Wissenschaftliches Arbeiten mit TEX/LATEX

Berndt Farwer (Hrsg.)

Preprint, 20. Februar 2004

Vorwort

Im wissenschaftlichen Betrieb sind schriftliche Darstellungen eines der wichtigsten Kommunikationsmittel überhaupt. Hierbei muß eine Vielzahl von Faktoren berücksichtigt werden. Zum Beispiel ist es in den Naturwissenschaften häufig erforderlich, mathematische Formeln in den Text zu integrieren. Hierzu sind spezielle Hilfsmittel erforderlich. In allen Bereichen des wissenschaftlichen Publizierens sind Diagramme zur Veranschaulichung zu erstellen und einzubinden. In den Musikwissenschaften werden Auszüge aus Partituren verwendet, so daß Notensatz erforderlich ist. Für den potentiellen Autoren stellt sich somit die schwierige Frage nach einem oder mehreren Hilfsmitteln bei der Erstellung von Seminararbeiten, Diplomarbeiten, Konferenzbeiträgen, o.ä.

Trotz (oder gerade wegen) der immer größer werdenden Komplexität von herkömmlichen (WYSIWIG-)Textsystemen, erfreut sich im wissenschftlichen Publizieren TEX und LATEX immer größerer Beliebtheit. Diese Textsatzsysteme machen den Autoren gleichzeitig zum Setzer, ohne jedoch eine Ausbildung im Layout oder Textsatz zu erfordern. Die strengen – in Dokumentenklassen und Stilvorlagen – vorgegebenen layouterischen Richtlinien sind meist vorgegeben, so daß der Autor sich um die Gestalt des Textes wenig zu kümmern braucht. Vielmehr kann er sich voll und ganz dem Inhalt widmen und das Layout dem Textsatzsystem überlassen.

Nicht zuletzt aus wirtschaflichen Gründen findet TEX bzw. LATEX auch im Verlagswesen immer mehr Anhänger. Wie bei jeder komplexen Software, ist aber auch beim Einsatz von TEX mit einigen Klippen und Fallgruben zu rechnen. In den hier zusammengestellten Artikeln werden einige dieser Schwierigkeiten aufgegriffen und praxisorientierte Lösungen angeboten.

Diese Sammlung von Beiträgen entstand im Wintersemester 2003/2004 im Rahmen eines Proseminars am Fachbereich Informatik der Universität Hamburg. Ein Hauptaugenmerk der Proseminars, nämlich das Abhalten eines Vortrages, kann naturgmäß in dieser schriftlichen Form nicht widergegeben werden. Die Beiträge erheben ferner nicht den Anspruch auf Vollständigkeit, noch kann eine Gewähr für die Korrektheit der Darstellungen übernommen werden. In den meisten Fällen sind die Quellen sowie Hinweise auf weiterführende Literatur angegeben.

Ich habe mich bei der Zusammenstellung der Beiträge bemüht, einige gewisse „Vereinheitlichung der Stile“ zu erzielen. Dies ist in der begrentzten mir zur Verfügung stehenden Zeit sicherlich nur in geringem Maße gelungen.

Hamburg im Februar 2004 Berndt Farwer

Inhaltsverzeichnis

Kapitel 1

Was unterscheidet TEX/LATEXvon Word & Co.?

Irina Cornies, Peer Springstübe

Vortrag vom 31.10.2003

1.1 Hintergrundinformationen

1.1.1 TEX

Der Erfinder von TEX ist Donald E. Knuth, Professor für Mathematik. Unzufriedenheit mit dem Layout seiner wissenschaftlichen Arbeiten bewegte ihn 1977 dazu, ein Programm – TEX – zu entwickeln, das seinen Vorstellungen einer gut gestalteten Dokumentation und den internationalen Standards entsprach. Dabei hatte er großen Wert auf tadellose Darstellung mathematischer Formeln gelegt.

1982, 5 Jahre später, erfolgte die Fertigstellung von TEX. TEX ist somit ein Grundprogramm, das druckreife wissenschaftliche Publikationen und Facharbeiten auf Desktop-Computersystemen erzeugte.

1.1.2 LATEX

LATEX als ein Aufsatz für TEX ist eine Entwicklung von Leslie Lamport. LATEX beinhaltet alle Befehle von TEX und darüber hinaus eine ganze Reihe an vordefinierten Layouts, was die Arbeit mit TEX erheblich erleichtert.

Seit 15 Jahren ist TEX in seinem Funktionsumfang gleich geblieben. Die intelligenten Schnittstellen ermöglichen eine Erweiterung um zusätzliche Pakete (vergleichbar mit Plug-Ins). Auf diese Weise konnten in den letzten Jahren ausschließlich Fehler im TEX-System ausgemerzt werden, was auch dessen Stabilität erklärt.

1.2 Ein Beispiel

Ein erster kleiner Einblick, wie ein LATEX-Dokument aussieht:

1.2.1 Eingabe:

```
\documentclass{article}
\begin{document}
   Ein Beispiel mit einer mathematischen Formel
\begin{equation}
c = \sqrt{ a^{2}+b^{2} }
\end{equation}
\end{document}
```

1.2.2 Ausgabe:

Ein Beispiel mit einer mathematischen Formel

$$c = \sqrt{a^2 + b^2} \tag{1.1}$$

1.2.3 Einige Erläuterungen zum Quelltext:

Es wird erst eine Dokumentklasse Artikel deklariert. In der nächsten Zeile beginnt das Dokument gefolgt mit einem Text. Mathematische Umgebung fängt an, in der Wurzel gezogen wird. Mathematische Umgebung und das Dokument werden geschlossen.

