`latex` *quelle*`.ins`

das Endergebnis für das zugehörige LaTeX-Werkzeug erstellt wird, wobei implizit auf das Entwicklungswerkzeug `docstrip.tex` zurückgegriffen wird. Dieser Bearbeitungsaufruf erfolgt zweckmäßigerweise aus dem Verzeichnis mit den zugehörigen Quellenfiles heraus. Damit werden die Bearbeitungsergebnisse zunächst auch in diesem Verzeichnis abgelegt. Von hier sind sie dann abschließend in die endgültigen Zielverzeichnisse zu verschieben, nach dem TDS-Vorschlag also nach

`.../texmf/tex/latex/`*ziel_verz*

wobei für *ziel_verz* der gleiche Name zu wählen ist, wie für das Verzeichnis der Quellenfiles. Für die LaTeX-Ergänzungen sind dies damit `required` und `contrib` als Parallelverzeichnis zu `base`, das bereits bei der Grundinstallation angelegt wurde. Die Zielverzeichnisse `required` und `contrib` werden evtl. weiter untergliedert, wobei die entsprechende Untergliederung der Quellenfiles zum Vorbild genommen werden kann.

Die implizite Verwendung von `docstrip.tex` greift auf dessen Standardeigenschaften zurück. Mit der Bereitstellung eines Files `docstrip.cfg` können diese Standardeigenschaften beim Anwender modifiziert werden, da innerhalb `docstrip.tex` nach der

Existenz eines Files `docstrip.cfg` gefragt und, wenn es existiert, eingelesen und damit genutzt wird. Innerhalb von `docstrip.tex` wird ein Befehl `\usedir{`*verzeichnis*`}` definiert, mit dem ein Teilpfad- oder Verzeichnisname vorgegeben werden kann, unter dem LaTeX die zur Bearbeitung erforderlichen Bearbeitungsfiles sucht, und zwar unterhalb dem mit `\BaseDirectory{`*TDS_eing*`}` vorgegebenen Eingang für das TeX-Filesystem. Mit den Angaben

```
\BaseDirectory{TDS_eing_pfad}
\UseTDS
```

in einem anwendereigenen `docstrip.cfg`-File kann die vom Systemverwalter gewählte TDS-Eingangsstruktur zur Suche und Ablage bei der Installation von LaTeX-Ergänzungspaketen automatisiert werden. Bei meinem LINUX-System habe ich für das `docstrip`-Konfigurationsfile das Befehlspaar `\BaseDirectory{/usr/lib/teTeX/texmf}` und `\UseTDS` gewählt. Für DOS oder WINDOWS *xx* würde sich hier z. B. das Befehlspaar `\BaseDirectory{`*D*`:/texmf}` `\UseTDS` anbieten. Die Installationsfiles der LaTeX-Standardergänzungen enthalten eine geeignete Vorgabe für `\usedir{`*teil_pfad*`}`, die mit der Vorgabe für den Eingangspfad aus `docstrip.cfg` beide zum Gesamtpfad zusammenfügt.

Bei der LaTeX-Bearbeitung der Installationsfiles entstehen neben den aufbereiteten LaTeX-Werkzeugen häufig noch Files mit dem gleichen Grundnamen des Installationsfiles und dem Anhang `.drv`. Dies sind dann sog. Treiberfiles, die zur Erstellung der Dokumentation genutzt werden können. Die Dokumentation wird mit der LaTeX-Bearbeitung des zugehörigen Treiberfiles erstellt. Der LaTeX-Bearbeitungsaufruf ist zunächst zweimal auszuführen, damit alle Querverweise aufgelöst werden sowie das Inhaltsverzeichnis erstellt werden kann.

Entstehen bei der LaTeX-Bearbeitung der Treiberfiles Files mit dem Anhang `.idx` und/oder `.glo`, so sind diese nach der zweiten LaTeX-Bearbeitung mit MakeIndex in der Form

`makeindx -s gind.ist` *dok_file*	für die `.idx`-Files und
`makeindx -s gglo.ist -o` *dok_file*`.gls` *dok_file*`.glo`	für die `.glo`-Files

zu bearbeiten. Hiermit wird aus dem `.idx`-File das `.ind`-Indexfile und aus dem `.glo`-File das `.gls`-Glossarfile gebildet, wobei *dok_file* für den Grundnamen des dokumentierten Makrofiles steht. Anschließend ist das Treiberfile ein drittes Mal mit LaTeX zu bearbeiten, womit dann das endgültige `.dvi`-File für die Dokumentation erstellt wird. Dieses schließt damit ein Stichwortregister und evtl. auch seine Entwicklungsgeschichte ein.

Entfällt die Erstellung eines Treiberfiles bei der LaTeX-Bearbeitung des Installationsfiles, dann kann die Dokumentation durch direkte dreifache LaTeX-Bearbeitung des zugehörigen dokumentierten Makrofiles erstellt werden, da die dokumentierten Makrofiles die Aufrufanweisungen für ein Treiberfile stets auch implizit enthalten.

Bei der Erstellung der Dokumentation mit einem Treiberfile oder direkt aus den `.dtx`-Files entsteht gewöhnlich die Gesamtdokumentation mit der Darstellung einer kurzen Nutzungserläuterung für das zugehörige Ergänzungspaket sowie der aufbereiteten Darstellung des gesamten Realisierungskodes. Letzterer ist für viele LaTeX-Anwender nicht von Interesse, da sein Verständnis vertiefte TeX- und LaTeX-Programmierkenntnisse voraussetzt.

In einigen Fällen enthält der Treiberkode (explizit oder implizit) den herauskommentierten Befehl `%\OnlyDescription`. Wird das vorangestellte Kommentarzeichen `%` entfernt, so bewirkt der LaTeX-Programmaufruf zur Erstellung der Dokumentation eine verkürzte Dokumentation, die sich auf die Nutzungserläuterungen beschränkt.

Der implizite Treiberkode in einem dokumentierten Makropaket ist daran zu erkennen, dass ihm die Kommentarzeile '`%<*driver>`' vorangestellt ist. Der Treiberkode reicht dann bis zu der abschließenden Kommentarzeile '`%</driver>`'. Der Treiberkode besteht gewöhnlich mindestens aus den Programmzeilen:

```
\documentclass{ltxdoc}
  \EnableCrossrefs
  %\DisableCrossrefs   % Say \DisableCrossrefs if index is ready
  \RecordChanges       % Gather update information
  %\OnlyDescription    % comment out for implementation details
\begin{document}
  \DocInput{makro_file.dtx}
\end{document}
```

Die LaTeX-Bearbeitungsklasse `ltxdoc` lädt implizit das Ergänzungspaket `doc.sty` hinzu, das die Dokumentationsaufbereitung bewirkt. Auch die anderen, evtl. unbekannt erscheinenden Befehle des Treiberkodes stammen aus diesem Ergänzungspaket. Fehlt die herauskommentierte Befehlszeile '`%OnlyDescription`', so kann sie vom Anwender unmittelbar vor dem Öffnungsbefehl '`\begin{document}`' angebracht werden. Durch Entfernen des Kommentarzeichens kann dann eine eingeschränkte Dokumentation erstellt werden. Ich empfehle jedem LaTeX-Anwender, sich zumindest diese eingeschränkte Dokumentation für alle bei ihm installierten Ergänzungspakete zu erstellen.

Die Aufbereitung und Dokumentation von dokumentierten Makrofiles erfolgt für alle Quellenfiles eines LaTeX-Gesamtsystem in der beschriebenen Form, so dass diese Erläuterung vorangestellt wurde, damit sie für die verschiedenen Teilpakete nicht jedes Mal neu anzugeben ist.

1.2.3 Die LaTeX-Standardergänzungen

Das eingangs erwähnte Quellenverzeichnis `.../required` ist dadurch ausgezeichnet, dass es LaTeX-Ergänzungen enthält, die von Mitwirkenden des LaTeX 3-Projekts stammen und von diesen regelmäßig gewartet und verbessert werden. Die hier bereitgestellten Ergänzungen stellen somit die LaTeX-Standardergänzungen dar. Das Eingangsverzeichnis `.../required` ist zunächst weiter untergliedert in die Unterverzeichnisse

`./amslatex` `./babel` `./cyrillic` `./graphics` `./psnfss` `./tools`

Das Unterverzeichnis `./cyrillic` enthält die Quellen der Ergänzungspakete zur Nutzung kyrillischer Schriften, die in eine herkömmliche LaTeX-Bearbeitung eingebunden werden sollen. Nutzungshinweise werden in 2.4 vorgestellt. `./graphics` und `./psnfss` stellen die Ergänzungspakete zur Nutzung von PostScript-Schriften und -Grafiken für eine LaTeX-Bearbeitung bereit. Deren Anwendung wird in Kapitel 5 näher vorgestellt. `./amslatex`, `./babel` und `./tools` werden noch in diesem Kapitel vorgestellt. Im weiteren Verlauf dieses Abschnitts wird nur der Inhalt der sechs Unterverzeichnisse aufgelistet und deren Installation beschrieben.

Das Unterverzeichnis `./cyrillic` gehört seit Dezember 1998 zu den LaTeX-Standardergänzungen. Bis dahin gab es dort das Verzeichnis `./mfnfss` zur Erstellung von Ergänzungspaketen zur Nutzung von TeX-Zusatzschriften, das seitdem entfällt. Die Erstellung von Ergänzungen für solche Zusatzschriften kann mit den Hinweisen aus 2.3 bei Bedarf manuell erfolgen.

1.2.4 Das Unterverzeichnis ./tools

Die Ergänzungen aus diesem Unterverzeichnis setzen lediglich die Installation des LaTeX-Standardpakets aus dem Eingangsverzeichnis `./latex/base` voraus. Sonstige Zusatzprodukte wie weitere Zeichensatzfiles oder bestimmte DVI-Treiber werden nicht benötigt. Dieses Unterverzeichnis stellt die LaTeX-Standardergänzungen im engeren Sinne bereit. Sie sollten bei allen LaTeX-Anwendern eingerichtet werden.

Das Originalquellenverzeichnis `./tools` enthält die dokumentierten Makrofiles

```
afterpage.dtx  enumerate.dtx    layout.dtx     tabularx.dtx
array.dtx      fileerr.dtx      longtable.dtx  theorem.dtx
bm.dtx         fontsapl.dtx     multicol.dtx   variofer.dtx
calc.dtx       fntright.dtx     rawfonts.dtx   verbatim.dtx
dcolumn.dtx    hhline.dtx       showkeys.dtx   xr.dtx
delarray.dtx   indentfirst.dtx  somedefs.dtx   xspace.dtx
```

sowie sein Inhaltsverzeichnis unter dem File `00Contents`, ein `readme.txt`-Textfile und das Installationsfile `tools.ins`. Soweit einige der vorstehenden Filegrundnamen aus mehr als acht Buchstaben bestehen, werden sie unter DOS auf die ersten acht Buchstaben gekürzt, z. B. auf `indentfi.dtx` für `indentfirst.dtx`.

Mit dem Bearbeitungsaufruf

```
latex tools.ins
```

aus dem Quellenverzeichnis `./tools` heraus entstehen dort für alle dokumentierten Makrofiles bis auf `fileerr.dtx` die Ergänzungspakete mit den gleichen Grundnamen und dem Anhang `.sty`. Für das Makropaket `theorem.dtx` entstehen neben `theorem.sty` noch `thb.sty`, `thc.sty`, `thcb.sty`, `thm.sty`, `thmb.sty` und `thp.sty`. Letztere sind keine eigenständigen Ergänzungspakete, sondern Optionsrealisierungen für `theroem.sty`. Für die Makropakete `fontsmpl.dtx` und `verbatim.dtx` entsteht neben `fontsmpl.sty` und `verbatim.sty` noch das interaktive Programm `fontsmpl.tex` sowie das Testfile `verbtest.tex`.

Für das Makropaket `fileerr.dtx` entstehen die sechs kleinen TeX-Files `e.tex`, `h.tex`, `q.tex`, `r.tex`, `s.tex` und `x.tex`. Sie werden eingelesen, wenn auf die Fehlermeldung

```
! LaTeX Error: File `name.anh' not found.
. . . . . .
Enter file name:
```

versucht wird, mit einer der Fehlerreaktionen `E`, `H`, `Q`, `R`, `S` oder `X` zu antworten. Die Eingabe wird an dieser Stelle von TeX als Filegrundname und nicht als Fehlerreaktion interpretiert. Mit den vorstehenden TeX-Hilfsfiles existieren Files mit diesen Namen und bewirken die gleiche Programmreaktion wie dies für sonstige Fehler mit der entsprechenden Anwenderreaktion erreicht wird.

Die erzeugten `.sty`- und `.tex`-Files sind abschließend in das Verzeichnis zu verschieben, unter dem TeX seine Makrofiles erwartet. Nach dem TDS-Vorschlag wäre dies `.../texmf/tex/latex/tools` (s. S. 6). In der Installationsdatei `tools.ins` wird `\usedir{tex/latex/tools}` (s. 1.2.2 auf S. 7) gesetzt, womit die Ergänzungspakete aus `tools` in diesem Verzeichnis unterhalb des mit dem `docstrip`-Konfigurationsfile

über `\BaseDirectory{`*TDS_eing_verz*`}` vorgegebenen Eingangsverzeichnis gemäß der Darstellung auf S. 7 festgelegten TDS-Eingangs abgelegt werden.

Eigenständige Treiberfiles zur Erstellung der Dokumention entstehen beim Installationsaufruf für das `./tools`-Verzeichnis nicht. Alle dokumentierten Makrofiles aus diesem Verzeichnis enthalten aber implizit den Treiberkode, so dass die wohlformatierte Dokumentation mit der direkten LaTeX-Bearbeitung gemäß S. 8 leicht zu erstellen ist. Die Nutzungsbeschreibung der Ergänzungspakete aus dem `./tools`-Verzeichnis erfolgt in den Abschnitten 1.3.1 bis 1.3.13.

1.2.5 Das Unterverzeichnis ./babel

Das Babel-System enthält die LaTeX-Werkzeuge für die Textbearbeitung nahezu aller europäischen und weiterer auf der lateinischen Schrift aufbauenden außereuropäischen Sprachen. Die derzeit (Ende 2001) aktuelle Version hat die Versionsnummer 3.7h und das Versionsdatum vom 1. März 2001. Der Hauptautor des Babel-Systems ist JOHANNES BRAAMS, Niederlande, unter Beteiligung weiterer in der internen Dokumentation genannten Sprach- und TeX-Experten. Die Nutzungsbeschreibung des installierten Babelsystems wird in 1.4 nachgereicht.

Das Originalverzeichnis `./babel` für die Quellenfiles enthält eine Vielzahl von dokumentierten Makrofiles, die ich hier nicht aufliste. Zusätzlich stellt es einige `.txt`-Files mit Inhalts-, Erläuterungs- und Installationshinweisen bereit, die mit dem Texteditor eingesehen werden können. Die Gesamtheit aller Bestandteile aus dem `./babel`-Installationspaket werden dort in `manifest.txt` aufgelistet.

Das Programmpaket `./babel` sollte eingerichtet werden, wenn neben den Sprachen Deutsch, Englisch und Französisch, die bereits mit dem Ergänzungspaket `german.sty` bzw. `french.sty` abgedeckt werden, weitere Sprachen beim LaTeX-Betreiber zur Anwendung kommen. Das Paket ist für multilinguale Anforderungen von Bedeutung, wenn diese die Standardeigenschaften von `german.sty` übersteigen bzw. mit dessen Erweiterungen gemäß den Hinweisen aus D.2.3 in Band 1 dieser Buchserie nicht zu erfüllen sind.

Das Quellenpaket enthält das Installationsfile `babel.ins`, dessen LaTeX-Bearbeitung die dokumentierten Makrofiles aufbereitet. Mit dem LaTeX-Bearbeitungsaufruf

```
latex babel.ins
```

entstehen zum einen das eigentliche Ergänzungspaket `babel.sty` und die Definitionsfiles `babel.def`, `plain.def` und `switch.def` sowie das Konfigurationsfile `hyphen.cfg`. Zum anderen entstehen eine Vielzahl Sprachdefinitionsfiles mit dem englischen Grundnamen für die entsprechende Sprache und dem Anhang `.ldf` (language definition file). Die hier gleichzeitig entstehenden Sprachoptionsfiles mit dem gleichen Grundnamen und dem Anhang `.sty` dienen nur für den Babel-Kompatibilitätsmodus mit LaTeX 2.09, wobei mit diesen `.sty`-Files die gleichnamigen `.ldf`-Files eingelesen werden. Zusätzlich entstehen noch die beiden Treiberfiles `babel.drv` und `user.drv` mit den MakeIndex-Formatierungsfiles `bbind.ist` und `bbglo.ist` zur Erstellung der Dokumentation.

Die Definitionsfiles mit den Anhängen `.def` bzw. `.ldf` werden vom Anwender nicht selbst angesprochen, sondern mit den Babel-Sprachoptionen intern aktiviert (s. u. zur Babelaktivierung mit `usepackage`). Die erstellten `.def`-, `.ldf`- und `.sty`-Files sind, wie üblich, abschließend in das Verzeichnis zu verschieben, unter dem TeX seine Makropakete erwartet. Nach dem TDS-Vorschlag könnte dies z. B. `.../texmf/tex/generic/babel` sein.

Das Installationsfile `babel.ins` enthält mit `\usedir{tex/generic/babel}` bereits eine passende Teilpfadvorgabe, die zusammen mit dem im `docstrip.cfg`-Konfigurationsfile mit dem Befehlspaar `\BaseDirectory{`*TDS_eing_pfad*`}` `\UseTDS` (s. 1.2.2 auf S. 7) vorgegebenen Pfadeingang das beim Anwender eingerichtete TDS-System für die Fileablage und -suche automatisch berücksichtigt.

Bei der LaTeX-Bearbeitung von `babel.ins` entstehen auch die Unterstützungswerkzeuge zur Bearbeitung griechischer Texte, und zwar das Dekodierfile `lgrenc.def`, die Stilfiles `athnum.sty` und `grmath.sty` sowie die Zeichensatz-Definitionsfiles `lgrcmr.fd`, `lgrcmro.fd`, `lgrcmss.fd`, `lgrcmtt.fd`, `lgrlcmss.fd` und `lgrlcmtt.fd`.

Die zugehörigen griechischen Zeichensätze findet man auf dem DANTE-Fileserver unter `/tex-archive/fonts/greek/babel-package` als `greek_fonts.zip`. Nach dem Entpacken stehen die Metrikfiles und die METAFONT-Quellenfiles zur Verfügung. Aus diesen sollte der Druckertreiber bei Bedarf die erforderlichen Druckerzeichensätze dynamisch erstellen, wie das z. B. für `dvips` oder die Druckertreiber aus dem emTeX-Paket der Fall ist.

Das aufbereitete Babel-System enthält mit *gus*`.ldf` und *gus*`.sty` mit `russianb` bzw. `ukraineb` für *gus* die Babel-Anpassungsmakros zur Bearbeitung russischer bzw. ukrainischer Texte. Für deren LaTeX-Bearbeitung und Druckausgabe werden passende Kodierungsattribute, Zeichensatzdefinitionsfiles (`.fd`-Files) und METAFONT-Quellenfiles für kyrillische Schriften benötigt, die nicht Bestandteil des Babelsystems und erst recht nicht des LaTeX-Grundsystems sind. Geeignete Makropakete für diese Aufgabe werden seit Dezember 1998 mit dem Verzeichnis `cyrillic` aus den LaTeX-Standardergänzungen bereitgestellt, über dessen Inhalt und Installation der nächste Unterabschnitt 1.2.6 unterrichtet.

Die Nutzung des Babel-Pakets verlangt die Bereitstellung eines oder mehrerer LaTeX-Formatfiles mit der Einbindung geeigneter Trennmusterfiles für die in Betracht kommenden Sprachen. Trennmusterfiles sind dem Babel-Paket nicht beigefügt. Solche findet man auf dem DANTE-Fileserver für eine Vielzahl von Sprachen unter `/tex-archive/languages`. Die Einbindung mehrerer Trennmusterfiles erfolgt am einfachsten durch die Bereitstellung eines Konfigurationsfiles `hyphen.cfg` im lokalen Verzeichnis, aus dem der INITEX-Aufruf erfolgt (s. [5a, Anh. F.2.1]).

Die prinzipielle Möglichkeit von TeX 3.x, bis zu 256 verschiedene Trennmusterfiles in ein Formatfile einzubinden, scheitert in der Praxis an der mit `trie_size` und `trie_op_size` vorgegebenen Größe für die zugehörigen Pufferspeicher (s. [5a, Anh. F.1.3]). Wurde TeX aus den `.web`-Quellenfiles mittels eigener Kompilierung eingerichtet, so können die Original-Größenvorgaben abgeändert werden, was jedoch erhebliche Kenntnisse über die Interna der TeX-Quellenstruktur verlangt.

Für ein fertiges TeX unter UNIX sind die internen Vorgaben ausreichend groß, um vier bis fünf verschiedene Trennmusterfiles in ein Formatfile einzubinden. EmTeX lässt die Festlegung diverser Pufferspeichergrößen mit Optionsangaben zur Laufzeit von INITEX zu, z. B. mit der Optionsangabe `-mt`$xxxx$. Für weitere Einzelheiten muss auf die Dokumentation von emTeX verwiesen werden. Für ein `web2c`-TeX, wie z. B. unter LINUX, kann ebenfalls eine Laufzeitanpassung mittels des zugehörigen Konfigurationsfiles `texmf.cnf` vorgenommen werden.

Stößt die Vergrößerung von `trie_size` und `trie_op_size` auf unüberwindbare Schwierigkeiten, z. B. weil nur ausführbare TeX-Programme beim Anwender existieren, die keine Laufzeitanpassungen erlauben, dann kann für TeX-Versionen wie unter UNIX ein Kompromiss angestrebt werden. Für diesen könnte man die Trennmusterfiles in Bearbeitungsgruppen aufteilen, z. B. in eine Basissprache wie Deutsch und jeweils drei bis vier Fremdsprachen,

die zu einer gemeinsamen Arbeitsgruppe zusammengefasst werden. Für jede dieser Sprachgruppen ist dann jeweils ein eigenes Formatfile mit jeweils einem eigenen `hypgen.cfg`-Konfigurationsfile zu erstellen, dessen Standardformatname `latex.fmt` anschließend geeignet umzubennen ist.

Mit diesem Kompromiss sollten selbst wissenschaftliche multilinguale Anforderungen weitgehend abgedeckt werden, da auch hierbei das gleichzeitige Auftreten von mehr als fünf verschiedenen Sprachen vermutlich ein Ausnahmefall sein wird. Die Erstellung mehrerer sprachgruppenspezifischer LaTeX-Formatfiles sollte mit den Hinweisen aus [5a, Anh. F.2.1] keine Schwierigkeiten bereiten.

Das Babel-System kann mit einem der angeregten LaTeX-Standardformatfiles mit einer zugehörigen Befehlsdatei (s. [5a, Anh. F.2.1]) durch Einbindung des Babel-Hauptstilfiles mit

`\usepackage[`*sprache_1*`,`*sprache_2*`,...,`*sprache_n*`]{babel}`

für die in der Optionsliste angegebenen Sprachen *sprache_x* aktiviert werden. Die hierfür verwendeten LaTeX-Formatfiles enthalten dabei keine internen Babel-Makrostrukturen. Es ist aber möglich, Makrostrukturen aus dem Babel-Kern mit in ein LaTeX-Formatfile einzubinden, wofür am Schluss dieses Unterabschnitts Hinweise gegeben werden. Der hiermit verbundene Zeitgewinn beim Einlesen dieser Babel-Makrostrukturen ist inzwischen jedoch vernachlässigbar, so dass für die Erstellung solcher Babel-Formatfiles kein wirklicher Bedarf besteht.

Das Babel-Paket enthält für seinen internen Makroaufbau und dessen Nutzung eine umfangreiche beigefügte Dokumentation, die mit den beiden Treiberfiles `babel.drv` und `user.drv` wohlformatiert erstellt werden kann, und zwar in seiner vollständigen Form mit der Darstellung aller seiner Makrodefinitionen (`babel.drv`) bzw. in einer verkürzten Form (`user.drv`) mit der Darstellung seiner Nutzungsbeschreibung und den zusätzlichen Nutzungshinweisen für die unterstützten Sprachen. Die Aufrufe

```
latex babel.drv  und  latex user.drv
```

sind zunächst zweimal auszuführen. Hiernach kann das erzeugte File `user.dvi` über den Druckertreiber als 33-seitige Dokumentation ausgedruckt werden. Die LaTeX-Bearbeitung von `babel.drv` erzeugt zusätzlich die MakeIndex-Formatfiles `babel.idx` und `babel.glo`. Diese sind mit den Formatfiles des Babel-Pakets `bbind.ist` und `bblo.ist` mit MakeIndex zu bearbeiten:

```
makeindx -s bbind.ist babel
makeindx -s bbglo.ist -o babel.gls babel.glo
```

Hiernach kann mit einer abschließenden LaTeX-Berbeitung von `babel.drv` das erzeugte File `babel.dvi` über den Druckertreiber als ca. 250-seitige Gesamtdokumentation ausgedruckt werden. Die Gesamtdokumentation ist für solche Anwender von Nutzen, die weitere Sprachanpassungsfiles für das Babel-System entwickeln wollen. Die Kurzbeschreibung aus `user.drv` sollte sich dagegen jeder Babel-Anwender erstellen. Sie enthält zusätzliche Informationen zu der Nutzungsbeschreibung des Babel-Pakets im Abschnitt 1.4 dieses Buches.

Einbindung von Babel-Strukturen in ein LaTeX-Formatfile

Bei der INITEX-Bearbeitung des LaTeX-Hauptmakrofiles `latex.ltx` trifft man gegen Ende dieses Files auf den Versuch, ein File mit dem Namen `hyphen.cfg` einzulesen. Existiert

ein solches File, so wird es eingelesen und seine Strukturen werden Bestandteil des LaTeX-Formatfiles. Üblicherweise enthält ein solches `hyphen.cfg`-File nur Sprachdefinitions- und Lesebefehle für die zugehörigen Trennmusterfiles.

Dem installierten Babel-Paket ist ein Konfigurationsfile `hyphen.cfg` beigefügt, das deutlich umfangreicher ist. Es enthält zunächst einige Makrodefinitionen aus dem Babel-Paket zusammen mit dem Lesebefehl für das dortige Makropaket `plain.def`. Beide enthalten eine Vielzahl von Makrostrukturen aus dem Babel-Kern, die damit Bestandteil des entstehenden Formatfiles werden. Zusätzlich enthält das obige `hyphen.cfg`-File einen Lesebefehl für das File `language.dat`, das seinerseits eigene Sprachdefinitionsbefehle und zugehörige Trennmusterfiles miteinander verknüpft. Das beigefügte File `languages.dat` ist eine Musterdatei, die vom Anwender nach seinen Bedürfnissen zu modifizieren ist. Mit den Angaben

```
american ushyph.tex
=USenglish
dutch    nehyph1.tex
french   frhyph.tex  french.exc
german   dehypht.tex
```

für `language.dat` wird der Sprachname 'american' mit dem Trennmusterfile `ushyph.tex` verknüpft, wobei für diesen Sprachnamen auch 'USenglish' gewählt werden kann. Weiterhin werden die Sprachnamen 'dutch', 'french' und 'german' mit den Trennmusterfiles `nehyph1.tex`, `frhyph.tex` und `dehypht.tex` verknüpft, wobei das französische Trennmusterfile zusätzlich durch das Ausnahmeverzeichnis `french.exc` ergänzt wird. Für die vorstehenden Sprachnamen werden intern die Sprachschalter `\l@american`, `\l@USenglish`, `\l@dutch`, `\l@french` sowie `\l@german` eingerichtet, denen die Sprachnummern 0, 1, 2 bzw. 3 entsprechen, da die beiden Sprachschalter `\l@american` und `l@USenglish` einander gleichgesetzt sind.

Bei der INITEX-Bearbeitung von `latex.ltx` entsteht mit diesem `hyphen.cfg` damit ein Formatfile, das einerseits eine Vielzahl von Makrostrukturen aus dem Babel-Paket wie andererseits die Verknüpfung von Sprachdefinitionsbefehlen mit zugehörigen Trennmusterfiles enthält. Mit dieser Verknüpfung von Babel-Kernstrukturen zusammen mit den herkömmlichen Aufgaben eines `hyphen.cfg`-Files sollten die ersteren für eine Babel-Aktivierung schneller bereitgestellt werden. Der damit erzielte Zeitgewinn kann bei der Leistungsfähigkeit moderner Pentium-Prozessoren jedoch vernachlässigt werden.

1.2.6 Das Programmverzeichnis ./cyrillic

Das Programmverzeichnis `./cyrillic` gehört seit Dezember 1998 zu den Standardergänzungen einer LaTeX-Installation. Es stellt die Zeichensatz-Makrodateien bereit, die zur Nutzung kyrillischer Schriften, z. B. zusammen mit dem Babel-Paket, bei Bearbeitung russischer oder ukrainischer Texte benötigt werden.

Das Installationsverzeichnis `./cyrillic` besteht aus den dokumentierten Makrofiles `cyinpenc.dtx`, `cyoutenc.dtx`, `lcy.dtx` und `ot2.dtx`, den Zeichensatz-Definitionsmasterfiles `lcycmlh.fdd`, `ot2cmams.fdd`, `ot2cmlh.fdd` und `t2lnfnt.fdd` sowie dem Installationsfile `cyrlatex.ins`. Zusätzlich enthält es noch die mit dem Editor einsehbaren Textfiles `00readme.txt`, `changes.txt` und `manifest.txt`. Als zusätzliche Information wird bereits mit der LaTeX-Grundinstallation das File `cyrguide.tex` angeboten, dessen LaTeX-Bearbeitung eine wohlformatierte Dokumentation bereitstellt.

Die LaTeX-Bearbeitung des Installationsfiles `cyrlatex.ins`, also der Aufruf

```
latex cyrlatex.ins
```

erzeugt die Nutzungswerkzeuge des `./cyrillic`-Verzeichnisses. Hiermit entstehen zum einen die Definitionsfiles *code*`enc.def` für das Kodierungsattribut mit dem Ergänzungspaket `fontenc.sty`, und zwar mit den Kodierungskennamen `t2a`, `t2b`, `t2c`, `x2`, `lcy` und `ot2` für *code*, also die Definitionsfiles `t2aenc.def`, ..., `ot2enc.def`.

Zum anderen entstehen die Definitionsfiles *eing_code*`.def` für den Eingabekode mit dem Ergänzungspaket `inpenc.sty`. Als Grundnamen *eing_code* für die entstehenden Eingabekodierfiles treten insgesamt auf

> `cp855`, `cp866`, `cp866cv`, `cp866mav`, `cp866nav`, `cp866tat`, `cp1251`, `ctt`, `dbk`, `iso88595`, `isoir111`, `koi8-r`, `koi8-ru`, `koi9-u`, `maccyr`, `macukr`, `mik`, `mls`, `mlk`, `mos`, `ncc`, `pt154`, `pt254`

denen die Eingabe-Kodierfiles `cp855.def`, ..., `pt254.def` entsprechen. Zur Bedeutung und Einstellwirkung dieser Definitionsfiles für die Zeichensatzkodierung bezüglich der Ein- und Ausgabe wird auf die Nutzungsbeschreibung der `cyrillic`-Werkzeuge in 2.4.4 sowie auf `cyrguide.tex`-aus der LaTeX-Grundinstallation verwiesen.

Neben diesen Definitionsfile entsteht mit dem obigen Installationsaufruf das Ergänzungspaket `lct.sty`, das seinerseits das Ergänzungspaket `fontenc.sty` zusammen mit der Optionsangabe `LCY` sowie das bei der Installation entstehende Makropaket `lcydefs.tex` jeweils implizit einliest.

Neben diesen aufgelisteten Kodierungs-Definitionsfiles und Makropaketen entstehen aus den Zeichensatz-Definitionsmasterfiles (`.fdd`-Files) eine große Zahl von Zeichensatz-Definitionsfiles (`.fd`-Files) mit der Namenssyntax *attr_code_tex_zs*`fd`. Hierin steht *attr_code* für einen der bereits oben aufgelisteten Namen für das Kodierungsattribut `t2a`, `t2b`, `t2c`, `x2`, `lcy` bzw. `ot2` und *tex_zs* für einen der bekannten TeX-Zeichensatz-Kennungsnamen wie `cmr`, `cmss`, `cmtt` u. a. Für das Kodierungsattribut `ot2` tritt für *tex_zs* zusätzlich noch `wncyr` und `wncyss` auf, was an die Namen der kyrillischen Zeichensätze der $\mathcal{AMS}$ erinnert. Für die Zeichensatz-Definitionsfiles treten damit Filenamen wie `t2acmr.fd`, `t2ccmss.fd`, `lcycmtt.fd`, `ot2cmfib.fd` und Ähnliche auf.

Diese mit dem Installationsaufruf entstandenen Makropakete sind abschließend in das Verzeichnis zu verschieben, unter denen LaTeX seine Makropakete erwartet. Nach dem TDS-Vorschlag wäre dies `.../texmf/tex/latex/cyrillic`. Das Installationsfile `cyrlatex.ins` enthält wie alle Installationsfiles aus dem `./required`-Verzeichnis mit `\usedir{tex/latex/cyrillic}` bereits eine passende Teilpfadvorgabe, die zusammen mit dem im `docstrip.cfg`-Konfigurationsfile mit dem Befehlspaar `\BaseDirectory{`*TDS_eing_pfad*`}` und `\UseTDS` (s. 1.2.2 auf S. 7) vorgegebenen Pfadeingang das beim Anwender eingerichtete TDS-System für die Fileablage und -suche automatisch berücksichtigt.

Die METAFONT-Quellenfiles für kyrillische Zeichensätze sind nicht Bestandteil der LaTeX-Standardergänzungen aus dem Verzeichnis `./cyrillic`. Man findet auf den TeX-Fileservern unter `/tex-archive/fonts/cyrillic/lh` die von der russischen TeX-Anwendervereinigung CyrTUG enwickelten kyrillischen METAFONT-Qellenfiles, die mit kyrillischen Nutzungs- und Definitionsmakros aus dem `./cyrillic`-Verzeichnis harmonisch zusammenarbeiten.

1.2.7 Das Programmverzeichnis ./amslatex

Die Installationshinweise für das `./amslatex`-Verzeichnis können kürzer ausfallen als diejenigen zum vorangegangenen `./babel`-Verzeichnis. Das `./amslatex`-Originalverzeichnis enthält zum einen die beiden Erläuterungsfiles `00readme.txt` und `install.txt` sowie die zwei Unterverzeichnisse `./classes`, und `./math`.

Das erste Unterverzeichnis `./classes` enthält die dokumentierten Makrofiles `amsclass.dtx`, `amsdtx.dtx` und `upref.dtx`, die Bibliografie-Stil- und -Definitionsfiles `amsalpha.bst`, `amsplain.bst` und `mrabbrev.bib`, das Installationsfile `ams-c1.ins`, die Dokumentations- und Testfiles `amsthdoc.tex`, `thmtest.tex` und `instr-l.tex` sowie die Textfiles `00readme.txt`, `diffs-c.txt`, `install.txt`, `manifest.txt` und `amsclass.faq`.

Das andere Unterverzeichnis `./math` enthält die dokumentierten Makrofiles `amsbsy.dtx`, `amscd.dtx`, `amsgen.dtx`, `amsmath.dtx`, `amsopn.dtx`, `amstext.dtx` und `amsxtra.dtx`, das Installationsfile `ams-m1.ins`, das $\mathcal{AMS}$-TeX-Ergänzungspaket `amstex.sty`, das Klassenfile `amsldoc.cls`, die Dokumentations- und Testfiles `amsldoc.tex`, `subeqn.tex`, `technote.tex` und `testmath.tex` sowie die Textfiles `amslatex.bug`, `amslatex.faq`, `diffs-m.txt`, `install.txt`, `manifest.txt` und `00readme.txt`

Die LaTeX-Bearbeitung von `amsdoc.tex` erzeugt eine wohlformatierte Dokumentation mit einem ausführlichen Inhalts- und Stichwortverzeichnis. Die LaTeX-Bearbeitung der Installationsfiles aus beiden Verzeichnissen `ams-c1.ins` und `ams-m1.ins` führt zur Aufbereitung der zugehörigen dokumentierten Makrofiles und damit zur Erzeugung der einzurichtenden Makropakete aus dem `./amslatex`-Verzeichnis.