1.3 Vorteile von LATEX

Im Gegensatz zu den meisten anderen verbreiteten Programmen zum Erstellen von Dokumenten übernimmt LATEX die Rolle eines Typographen. Der Autor wählt lediglich die Art des zu erstellenden Dokumentes aus und gib den Inhalt nebst der gedanklichen Gliederung (Kapitel, Absätze etc) an; Schrifarten und Anordnung des Textes werden ihm vom LATEX-Programm abgenommen. Hierdurch kann auch eine Person, die sich mit Typographie nicht auskennt, gut lesbare Dokumente unterschiedlichster Art erstellen.

LATEX ist auf ein langbewährtes Grundsystem aufgebaut, das seit Jahren, abgesehen von Fehlerbehebung, nicht mehr verändert wurde. Hierin begründet sich eine hohe Stabilität des Programmes (es ist quasi bugfrei). Außerdem ist es mitlerweile auf nahezu jeder Plattform kostenlos verfügbar und es steht eine große Zahl an zusätzlichen Paketen zur Auswahl.

LATEX-Quelldateien sind einfache Textdateien, und es ist möglich, jede Funktionalität allein mit ASCII-Zeichen zu nutzen. Hieraus ergibt sich maximale Portabilität von LATEX-Dokumenten, nicht nur auf unterschiedliche Betriebssysteme, sondern auch in unterschiedliche Länder mit verschiedenen Zeichensätzen (z.B. Umlauten). Die Quelldateien können mit jedem beliebigen Texteditor bearbeitet werden, so dass man nicht an die

Möglichkeiten eines Programmes gebunden ist, sondern seinen Lieblingseditor verwenden kann und auch andere Werkzeuge, wie beispielsweise grep, auf die Dateien anwenden kann.

Dokumente können einfach auf mehrere Dateien aufgeteilt werden, indem man einfach in einer Datei angibt, wo eine andere eingebunden werden soll. Somit kann auch bei großen Projekten die Übersicht gewahrt werden.

Aus vielen dieser Vorteile ergibt sich noch eine weiterer: Durch das Aufteilen in kleine Teile, das einfache Dateiformat und das vor Jahren entwickelte Grundsystem, ergibt sich eine geringe Anforderung an Camputerhardware, so dass LaTeX auch auf alten oder langsamen Systemen benutzt werden kann.

Bei Alldem bietet LaTeX viele Funktionen, die man beim Erstellen von Dokumenten benötigt. Ein zentrales Element bildet die einfache Integration auch komplexer mathematischer Formeln, da dies eine der Grundideen von TeX, auf dem LaTeX aufgebaut ist, war.

Da dem LaTeX-System der Aufbau des Dokuments mitgeteilt wird, kann es dem Autor Aufgaben abnehmen, wie zum Beispiel das Erstellen eines Inhaltsverzeichnises, das Durchnummerieren von Tabellen oder Graphiken oder Seitenangaben bei Querverweisen. Diese werden beim späteren Verändern des Dokuments automatisch aktualisiert, sodass Nummerierungsfehler verhindert werden.

1.4 Nachteile von LaTeX

Auf der anderen Seite stehen auch ein paar Nachteile. Für viele Leute ist es sicher ein ungewohntes System, Text-Quelldateien zu schreiben und dann zu compilieren, wenn sie von einem graphischen System umsteigen.

Es ist relativ schwierig, mit LaTeX Dokumente zu erstellen, die stark von den (zahlreichen) Vorlagen abweichen.

1.5 Was benötige ich, um LaTeX zu benutzen?

Um ein Dokument mit LaTeX zu erstellen, benötigt man mehrere Programme:

- einen Texteditor, mit dem man die Quelldateien erstellt
  (z.B. vi, emacs, notepad...)
- LaTeX/TeX
  die eigentlichen Programme, die Textdateien in eine dvi-Datei umwandeln

  TeX erzeugt das gesetzte Schriftstück in einem Ausgabeformat namens „DeVice Independent“ (DVI). Die erzeugte Datei enthält die Positionierungsanweisungen,

Verweise auf Schriften, Lettern und Linien in einem Format, das unabhängig vom Ausgabegerät. Diese Datei muß zur Ansicht oder zum Druck anschließend in das Format des jeweiligen Ausgabegerätes gewandelt werden.

- ein Programm, um sich die dvi-Datei anzuschauen, damit man sieht, wie das Dokument compiliert aussieht
  (z.B. xdvi für ein X-Window System)

- wenn man das Dokument auf Papier haben will, benötigt man noch ein Programm, das die dvi-Datei ausdruckt
  (z.B. dvihp)

Alternativ hierzu gibt es auch graphische Oberflächen (z.B. kile unter Linux), die im Hintergrund diese Programme aufrufen, und Umsteigern von anderen Programmen den Einstieg in LATEX erleichtern können.

1.6 Wie benutze ich LATEX?

Ein einfaches Beispiel, wie man an einer einfachen Linuxkonsole ein LATEX-Dokument erstellt:

- `prompt$ vi bsp.tex`
  (Erstellen des Quelldokuments)

- `prompt$ latex bsp.tex`
  (Konvertieren in geräteunabhängige dvi-Datei)

- `prompt$ xdvi bsp.dvi`
  (Ansehen des Dokumentes)

1.7 Wo bekomme ich TEX/LATEX?

- Die internationale TEX-User-Group namens TUG
  `http://www.tug.org`

- Die deutschsprachige User-Group namens DANTE oder FAQ TEX
  `http://www.dante.de`
  `http://www.dante.de/faq/de-tex-faq/`

1.8 Hilfe zu TEX/LATEX

- Kochbuch für LATEX
  `http://www.uni-giessen.de/hrz/tex/cookbook/first.html`
- LATEX – eine Einführung und ein bisschen mehr
  `ftp://ftp.fernuni-hagen.de/pub/pdf/`
  `urz-broschueren/broschueren/a0260003.pdf`
- LATEX – Fortgeschrittene Anwendungen (FernUniversität Hagen)
  `ftp://ftp.fernuni-hagen.de/pub/pdf/`
  `urz-broschueren/broschueren/a0279510.pdf`

Kapitel 2

Logische Dokumentenstruktur und logische LaTeX-Dokumente

Mirko Heger

Vortrag vom 7.11.2003

[Hier fehlt leider noch ein Beitrag]

Kapitel 3

Erweiterte Dokumentenklassen und Packages

Ilker Daricili, Salim Rechenberg

Vortrag vom 21.11.2003

3.1 Was ist KOMA-Script?

Das KOMA-Script ist eine Sammlung von Klassen, die äquivalent zu den LaTeX Standardklassen angelegt wurden. So gibt es jeweils zu den Klassen `book`, `report`, `article` und `letter` angepasste Varianten. Ziel ist es, die für die amerikanische Typographie erstellten Klassen an die in Europa verbreiteten Konventionen anzupassen. Ein Beispiel dafür ist das DIN Papierformat, welches in den KOMA-Klassen voreingestellt ist.

Ihre Namen sind durch das Präfix „scr“ und den ggf. auf maximal 5 Zeichen gekürzten Namen der Standardklassen zusammengesetzt, beispielsweise „scrbook“ oder „scrartcl“. Generell sind die meisten Optionen der Standardklassen, genauso oder ähnlich auch in den KOMA-Klassen zu finden.

Geschichtliches zu KOMA-Script

Wie bereits angedeutet sind die KOMA-Klassen aus dem Wunsch nach an europäische Konventionen angepasste Dokumente entstanden. Frank Neuman hat Anfang der 90er festgestellt, dass die vorhandenen Dokumentenstile[1] nicht seinen „europäischen“ Ansprüchen genügten. Er machte sich dann daran, eine Stilfamilie zu erstellen, die den Regeln der europäischen Typographie entsprach. In den folgenden Jahren wurde „Script“ zusammen mit Markus Kohm weiterentwickelt. Nachdem 1994 LaTeX2ϵ herausgekommen ist, wurde Script weiterentwickelt und kam wenig später dann unter dem neuen Namen „KOMA-Script“ heraus. Von da an entwickelte es sich stetig weiter, so dass es heute weit über Deutschland, sogar über Europa hinaus bekannt ist und auch eingesetzt wird. Seine Zielsetzung ist deshalb ein wenig verändert: nun geht es um „Flexibilisierung durch

[1] damals unterteilte man nicht in Klassen und Pakete sondern alles waren „Stile“

Variabilisierung“. Das bedeutet, dass dem Benutzer mehr Möglichkeiten zum Verändern des Erscheinungsbilds gegeben werden soll.

3.2 KOMA-Script – Die wichtigsten Optionen

3.2.1 Satzspiegel – Grundlegendes zum Seitenlayout

Jede Textseite besteht aus Rändern, dem Textkörper, einem Kopf- und einem Fußbereich. Die Aufteilung und die Anordnung dieser Bereiche nennt man Satzspiegel. Dieser ist durch Richtwerte, die ein angenehmes Lesen ermöglichen, festgelegt. Dazu gehört zum Beispiel, dass die Seitenverhältnisse des Textkörpers, den der Seite entsprechen. Weil es dabei auf den sichtbaren Bereich ankommt, muss eine Bindekorrektur durchgeführt werden. Der Rand, an dem gebunden wird, muss entsprechend vergrößert werden, damit auch im nachhinein das Verhältnis noch stimmt. Dies und auch das Ermitteln des Satzspiegels, wird vom Typearea-Paket erledigt. Mit dem Befehl `\usepackage[BCOR10mm]{typearea}` würde man LaTeXveranlassen, den Rand auf der gebundenen Seite um 10mm größer zu machen, weil diese durch das Binden verloren gehen. `typearea` wird bei allen KOMA-Klassen automatisch geladen und man kann die Bindekorrektur einfach als Option der Dokumentenklasse angeben, sie wird dann automatisch an `typearea` weitergegeben. Das sieht dann zum Beispiel so aus:
`\documentclass[a4paper,BCOR8.25mm]{scrreprt}`

Eine weitere Option, die man bei `typearea` angeben kann, und somit auch bei allen KOMA-Klassen in der eben gezeigten Form, ist der DIV-Wert. Dieser Faktor gibt an, wie groß der Text- und Randbereich sein soll. Ein kleiner Wert, zum Beispiel 4 (das ist der kleinste), produziert einen sehr kleinen Textbereich mit sehr großen Rändern. Bei Werten über 15, ist schon nur noch sehr wenig Rand zu sehen. Extrem große Werte können dazu führen, dass die Kopfzeile außerhalb der Seite liegt, hier ist also Vorsicht angebracht. Bei DIN-A4 Seiten sind standardmäßig günstige DIV-Werte eingetragen. Für alle anderen Papiergrößen kann man mit Hilfe von `DIVcalc` einen günstigen Wert berechnen lassen. `DIVclassic` benutzt ein Wert, wie er im Mittelalter beim Buchdruck verwendet wurde.

Mit dem Befehl `\typearea[current]{calc}` kann man, wenn man zum Beispiel im Laufe seines Dokuments die Schriftart wechselt, erneut einen passenden Satzspiegel berechnen oder auch explizit angeben. Der optionale Parameter steht in dem Befehl für die Bindekorrektur. Man hätte also die Möglichkeit auch diese im Laufe des Dokuments anzupassen.