Mit der LaTeX-Bearbeitung von `ams-c1.ins`, also dem Aufruf '`latex ams-c1.ins`' aus dem Verzeichnis `./classes` heraus, entstehen dort die beiden Ergänzungspakete `amsthm.sty` und `upref.sty` sowie die Klassenfiles `amsart.cls`, `amsbook.cls` und `amsproc.cls`, die in Analogie zu den LaTeX-Standardklassen `article`, `book` und `proc` stehen, sowie das Klassenfile `amsdtx.cls`. Mit dem LaTeX-Bearbeitungsaufruf '`latex ams-m1.ins`' aus dem Verzeichnis `./math` heraus, entstehen dort die Ergänzungspakete `amsbsy.sty`, `amscd.sty`, `amsgen.sty`, `amsintx.sty`, `amsmath.sty`, `amsopn.sty`, `amstext.sty`, `amstex.sty` und `amsxtra.sty`. Die erzeugten Klassenfiles und Ergänzungspakete aus beiden Installationsfiles sind abschließend in ein Verzeichnis zu verschieben, unter dem TeX seine Makrofiles erwartet. Nach dem TDS-Vorschlag wäre dies `.../texmf/tex/latex/amslatex` (s. S. 6).

Die beiden Klassenfiles `amsldoc.cls` und `amsdtx.cls` werden vom Anwender kaum selbst angesprochen. Sie werden intern aktiviert, wenn die beigefügte Dokumentation `amsldoc.tex` bzw. die dokumentierten Makrofiles mit ihrem impliziten Treiberkode mit LaTeX zur Erstellung einer wohlformatierten Dokumentation bearbeitet werden.

Die Nutzung des $\mathcal{AMS}$-LaTeX-Pakets erfordert die Einrichtung weiterer mathematischer Zeichensätze, die von der $\mathcal{AMS}$ (American Mathematical Society) entwickelt und der Allgemeinheit zur Verfügung gestellt wurden. Auf dem DANTE-Fileserver findet man hierzu das Eingangsverzeichnis `/tex-archive/fonts/amsfonts`, das in die weiteren Unterverzeichnisse `./latex`, `./tfm` und `./sources` untergliedert ist. Das letzte Unterverzeichnis enthält eine weitere Untergliederungsebene, von der zur Nutzung des `amslatex`-Pakets nur die METAFONT-Quellenfiles aus den Unterverzeichnissen `./euler`, `.extracm` und `./symbols` benötigt werden.

Das darüber liegende Parallelverzeichnis `./latex` enthält weitere `.sty`- sowie `.fd`-Files, die in das gleiche Zielverzeichnis wie die `.cls`- und `.sty`-Files aus dem `amslatex`-Paket zu verschieben sind, also entsprechend dem TDS-Vorschlag nach `.../texmf/tex/latex/required/amslatex`.

Die Nutzungsbeschreibung der $\mathcal{AMS}$-LaTeX-Werkzeuge erfolgt mit vielen Beispielen in 1.5, wozu die mit LaTeX aufbereitete englischsprachige Dokumentation aus `amsldoc.tex` und `instr-l.tex` des `amslatex`-Pakets zur Begleitung dienen kann. Eine Nutzungsbeschreibung von zusätzlichen $\mathcal{AMS}$-Zeichensätzen mit einer herkömmlichen Standard-LaTeX-Bearbeitung erfolgt in 2.2.1.

1.2.8 Das Unterverzeichnis ./graphics

Das Originalquellenverzeichnis `./graphics` besteht aus den dokumentierten Makrofiles `color.dtx`, `drivers.dtx`, `epsfig.dtx`, `graphics.dtx`, `graphicx.dtx`, `keyval.dtx`, `lscape.dtx`, `pstcol.dtx` und `trig.dtx`, dem Installationsfile `graphics.ins`, den Definitionsfiles `dvipdfm.def`, `pdftex.def`, `textures.def` und `vtex.def`, den Textfiles `changes.txt` und `00readme.txt` sowie dem Dokumentationsfile `grfguide.tex`, zusammen mit seinem aufbereiteten PostScript-File `grfguide.ps`.

Die LaTeX-Bearbeitung des Installationsfiles `graphics.ins` erzeugt die Ergänzungspakete

```
color.sty      epsfig.sty   graphics.sty   graphicx.sty
keyval.sty     lscape.sty   pstcol.sty     trig.sty
```

sowie die Treiber-Definitionsfiles (Stand Januar 2000)

```
dvipdf.def     dvips.def      dvipsnam.def   dvipsone.def
dviwin.def     emtex.def      pctex32.def    pctexhp.def
pctexps.def    pctexwin.def   tcidvi.def     truetex.def
```

wozu sich bei zukünftigen Versionen evtl. weitere gesellen.

Abschließend sind die erstellten `.def`- und `.sty`-Files wie bei allen LaTeX-Ergänzungen in ein Verzeichnis zu verschieben, unter dem TeX seine Makropakete erwartet. Nach dem TDS-Vorschlag ist dies `.../texmf/tex/latex/packages/graphics` und für emTeX vermutlich `\emtex\texinput\latex2e`, falls das letzte Unterverzeichnis dort nicht weiter untergliedert ist.

Mit dem `graphics`-Paket werden Werkzeuge bereitgestellt, mit denen spezielle Grafikeigenschaften verschiedener Drucker unter einheitlichen Befehlsnamen aus LaTeX heraus angesprochen und genutzt werden. Die Aufgabe der LaTeX-Bearbeitung ist es dann, die treiberspezifischen `\special`-Befehle zu erstellen und im `.dvi`-File abzulegen, ohne dass sich der Anwnender mit diesen `\special`-Befehlen selbst befassen muss.

Die Vorstellung und Nutzungsbeschreibung dieser Grafikwerkzeuge erfolgt in den Abschnitten 5.3.2 bis 5.3.8. Es empfiehlt sich, hierzu die dem `graphics`-Paket beigefügte Dokumentation `grfguide.tex` mit LaTeX aufzubereiten und den Ausdruck zur Begleitung heranzuziehen. Steht beim Anwender ein PostScript-Drucker zur Verfügung, dann kann das beigefügte PostScript-File `grfguide.ps` auch direkt ausgedruckt werden.

1.2.9 Das Unterverzeichnis ./psnfss

Dieses Verzeichnis enthält die Ausgangsquellen zur Nutzung von PostScript-Schriften für eine LaTeX-Bearbeitung. Dies setzt die Verfügbarkeit eines PostScript-Druckers voraus. Die Aufbereitung der dokumentierten Makrofiles bereitet mit den beigefügten Installationsfiles keine Schwierigkeiten. Da die Nutzung von PostScript-Druckern die Einrichtung von weiteren Programmen, z. B. eines geeigneten DVI-Treibers sowie evtl. die Erstellung oder Beschaffung von `.tfm`- und `.vf`-Metrikfiles für die PostScript-Schriften, voraussetzen, erfolgt die Beschreibung für die Installation und einer möglichen Auswahl dieser Werkzeuge gemeinsam in 5.1.

1.2.10 Weitere LaTeX-Ergänzungen

Wie bereits in der Einleitung dieses Gesamtabschnitts vermerkt, gliedert sich das LaTeX-Gesamtpaket auf den offiziellen TeX-Fileservern in die drei Eingangsverzeichnisse `./base`, `./required` und `./contrib`. Die in den vorangegangenen Unterabschnitten vorgestellten Bestandteile aus `./required` stellen im weiteren Sinne die LaTeX-Standardergänzungen bereit, wobei die Werkzeuge aus dem dortigen Unterverzeichnis `./tools` als die Standardergänzungen im engeren Sinne bezeichnet werden können.

Das dritte Eingangsverzeichnis `./contrib` stellt eine große Zahl von weiteren LaTeX-Ergänzungen bereit. Es ist zunächst zweifach untergliedert in `./supported` und `./other`, was keine Bedeutungswürdigung darstellt, sondern nur darauf verweist, dass die Werkzeuge aus `./supported` einer Wartung und damit ggf. einer Verbesserung durch ihre Autoren unterliegen.

Der Unterverzeichniszweig `./supported` besteht zunächst ausschließlich aus weiteren Unterverzeichnissen, deren aktuelle Anzahl (Stand Dezember 1996) 142 beträgt. Viele dieser Unterverzeichnisse sind weiter untergliedert und enthalten oft ganze Gruppen von LaTeX-Ergänzungen. Einige dieser Unterverzeichnisse tragen selbstbeschreibende Namen, die die Zweckbestimmung erahnen lassen. Andere tragen Autorennamen, die ebenfalls Aufgabenziele erkennen lassen, wenn die Programmschwerpunkte des Autors bekannt sind. Ansonsten müssen die Unterverzeichnisse genauer eingesehen werden. Die meisen von ihnen enthalten ein `README`-File, das häufig eine Zweckbestimmung wiedergibt.

Angesichts der Vielzahl und des Wandels und Wachstums dieser Verzeichnisse unterbleibt hier eine Inhalts- und Aufgabenauflistung. Das gilt auch für den anderen Verzeichniszweig `./other`, der in seiner ersten Ebene, von einer Ausnahme abgesehen (`multfoot.sty`), ebenfalls nur weitere Unterverzeichnisse enthält, und zwar derzeit 50.

LaTeX-Anwender mit speziellen Formatierungsanforderungen sollten diese Verzeichnisse durchmustern. In den meisten Fällen werden sie ein geeignetes Werkzeug finden. Bleibt die Suche erfolglos oder erinnert sich der Anwender an Ergänzungswerkzeuge einer früheren LaTeX 2.09-Installation, so sollte auch das Verzeichnis `./macros/latex209/contrib` durchmustert werden.

Die dortigen Ergänzungen wurden für die Nutzung mit LaTeX 2.09 entwickelt. Viele von ihnen können aber problemlos auch mit LaTeX 2_ε über dessen `\usepackage`-Aktivierungsbefehl genutzt werden. Das gilt z. B. für die bei vielen LaTeX-Anwendern genutzten Erweiterungen der `picture`-Umgebung mit dem `epic.sty`- bzw. `eepic.sty`-Paket.

1.3 Vorstellung der LaTeX-Standardergänzungen

Gemeint sind hiermit die LaTeX-Standardergänzungen im engeren Sinne, die mit dem Quellenverzeichnis `./latex/required/tools` bereitgestellt werden und mit den Installationshinweisen aus 1.2.4 aufbereitet und eingerichtet wurden. Bei dieser Installation entstanden die Ergänzungspakete

`afterpage.sty`	`enumerate.sty`	`longtable.sty`	`theorem.sty`
`array.sty`	`fontsmpl.sty`	`multicol.sty`	`varioref.sty`
`bm.sty`	`ftnright.sty`	`rawfonts.sty`	`verbatim.sty`
`calc.sty`	`hhline.sty`	`showkeys.sty`	`xr.sty`
`dcolumn.sty`	`indentfirst.sty`	`somedefs.sty`	`xspace.sty`
`delarray.sty`	`layout.sty`	`tabularx.sty`	

die alle mit dem Vorspannbefehl `\usepackage` in die LaTeX-Bearbeitung eingebunden werden können. Das Ergänzungspaket `theorem.sty` greift seinerseits auf die Zusatzfiles `thb.sty`, `thc.sty`, `thcb.sty`, `thm.sty`, `thmb.sty` und `thp.sty` zurück, ohne dass hierzu eigene Anwendervorgaben vorzunehmen sind.

Zusätzlich entstanden bei der Installation noch die sechs kleinen TeX-Files `e.tex`, `h.tex`, `q.tex`, `r.tex`, `s.tex` und `x.tex` sowie das Testfile `verbtest.tex` und das interaktive Programm `fontsmpl.tex`. All diese Werkzeuge werden im Verlauf dieses Abschnitts vorgestellt, wobei empfohlen wird, die gemäß 1.2.2 aufbereitete Dokumentation aus den zugehörigen `.dtx`-Files zur Begleitung zu nutzen.

1.3.1 Verbesserte und erweiterte Tabellenumgebungen

Die Ergänzungspakete `array.sty` von FRANK MITTELBACH sowie `dcolumn.sty`, `delarray.sty`, `hhline.sty` und `tabularx.sty` von DAVID P. CARLISLE, Universität Manchester, stellen Verbesserungen und Ergänzungen zur LaTeX-`tabular`-Umgebung dar, die teilweise aufeinander aufbauen und deshalb in diesem Unterabschnitt gemeinsam vorgestellt werden.

1.3.1.1 Das Ergänzungspaket array.sty

FRANK MITTELBACH stellt mit `array.sty` ein Ergänzungspaket mit erweiterten Tabellenstrukturen gegenüber dem LaTeX-Original bereit. Es gestattet, Tabellenstrukturen wie bisher mit

`\begin{array}[`*pos*`]{`*sp_form*`}`	*Zeilen*	`\end{array}`	bzw.
`{begin{tabular}[`*pos*`]{`*sp_form*`}`	*Zeilen*	`\end{tabular}`	oder
`\begin{tabular&}{`*breite*`}[`*pos*`]{`*sp_form*`}`	*zeilen*	`\end{tabular*}`	

zu erzeugen. Wirkung und Syntax der Parameter *pos* und *breite* entsprechen vollständig denen der Standard-Tabellenumgebungen (s. [5a, Abschn. 4.8.1]). Der Spaltenformatierungsparameter *sp_form* wurde erweitert. Die zulässigen Einträge und deren Wirkungen demonstriert die Tabelle auf der nächsten Seite.

Die Standarderklärungen für das Tabellenlayout mit `\tabcolsep`, `\arraycolsep`, `\arrayrulewidth`, `\doublerulesep` und `\arraystretch` (s. [5a, Abschn. 4.8.2] wurden ebenfalls um einige zusätzliche Erklärungen erweitert, die weiter unten vorgestellt werden.

Unveränderte Spaltenformatierungsparameter	
`l c r`	linksbündiger (`l`), zentrierter (`c`) oder rechtsbündiger (`r`) Spalteneintrag wie beim `tabular`-Original
`p{`*breite*`}`	Definiert eine Spalte für mehrzeiligen Eintrag, der entsprechend der eingestellten *breite* umbrochen wird. Mehrzeiliger Text wird in Bezug auf die Nachbarspalten auf die *erste* Zeile ausgerichtet. Entspricht der Wirkung von `\parbox[t]{`*breite*`}`.
`@{`*erkl*`}`	Entfernt den Standardzwischenraum und ersetzt ihn durch den Inhalt von *erkl*. Beispiel: `r@{.}l` formatiert die benachbarten Spalten für die Eingabe `1 & 5` als 1.5. (Originalbefehl)

Zusätzliche oder geänderte Spaltenformatierungsparameter	
`m{`*breite*`}`	Wie `p`, mehrzeiliger Text wird mit der *vertikalen Mitte* auf die Eintragungen der Nachbarspalten ausgerichtet. Entspricht der Wirkung von `\parbox{`*breite*`}`.
`b{`*breite*`}`	Wie `p`, mehrzeiliger Text wird mit der *letzten* Zeile auf die Eintragungen der Nachbarspalten ausgerichtet. Entspricht der Wirkung von `\parbox[b]{`*breite*`}`.
`>{`*erkl*`}`	Kann direkt *vor* eine `l`-, `c`-, `r`-, `p{..}`-, `m{..}`- oder `b{..}`-Formatierungsangabe gesetzt werden und fügt den Inhalt von *erkl* vor jedem Spalteneintrag ein. Die Erklärung *erkl* steht hierbei für gewöhnlichen Text, Befehlsfolgen oder eine Mischung aus beiden. Beispiel: `>{\bfseries}c` formatiert die entsprechende Spalte mit einem horizontal zentrierten Eintrag mit dem Schriftattribut `\bfseries`.
`<{`*erkl*`}`	Kann direkt *nach* einer `l`-, `c`-, `r`-, `p{..}`-, `m{..}`- oder `b{..}`-Formatierungsangabe gesetzt werden und fügt den Inhalt von *erkl* nach jedem Spalteneintrag ein. Beispiel: `l<{!!!}` linksbündiger Spalteneintrag, an den jeweils !!! angehängt wird.
`|`	Fügt eine vertikale Linie ein. Der standardmäßige Spaltenzwischenraum wird hierbei im Gegensatz zum Original um die Breite der Linie vergrößert.
`||`	Fügt eine vertikale Doppellinie ein und vergrößert den Spaltenzwischenraum um die Breite der Doppellinie.
`!{`*erkl*`}`	Fügt den Inhalt von *erkl* zwischen benachbarten Spalten bzw. vor der ersten oder nach der letzten Spalte hinzu, *ohne* den zusätzlichen Spaltenzwischenraum zu entfernen.

Die vorstehende Tabelle wurde demgemäß mit

`\begin{tabular}{|>{\ttfamily}c||m{105mm}|}` *Tabellentext* `\end{tabular}`

erzeugt. Für die linke Spalte wird damit als Schriftattribut `\ttfamily` gewählt. Um die kursiven Textteile innerhalb dieser Spalte zu erzeugen, wurde lokal auf `\emph` umgeschaltet: `m\{\emph{breite}\}` ⇒ `m{`*breite*`}`[1].

Die rechte Spalte ist als `m{105mm}` erklärt worden. Mehrzeiliger Text wird gegenüber der linken Spalte auf die vertikale Mitte zentriert. Das gilt natürlich auch für die Angaben

[1] Genau genommen wurde `m\symbol{123}\emph{breite}\symbol{125}` eingegeben, da `\{` bzw. `\}` auch innerhalb des `\ttfamily`-Zeichensatzes die geschweiften Klammern dem Standard-Roman-Zeichensatz entnehmen.

`p{`*breite*`}` bzw. `b{`*breite*`}`. In der vorstehenden Tabelle wurde die vertikale Verschiebung durch zugefügte `\raisebox`-Befehle bewirkt, um die Wirkung der `p`- bzw. `b`-Formatierung zu demonstrieren.

Der Nutzen des Formatierungsparameters `>{`*erkl*`}` wird mit dem in der Tabelle angegebenen Beispiel sofort offenkundig. Dagegen erscheint das Beispiel für das `<{`*erkl*`}`-Format etwas gekünstelt. Tatsächlich kann man mit der Kombination beider Parameter erstaunlich geschickte Formatierungsmöglichkeiten erschließen.

Soll z. B. für einzelne Spalten einer `tabular`-Umgebung in den mathematischen Bearbeitungsmodus umgeschaltet werden, in dem die Textformeln horizontal zentriert erscheinen sollen, so kann das für diese Spalte mit der Vorgabe `>{$}c<{$}` im Spaltenformatierungsfeld erreicht werden. Bei einer `array`-Umgebung bewirkt die gleiche Vorgabe die Umschaltung in den Text-LR-Modus, so dass in dieser Spalte normaler Text erscheint. Das anfängliche `$`-Zeichen hebt hierbei das implizite `$`-Zeichen der `array`-Umgebung auf. Am Ende der Spalte bewirkt das dort zugefügte `$`-Zeichen die Rückschaltung in den mathematischen Modus der `array`-Umgebung. Weitere Beispiele folgen am Ende dieses Unterabschnitts.

Bei den Standardtabellenumgebungen wird die Strichstärke von vertikalen Linien beim Spaltenabstand nicht berücksichtigt. Dies kann, insbesondere bei stärkeren Linien, dazu führen, dass das letzte bzw. das erste Zeichen benachbarter Spalten zu eng zum vertikalen Trennstrich erscheint und in Extremfällen den Trennstrich berührt oder sogar in diesen hineinragt. Diese Schwäche wird mit `array.sty` beseitigt, allerdings mit der Folge, dass Tabellen mit vertikalen Trennstrichen nun breiter ausfallen als äquivalente Tabellen ohne vertikale Trennstriche.

Die Nichtberücksichtigung der Strichstärke bei den Standardtabellenumgebungen führt bei umrandeten Tabellen zu einer weiteren Unzulänglichkeit, die bei stärkeren Linien sichtbar wird, wie das nachfolgende Beispiel zeigt.

```
\setlength{\arrayrulewidth}{5pt}
\begin{tabular}{|l|}
   \hline AAA\\ \hline
\end{tabular}
```

erzeugt das nebenstehende obere Ergebnis, bei dem die horizontalen Linien in der Mitte der vertikalen Linien enden.

`array.sty` beseitigt diese Schwäche. Das gleiche Beispiel erscheint damit als

Horizontale Linien oberhalb und unterhalb von Tabellen können mit Standard-LaTeX zu Fehlpositionierungen führen, und zwar dann, wenn der `tabular`-Umgebung horizontaler Text vorangeht und/oder nachfolgt. Ein solcher umgebender Text wird standardmäßig auf die vertikale Tabellenmitte ausgerichtet. Mit dem Positionierungsparameter *pos* als `t` oder `b` kann der umgebende Text auf die erste bzw. letzte Tabellenzeile ausgerichtet werden.

Die Ausrichtung der Tabelle zum umgebenden Text erfolgt korrekt, wenn der ersten und/oder der letzten Tabellenzeile kein `\hline`-Befehl vorausgeht bzw. nachfolgt. Anderenfalls kann es zu Fehlpositionierungen kommen, wie das nachfolgende Beispiel zeigt.

```
Tabellen
\begin{tabular}[t]{l}
  ohne \verb|\hline|-Befehle\\
  erscheinen korrekt\\
  zum umgebenden\\
\end{tabular} Text.
```

Tabellen ohne `\hline`-Befehle Text.
erscheinen korrekt
zum umgebenden

```
Tabellen
\begin{tabular}[t]{|l|}\hline
  mit einleitenden oder\\
  abschlie"senden \verb|\hline|-\\
  Befehlen erscheinen\\
  dagegen\\ \hline
\end{tabular} fehlpositioniert.
```

Tabellen

mit einleitenden oder abschließenden \hline-Befehlen erscheinen dagegen

fehlpositioniert.

Der gleiche Eingabetext führt auch mit `array.sty` zur entsprechenden Fehlpositionierung. Hier kann jedoch mit `\firsthline` bzw. `\lasthline` an Stelle des einleitenden bzw. abschließenden `\hline`-Befehls Abhilfe geschaffen werden:

```
Tabellen
\begin{tabular}[t]{|l|}
  \firsthline mit Umrandungen\\
  erscheinen korrekt\\
  zum umschlie"senden\\
  Text, wenn sie mit\\
  \verb|\firsthline| und \\
  \verb|\lasthline|\\
  \lasthline
\end{tabular} umschlossen werden.
```

Tabellen

mit Umrandungen erscheinen korrekt zum umschließenden Text, wenn sie mit \firsthline und \lasthline

umschlossen werden.

Änderung des Tabellenstils: Mit `\setlength`-Zuweisungen für die Längenerklärungen

`\tabcolsep` ⇒ halbe Breite des Spaltenzwischenraums für `tabular`-Umgebungen

`\arraycolsep` ⇒ halbe Breite des Spaltenzwischenraums für `array`-Umgebungen

`\arrayrulewidth` ⇒ Strichstärke von horizontalen und vertikalen Linien in einer Tabelle

`\doublerulesep` :⇒ Abstand von Doppellinien

kann der Anwender eigene Einstellwerte gegebenüber den Standardvorgaben vorgeben. Diese Längenerklärungen sind außerhalb der Tabellenumgebungen vorzunehmen und gelten für alle nachfolgenden Tabellenumgebungen, bis sie durch neue entsprechende Längenerklärungen abgelöst werden bzw. bis zum Ende einer evtl. umschließenden Umgebung.

Diese Erklärungen gelten sowohl für Standard-LaTeX als auch für `array.sty`. Für dieses Ergänzungspaket gibt es zusätzlich die Längenerklärung:

`\extrarowheight` ⇒ fügt den hiermit eingestellten Zusatzabstand der normalen Zeilenhöhe einer Tabelle hinzu. Ein Zusatzabstand von mindestens 1 pt sollte bei allen Tabellen mit horizontalen Trennlinien vorgenommen werden, da sonst die Oberlängen der Zeichen der ersten Folgezeile zu dicht an die darüber liegende Trennlinie heranreichen. Für die Erläuterungstabelle der Spaltenformatierungsparameter auf S. 19 war `\setlength{\extrrowheight}{1pt}` wirksam.

Schließlich gibt es für `array.sty` noch den Definitionsbefehl

`\newcolumntype{`*typ_kennung*`}{`*defintion*`}` z. B. in der Form
`\newcolumntype{`x`}{>{`*anf_erkl*`}c<{`*end_erkl*`}}`

Mit der Definition der zweiten Zeile steht für anschließende Tabellenumgebungen als Spaltenkennungstyp das gewählte Zeichen x zur Verfügung, das entsprechend seiner Definition die zugehörige Spalte gemäß `>{`*anf_erkl*`}c<{`*end_erkl*`}` horizontal zentriert, wobei jedem Eintrag für diese Spalte die Vorgaben *anf_erkl* und *end_erkl* voran- bzw. nachgestellt werden.

Die Einrichtung eigener Spaltentypkennungen ist stets dann zu empfehlen, wenn mehrere Tabellen mit gleichartigen komplexeren Spaltenformaten erstellt werden sollen. Dies erspart Schreibarbeit und vermindert Formatierungsfehler gegenüber expliziten Formatierungsangaben im Formatierungsfeld bei jeder einzelnen Tabellenumgebung. So könnten für Tabellen mit gemischten Text- und Formeleinträgen in Ergänzung zu den Standardspaltenkennungen `c`, `l` und `r` mit

```
\newcolumntype{C}{>{$}c<{$}}
\newcolumntype{L}{>{$}l<{$}}
\newcolumntype{R}{>{$}r<{$}}
```

als zusätzliche Spaltenkennungen `C`, `L` und `R` eingerichtet werden, mit denen bei `tabular`-Umgebungen entsprechende Spalten zur Ausgabe von Textformeln bzw. bei `array`-Umgebungen zur Ausgabe von normalen Texten gewählt werden.

Der Definitionsbefehl `\newcolumntype` erlaubt zusätzlich noch ein erstes optionales Argument in Form einer Zahlenangabe von 1 bis 9, mit der dem einzurichtenden neuen Spaltentyp ein bis neun freie Parameter `#1` bis `#9` zugewiesen werden, deren aktuelle Werte dann beim jeweiligen Aufruf des Spaltentyps als dessen Argumente übergeben werden, wie dies aus der LaTeX-Befehlsdefinition `\newcommand` her bekannt ist.

Mit dem Befehlsaufruf `\showcols` im Eingabefile erfolgt eine Bildschirmprotokollierung aller zum Zeitpunkt dieses Befehls mit `\newcolumntype` eingerichteten Spaltentypkennungung mit gleichzeitiger Ablage im Protokollfile.

Die Dokumentation für `array.sty` enthält einige weitere Beispiele zur Einrichtung von eigenen Spaltentypen mit zugehörigen Hilfsmakros, die ich hier wiedergebe, da sie das Realisierungsprinzip für das Ergänzungspaket `dcolumn.sty` widerspiegeln. Zunächst wird mit

```
\newcolumntype{d}{>{\centerdots}c<{\endcenterdots}}
```

der neue Spaltentyp `d` vorgeschlagen, dessen interne Markos mit

```
{\catcode‘\.=\active\gdef.{\egroup\setbox2=\hbox\bgroup}}
\def\centerdots{\catcode‘\.=\active\setbox0=\hbox\bgroup}
\def\endcenterdots{\egroup\ifvoid2 \setbox2=\hbox{0}\fi
   \ifdim \wd0>\wd2 \setbox2=\hbox to\wd0{\unhbox0\hfill}
   \else  \setbox0=\hbox to\wd2{\hfill\unhbox0}\fi
   \catcode‘\.=12 \box0.\box2}
```

einzurichten sind. Die hier mit dem Spaltentyp `d` eingerichteten Spalten sind zur Aufnahme von Dezimalzahlen geeignet, bei denen die übereinander stehenden Dezimalzahlen nach dem Dezimalpunkt ausgerichtet werden, wobei der Dezimalpunkt genau in der Spaltenmitte steht.

Die horizontale Zentrierung des Dezimalpunkts wirkt ungefällig für Spalten, bei denen die Zahl der Stellen vor dem Dezimalpunkt sich deutlich von denen nach dem Dezimalpunkt unterscheidet. Für solche Spalten wird deshalb `d` mit

```
\newcolumntype{d}[1]{>{\rightdots{#1}}r<{\endrightdots}}
```

vorgeschlagen. Hiermit wird der Spaltenkennungstyp `d` mit einem Argument eingerichtet, mit dem die maximale Anzahl von Stellen nach dem Dezimalpunkt vorgegeben werden kann. Die Realisierungsmakros `\rightdots` und `\endrightdots` sind mit

```
\def\coldot{.}
{\catcode‘\.=\active
   \gdef.{$\egroup\setbox2=\hbox to \dimen0 \bgroup$\coldot}}
```

```
\def\rightdots#1{\setbox0=\hbox{$1$}\dimen0=#1\wd0
   \setbox0=\hbox{$\coldot$}\advance\dimen0 by \wd0
   \setbox2=\hbox to \dimen0 {}%
   \setbox0=\hbox\bgroup\mathcode`\.="8000 $}
\def\endrightdots{$\hfil\egroup\box0\box2}
```

zu definieren. Nun kann mit der Angabe `d{`n`}` im Spaltenformatierungsfeld einer Tabellenumgebung für die entsprechende Spalte eine Ausrichtung des Dezimalpunkts vorgenommen werden, wobei die Dezimalpunkte so ausgerichtet werden, als hätten alle Zahle genau n Stellen nach dem Dezimalpunkt.

Der Einrichtungsbefehl `\newcolumntype` kann auch dazu genutzt werden, um unter einer Einzeichenkennung mehrere Spalten einzurichten. So wird z. B. mit

```
\newcolumntype{X}{lcr}
\begin{tabular}{X} ... \end{tabular}
```

eine dreispaltige Tabelle eingerichtet, deren erste Spalte linksbündig, zweite Spalte zentriert und dritte Spalte rechtsbündig ausgerichtet ist.

1.3.1.2 Das Ergänzungspaket dcolumn.sty

Das Paket `dcolumn.sty` von DAVID CARLISLE dient zur Ausrichtung von Spalten mit Dezimalzahlen nach dem Dezimaltrennzeichen. Es stellt zunächst den Spaltenformatierungsparameter `D` mit der Syntax

`D{`*eing_zeichen*`}{`*ausg_zeichen*`}{`*dez_stellen*`}`

bereit. Hierin steht *eing_zeichen* für das Eingabezeichen, das im Text verwendet wird, und *ausg_zeichen* für das Ausgabezeichen, das bei der Textausgabe verwendet wird und nach dem die Ausrichtung innerhalb der mit `D` gekennzeichneten Spalte erfolgt. Die maximale Anzahl von Stellen nach dem Dezimalzeichen kann mit *dez_stellen* vorgegeben werden, wobei hier die Angabe einer negativen Zahl bei der Texteingabe eine beliebige Anzahl von Stellen nach dem Dezimalzeichen erlaubt. Mit `D{.}{\cdot}{3}` ist das Ausrichtungszeichen bei der Eingabe der normale Punkt, der bei der Ausgabe als hochgestellter Punkt erscheint, wobei bis zu drei Stellen nach dem Punkt bei der Eingabe erlaubt sind: `345.12` erscheint damit als $345{\cdot}12$.

Die Formatierungsangabe kann mit `D` im Formatierungsfeld der `tabular`- oder `array`-Umgebung in der beschriebenen Form erfolgen. Bei häufiger Verwendung gleicher oder ähnlicher Spaltenformatierung empfiehlt sich die Bereitstellung spezieller Formatierungszeichen mit `\newcolumntype`. Mit

```
\newcolumntype{d}[1]{D{.}{\cdot}{#1}}
\newcolumntype{.}{D{.}{.}{3}}
\newcolumntype{,}{D{,}{,}{2}}
```

stehen ‘`d`, ‘`.`’ und ‘`,`’ als neue Spaltenformatierungszeichen zur Verfügung. Mit ihnen wurde

$1{\cdot}2$	$1{\cdot}2$	1.2	1,2
$1{\cdot}23$	$1{\cdot}23$	2.4	100,00
$12345{\cdot}678$	$12345{\cdot}67$	10.000	1,09
$-0{\cdot}01$	$-{\cdot}01$	$-.999$	$-$,99
100	100	25	25

durch `\begin{tabular}{|d{-1}|d{2}|.|,|}` erzeugt, wobei die Spalteneingaben mit `1.2`, ... `1,2` usw. erfolgten. Bei der ersten Spalte fällt auf, dass rechts ein größerer freier Platz als bei den anderen Spalten erscheint. Der Grund liegt in den unterschiedlichen Zuordnungen für negative bzw. positive Stellenangaben. Bei einer negativen Angabe realisiert `dcolumn.sty` die Spalte mit `>{\centerdots}c{\endcenterdots}` aus `array.sty`. Hiermit erscheint der Dezimalpunkt in der Spaltenmitte. Treten in einer solchen Spalte vor dem Dezimalpunkt mehr Stellen auf als nach ihm, so wird der links vom Dezimalpunkt benötigte Platz auch für den Platz rechts vom Dezimalpunkt bereitgestellt und umgekehrt.

Bei einer positiven Angabe für die Stellen nach dem Dezimalzeichen erfolgt die interne Realisierung mit `>{\rightdots}r<{\endrightdots}` aus `array.sty`. Hiermit wird rechts vom Dezimalzeichen der angeforderte Maximalplatz eingerichtet, unabhängig davon, wie viele Stellen vor dem Dezimalzeichen verwendet werden.

Ab Version v1.03 von `dcolumn.sty` kann das dritte Argument des Spaltenformatierungsparameters D *dez_stellen* auch als Dezimalzahl erfolgen, z. B. als `D{`e`}{`a`}{`$v.n$`}`. Die Ausrichtung des Dezimalpunkts erfolgt dann so, als hätten alle Zahlen dieser Spalte v Stellen *vor* und n Stellen *nach* dem Dezimalpunkt.

Die getrennte Vorgabe für Eingabe- und Ausgabezeichen kann dazu genutzt werden, einheitliche Datensätze für umfangreiche Zahlentabellen, z. B. die Ergebnisse eines Rechenprogramms für deutsch- und englischsprachige Veröffentlichungen, aufzubereiten. Angenommen, die Ergebnisse eines Rechenprogramms sind in einem aufbereiteten Datenfile `math.dat` abgelegt, wobei alle Dezimalzahlen entsprechend der üblichen Konvention von Rechenprogrammen mit einem Dezimalpunkt abgelegt wurden. Mit der Definition eines Spaltenparameters, z. B. `\newcolumntype{.}{.}{`$v.n$`}`, werden die Spalten der entsprechenden Tabelle nach dem Dezimalpunkt ausgerichtet, der gleichzeitig als Ein- und Ausgabezeichen zur Anwendung kommt. Der Datensatz kann dabei mit `\input{math.dat}` innerhalb der zugehörigen Tabellenumgebung eingelesen werden.

Wird nunmehr `\newcolumntype{.}{,}{`$v.n$`}` gewählt, so erscheint in der Ausgabetabelle als Dezimaltrennzeichen das Komma, ohne dass an den Eingabedaten irgend etwas zu ändern wäre. In dieser Weise können die Ergebnisse von Rechenprogrammen ganz einfach für deutschsprachige Veröffentlichungen genutzt werden, ohne dass mühevolle Editierarbeiten an den Programmergebnissen erforderlich werden.

1.3.1.3 Das Ergänzungspaket delarray.sty

Diese Ergänzung ist vorrangig für die `array`-Umgebung und damit für den Formelsatz zur Erzeugung von Feldstrukturen [5a, 5.4.3] gedacht. Dabei ist die Syntax der bisherigen `array`-Umgebung verändert worden. Sie lautet nun

`\begin{array}[`*pos*`]` *lkl* `{`*sp_form*`}` *rkl* *Zeilen* `\end{array}`

mit der gleichen Wirkung, als hätte man bei der `array`-Standardumgebung geschrieben

`\left` *lkl* `\begin{array}[`*pos*`]{`*sp_form*`}` *Zeilen* `\end{array} \right` *rkl*

mit *lkl* und *rkl* für ein linkes und ein rechtes mathematisches Klammersymbol. Die Eingabe

```
$ \begin{array}({cc}) a&b\\ c&d \end{array} $
```

erzeugt die Formel $\left(\begin{array}{cc} a & b \\ c & d \end{array} \right)$ und nach `\newcolumntype{L}{>{$}l<{$}}` erhält man mit

```
\[ f(x) = \begin{array}\{{lL}.
    1 & wenn $x=0$\\ \sin(x)/x & andernfalls \end{array} \]
```

die abgesetzte Formel

$$f(x) = \begin{cases} 1 & \text{wenn } x = 0 \\ \sin(x)/x & \text{andernfalls} \end{cases}$$

Als Folge der `\left`-`\right`-Paarbedingung [5a, 5.4.1] muss der linken öffnenden Klammer `\{` eine *unsichtbare* rechte schließende Klammer zugeordnet werden, was beim vorstehenden Spaltenfeld mit dem nachfolgenden '`.`' geschieht. Der dort eingeführte Spaltenpositionierungsparameter `L` beendet mit dem anfänglichen `$`-Zeichen den mathematischen Bearbeitungsmodus innerhalb der `array`-Umgebung. Mit dem nachfolgenden `$`-Zeichen wird anschließend wieder in den mathematischen Modus zurückgeschaltet.