Auch wenn man den Zeilenabstand (auch Durchschuss genannt) verändert, sollte man den `\typearea` Befehl neu aufrufen um den Satzspiegel neu berechnen zu lassen. Der Standardwert für den Zeilenabstand ist bei 10pt Schriftgröße 2pt Durchschuss (1.2 Zeilenabstand). Möchte man ihn verändern, tut man das mit dem Befehl `\linespread{Dehnfaktor}`. Der Dehnfaktor ist der Wert mit dem der Standard (1.2) multipliziert werden muss, um den gewünschten Zeilenabstand zu erhalten. Um von 1.2

auf 2.0 zu kommen muss man also `\linespread{1.66}` eingeben. Um die Veränderung anzuwenden muss man noch einmal `\selectfont` aufrufen, eigentlich ist es jedoch so gedacht, dass man diese ganzen Einstellungen *einmal* am Anfang tätigt.

Mit den eben genannten Optionen hat man die wichtigsten Funktionen des `typearea` abgedeckt, was jedoch nicht bedeutet, dass es nicht noch mehr gibt. Es gibt noch weitere Einstellungen, wie zum Beispiel, ob Kopf- und Fußzeile zum Rand gehören und wie viel Platz für Notizen im Rand bereitgehalten wird.

3.2.2 Hauptklassen

Die KOMA-Scripte beinhalten eine Vielzahl der Optionen, die auch bei den Standardklassen vorhanden sind, teilweise abgeändert, teilweise genau übernommen. Im Folgenden wird eine Auswahl von ihnen vorgestellt. Dabei wird versucht einen Kompromiss aus Optionen die neu hinzugekommen sind, welchen die verändert sind und welchen die sehr wichtig sind zu machen.

Papierformat

Im Gegensatz zu den Standardklassen, welche auf das amerikanische Papierformat eingestellt sind, ist bei den KOMA-Klassen das A4 Format voreingestellt. Man kann bei der Dokumentenklasse zum Beispiel die Option `a5paper` angeben wenn man A5 Papier verwenden möchte. Unterstützt werden außer dem DIN A Format noch die DIN B, C, D Formate sowie einige andere, welche dann auch mit einem Satzspiegel versehen werden können, der den europäischen Konventionen entspricht.

Layout

Die Optionen `oneside`, `twoside` sowie `onecolumn` und `twocolumn` sind wie in den Standardklassen zu gebrauchen. Das zweiseitige Layout wird in der `scrbook`-Klasse standardmäßig ausgewählt, in den anderen `oneside`. Mit `twoside` wird auch automatisch ein doppelseitiger Satzspiegel gewählt, so dass zum Beispiel die Seitenzahlen jeweils außen sind. Auch in den KOMA-Klassen kann man sich überlegen, ob man ein neues Kapitel immer auf der rechten Seite beginnen möchte (`openright`) oder nicht (`openany`).

Möchte man in einem Buch oder einem Report die Kapitelüberschriften mit dem Wort „Kapitel“ gefolgt von dem Namen des Kapitels haben, dann benutzt man die Option `chapterprefix`. Will man aber nur die Kapitelnummer ohne „Kapitel“, dennoch gefolgt von dem Namen des Kapitels, dann benutzt man die voreingestellte Option `nochapterprefix`. Zusammen mit der *-Variante des `\chapter`-Befehls, kann man so alle Möglichkeiten abdecken.

In den KOMA-Klassen werden die Überschriften von Tabellen und Abbildungsbezeichnungen standardmäßig zentriert, solange sie die Länge einer Zeile nicht überschreiten.

Mit der Option `noonelinecaption` kann man diese Besonderheit ausschalten, alle Überschriften werden nun ganz normal linksbündig im Blocksatz gesetzt.

3.2.3 Schriftgröße

Generell kann man im KOMA-Script wie auch in den Standardklassen die Schriftgrößen 10pt, 11pt und 12pt wählen, jedoch ist es auch möglich, sich selber welche zu definieren.

Die Größe der Überschriften sind in den KOMA- wie auch in den Standardklassen relativ groß gewählt. Das KOMA-Script bietet nun die Möglichkeit mit den Optionen `smallheadings`, `normalheadings` und `bigheadings` die Größe der Überschriften anzupassen, wobei `bigheadings` der Standard ist, man kann also nur kleiner, nicht *noch* größer machen.

3.2.4 Inhaltsverzeichnis

Bezüglich des Inhaltsverzeichnises bietet das KOMA-Script einige Erweiterungsmöglichkeiten. So ist es zum Beispiel möglich Index und Literaturverzeichnis mit in das Inhaltsverzeichnis aufzunehmen. Ferner bieten sowohl die Standard- als auch die KOMA-Klassen die Möglichkeit, die Tiefe des Inhaltsverzeichnis einzustellen. Konkret kann man mit dem Befehl `\setcounter{tocdepth}{Wert}` einstellen, wie viele Ebenen angezeigt werden sollen. Setzt man die Variable tocdepth auf diese Weise auf den Wert 1, wird zum Beispiel in einer book-Klasse parts (Teile,Abschnitte), chapter und sections angezeigt. Bei 2 würden dann die subsections auch angezeigt und bei größeren Werten dann entsprechend die subsubsections und so weiter. In der Artikel-Klasse gibt es keine parts und chapters, auch hier steht 1 für „sections anzeigen“. Wenn man den absoluten Wert nicht kennt, kann man den Wert der Variable auch relativ verändern, zum Beispiel könnte man `addtocounter{tocdepth}{-1}` benutzen, um eine Ebene weniger angezeigt zu bekommen.

3.2.5 Formatierung

Die Optionen `tablecaptionbelow` und `tablecaptionabove` werden im KOMA-Script verwendet, um den Abstand zwischen Tabellenüberschrift/-unterschrift und der eigentlichen Tabelle anzupassen.

Wenn man mit `longtable` arbeitet und verhindern möchte, dass die Optionen die man gesetzt hat auch für diese gelten, erreicht man das mit der Option `origlongtable`.

Eine nützliche Option ist `draft`. Damit versetzt man LaTeXin einen Entwurfsmodus. Es werden alle zu vollen Boxen mit einem schwarzen Kästchen in der Ausgabe markiert, so dass man sie gezielt nachbessern kann. Auch andere Pakete überprüfen diese Option, beispielsweise `graphics` veranlasst, dass im Entwicklungsmodus die Bilder durch Rahmen

dargestellt werden, anstatt des Bildes selber. Mit `final` oder indem man `draft` einfach weglässt, erhält man wieder die gewohnte Ausgabe.

3.3 weitere Pakete

Einbindung in LaTeX

Im Folgenden werden einige Pakete vorgestellt. Generell werden sie mit `\usepackage[optionen]{package}` geladen. Sollen mehrere Optionen übergeben werden, listet man sie durch Kommata getrennt auf. Die letzte Option einer solchen Aufzählung ist die, die dann erst einmal aktiv ist.

3.3.1 babel – verschiedene Sprachunterstützungen

Das Paket `babel` ist entwickelt worden um LaTeXauch für nicht amerikanischsprachige Einsatzgebiete erschließbar zu machen. Es stellt zum Beispiel die Regeln für die Silbentrennung oder die Übersetzung von chapter auf die ausgewählte Sprache um. Es unterstützt über 40 Sprachen, teilweise mit mehreren Dialekten. Man lädt es mit `\usepackage[sprache]{babel}` ein. Eine wichtige Erweiterung ist, dass mit Hilfe von `babel` innerhalb eines Dokumentes die Sprache geändert werden kann. Hier zum Beispiel ist die Sprache auf `ngerman` (neue deutsche Rechtschreibung) gestellt. Mit dem Befehl `\languagename` kann man auf die aktuell verwendete Sprache ausgeben. Wir verwenden zur Zeit ngerman.

`\today` gibt folgendes aus: 20. Februar 2004

Wechselt man nun die Sprache mit `\selectlanguage{polish}` auf polnisch, erhält man mit `\languagename` die nun eingestellte Sprache: polish.

`\today` gibt *jetzt* folgendes aus: 20 lutego 2004

Man sieht also sehr schön, dass nun nicht mehr das deutsche Datum, sondern das Datum in polnischer Sprache dargestellt wird. Andere „feste“ Begriffe, wie „Tabelle“ werden ebenfalls übersetzt.

Außerdem haben die meisten Sprachen wie angedeutet eigene Silbentrennregeln. Möchte man die alte deutsche Rechtschreibung verwenden, benutzt man `german`, für die neue `ngerman`. Wenn man bei german möchte, dass Wörter wie Schiffahrt (alte Rechtschreibung) korrekt getrennt werden, muss man sie `Schi"ffahrt` schreiben, dann fügt LaTeX das gegebenenfalls benötigte f hinzu. Bei `ngerman` entfällt das, man sollte es hier nicht benutzen. Eine weitere Möglichkeit die Sprache zu verändern ist folgende: `\begin{otherlanguage}{german}` Jetzt befindet man sich in der Umgebung `german` und

nicht mehr in `ngerman`, bis man sie mit `\end{otherlanguage}` wieder verlässt. Hier gibt es wieder eine recht nützliche *-Variante, in der dann die Namen wie z.B. „Tabelle“ weiterhin wie außerhalb der Umgebung benannt sind. Sehr praktisch wenn man einige Seiten in einer anderen Sprache schreiben möchte, aber eben weiterhin einheitlich „Tabelle“ über jeder Tabelle stehen haben möchte.

In den neueren LaTeXVersionen ist es auch möglich, die Sprachen als Optionen der Dokumentenklasse anzugeben, man kann also auch schreiben: `\documentclass[german,polish,ngerman]{scrbook}`

3.3.2 inputenc – Eingabekodierung

Mit Hilfe des Pakets `inputenc` hat man die Möglichkeit, die Eingabecodierung passend zu seinem System einzustellen. Ein Mac-Benutzer kann einen entsprechenden Zeichensatz wählen und so dann ebenfalls die Umlaute direkt eingeben. Mit Hilfe von folgendem Befehl kann man das machen: `\usepackage[latin9]{inputenc}` oder alternativ andere Zeichensätze als latin9, z.B. applemac.

3.3.3 theorem – Theoreme und Definitionen

Lemma 1
Das Package `theorem` *ermöglicht einen flexibleren Umgang mit Theoremen. So ist es im Gegensatz zum Standardbefehl* `\newtheorem` *vielseitiger. Es bietet die Möglichkeit, die Schriftart auszuwählen und zwar sowohl für den Kopf als auch den Körper. Außerdem lässt es einen entscheiden, ob man eine Nummer vor und hinter der Bezeichnung stehen haben möchte. Man definiert eine Theorem im Dokumentenkopf wie folgt:*

```
\usepackage{theorem}
\theoremstyle{break}
\newtheorem{Lem}{Lemma}
```

2 Definition Diesmal wird eine alternative Theorem Umgebung verwendet. Sie wurde wie folgt erstellt:

```
\theorembodyfont{\upshape}
\theoremstyle{change}
\newtheorem{Def}[Lem]{Definition}
```

Zu beachten bleibt hierbei, der optionale Parameter [Lem], welcher dafür sorgt, dass beide die gleiche Zählervariable besitzen. Ob das im Fall von Definitionen und Lemmas sinnvoll ist, sei dahingestellt. Der Befehl `\theorembodyfont{\upshape}` setzt die Schriftart der danach folgenden newtheorems auf `upshape`. Das Paket kennt die folgenden theoremstyles:

plain, break, marginbreak, changebreak, change, margin. Sie stehen für verschiedene Kombinationen, welche sich aus einem Zeilenumbruch hinter dem Theoremkopf (durch `break` erreicht), der Positionierung der Zahl vor bzw. hinter dem Kopf und der Möglichkeit, die Zahl in den Rand hinein zu rücken ergeben (`margin`).