Der optionale Parameter *pos* gestattet mit den Werten '`t`' und '`b`' eine vertikale Ausrichtung nebeneinander stehender `array`-Umgebungen bezüglich der obersten oder untersten Feldzeile [5a, 4.8.1]. Dies gilt auch für das Ergänzungspaket `tabularx.sty`:

```
\begin{array}[t] ({c}) 1\\2\\3 \end{array}
\begin{array}({c})      1\\2\\3 \end{array}
\begin{array}[b] ({c}) 1\\2\\3 \end{array}
```

$$\begin{array}{t} \left(\begin{array}{c} 1\\2\\3 \end{array}\right) \end{array}
\begin{array}{c} \left(\begin{array}{c} 1\\2\\3 \end{array}\right) \end{array}
\begin{array}{b} \left(\begin{array}{c} 1\\2\\3 \end{array}\right) \end{array}$$

Bei Verwendung des vertikalen Positionierungsparameters unterscheidet sich das Ergebnis gegenüber der Standardumgebung, die mit `\left(` und `\right)` umschlossen wird:

```
\left(\begin{array}[t]{cc} 1\\2\\3 \end{array}\right)
\left(\begin{array}{cc}    1\\2\\3 \end{array}\right)
\left(\begin{array}[b]{cc} 1\\2\\3 \end{array}\right)
```

$$\left(\begin{array}{t} \\ 1\\2\\3 \end{array}\right)
\left(\begin{array}{c} 1\\2\\3 \end{array}\right)
\left(\begin{array}{b} 1\\2\\3 \\ \end{array}\right)$$

Alle hier vorgestellten Beispiele zur Verwendung von `delarray` wurden mit freundlicher Genehmigung durch den Autor, DAVID CARLISLE, aus der Originaldokumentation `delarray.drv` übernommen.

1.3.1.4 Das Ergänzungspaket hhline.sty

Das Paket gestattet flexiblere Einfach- und Doppelumrandungen innerhalb von Tabellen und Feldern, als dies mit den Standardeinstellungen `|`, `||`, `\vline`, `\hline` und `\cline` [5a, 4.8.1] möglich ist. Das Paket stellt für die `array`- und `tabular`-Umgebung zusätzlich den Befehl `\hhline{`*lsp_form*`}` bereit, in der *lsp_form* für die Spalten- und Zwischenspaltengestaltung der damit erzeugten Linienstruktur steht. Zum Verständnis möge man annehmen, dass das Spaltenformatierungsfeld die spaltenweise Gestaltung der herkömmlichen Zeileneinträge bestimmt. Mit `\begin{tabular}{||cc||c|c||}` wird eine Tabelle erzeugt, deren einzelne Zeilen von vertikalen Doppellinien umschlossen werden. Zwischen Spalte 2 und 3 wird eine weitere vertikale Doppellinie und zwischen Spalte 3 und 4 eine einfache vertikale Linie eingefügt. Der Eintrag `a & b & c & d \\` erscheint in der Tabelle damit als ‖ a b ‖ c | d ‖.

Die mit `\hhline{`*lsp_form*`}` erzeugte Linienstruktur möge man sich nun als eigenständigen Zeileneintrag vorstellen, und zwar zusätzlich zu den herkömmlichen Textzeilen der Tabelle. Ein solcher zusätzlicher Zeileneintrag muss ebenfalls eine spaltenweise Kennung erhalten,

die unabhängig von den Vorgaben aus dem Spaltenformatierungsfeld für die herkömmlichen Texteinträge ist. Als Kennungszeichen stehen zur Verfügung:

`=` erzeugt eine horizontale Doppellinie von der Breite der zugehörigen Spalte.

`-` erzeugt eine horizontale Einfachlinie von der Breite der zugehörigen Spalte.

`~` unterdrückt die horizontale Linienstruktur für die zugehörige Spalte.

`|` erzeugt eine vertikale Linie zwischen zwei Spalten von der Höhe der horizontalen Doppellinie.

`:` erzeugt eine *unsichtbare* vertikale Linie zwischen zwei Spalten von der Höhe einer horizontalen Doppellinie, an die sich die nächste horizontale Linie anschließt oder an der die vorangehende horizontale Linie endet.

`#` erzeugt ein horizontales Doppelliniensegment von der Breite einer vertikalen Doppellinie.

`t` erzeugt die obere Linie eines horizontalen Doppelliniensegments von der Breite einer vertikalen Doppellinie.

`b` wie `t`, jedoch für die untere Linie des horizontalen Doppelliniensegments

`*` Mit `*{`n`}{`*tsp*`}` wird das angegebene Teilspaltenformat *tsp* n-mal wiederholt: `*{3}{|==|}` steht also für `|==||==||==|`.

Die vorstehende Beschreibung klingt komplizierter, als es sich nach kurzer Nutzung erweist. Die Angaben '`=`', '`-`' und '`~`' dienen zur Erzeugung horizontaler Linien innerhalb einer Spalte, die Angaben '`|`' und '`:`' zur Erzeugung vertikaler Strukturen zwischen zwei Spalten und '`#`', '`t`' sowie '`b`' zur Erzeugung kurzer horizontaler Linien für die Zwischenspaltenstrukturen. Die Kombination `|t:` und `|b:` erzeugt die kleine linke obere bzw. linke untere Ecke für eine evtl. anschließende horizontale Doppellinie; mit `:t|` und `:b|` wird die rechte obere bzw. rechte untere Ecke zum Abschluss der vorangehenden horizontalen Doppellinie erzeugt. Mit diesen Erläuterungen sollte das Verstehen des abschließenden Beispiels aus `hhline.dtx` nicht schwerfallen:

```
\begin{tabular}{||cc||c|c||}
  \hhline{|t:==:t:==:t|} a&b&c&d\\
  \hhline{|:==:|~|~||}   1&2&3&4\\
  \hhline{#==#~|=#}      i&j&k&l\\
  \hhline{||--||--||}    w&x&y&z\\
  \hhline{|b:==:b:==:b|}
\end{tabular}
```

a	b	c	d
1	2	3	4
i	j	k	l
w	x	y	z

Zur Übung möge der Leser die einzelnen `\hhline`-Befehle nacheinander aktivieren und das Formatierungsfeld bei der `tabular`-Umgebung zunächst nur als `{cccc}` wählen. Die Wirkung der einzelnen `\hhline`-Befehle wird dann rasch klar. So erzeugt die erste Tabelleneingabezeile mit dem verkürzten Formatierungsfeld a b c d. Wird nun das Formatierungsfeld auf `{||cc||c|c||}` erweitert, dann werden der Textzeile die vertikalen Doppel- und Einfachstriche zugefügt, die sich mit der vorangehenden *Linien*zeile korrekt ergänzen: a b c d.

1.3.1.5 Das Ergänzungspaket tabularx.sty

Die LaTeX-Standardumgebung `tabular` erlaubt in der *-Form die Vorgabe für die Gesamtbreite der Tabelle in der Form [5a, 4.8.1]

`\begin{tabular*}{`*breite*`}[`*pos*`]{`*sp_form*`}` *Zeilen* `\end{tabular*}`

Die Spaltenbreiten werden dabei in der gewohnten Form ermittelt bzw. für absatzartige Spalteneinträge mit `p{`*sp_breite*`}` vorgegeben. Die *-Form der `tabular`-Umgebung beginnt im Spaltenformatierungsfeld gewöhnlich mit der Angabe `@{\extracolsep\fill}`, womit ein beliebig dehnbarer Spaltenzwischenraum gewählt wird, mit dem die vorgegebene Tabellenbreite aufgefüllt wird.

Die `tabularx`-Umgebung hat dieselbe Syntax wie die *-Form der `tabular`-Umgebung (natürlich mit dem eigenen Umgebungsnamen `tabularx`). Zusätzlich stellt sie den Spaltenformatierungsparameter '`X`' bereit. Dieser ist für absatzartige Spalteneinträge vorgesehen, wobei die Spaltenbreite so ermittelt wird, dass die vorgegebene Tabellenbreite ohne zusätzlichen Spaltenzwischenraum erreicht wird. Mit `\begin{tabularx}{120mm}{lXcXr}` wird eine 120 mm breite Tabelle erzeugt, deren erste, dritte und fünfte Spalte für einzeilige Spalteneinträge mit linksbündiger, zentrierter bzw. rechtsbündiger Anordnung vorgesehen ist. Die Breiten dieser Spalten werden durch die jeweils breitesten Einträge bestimmt. Die Spalten zwei und vier werden durch absatzartige Spalteneinträge geprägt, wobei die Breite so gewählt wird, dass die Tabellenbreite von 120 mm gerade ausgefüllt wird. Der Zeilenumbruch erfolgt entsprechend der errechneten Breite automatisch und beidseitig bündig.

Ein bündiger Zeilenumbruch ist bei schmalen Spalten oft schwer zu verwirklichen. Der Formatierungsparameter '`X`' wird deshalb gern mit zusätzlichen Vorgaben verknüpft. Mit '`>{\small}X`' wird für die absatzartigen Spalteneinträge als Schriftgröße `\small` gewählt. Mit `>{\raggedright}X`' erfolgt der Umbruch in der nachfolgenden Spalte nur linksbündig. Diese Wahl für die Spaltenformatierung führt dann aber beim Auftreten von `\\` zur Beendigung der gesamten Tabellenzeile zu einem Bearbeitungsfehler, da der LaTeX-Befehl `\raggedright` (ebenso wie `\raggedleft` und `\centering`) den Zeilenendbefehl `\\` umdefiniert. Dies kann mit der Zusatzangabe `\arraybackslash`, also mit der Spaltenvorgabe '`>{\raggedright\arraybackslash}X`' (und entsprechend für `\raggedleft` bzw. `\centering`) korrigiert werden. Werden solche Vorgaben häufiger benötigt, so empfiehlt es sich, diese mit z. B.

```
\newcolumntype{R}{>{\raggedright\arraybackslash}X}
```

als eigene Formatierungsparameter, wie hier für `R`, bereitzustellen. Hiernach kann `R` als Kurzform im Spaltenformatierungsfeld der `tabularx`-Umgebung mit der entsprechenden Wirkung angegeben werden.

Der Formatierungsparameter `X` stellt Spalten mit absatzartigen Einträgen für Tabellen bereit. Dabei wird die jeweils erste Zeile eines Absatzeintrages auf die einzeiligen Einträge der anderen Spalten ausgerichtet. Diese Eigenschaft wird durch die interne Definition von `\tabularxcolumn` bestimmt, die mit `\newcommand{\tabularxcolumn}[1]{p{#1}}` vorgegeben ist. Mit den Definitionsänderungen außerhalb der `tabularx`-Umgebung

```
\renewcommand{\tabularxcolumn}[1]{m{#1}}  bzw.
\renewcommand{\tabularxcolumn}[1]{b{#1}}  oder auch
\renewcommand{\tabularxcolumn}[1]{>{\small}b{#1}}
```

erfolgt die Ausrichtung durch den Parameter X auf die vertikale Mitte oder auf die unterste Zeile der Spaltenabsätze, wobei mit der letzten Form gleichzeitig auf die Schriftgröße `\small` umgeschaltet wird.

Die Breite der mit `X` eingerichteten Spalten sind innerhalb der Tabelle gleich. Die intern errechnete Breite wird in dem Register `\hsize` abgelegt. Mit Einträgen wie

```
...>{\hsize=.6667\hsize}X ... >{\hsize=1.3333\hsize}X ...
```

im Spaltenformatierungsfeld kann man für die verschiedenen X-Spalten unterschiedliche Breiten verlangen; bei diesem Beispiel wird die erste X-Spalte halb so breit wie die zweite X-Spalte gewählt. Bei einer solchen Vorgabe ist darauf zu achten, dass die Summe der Breiten der unterschiedlichen X-Spalten genau $n \times$ `\hsize` ergibt, wenn die Tabelle insgesamt n X-Spalten enthält. Bei unterschiedlichen Breiten für die X-Spalten sollten zudem keine `\multicolumn`-Befehle auftreten, die X-Spalten einschließen.

Mit der Vorspannerklärung `\tracingtabularx` erreicht man, dass bei der Bearbeitung von `tabularx`-Umgebungen die errechneten Breiten als Bildschirmnachrichten ausgegeben und zusätzlich im Protokollfile abgelegt werden. Abschließend sei noch vermerkt, dass `tabularx.sty` und `delarray.sty` ihrerseits das Ergänzungspaket `array.sty` einlesen, das somit zur Nutzung von `tabularx` und `delarray` verfügbar sein muss. Damit stehen alle Erweiterungen aus `array.sty` auch in `tabularx` und `delarray` zur Verfügung.

Trotz der gleichen Syntax für die `tabularx`- und die `tabular*`-Umgebung unterscheiden sich beide strukturell. Die Standardumgebungen `tabular` und `tabular*` können beliebig verschachtelt werden. Bei verschachtelten `tabularx`-Umgebungen ist darauf zu achten, dass die inneren Umgebungen ihrerseits jeweils in ein `{ }`-Paar eingeschlossen werden müssen! Die Verwendung von `\verb`-Befehlen führt innerhalb der `tabularx`-Umgebung nur mit Einschränkungen zu richtigen Ergebnissen. Eingeschlossene Leerzeichen werden teilweise unterdrückt, `%`-Zeichen sind nicht erlaubt, und geschweifte Klammern dürfen nur paarweise auftreten. Die `tabularx`-Umgebung erlaubt in `X`-Spalten die Verwendung von `\footnote`-Befehlen, was in der `tabular*`-Umgebung nicht möglich ist.

1.3.2 Mehrseitige Tabellen mit longtable.sty

Die Erstellung von Tabellen ist mit der `tabular`-Standardumgebung auf Tabellen beschränkt, deren Länge die vorgegebene Seitenhöhe nicht überschreitet. Längere Tabellen müssen auf mehrere `tabular`-Umgebungen aufgeteilt werden. Das verlangt gewöhnlich mehrere Versuche, da die passende Tabellenlänge für die einzelne Seite beim Eingabetext zu Beginn kaum richtig abgeschätzt und erst nach mehreren Probeausdrucken optimal ermittelt werden kann. Dabei tritt eine weitere Unzulänglichkeit auf: Die zusammengehörenden Spalten unterscheiden sich häufig auf den verschiedenen Seiten, da die Spaltenbreite durch den maximalen Eintrag der jeweiligen Teiltabelle bestimmt wird, der auf verschiedenen Seiten sehr unterschiedlich ausfallen kann. Dasselbe Problem tritt auch bei kürzeren Tabellen auf, wenn diese innerhalb der umgebenden Texte auf der Seite umbrochen werden müssen.

Das Ergänzungspaket `longtable.sty` (unter DOS `longtabl.sty`) von D. P. CARLISLE beseitigt diese Schwäche. Der Aufruf erfolgt mit

`\begin{longtable}{`*sp_form*`}` *Zeilen* `\end{longtable}`

genauso wie mit der einfachen `tabular`-Umgebung, wobei im Spaltenformatierungsfeld *sp_form* dieselben Angaben wie bei jener zulässig sind.

Bei mehrseitigen Tabellen besteht häufig die Forderung, der Tabelle eine Überschrift voranzusetzen, die sich in einer evtl. modifizierten Form oberhalb der Folgeseiten mit den Teiltabellen wiederholt. Ebenso sollen der Tabelle auf der Anfangs- und den Folgeseiten oft eine oder mehrere gleichartige Kopfzeile(n) vorangestellt werden. Umgekehrt sollen die einzelnen Teiltabellen am unteren Seitenende häufig mit einem speziellen Tabellenfuß enden, wobei dieser am Tabellenende ggf. nochmals modifiziert werden soll.

Beide Aufgaben werden von der `longtable`-Umgebung unterstützt. Dazu kann die `longtable`-Umgebung mit einem Kopf- und Fußdoppelvorspann eingeleitet werden. Die Syntax dieser Vorspannstrukturen lautet:

anf_vorspann	`\endfirsthead`	*folge_vorspann*	`\endhead`
standard_fuß	`\endfoot`	*abschl_fuß*	`\endlastfoot`

Der Kopfvorspann der ersten Zeile kann in *anf_vorspann* und *folge_vorspann* den Befehl `\caption` enthalten. Dieser sollte dem Anwender von den Gleitumgebungen `table` und `figure` her bekannt sein. Er erzeugt mit `\caption{`*überschrift*`}` oberhalb der Tabelle eine Überschrift der Form „Tabelle *n*: *überschrift*“ mit dem als *überschrift* übergebenen Text für die Überschrift und der laufenden Tabellennummer *n*.

Beide Vorspannteile *anf_vorspann* und *folge_vorspann* können weiterhin einen Tabellenkopf enthalten, der gewöhnlich über die gesamte Tabelle reicht und mit einem oder mehreren `\multicolumn{`*n*`}{c}{`*kopf_zeile*`}`-Befehlen erzeugt wird. Für *n* wird bei dieser Form die Gesamtspaltenanzahl der Tabelle eingesetzt, damit die Kopfzeile über die gesamte Tabellenbreite eingerichtet wird. Der Formatierungsparameter `c` könnte auch als `l` oder `r` gewählt werden, obwohl der zentrierte Text aus *kopf_zeile* der Regelfall sein wird. Eine andere Form der Kopfzeile ist bei Beibehaltung der einzelnen Spalten eine kurze Überschrift für jede einzelne Spalte. Der oder den Kopfzeilen gehen evtl. `\hline`-Befehle voran und/oder folgen ihr. In der `longtable`-Umgebung wird der erste Vorspannteil *anf_vorspann* mit dem Befehl `\endfirsthead` und der darauffolgende zweite Vorspannteil *folge_vorspann* mit `\endhead` abgeschlossen.