3.3.4 longtable – Tabellen über mehrere Seiten

Die `longtable`-Erweiterung ermöglicht andere Varianten bei der Erstellung von Tabellen. Die Standardbefehle packen die Tabellen in eine Art Box, welche nicht gebrochen werden kann. Für TEXbesteht dann kein Unterschied mehr zu einem riesig großen Zeichen. Das hat Vor- und Nachteile. Ein Vorteil, welcher auch gegenüber `longtable` besteht, ist dass sie frei ausgerichtet werden können, zum Beispiel mit der `center` Umgebung. Der sehr gravierende Nachteil ist, dass sich keine Tabellen erstellen lassen, die größer als eine Seite sind, bzw. dass sie dann ggf. von Hand gebrochen werden müssen. Dies widerspricht nicht nur der Grundidee von LATEX, es ist auch lästig, wenn sich bei großen Projekten im Laufe der Zeit die Position verschiebt und so immer wieder der Umbruch angepasst werden muss. Außerdem bietet `longtable` Möglichkeiten, die Kopf- und Fußzeilen der Tabellen zu verändern, insbesondere was auf jeder Seite als Kopfzeile stehen soll. Dies geschieht, indem man das, was auf der ersten Seite als erster Kopf der Tabelle erscheinen soll, ganz normal hinschreibt, wie man es auch mit den anderen Zeilen tut, gefolgt von dem Befehl `\endfirsthead`. Das sieht dann so aus:

```
\caption{Name}\\
\hline
Eigenschaft & KOMA-Script  &Standard\\ \hline \hline
\endfirsthead
```

Möchte man, dass dieser Kopf oder ein anderer Kopf auf jeder Seite am Anfang der Tabelle steht, benutzt man `\endhead`. Analog dazu, verwendet man `\endfoot` um auf jeder Seite die letzte Zeile so, wie davor beschrieben, darzustellen. Möchte man auf der allerletzten Seite ganz unten wiederum noch was anderes stehen haben, kann man dafür `\endlastfoot` benutzen. Wenn man einige oder alle dieser vier Befehle eingefügt hat, fängt man an, den eigentlichen Inhalt der Tabelle hinzuschreiben. LATEXsorgt dann dafür, dass auf jeder Seite die Kopf- und Fußzeilen entsprechend vorhanden sind.

Tabelle 3.1: diese Tabelle zeigt longtable

Code	Ausgabe
`\begin{longtable}{|l|c|}`	eine zweispaltige Tabelle wird erstellt.
`\caption{Name}`	erstellt eine Überschrift, mit dem angegebenen Namen

Tabelle 3.1: - Fortsetzung

Code	Ausgabe
`\multicolumn{2}{|c|}{text}`	fügt zwei Zellen zu einer zusammen...
so das so etwas hier entsteht ...	
oder aber so etwas wie hier...	

Nicht unerwähnt bleiben sollte die Tatsache, dass `longtable` mit den Standardtabellen die gleiche Zählervariable teilt, so dass wenn man beide Varianten benutzt, die Nummerierung trotzdem korrekt ist. Einen ähnlichen Funktionsumfang bietet auch das Paket `supertab`.

3.3.5 color

Das Paket `color` ermöglicht die Benutzung und Neudefinition von Farben. Es wird wie jedes andere Paket eingebunden. Seine eigenen Farben definiert man sich wie folgt: `\definecolor{name}{modell}{farbspezifikationen}` wobei man den Namen frei wählen kann. Das Farbmodell ist entweder `rgb`, `cmyk`, `gray` oder `named`. RGB besteht aus 3 Zahlenwerten von 0 bis 1, die für rot, grün, blau stehen. CMYK ist ähnlich aufgebaut, nur das die Werte für cyan, magenta, gelb und schwarz stehen. Der `gray` Wert ist eine Zahl von 0 bis 1. Und bei `named` kann man aus einer Liste von vordefinierten Namen einen auswählen, zum Beispiel JungelGreen. Wenn man sich dann seine eigene Farbe definiert: `\definecolor{komischefarbe}{rgb}{0.53,0.48,0.18}` kann man anschließend eben diese mit `\textcolor{komischefarbe}{Text}` auswählen, und erhält dann diese hier. Normalerweise gibt man die RGB-Werte als Zahl von 0 bis 255 an. Will man diese Werte jetzt umrechnen, teilt man den Wert den man haben möchte durch 255. Eigentlich definiert man alle verwendeten Farben im Dokumentenkopf, es besteht trotzdem auch die Möglichkeit, die Farben direkt einzugeben.

`\color[rgb]{0.53,0.48,0.18}` setzt also ohne das es vorher definiert wurde, die Farbe der aktuellen Umgebung auf den angegebenen Farbton. Um die Hintergrundfarbe auf rot zu setzen benutzt man zum Beispiel `\pagecolor{red}`. Wenn man den Text wie hier darstellen will, macht man das auf die gleiche Art wie wenn man die Textfarbe verändern will, nämlich mit `\colorbox{yellow}{text}`. Man kann auch eine Box erzeugen, die einen farbigen Rahmen hat, hierfür benutzt man `\fcolorbox{farbe1}{farbe2}{text}`, wobei `farbe1` die Farbe für den Rahmen und `farbe2` die Füllfarbe ist. In beiden Fällen kann man, wie auch bei der Textfarbe, auch direkt die Farbwerte spezifizieren, muss dann natürlich als ersten Parameter das Farbmodell übergeben.