Der evtl. `\caption`-Befehl sowie die Kopfzeilen aus dem ersten Vorspannteil *anf_vorspann* erscheinen als Überschrift und Kopfzeilen auf der ersten Tabellenseite. Überschrift und Kopfzeilen der Folgeseiten der Tabelle entstammen dagegen den entsprechenden Angaben aus *folge_vorspann*.

Auf den Kopfdoppelvorspann kann ein Fußdoppelvorspann entsprechend der obigen Syntaxvorstellung folgen. Er besteht aus einer oder mehreren Tabellenzeilen, die mit dem Befehl `\endfoot` abgeschlossen werden. Diese Tabellenzeilen erscheinen am unteren Tabellenende jeder Teiltabelle auf der ersten bis zur vorletzten Tabellenseite. Weitere Tabellenzeilen, die mit `\endlastfoot` abgeschlossen werden, erscheinen schließlich am Ende der Tabelle auf der letzten Tabellenseite.

Fehlt im Kopfvorspann der Anteil *anf_vorspann*, was an dem fehlenden `\endfirsthead`-Befehl zu erkennen ist, dann erscheinen die Angaben aus *folge_vorspann* auch auf der ersten Tabellenseite. Fehlt umgekehrt im Fußvorspann der Anteil *abschl_fuß*, erkennbar an dem dann fehlenden `\endlastfoot`-Befehl, so wird auch die letzte Tabellenseite mit den Angaben aus *standard_fuß* abgeschlossen.

Erst hiernach folgen die laufenden Tabelleneinträge, die vollständig denen der herkömmlichen `tabular`-Umgebung entsprechen. Die `longtable`-Umgebung darf aber beliebig viele Tabellenzeilen enthalten, da der Tabellentext bei Bedarf automatisch in Seiten umbrochen

wird. Bei der LaTeX-Behandlung erscheinen eventuell Bildschirmwarnungen mit dem Hinweis, dass bei einer gewissen Zeilennummer die Spaltenbreite verändert wurde. Dahinter verbirgt sich die Absicht, den erforderlichen Speicherbedarf möglichst gering zu halten. Bei der Bearbeitung der Tabelle werden zunächst standardmäßig bis zu 20 Tabellenzeilen als Bearbeitungseinheit zusammengefasst, die Spaltenbreiten für die Zeilengruppen aus den jeweils längsten Spalteneinträgen ermittelt und die Teiltabelle hiermit erstellt. Die Spaltenbreite für die nächste Gruppe mag von der vorangehenden abweichen, was mit einer Bildschirmwarnung mitgeteilt wird.

Die Zeilenanzahl, die bei der Tabellenkonstruktion als Bearbeitungseinheit zusammengefasst wird, kann vom Anwender mit der Anweisung `\setcounter{LTchunksize}{`n`}` mit einem beliebigen Wert für n verändert werden. Voreingestellt ist $n = 20$. Bei einer geänderten Zuweisung sollte n nicht kleiner als die Spaltenanzahl der Tabelle gewählt werden. Mit BigTeX und ausreichender Hauptspeicherkapazität kann ein deutlich größerer Wert, z. B. 50 oder 100, gewählt werden. Größere Vorgaben beschleunigen die Bearabeitung. Die gruppenweise ermittelten Spaltenbreiten werden im `.aux`-File abgespeichert.

Bei der anschließenden Bearbeitungswiederholung werden die unterschiedlichen Spaltenbreiten aufeinander abgeglichen, wobei es gewöhnlich nach drei, spätestens jedoch nach vier Durchläufen zum endgültigen Abgleich kommt. Damit wird die Tabelle endgültig mit den breitesten Spaltenwerten der Gesamttabelle erstellt, so dass die Breiten der korrespondierenden Spalten auf allen Tabellenseiten übereinstimmen.

Dieser Abgleich erfolgt ab Version 4.0 automatisch mit den Bearbeitungswiederholungen. Dies war bei den vorangegangenen Versionen nicht der Fall. Der endgültige Abgleich verlangte bei den Versionen 3.x die Aktivierung des Vorspannbefehls `\setlongtables`. Bei noch älteren Versionen mussten die Breiten der einzelnen Spalten sogar mit einer Musterzeile vorgegeben werden, die anschließend mit dem Befehl `\kill` wieder zu deaktivieren war. Der Befehl `\setlongtables` sowie die Angabe einer Musterzeile mit dem Deaktivierungsbefehl `\kill` sind aus Kompatibilitätsgründen auch bei der aktuellen Version 4.04 vom 1. 6. 1996 erlaubt. Der Befehl `\setlongtables` ist hier aber ein Leerbefehl, der lediglich eine Fehlermeldung bei Auftreten von `\setlongtables` vermeidet.

Das Ergänzungspaket `longtable.sty` gestattet zur Tabellenerzeugung einige weitere Feinabstimmungen, mit denen vorgegebene Einstellungen verändert werden können. Der rechte und linke Rand zu beiden Seiten der Tabelle wird durch `\LTleft` und `\LTright` bestimmt. Diese internen Register sind mit `\fill` voreingestellt, so dass die Tabelle standardmäßig horizontal zentriert erscheint. Dies kann vom Anwnder mit einer elastischen Maßzuweisung geändert werden. Nach `\setlength{\LTleft}{\parindent}` wird als linker Rand die Tiefe der Zeileneinrückung der ersten Zeile eines Absatzes gewählt.

Die `longtable`-Umgebung kann auch mit `\begin{longtable}[`*pos*`]{`*sp_form*`}` eingeleitet werden. Als Positionierungsparameter darf für *pos* `l`, `c` oder `r` gewählt werden, womit die Tabelle auf den einzelnen Seiten linksbündig, zentriert oder rechtsbündig angeordnet wird. Ohne Angabe des optionalen Positionierungsparameters *pos* erfolgt die Anordnung entsprechend den Einstellungen für `\LTleft` und `\LTright`.

Zwischen dem vorangehenden und dem nachfolgenden Text und der Tabelle wird vertikaler elastischer Zwischenraum eingefügt, der durch `\LTpre` und `\LTpost` festgelegt wird. Diese internen Maßregister sind mit `\bigskipamount` voreingestellt, womit vor und nach der Tabelle vertikaler Zwischenraum eingefügt wird, so als wäre an diesen Stellen der Befehl verb== angebracht worden. Auch diese Voreinstellungen können vom Anwender mit eigenen

Längenzuweisungen geändert werden.

Die Breite für eine etwaige Tabellenüberschrift mit dem `\caption`-Befehl wird durch `\LTcapwidth` festgelegt. Voreingestellt ist hierfür 4 in (4 Zoll). Mit der Zuweisung

```
\setlength{LTcapwidth}{\textwidth}
```

wird sie gleich der Seitenbreite für den umgebenden Text gewählt.

Innerhalb der `longtable`-Umgebung ist der Befehl `\newpage` erlaubt. Er erzwingt an der Stelle seines Auftretens einen Seitenumbruch in der Tabelle. Außerdem sind in absatzartigen Spalten `\footnote`-Befehle zur Erzeugung und Markierung von Fußnoten erlaubt, was die Standardtabellenumgebung nicht gestattet.

Bei der Standard-`tabular`-Umgebung kann mit deren *-Form die Tabellenbreite vorgeschrieben werden, was dann gleichzeitig den Anfangseintrag `@{\extracolsep\fill}` im Spaltenformatierungsfeld verlangt. So wird mit

```
\begin{tabular*}{\textwidth}{@{\extracolsep\fill}...}
```

eine Tabelle erzeugt, deren Breite der Seitenbreite entspricht. Bei der `longtable`-Umgebung kann das mit

```
\setlength{\LTleft}{0pt}\setlength{\LTright}{0pt}
\begin{longtable}{@{\extracolsep\fill}...}
```

erreicht werden.

Das nachfolgende Beispiel erzeugt die dreiseitige Tabelle der nächsten Seite. Es enthält Anfangs- und Fortsetzungsüberschriften sowie zugehörige Kopfzeilen, Standard- und Abschlussfüße.

```
\begin{longtable}{|lr|}
   \caption*{\bfseries Naturparke in den alten Bundesl"andern}\\ \hline
   Bezeichnung & Fl"ache\\ & $(\mbox{km}^2)$\\ \hline
   \endfirsthead
   \caption*{\bfseries Naturparke -- Fortsetzung}\\ \hline
   Bezeichnung & $(\mbox{km}^2)$\\ \hline
   \endhead
   \hline\multicolumn{2}{|r|}{\slshape
          Fortsetzung auf der n"achsten Seite}\\ \hline
   \endfoot
   \hline\multicolumn{2}{l}{\textit{Quelle}:
          Verband Deutscher Naturparke, 1987}
   \endlastfoot
   \slshape Baden-W"urttemberg: & \\
   Neckar-Odenwald              & 1300\\
   Obere Donau                  & 825\\
   . . . . . .
   Westenseee                   & 260\\[1ex]
   \slshape Saarland:           & \\
   Saar-Hunsr"uck               & 1662
\end{longtable
```

Die hier verwendete *-Form des `\caption`-Befehls vermeidet die zusätzliche Angabe „Tabelle n“ vor der übergebenen Überschrift.

Naturparke in den alten Bundesländern

Bezeichnung	Fläche (km^2)
Baden-Württemberg:	
Neckar-Odenwald	1300
Obere Donau	825
Schönbuch	155
Schwäb.-Fränk. Wald	900
Stromberg-Heuchelberg	328
Bayern:	
Altmühltal	2908
Augsburg Westl. Wälder	1175
Bayerische Rhön	1240
Bayerischer Spessart	1710
Bayerischer Wald	2068
Fichtelgebirge	1004
Fränkische Schweiz	2348
Frankenhöhe	1070
Frankenwald	1116
Haßberge	823
Hessenreuter Wald	270
Nördl. Oberpfälzer Wald	643
Oberer Bayerischer Wald	1801
Oberpfälzer Wald	723
Steigerwald	1280
Steinwald	232
Hamburg:	
Harburger Berge	38

Fortsetzung auf der nächsten Seite

Naturparke – Fortsetzung

Bezeichnung	(km^2)
Hessen:	
Bergstraße-Odenwald	1603
Diemelsee	334
Habichtswald	471
Hessische Rhön	700
Hessischer Spessart	710
Hochtaunus	1201
Hoher Vogelsberg	384
Meißner-Kaufunger Wald	429
Rhein-Taunus	807
Niedersachsen:	
Dümmer	472
Elbufer-Drawehn	750
Elm-Lappwald	340
Harz	950
Lüneburger Heide	200
Münden	370
Nördl. Teutoburger Wald	1112
Solling-Vogler	527
Steinhuder Meer	310
Südheide	500
Weserbergland Schaumburg-Hameln	1116
Wildeshauser Geest	965
Nordrhein-Westfahlen:	
Arnsberger Wald	447
Bergisches Land	1916
Ebbegebirge	777

Fortsetzung auf der nächsten Seite

Naturparke – Fortsetzung

Bezeichnung	(km^2)
Eggegebirge u. südl. Teutoburger Wald	593
Hohe Mark	1009
Homert	550
Kottenforst-Ville	770
Nordeifel	1767
Rothaargebirge	980
Siebengebirge	435
Schwalm-Nette	435
Rheinland-Pfalz:	
Nassau	590
Pfälzerwald	1843
Rhein-Westerwald	446
Saar-Hunsrück	918
Südeifel	426
Schleswig-Holstein:	
Aukrug	380
Hüttener Berge Wittensee	260
Holsteinische Schweiz	523
Lauenburgische Seen	444
Westensee	260
Saarland:	
Saar-Hunsrück	1672

Quelle: Verband Deutscher Naturparke, 1987

1.3.3 Seitensteuerung mit afterpage.sty

Das Ergänzungspaket `afterpage.sty` (Autor: DAVID P. CARLISLE) stellt den Befehl

`\afterpage{`*bef_ und text_strukt*`}`

bereit, dessen als Argument übergebene Befehls- und Textstrukturen nach Ausgabe der laufenden Seite als Erstes ausgeführt werden. Der Befehl wurde primär dazu entwickelt, um Schwächen des internen LaTeX-Mechanismus zur Behandlung von Gleitobjekten zu begegnen. Bei mehreren, rasch aufeinander folgenden Gleitobjekten im Eingabetext passiert es häufig, dass diese Gleitobjekte auf eigenen Seiten gesammelt werden und erst am Ende des gesamten Eingabetextes bei der Ausgabe erscheinen. Mit dem Befehl `\clearpage` kann man zwar die unverzügliche Ausgabe aller bis dahin angesammelten Gleitobjekte erzwingen, doch beendet dieser Befehl auch die laufende Seite, ohne sie mit nachfolgendem Text aufzufüllen.

Mit `\afterpage{\clearpage}` wird dagegen die laufende Seite mit nachfolgendem Text aufgefüllt und nach der Seitenausgabe der Befehl `\clearpage` ausgeführt, der dann die unmittelbare Ausgabe aller anstehenden Gleitobjekte erzwingt. Bei langen Tabellen, die mit der `longtable`-Umgebung erstellt werden, wünschen Anwender häufig, dass die Tabelle mit Beginn der nächsten Seite startet. Die Behandlung einer `longtable`-Umgebung als Gleitobjekt innerhalb einer `table`-Umgebung scheitert häufig an fehlendem Speicherplatz, da der Inhalt der `table`-Umgebung in den dafür vorgesehenen Pufferspeicher passen muss, der von einer langen Tabelle leicht überschritten wird. Wird die `longtable`-Umgebung mit ihrer Tabelle in einem eigenen File *ltab*`.tex` abgelegt, dann kann mit

```
\afterpage{\clearpage\input{ltab.tex}}  oder
\afterpage{\clearpage\input{ltab.tex}\clearpage}
```

das gewünschte Ziel erreicht werden. Die laufende Seite wird zunächst mit nachfolgendem Text aufgefüllt; die nächste Seite beginnt evtl. mit unbearbeiteten Gleitobjekten, gefolgt von der langen Tabelle. Die letzte Tabellenseite wird bei der ersten Form mit weiterem nachfolgenden Text bis zum Seitenumbruch aufgefüllt. Bei der zweiten Form startet der nachfolgende Text eine neue Seite nach der Tabellenausgabe.

1.3.4 Das Ergänzungspaket bm.sty

Das Ergänzungspaket `bm.sty` (Autoren: DAVID P. CARLISLE und FRANK MITTELBACH) erleichtert den Fettdruck in mathematischen Formeln oder Teilformeln durch Bereitstellung des mathematischen Umschaltbefehls

`\bm{`*formel_teil*`}`

Damit erscheint der eingeschlossene Formelteil, der die ganze Formel oder eine Teilformel sein kann, in Fettschrift. Die physikalische Aussage „Kraft gleich Masse mal Beschleunigung“ wird formelmäßig als $\boldsymbol{P} = m\boldsymbol{b}$ dargestellt, worin die Kraft $\boldsymbol{P}$ und die Beschleunigung $\boldsymbol{b}$ als gerichtete Größen in Fettdruck erscheinen, um ihre Vektoreigenschaft zu kennzeichnen, während die Masse m als Skalar in normaler Stärke gekennzeichnet wird. Mit `\bm` als mathematischer Umschaltbefehl verlangt die Formel $\boldsymbol{P} = m\boldsymbol{b}$ als Eingabe

```
$ \bm{P} = m\bm{b} $
```

Um mit den LaTeX-Standardmethoden mathematische Formeln in Fettdruck erscheinen zu lassen, ist vor Eintritt in den mathematischen Bearbeitungsmodus noch im Textmodus die mathematische *Version* auf ‚fett' zu schalten, was mit der Erklärung `\boldmath` oder `\mathversion{bold}` geschieht. Hiernach erscheinen alle nachfolgenden mathematischen Formeln in Fettdruck, bis nach Rückkehr in den Textmodus die aktuelle Version mit `\unboldmath` oder `\mathversion{normal}` wieder auf ‚normal' zurückgeschaltet wird.

Die Mischung von Fett- und Normaldruck innerhalb einer Formel führt bei Standard-LaTeX zu der Komplikation, dass nach der `\boldmath`-Erklärung die gesamte Formel in Fettdruck erscheint, da die Rückschaltung auf Normaldruck mit `\unboldmath` erst wieder im Textmodus erlaubt ist. Zur Mischung von Fett- und Normaldruck muss deshalb innerhalb eines mathematischen Bearbeitungsmodus *lokal* mit `\mbox{...}` auf die mathematische Normalversion zurückgeschaltet und dann innerhalb des lokalen Textmodus der mathematische Bearbeitungsmodus erneut lokal aktiviert werden, s. [5a, Kap. 5.4.9]. Zur Erzeugung von $\boldsymbol{P} = m\boldsymbol{b}$ müsste mit den LaTeX-Standardmethoden

```
\boldmath \( P = \mbox{\unboldmath $m$} b \) \unboldmath
```

eingegeben werden, was die Einfachheit und Übersichtlichkeit von `\bm{...}` (s. o) gegenüber der LaTeX-Standardmethode selbst bei einer so einfachen Formel wie bei $\boldsymbol{P} = m\boldsymbol{b}$ hinreichend demonstriert. Bei genauem Hinsehen fällt vielleicht auf, dass sich die mit der LaTeX-Standardmethode erzeugte Formel von der mit `\bm` erzeugten Formel geringfügig unterscheidet, weil bei ersterer das Gleichheitszeichen in Fettschrift und bei letzterer in Normalschrift erscheint, wobei die Unterschiede beim Gleichheitszeichen nur sehr gering sind. Eine identische Formelausgabe ist auch in Standard-LaTeX möglich, doch wird das Verständnis für die dazu erforderliche Einabe noch undurchsichtiger.

Neben der verständlicheren und übersichtlicheren Eingabe von Formeln mit gemischten Normal- und Fettdruckanforderungen von Formelteilen hat `\bm{...}` gegenüber den LaTeX-Standardmethoden noch eine weitere Umschalteigenschaft, die Letztere gar nicht kennt. Nach der Versionsumschaltung mit `\boldmath` werden Symbole, die in unterschiedlichen Größen existieren, wie Integral-, Summen- und Wurzelzeichen sowie die verschieden großen Klammersymbole, nur in der normalen Standardstärke ausgegeben, weil diese dem mathematischen Zeichensatz `cmex` entnommen werden, dessen Symbole nur einheitlich in Normalstärke vorhanden sind. Mit `\bm` werden dagegen auch solche Symbole verstärkt, wie das Beispiel

```
$ \bm{\left<\int\sum\prod\right>}
  \ne \left<\int\sum\prod\right>} $
$ \bm{\sqrt{\alpha\beta\gamma}}
  \ne \sqrt{\alpha\beta\gamma} $
```

$$\langle \int \sum \prod \rangle \ne \langle \int \sum \prod \rangle$$

$$\sqrt{\boldsymbol{\alpha\beta\gamma}} \ne \sqrt{\alpha\beta\gamma}$$

bzw. für abgesetzte Formeln

```
\[ \bm{\int\sqrt{\frac{x-y}{x+y}}\,dx}
   \ne \int\sqrt{\frac{x-y}{x+y}}\,dx \]
```

$$\int \sqrt{\frac{\boldsymbol{x-y}}{\boldsymbol{x+y}}}\,\boldsymbol{dx} \ne \int \sqrt{\frac{x-y}{x+y}}\,dx$$

zeigt. Existieren beim Anwender entsprechende mathematische fette Zeichensätze mit den angeforderten fetten Symbolen, so kommen diese innerhalb `\bm{...}` zur Anwendung. Anderenfalls werden die normalstarken Symbole bei Verwendung des TeX-Standardzeichensatzes `cmex` durch mehrfaches Überdrucken mit geringfügigen Verschiebungen verstärkt, wodurch das fette Aussehen nachgebildet wird.

Das Ergänzungspaket `bm.sty` gestattet mit der Definitionsstruktur

`\DeclareBoldMathCommand[`*math_vers*`]{\`*bef_name*`}{`*math_ausdr*`}`

fette mathematische Symbole oder Teilformeln, die mit *math_ausdr* erklärt werden, unter dem Befehlsnamen `\`*bef_name* einzurichten. Entfällt der optionale Parameter *math_vers*, so wird hierfür standardmäßig `bold` mit der lokalen Wirkung von `\boldmath` verwendet. Bei Beschränkung auf die mathematischen TeX-eigenen Zeichensätze kann für *math_vers* explizit nur `bold` und `heavy` mit gleicher Wirkung eingesetzt werden.

Mit `\DeclareBoldMathCommand{\balpha}{\alpha}` wird der Befehl `\balpha` eingerichtet, der dann mit `$\alpha\ne\balpha$` $\alpha \neq \boldsymbol{\alpha}$ erzeugt. Dieser Definitionsbefehl wird auch in einer analogen Kurzform als `\bmdefine{\`*bef_name*`}{`*math_ausdr*`}` angeboten, der `\DeclareBoldMathCommand[bold]{\`*bef_name*`}{`*math_ausdr*`}` entspricht.

Werden bei einer Textbearbeitung des Anwenders fette mathematische Symbole oder Teilformeln mehrfach angefordert, so sollten hierfür geeignete Befehlsnamen mit `\bmdefine` oder `\DeclareBoldMathCommand` eingerichtet und diese fetten Symbole und Teilformeln dann über ihre so definierten Befehlsnamen angefordert werden. Dies führt zu einer schnelleren Ausgabe als ihre Anforderung über jeweils gleiche `\bm`-Befehle.

Kommen beim Anwender mathematische Zeichensätze zur Anwendung, die zusätzlich extrastarke Zeichen anbieten, wie z. B. bei dem Schriftsatzpaket ‘maththime plus’, so sollte hierfür ein geeigneter Versionsname mit `\DeclareMathVersion{`*vers_name*`}` wie z. B `heavy` für *vers_name* erklärt werden. Hiermit können dann mit dieser Versionsangabe in `\DeclareBoldMathCommand` unter den dort definierten Befehlsnamen die extrastarken Zeichen oder Teilformeln ausgegeben werden. Mit einem so erklärten Versionsnamen `heavy` steht dann auch der Definitionsbefehl `\hmdefine` als analoge Kurzform für `\DeclareBoldMathCommand[heavy]` zur Verfügung, wie oben entsprechend für `\bmdefine` vorgestellt.

Ohne explizite Erklärung des Versionsnamen `heavy` mit `\DeclareMathVersion {heavy}` wird dieser im `bm.sty`-Ergänzungspaket mit `bold` gleichgesetzt, so dass die Versionsangabe `heavy` oder der Befehl `\hmdefine` zum gleichen Ergebnis führt, wie dies mit Versionsangabe `bold` oder `\bmdefine` der Fall ist, d. h., es werden nur die fetten Standardschriften verwendet.

Für weitere Nutzungserläuterungen des `bm.sty`-Ergänzungspakets möge sich der Anwender die interne Dokumentation mit der LaTeX-Bearbeitung von `bm.dtx` zumindest in der Kurzform gemäß 1.2.2 auf S. 8 aufbereiten und ausdrucken.

Bei deutschen LaTeX-Betreibern ist darauf zu achten, dass bei gleichzeitiger Anforderung von `bm.sty` und `german.sty` beim Einbinden beider Ergänzungspakete die Reihenfolge `...\usepackage{bm}...\usepackage{german}...` einzuhalten ist, das Ergänzungspaket `bm.sty` also *vor* dem deutschen Anpassungspaket `german.sty` zu laden ist!

1.3.5 Das Ergänzungspaket calc.sty

Das LaTeX-Ergänzungspaket `calc.sty` stammt von KARSTEN KRAB THORUP und FRANK JENSEN unter Mitwirkung von CHRIS ROWLEY. Es erlaubt, mit LaTeX einfache arithmetische Ausdrücke mittels der vier Grundrechenarten zu bilden, die innerhalb der LaTeX-Zuweisungsbefehle `\setcounter` und `\addtocounter` sowie `\setlength` und `\addtolength` auftreten dürfen. Dazu wurde mit `cald.sty` die Syntax der vorstehenden

Zuweisungsbefehle verändert, die nunmehr eine deutlich leistungsfähigere und einsichtigere Zuweisungsarithmetik erlauben.

Die allgemeine Syntax der vorstehenden Zuweisungsbefehle lautet dabei nach wie vor vertraut (s. [5a, Kap. 7.1.3 und 7.2]):

`\setcounter{`*zähler*`}{`*zahl_ausdr*`}` Dem Zähler mit dem Namen *zähler* wird der Wert des Zahlenausdrucks *zahl_ausdr* zugewiesen.

`\addtocounter{`*zähler*`}{`*zahl_ausdr*`}` Der aktuellen Wert des Zählers mit dem Namen *zähler* wird um den Wert des Zahlenausdrucks *zahl_ausdr* verändert, und zwar *vergrößert*, wenn der Wert von *zahl_ausdr* positiv ist (Addition), bzw. *verkleinert*, wenn der Wert von *zahl_ausdr* negativ ist (Subtraktion).

`\setlength{\`*längen_bef*`}{`*maß_ausdr*`}` Dem LaTeX- oder anwendereigenen Längenbefehl `\`*längen_bef* wird der Wert des Längenmaßausdrucks *maß_ausdr* zugewiesen.

`\addtolength{\`*längen_bef*`}{`*maß_ausdr*`}` Der aktuelle Wert des Längenbefehls mit dem Namen *längen_bef* wird um den Wert des Längenmaßausdrucks *maß_ausdr* verändert, und zwar *vergrößert*, wenn der Wert von *maß_ausdr* positiv ist (Addition), bzw. *verkleinert*, wenn der Wert von *maß_ausdr* negativ ist (Subtraktion).

Bei den vorstehend angeführten Ausdrücken *zahl_ausdr* oder *maß_ausdr* erfolgen Additionen und Subtraktionen in der vertrauten Syntax: $a_1 \pm a_2 \pm \cdots \pm a_n$, worin $\pm$ für einen der Additions- oder Subtraktionsoperatoren $+$ bzw. $-$ und a_i für eine Zahlen- oder Maßangabe wie `3` oder `4cm` steht, wobei in den Summen- und Differenzketten alle Terme vom gleichen Typ sein müssen. Die Angabe `2 + 4` oder `2cm + 4cm` ist damit zulässig, `2cm + 4` dagegen nicht, da letztere eine Zahl mit einer Länge vermischen würde.

Innerhalb von Längensummen oder Differenzen dürfen die Maßeinheiten jedoch wechseln, wie z. B. `2in+5cm`, da alle Längenangaben intern vorab in die Maßeinheit 'sp' umgewandelt werden: $1\,\text{pt} = 65636\,\text{sp}$, $1\,\text{in} = 72{,}27\,\text{pt}$, $1\,\text{in} = 2.54\,\text{cm}$ usw.). Beim vorstehenden Beispiel führt die interne Umrechnung zu: $2\,\text{in} + 5\,\text{cm} = (2 + 5/2.54) \times 72.27\text{pt} = 286.80378 \times 65636 = 18\,795\,972\,\text{sp}$.

Für a_i können direkte Zahlen- und Längenangaben wie `10` und `5mm` oder Befehlsnamen stehen, wenn letztere wiederum für reine Zahlen- oder Längenangaben stehen. Mit

```
\newcommand{\ten}{10}          \newcommand{five}{5}
\newcommand{\twomm}{2mm}  \newcommand{fourpt}{4pt}
```

führt `\ten + \five` zu $10 + 5$ und damit zum Ergebnis 15 und `\twomm - \fourpt` zu $2\,\text{mm} - 4\,\text{pt}$, was wegen der internen Umrechnung zu $(5.69055 + 4)\,\text{pt} = 636\,049\,\text{sp}$ führt.

Für a_i kann bei Bezug auf extistierende Zähler `\value{`*zähler*`}` eingesetzt werden. Werden z. B. mit `\newcounter{prevpage}` und `\newcounter{\pastpage}` die anwendereigenen Zähler `prevpage` und `pastpage` eingerichtet, dann können diese z. B. mit

```
\setcounter{prevpage}{\value{page} - 5}
\setcounter{pastpage}{\value{page} + 2}
```

auf den um 5 verminderten bzw. um 2 vergrößerten Wert des aktuellen LaTeX-Seitenzählers `page` gesetzt werden.

Bei Längenangaben dürfen für die a_i auch Namen von Längenregistern (Längenbefehlen) stehen. Mit `\newlength{\bigparkskip}` kann diesem anwendereigenen Längenbefehl mit

```
\setlength{\bigparskip}{\bigsip + \parsip}
```

die Summe der LaTeX-eigenen elastischen Längenmaße `\bigslip` und `\parskip` zugewiesen werden.

Bei den Längenangaben in arithmetischen Ausdrücken können sowohl feste als auch elastische Maße angegeben werden. Bei der Summen- bzw. Differenzbildung von festen Maßen ist deren Bedeutung selbstverständlich, bei elastischen werden Soll-, Dehn- und Stauchwerte jeweils für sich addiert bzw. subtrahiert. Bei der Kombination von festen mit elastischen Maßen, die zulässig ist, ist das Ergebnis so, als würde das feste Maß in ein elastisches erweitert, bei dem die Dehn- und Stauchwerte jeweils mit 0 sp gesetzt sind.

Zahlen und Zahlenbefehle sowie Längen und Längenbefehle können mit Zahlen multipliziert oder dividiert werden. Angaben wie `3 * 4` oder `2cm * 5` und `10/2` oder `20mm/5` sind erlaubt und führen zum erwarteten Ergebnis. Bei der Multiplikation und Division von Längen oder Längenbefehlen mit Zahlen muss die Längenangabe dem Zahlenfaktor oder Divisor vorangehen: *länge*∗*faktor* und *länge*/*divisor* bzw. *längen_bef*∗*faktor* und *längen_bef*/*divisor*.

Bei der Division wird der ganzzahlige Anteil zurückgeliefert: `12/5` ergibt als Ergebnis 2. Bei der Division von Längen oder Längenbefehlen durch Zahlen bezieht sich diese Rundung auf die interne Maßeinheit ‘sp’, deren Rückwandlung in ‘cm’ oder ‘pt’ hierfür wegen der Kleinheit von ‘sp’ nahezu korrekte Bruchteile für die Ausgangsmaße zurückliefern.

Mehrfache Multiplikationen und Divisionen oder deren Mischung sind ebenfalls erlaubt. So ergibt `6*3*2` erwartungsgemäß `36` und `6*3/2` führt zu `9`. Bei der Kombination von Summen oder Differenzen mit Produkten und Quotienten gilt die herkömmliche Prioritätsregel (Punkt- vor Strichrechnung): `5*2+3` ergibt `13`. Durch explizite Klammerung mit runden Klammern kann die eingebaute Prioritätsregel überspielt werden: `5*(2+3)` ergibt dann `25`.

Bei der Multiplikation oder Division von Längen mit Zahlen dürfen für die Längen auch elastische Maße stehen. Die Multiplikation oder Division bezieht sich dann gleichzeitig auf die Soll-, Dehn- und Schrumpfwerte. Ist z. B.

```
\setlength{\parskip}{5pt plus 2pt minus 1pt}
```

gewählt worden, so ergibt `\parskip * 3`: `15pt plus 6pt minus 3pt`.

Das Ergänzungspaket `calc.sty` erlaubt als Zahlenfaktoren oder Dividenden auch natürliche oder dezimale Brüche. Diese müssen in der Form `\ratio{`*Zähler*`}{`*Nenner*`}` bzw. `\real{`*Dezimalbruch*`}` angegeben werden.

```
\setcounter{x}{100 * \real{1.75}}
\setcounter{y}{10 / \ratio{2}{3}}
```

ergibt für die LaTeX-Zähler x und y die Zuweisung von 175 bzw. 15. Das Ergebnis wird hierbei auf seinen ganzzahligen Anteil abgerundet. So ergibt `\setcounter{x}{3 * \real{1.6}}` für x die Zuweisung von 4. Bei mehrfachen Produkten erfolgt die Rundung jeweils für einzelne Zwischenergebnisse von links nach rechts.

```
\setcounter{x}{3 * \real{1.6} * \real{1.7}
```

ergibt für x die Zuweisung von 6, da das erste Zwischenergebnis von $3 \times 1.6 = 4.8$ auf 4 abgerundet und dieser Wert dann mit $4 \times 1.7 = 6.8$ auf 6 abgerundet wird.

Die Multiplikation und Division von festen Längenmaßen mit natürlichen oder Dezimalbrüchen erfolgt erwartungsgemäß, wenn man beachtet, dass die übergebenen Längenmaße vorab in die Maßeinheit 'sp' umgewandelt werden, die dann mit den übergebenen Bruchstrukturen multipliziert bzw. dividiert werden. Das interne Ergebnis in 'sp' beschränkt sich auf deren ganzteiligen Anteil, dessen Rückwandlung in die Ausgangseinheiten wie 'in', 'cm', 'pt' u. a. für diese fast exakte Bruchanteile zurückliefern (s. o. zur Division von Längenmaßen durch Ganzzahlen).

Sind `\figwidth` und `\figheight` Breite und Höhe einer Figur, die so vergrößert werden soll, dass sie die Textbreite der Seite ausfüllt, dann kann deren Höhe mit

```
\setlength{\scaledht}{\figheight*\ratio{\textwidth}{figwidth}}
```

entsprechend skaliert werden.

Bei der Multiplikation von elastischen Längenmaßen mit der `\real`-Funkion verschwinden jedoch die elatischen Anteile. Die LaTeX-Zuweisung

```
\setlength{\parskip}{5pt plus2pt minus1pt * \real{1.5}}
```

ergibt für `\parskip` den Wert von `7.5pt` ohne zusätzliche Dehn- und Schrumpfanteile.

Im Juni 1998 wurden dem Ergänzungspaket `calc.sty` als weitere Bestimmungsstrukturen für Textabmessungen

```
\widthof{text}  \heightof{emphtext}  \depthof{ text}
```

zugefügt, die Breite, Höhe und Tiefe des übergebenen Textes *text* zurückliefern, die dann als entsprechende Längen in den vorab beschriebenen arithmetischen Ausdrücken auftreten dürfen, z. B.

```
\setlength{\columnwidth}{\widthof{Dieser Beispieltext}
                                        * \real{0.667}}
```

1.3.6 Ergänzung der enumerate-Umgebung

Das Ergänzungspaket `enumerate.sty` (Autor: DAVID P. CARLISLE) gestattet eine flexiblere Handhabung der `enumerate`-Umgebung. Die `enumerate`-Umgebung des Ergänzungspakets kennt einen optionalen Parameter der Form

```
\begin{enumerate}[{opt_atxt} k {opt_etxt}] ... \end{enumerate}
```

Hierin steht *k* für eines der Kennzeichen `A`, `a`, `I`, `i` oder `1`, womit die Zählweise der eingeschlossenen `\item`-Befehle mit fortlaufenden Großbuchstaben (`A`), Kleinbuchstaben (`a`), großen römischen Zahlen (`I`), kleinen römischen Zahlen (`i`) oder mit arabischen Zahlen (`1`) erfolgt. Der Zählerkennung kann ein optionaler Text vorangestellt {*opt_atxt*} und/oder nachgestellt {*opt_etxt*} werden, der dann ebenfalls mit allen `\item`-Befehlen erscheint. Textzeichen in solchem optionalen Text, die mit den Zeichen für die Zählerkennung übereinstimmen, müssen in geschweiften Klammern eingegeben werden. Es empfiehlt sich deshalb, solchen optionalen Text stets in `{}`-Paare einzuschließen.

```
\begin{enumerate}[{Beisp.} a)]   ... \end{enumerate}  und
\begin{enumerate}[1. {Beispiel}] ... \end{enumerate}
```

gibt im ersten Fall die `\item`-Befehle nacheinander als Beisp. a), Beisp. b), Beisp. c), ... aus, während sie im zweiten Fall als 1. Beispiel, 2. Beispiel, 3. Beispiel, ... erscheinen.

Innerhalb der erweiterten `enumerate`-Umgebung können, wie bei ihrer Standardform, mit `\label`-Befehlen Markierungen angebracht werden, die sich auf den Zählerstand des jeweils vorangehenden `\item`-Befehls beziehen. Referenzierungen mit `\ref`-Befehlen erscheinen in der eingestellten Zählweise, so beim ersten Beispiel als a, b, c, ..., bzw. als 1, 2, 3, ... beim zweiten Beispiel. Die optionalen Texteinträge erscheinen bei den Referenzierungen dagegen nicht. Sie müssten bei Bedarf manuell zugefügt werden, z. B. wie `s.~Beisp.~\ref{...}`.