TEXwurde nicht für den Umgang mit verschiedenen Farben entwickelt und aus diesem Grund werden viele Aufgaben von den Treibern übernommen, was jedoch nicht bei allen

Treibern gleich gut klappt. Bei einigen gibt es Probleme, dass die Farbe bei einem Seitenwechsel verloren geht oder es kann dazu kommen, dass Absätze bei denen ein Farbwechsel stattfindet nicht wie gewünscht angezeigt werden.

Kapitel 4

Mathematische Formeln in LaTeX

Sören Glimm, Kai Kudlek

Vortrag vom 14.11.2003

Mathematische Formeln werden in LaTeX durch einen die Formeln beschreibenden Text erzeugt. Hierzu muß LaTeX einen Hinweis bekommen, dass der folgende Text als Formeln zu interpretieren ist. Die folgenden mathematischen Umgebungen sind so ein Hinweis und versetzen LaTeX in den *mathematischen Bearbeitungsmodus*.

4.1 Mathematische Umgebungen

Mathematische Formeln können in LaTeX innerhalb von Textzeilen auftreten, oder abgesetzt erscheinen.

4.1.1 Textformeln

Textformeln werden durch folgende Befehle erzeugt:

- `\begin{math}` *Formeltext* `\end\{math}`
- `\(` *Formeltext* `\)`
- `$` *Formeltext* `$`

Ein Beispiel hierzu:
Diese Formel $(a + b)^2 = a^2 + 2ab + b^2$ tritt interhalb des Textes auf.

4.1.2 abgesetzte Formeln

Abgesetzte Formeln können mit fortlaufender Formelnummer, oder ohne erzeugt werden:

- ohne Formelnummer:

  - `\begin{displaymath}` *Formeltext* `\end{displaymath}`
  - `\[` *Formeltext* `\]`

  Beispiel:

$$2\sum_{i=1}^{n} a_i \int_a^b f_i(x)g_i(x)\,dx$$

- mit Formelnummer:

  - `\begin{equation}` *Formeltext* `\end{equation}`

  Beispiel:

$$2\sum_{i=1}^{n} a_i \int_a^b f_i(x)g_i(x)\,dx \tag{4.1}$$

Abgesetzte Formeln werden standardmäßig horizontal zentriert und eine evtl. Formelnummer erscheint rechtsbündig. Durch das Verwenden der Dokumentenklassenoption **fleqn** werden die Formeln linksbündig angeordnet.

Die Einrücktiefe kann jederzeit mit:

`\setlength\{\mathindent}{`*Einrücktiefe*`}` geändert werden.

Linksbündiges Anordnen der Formelnummern erreicht man mit der Dokumentenklassenoption **leqno**.

4.1.3 Formelgruppen

Zum Erzeugen von Formelgruppen stehen abschließend noch folgende Umbebungen zur Verfügung:

- `\begin{eqnarray}` *Formeltext* `\end{eqnarray}`
- `\begin{eqnarray*}` *Formeltext* `\end{eqnarray*}`

Bei der *-Form entfallen die fortlaufend erzeugten Formelnummern.

4.2 amsmath.sty

AMS (American Mathematical Society) hat mit ihrem Macropaket **amsmath** das Formellayout von LaTeX weiter verbessert.

Eingebunden wird das Paket mit `\usepackage[opt]{amsmath}`. Auf die Unterschiede im einzelnen wird an den entsprechenden Stellen eingegangen.

4.3 Gerahmte Formeln

Um mathematische Formeln zur besonderen Hervorhebung einzurahmen, sind keine neuen Konstruktionselemente erforderlich. Für die Einrahmung von Textformel $\boxed{a+b}$ wird die Textformel einfach in \fbox{$ a+b $} eingebettet.

Abgesetzte, gerahmte Formeln erfordern das „Einschachteln“ der abgesetzten Formeln in eine `\parbox` oder minipage-Umgebung geeigneter Breite, die dann in den `\fbox`-Befehl gepackt wird.

`\fbox{\parbox{5cm}{\[\frac{a+b}{2}\]}}` erzeugt z.B. folgende Formel:

$$\frac{a+b}{2}$$

Lästig ist in diesem Fall das „Abschätzen“ der Breite der Formel. Die richtige Größenangabe für die Box läßt sich oft nur durch Probieren herrausfinden. Geeigneter erscheint da folgender Befehl für das Erzeugen von gerahmten Formeln, ohne Größenangabe für die Umrandung.

- \[\fbox{$ \displaystyle *Formeltext* $} \]

$$\boxed{\frac{a+b}{2}}$$

  - \setlength{\fboxrule}{*Rahmenstärke*}
  - \setlength{\fboxsep}{*Rahmenabstand*}

4.4 Konstanten und Variablen

In Formeln auftretende Zahlen sind Konstanten. Einfache Variablen sind einzelne Buchstaben. Es ist weltweiter Standard, daß in mathemaschinen Formeln Konstanten in der Schriftart Roman und Variablen in der Schriftart Italic gesetzt werden. Dies wird im mathematischen Modus von LaTeX automatisch berücksichtig. Die Abstände zwischen Variablen, Konstanten und eventuellen Verknüpfungszeichen, wie +,-,= und anderen, werden automatisch gewählt.

4.5 Hoch- und Tiefstellung von Zeichen

Mathematische Formeln enthalten häufig Exponenten oder Indizes. Diese hoch- bzw. tiefgestellten Zeichen erscheinen in kleinerer Schrift als das Zeichen, an dem sie hoch- oder tiefgestellt sind.