1.3.7 Das Ergänzungspaket indentfirst.sty

Der Name dieses Ergänzungspakets beschreibt bereits seine Wirkung. Mit ihm erscheinen alle ersten Absätze nach einer Gliederungsüberschrift mit den Befehlen `\chapter{...}`, `\section{...}`, ... ebenfalls eingerückt, was standardmäßig nicht der Fall ist, da die vorangehende Gliederungsüberschrift den nachfolgenden Text hinreichend deutlich als neuen Absatz kennzeichnet. Soll der erste Absatz trotzdem eingerückt erscheinen, wie nach der vorangehenden Überschrift, so ist `\usepackage{indentfirst}` im Vorspann anzugeben. Eine direkte Lösung ist sonst nur mit geänderten Makrodefinitionen für die Gliederungsbefehle zu erreichen, was die Mehrzahl der Anwender sicher überfordert.

1.3.8 Mehrspaltenformatierungen mit multicol.sty

Das Ergänzungspaket `multicol.sty` von FRANK MITTELBACH wurde bereits als Stilfile für LaTeX 2.09 bereitgestellt und ist von dorther vielleicht bekannt. Es stellt eine Verallgemeinerung des LaTeX-Befehls `\twocolumn` dar, mit der

1. eine n-spaltige Seitenaufteilung erreicht werden kann, bei der
2. der Text *gleichmäßig* auf die n Spalten aufgeteilt wird und
3. auf einer Seite mehrfach zwischen ein- und mehrspaltiger Formatierung umgeschaltet werden kann.

Die beiden nachfolgenden Beispiele demonstrieren und erläutern die Haupteigenschaft der mit `multicol.sty` bereitgestellten `multicols`-Umgebung.

Beispiel für dreispaltigen Textsatz:

Der mehrspaltig zu formatierende Text ist mit

`\begin{multicols}{`n`}`
mehrspaltiger Text
`\end{multicols}`

einzuschließen, wobei die Zahlenangabe für n die Anzahl der Spalten bestimmt. (In diesem Beispiel $n = 3$.)

Der Aufruf kennt einen optionalen Parameter im Anschluss an den Spaltenparameter `{`n`}`

`{`n`}[`*volltext*`]`

mit dem der Inhalt von *volltext* über die volle Seitenbreite oberhalb der b Spalten angeordnet wird.

Der erzeugende Text für diese Spalten braucht nicht erläutert zu werden. Ausnahme: Der optionale Text ist bei einer Schriftänderung, wie hier für die Fettschrift

`[{{\bf Beispiel...}}]`

in geschweifte Doppelklammern einzuschließen!

Beispiel für zweispaltigen Textsatz:

Ein guter Zeilenumbruch wird für schmale Spalten sehr erschwert. Der seit TeX 3.0 neue Befehl

`\emergencystretch`*maß*

kann sich hier als nützlich erweisen. Wird ihm ein Maß größer 0 pt zugewiesen, so führt TeX bei der Absatzformatierung einen dritten Umbruchversuch durch, wenn der Satz mit den Standardregeln nicht ordnungsgemäß zu erreichen ist. Bei diesem dritten Umbruchversuch kann der Gesamtzwischenraum einer Zeile bis zu dem an `\emergencystretch` übergebenen Maß zusätzlich erweitert werden.

Das Ergänzungspaket `multicol.sty` macht intern hiervon bereits Gebrauch, indem `\emergencystretch` auf $4 \times n$ pt gesetzt wird, wobei n die übergebene Spaltenanzahl bedeutet. Frühere TeX-Versionen bleiben hiervon unbeeinflusst.

Der Erzeugungstext für die vorangehenden drei- und zweispaltigen Beispiele wurde im laufenden Text als

```
\begin{multicols}{3}[\noindent\textbf{Beispiel f"ur dreispaltigen ...}]
  fortlaufender Spaltentext  \end{multicols}
\begin{multicols}{2}[\nonident\textbf{Beispiel f"ur zweispaltigen ...}]
  fortlaufender Spaltentext  \end{multicols}
```

eingegeben[2]. Bei Eintritt in die `multicols`-Umgebung wird geprüft, ob auf der laufenden Seite noch vertikaler Platz vom Betrag `\premulticols` (= 50 pt) zur Verfügung steht. Ist das nicht der Fall, so findet für den vorangehenden Text ein Seitenumbruch statt. Anderenfalls wird zwischen dem vorangehenden Text und dem anschließenden n-spaltigen Text der vertikale Zwischenraum `\multicolsep` (= 12 pt plus 4 pt minus 3 pt) eingefügt. Am Ende der `multicols`-Umgebung wird geprüft, ob auf der laufenden Seite noch vertikaler Platz vom Betrag `\postmulticols` (= 20 pt) vorhanden ist. Falls nicht, findet ein Seitenumbruch vor dem nachfolgenden Text statt, anderenfalls wird wiederum vertikaler Zwischenraum vom Betrag `\mulicolsep` zum nachfolgenden Text eingefügt.

Die obigen Standardwerte für `\premulticols`, `\postmulticols` und `\multicolsep` können vom Anwender vor Eintritt in die `multicols`-Umgebung durch Längenzuweisungen geändert werden. Dies wird häufig bei gemeinsamen Überschriften mit dem optionalen Überschriftparameter erforderlich, insbesondere wenn die Überschrift sehr kurz oder sehr lang ist, so dass der Standardwert `\premulticols` (= 50 pt) hierfür eigentlich zu groß oder zu klein ist. Zur Vereinfachung kennt die `multicols`-Umgebung einen zweiten optionalen Parameter

`\begin{multicols}{`n`}[`*überschrift*`][`*maß_angabe*`]`

mit einer Maßangabe für *maß_angabe*. Sie bewirkt, dass das übergebene Maß so verwendet wird, als wäre für diese Umgebung der Wert für `\premulticols` lokal mit *maß_angabe* eingestellt worden, während für alle anderen `multicols`-Umgebungen die globale Vorgabe für `\premulticols` gilt.

Mit den LaTeX-Erklärungen `\columnsep` und `\columnseprule` kann der Spaltenzwischenraum bzw. die Stärke eines vertikalen Trennstrichs zwischen den Spalten eingestellt

[2] Die Schriftumschaltung verlangte in LaTeX 2.09 den Einschluss in ein doppeltes geschweiftes Klammerpaar als `\begin{multicols}[\noindent{{\bf Beispiel für ... }}]`

da eine Schriftumschaltung innerhalb der optionalen Gesamtüberschrift global und damit auch für den nachfolgenden Text wirkt. Erst bei Einschluss in ein geschweiftes Doppelklammerpaar beschränkte sich die Schriftumschaltung auf die Überschrift! Mit den LaTeX 2ε-Schriftbefehlen `\text`xx{*text_arg*} entfällt dieses Problem.

werden, die standardmäßig mit 10 pt für den Zwischenraum und 0 pt für die Strichstärke besetzt sind.

Randnotizen und spaltenweise Gleitumgebungen, also `\marginpar`-Befehle sowie `table`- und `figure`-Umgebungen, sind in ihrer Standardform in einer `multicols`-Umgebung nicht erlaubt. Dagegen sind Gleitumgebungen in der *-Form zulässig, da deren Gleitobjekte am Seitenanfang gemeinsam über alle Spalten reichen.

Das Ausbalancieren eines Textes auf gleiche Höhen für alle Spalten gelingt in einer `multicols`-Umgebung nicht immer, insbesondere wenn der eingeschlossene Text über keinen oder nur geringen vertikalen Zwischenraum verfügt. Man stelle sich z. B. vor, dass der eingeschlossene Text einer dreispaltigen Umgebung insgesamt zehn Zeilen erzeugt, zwischen denen keine Absatzunterteilungen auftreten. Diese lassen sich auf drei Spalten nur als zweimal drei und einmal vier Zeilen aufteilen.

Bei mehrspaltigen Strukturen mit relativ vielen Zeilen kann das vertikale Ausbalancieren durch Zuweisung eines elastischen Dehnmaßes an `\multicolbaselineskip` erleichtert werden. Dieses spezielle Register ist mit 0 pt vorbesetzt und wird dem Standardzeilenabstand `\baselineskip` innerhalb der `multicols`-Umgebung zuaddiert. Mit `\setlength{\multicolbaselineskip}{0pt plus1pt}` erhält der Zeilenabstand eine Dehnelastizität von 1 pt, was das vertikale Ausbalancieren erleichtert.

Das `multicol`-Ergänzungspaket erlaubt die lokalen Optionsangaben `errorshow`, `infoshow`, `balancingshow`, `markshow` und `debugshow` beim `\usepackage`-Befehl. Damit können unterschiedlich umfangreiche Terminalmeldungen während der Bearbeitung generiert werden, die die internen Abläufe bei der Einrichtung der Einzelspalten widerspiegeln. Der Protokollierungsumfang steigt mit der angegebenen Optionsreihenfolge. So schließt die letzte Option `debugshow` alle vorangehenden mit ein.

1.3.9 Das Ergänzungspaket ftnright.sty

Fußnoten bei der Klassenoption `twoside` oder innerhalb der `multicols`-Umgebung werden am unteren Ende der Seite gesammelt und reichen über die ganze Seitenbreite.[3] Als Alternative hat FRANK MITTELBACH das Ergänzungspaket `ftnright` bereitgestellt. Dieses war ursprünglich für die LaTeX-Klassenoption `twocolumn` entwickelt worden und bewirkt, dass Fußnoten auf jeder Seite am unteren Ende der rechten Spalte eingerichtet werden.[3] `ftnright` kann aber auch mit dem Ergänzungspaket `multicol` verknüpft werden, wenn eine `multicols`-Umgebung über mehrere Seiten reicht oder eine Seite bis zum unteren Seitenende prägt.

Die Anordnung der Fußnoten bei mehrspaltigem Text in dieser Form wird vielen Anwendern zweckmäßiger erscheinen.[4] Der Leser möge dies selbst beurteilen. Zur Demonstration sind die Fußnoten hier zweimal platziert worden, einmal in der Standardform am unteren Seitenende und zum anderen am Ende der rechten Spalte, wie es durch das Ergänzungspaket `fntright` erreicht wird.

Mit `\columnseprule0.4pt` wird zusätzlich der sichtbare Spaltentrennstrich demonstriert.

3. Was mit der alternativen Platzierung der Fußnote an dieser Stelle demonstriert wird.

4. Dies hat auch strukturelle Vorteile. Beim Auftreten vieler Fußnoten kann es bei der Standardanordnung gelegentlich vorkommen, dass als Folge des komplizierten Balancierungsvorgangs zur gleichmäßigen vertikalen Ausfüllung der Spalten mit gleichzeitigem Seitenumbruch eine Fußnote fehlerhaft erst auf der nächsten Seite erscheint. Diese Schwäche wird mit dem Ergänzungspaket `ftnright.sty` vermieden.

[3] Was mit der Fußnote an dieser Stelle demonstriert wird.

[4] Dies hat auch strukturelle Vorteile. Beim Auftreten vieler Fußnoten kann es bei der Standardanordnung gelegentlich vorkommen, dass als Folge des komplizierten Balancierungsvorgangs zur gleichmäßigen vertikalen Ausfüllung der Spalten mit gleichzeitigem Seitenumbruch eine Fußnote fehlerhaft erst auf der nächsten Seite erscheint. Diese Schwäche wird mit dem Ergänzungspaket `ftnright.sty` vermieden.

1.3.10 Erweiterte Regelsätze mit theorem.sty

Das Ergänzungspaket `theorem.sty` von FRANK MITTELBACH ermöglicht variablere Regelsatzstrukuren gegenüber dem LaTeX-Standard. Beim Durchlesen dieses Abschnitts sollte sich der Leser nochmals die Eigenschaften des LaTeX-Definitionsbefehls `\newtheorem` vor Augen führen. Diese sind in [5a, Abschn. 4.5] dargestellt. Der Befehl kennt drei Syntaxformen

`\newtheorem{`*strukt_name*`}{`*regel_begriff*`}`
`\newtheorem{`*strukt_name*`}{`*regel_begriff*`}[`*zusatz_zähler*`]`
`\newtheorem{`*strukt_name*`}[`*ersatz_zähler*`]{`*regel_begriff*`}`

Die Befehle richten neue Umgebungen mit dem Namen *pstrukt_name* ein, die dann mit

`\begin{`*strukt_name*`}[`*zusatz*`]` *regel_text* `\end{`*strukt_name*`}`

angesprochen werden. Mit `\newtheorem{satz}{Satz}` werden Regelsätze der Form

Satz 1 (Bolzano-Weierstraß). *Jede beschränkte unendliche Punktmenge besitzt mindestens einen Häufungspunkt.*

möglich, die mit der Eingabe wie

```
\begin{satz}[Bolzano-Weierstra"s] Jede beschr"ankte unendl...\end{satz}
```

erzeugt werden.

Mit jedem neuen Aufruf `\begin{satz}` wird die laufende Satznummer um eins erhöht und dem Regelbegriff **Satz** nachgefügt. Enthält der Aufruf die optionale Angabe `[`*zusatz*`]` wie bei dem Beispiel `Bolzano-Weierstra"s`, so folgt dieser Zusatz in Klammern und ebenfalls in Fettdruck dem durchnummerierten Regelbegriff. Der Regeltext erscheint in der Schriftart `\it` bzw. in LaTeX 2_ε mit dem Formattribut `itshape`.

Bei der zweiten Syntaxform, bei der *zusatz_zähler* ein Gliederungszähler wie `chapter` oder `section` ist, wird der Inhalt dieses Gliederungszählers dem Regelzähler, durch einen Punkt getrennt, vorangestellt. Bei der dritten Syntaxform wird als Regelzähler der Zähler einer anderen Regelstruktur verwendet. Mit

```
\newtheorem{hilfs}[satz]{Hilfssatz}
```

wird die Regelumgebung `hilfs` eingerichtet, mit deren Aufruf als Regelbegriff **Hilfssatz** erscheint, dessen Nummerierung aber durch den Regelzähler von `satz` erfolgt.

Mit dem Ergänzungspaket `theorem.sty`, das mit `\usepackage{theorem}` zusammen mit seinen weiteren Hilfs-`.sty`-Files aktiviert wird, können diese Regelstrukturen gegenüber dem Standard weitgehend verändert werden. Mit den Erklärungen

`\theorembodyfont{\`*schrift*`}` *und* `\theoremheaderfont{\`*schrift*`}`

können die Schriften für den Regeltext und den Regelbegriff eingestellt werden. Für den Regeltext wird die Schrift verwendet, die zum Zeitpunkt der zugehörigen Definition mit `\newtheorem` aktiv war. Mit

```
{\theorembodyfont{\sf}  \newtheorem{satz}{Satz}}
{\theorembodyfont{\sl}  \newtheorem{hilfs}[satz]{Hilfssatz}}
```

erscheint der Regeltext der Regelumgebung `satz` in `\sf` und der der Regelumgebung `hilfs` in `\sl`. Mit LaTeX 2_ε wird man hier zweckmäßiger `\sffamily` bzw. `\slshape` einsetzen. Die Schrift für den Regelbegriff kann für alle Regelstrukturen nur *einheitlich* gewählt werden.

Ihre Erklärung darf nur *einmal* im Vorspann erscheinen. Mit `\theoremheaderfont{\sc}` (mit LaTeX 2_ε wird `\sc` besser durch `\scshape` ersetzt) erscheinen alle Regelbegriffe in der Schriftart `\sc` (bzw. mit LaTeX 2_ε mit dem Formattribut `scshape`).

Die Anordnung des Regelbegriffs mit der Nummerierung und bezüglich des nachfolgenden Regeltextes kann mit Erklärungen von

```
\theoremstyle{stil}
```

gesteuert werden. Für *stil* können gewählt werden:

`plain`	Bewirkt Regelsätze entsprechend dem LaTeX-Standard.
`break`	Der Regeltext beginnt in einer neuen Zeile unterhalb des darüber stehenden Regelbegriffs.
`marginbreak`	Die laufende Nummer des Regelbegriffs erscheint im linken Rand vor dem linksbündigen Regelbegriff. Der Regeltext beginnt mit einer neuen Zeile wie bei `break`.
`changebreak`	Die laufende Nummer erscheint linksbündig *vor* dem Regelbegriff. Der Regeltext beginnt mit einer neuen Zeile wie bei `break`.
`change`	Die laufende Nummer erscheint *vor* dem Regelbegriff, an den sich der Regeltext in der gleichen Zeile anschließt.
`margin`	Die laufende Nummer erscheint im linken Rand vor dem linksbündigen Regelbegriff. Der Regeltext füllt zunächst die gleiche Zeile bis zum evtl. rechtsbündigen Umbruch wie bei `plain` und `change`.

Die jeweils aktuelle Einstellung von `\theoremstyle` bestimmt das Verhalten der anschließend mit `\newtheorem` eingerichteten Regelumgebungen. Mit

```
\theoremstyle{break}  \newtheorem{satz}{Satz}
                      \newtheorem{lem}{Lemma}[chapter]
\theoremstyle{change} \newtheorem{hilfs}[satz]{Hilfssatz}
```

werden die Regelumgebungen `satz` und `lem` im Stil `break` und die Umgebung `hilfs` im Stil `change` gesetzt. Die Einstellung der Schriften für die Regeltexte kann hiervon unabhängig mit `\theorembodyfont` erfolgen, wie bereits oben dargestellt.

Die Erklärungen von `\newtheorem`, `\theoremstyle`, `\theorembodyfont` und `\theoremheaderfont` sind nur im Vorspann eines LaTeX-Files erlaubt, wobei die letzte nur einmal auftreten darf. Ohne eine explizite Angabe der Einstellerklärungen werden Standardeinstellungen gewählt, z. B. `plain` für die Stileinstellung. Ohne explizite Einstellungen der Schriften für den Regeltext werden hierfür Standardschriften mit den Stileinstellungen vorgegeben, nämlich `\itshape` für den Stil `plain` und `\slshape` für alle anderen Stilarten. Das Standardschriftattribut für die Regelbegriffe ist `\bfseries`.

`theorem.sty` kennt noch zwei weitere Einstellbefehle für den vertikalen Zwischenraum *vor* und *nach* einer Regelumgebung zum umgebenden Text. Dies sind (mit gleichzeitiger Angabe ihrer Standardwerte):

```
\theorempreskipamount  = 12pt plus5pt minus3pt  und
\theorempostskipamount =  8pt plus3pt minus1.5pt
```

Änderungserklärungen mit diesen Befehlen sind an beliebigen Stellen im Text erlaubt. Die elastischen Maßzuweisungen erfolgen mit `\setlength`. Die neuen Einstellungen gelten für *alle* nachfolgenden Regelumgebungen.