Hoch- und Tiefstellungen lassen sich wie folgt erzeugen:

- `^` bewirkt das Hochstellen von Zeichen

  x^2 `x^2`
  x^{2n} `x^{2n}`

- `_` bewirkt das Tiefstellen von Zeichen

  x_i `x_i`
  x_{id} `x_{id}`

- eine Kombination aus Beidem

  $B^{y_j^2}_{n^3 m_p}$ `B^{y_j^2}_{n^3m_p}`

4.6 Brüche und Wurzeln

4.6.1 Brüche

Für kurze Brüche, insbesondere in Textformeln, wird als Bruchzeichen meistens der / verwendet. `$(n+m)/2$` gibt $(n + m)/2$. Für umfangreichere Brüche steht der Befehl `\frac{`*Zähler*`}{`*Nenner*`}` zur Verfügung. Dieser erzeugt einen Bruchstrich von der Breite der jeweils längsten Komponennte.

Beispiel:

$$\frac{(a^2 + b^3 + c^4)^2}{5c}$$

Brüche können beliebig in einander verschachtelt werden.

4.6.2 Wurzeln

Wurzelausdrücke werden mit dem Befehl `\sqrt[n]{arg}` erzeugt. Ohne den optionalen Parameter n wird die Standardform der Quadratwurzel erzeugt.

Beispiel:

`\sqrt[5]{\sqrt[4]{a+b*(c*\sqrt[2]{9})}}` erzeugt:

$$\sqrt[5]{\sqrt[4]{a + b * (c * \sqrt[2]{9})}}$$

4.7 AMS-Brüche

Neben dem LaTeX-Standardbefehl `\frac{Zähler}{Nenner}` stellt `amsmath.sty` zusätzlich die Bruchbefehle `\dfrac` und `\tfrac` mit der gleichen Syntax zur Verfügung. Diese

Zusatzbefehle sind nichts weiter als `\frac`-Befehle denen `\displaystyle` bzw. `\textstyle` vorangestellt wird.

Beispiel:

- \tfrac{*Zähler*}{*Nenner*} (textstyle fraction)

$$\frac{25}{k}\sin x \qquad \tfrac{25}{k}\sin x$$

- \dfrac {*Zähler*}{*Nenner*} (displaystyle fraction)

$$\sqrt{\frac{25}{k}\sin x} \quad \sqrt{\dfrac{25}{k}\sin x}$$

4.8 AMS-Binominal-Ausdrücke

$$\binom{n+1}{k}$$

Binominalausdrücke entsprechen optisch einem Bruch, der von einem großen, runden Klammerpaar umschlossen ist und bei dem der Bruchstrich entfernt wird. Der Grundbefehl dafür lautet: `\binom{`*oben*`}{`*unten*`}`.

Analog zu den Brüchen gibt es auch hier `\dbinom` und `\tbinom`.

4.9 Mathematische Symbole

Mathematischer Text kennt eine große Vielfalt von Symbolen. Nur ganz wenige davon stehen direkt auf der Tastatur zur Verfügung. LaTeX stellt nahezu jedes erdenkliche, mathematische Symbol unter einem Symbolnamen, dem ein \ vorangestellt ist, bereit.

4.9.1 Griechische Buchstaben

Kleinbuchstaben:

α	`\alpha`	θ	`\theta`	o	`o`	τ	`\tau`
β	`\beta`	ϑ	`\vartheta`	π	`\pi`	υ	`\upsilon`
γ	`\gamma`	ι	`\iota`	ϖ	`\varpi`	ϕ	`\phi`
δ	`\delta`	κ	`\kappa`	ρ	`\rho`	φ	`\varphi`
ϵ	`\epsilon`	λ	`\lambda`	ϱ	`\varrho`	χ	`\chi`
ε	`\varepsilon`	μ	`\mu`	σ	`\sigma`	ψ	`\psi`
ζ	`\zeta`	ν	`\mu`	ς	`\varsigma`	ω	`\omega`
η	`\eta`	ξ	`\xi`				

Großbuchstaben:

Γ	`\Gamma`	Λ	`\Lambda`	Σ	`\Sigma`	Ψ	`\Psi`
Δ	`\Delta`	Ξ	`\Xi`	Υ	`\Upsilon`	Ω	`\Omega`
Θ	`\Theta`	Π	`\Pi`	Φ	`\Phi`		

4.9.2 Symbole

Symbolnamen sind nicht immer „intuitiv“ , können aber in entsprechenden Tabellen nachgeschlagen werden.

Hier nun ein paar Beispiele:

$\rightarrow$	`\rightarrow`	$\Rightarrow$	`\Rightarrow`	$\rightleftharpoons$	`\rightleftharpoons`
$\subset$	`\subset`	$\supset$	`\supset`	$\in$	`\in`

4.9.3 Kalligraphische Buchstaben

Folgende kalligraphische Großbuchstaben können im mathematischen Modus erzeugt werden.
$\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{H}, \mathcal{I}, \mathcal{J}, \mathcal{K}, \mathcal{L}, \mathcal{M}$
$\mathcal{N}, \mathcal{O}, \mathcal{P}, \mathcal{Q}, \mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}, \mathcal{V}, \mathcal{W}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}$
der Befehl dafür lautet: `$...\mathcal{A,B,C,...,Z}...$`

4.9.4 Mengensymbole

Analog gibt es dazu `$...\mathbbm{A,B,B,...,Z}...$` zur Erzeugung von Mengensymbolen:

$\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}, \mathbb{E}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathbb{I}, \mathbb{J}, \mathbb{K}, \mathbb{L}, \mathbb{M}$
$\mathbb{N}, \mathbb{O}, \mathbb{P}, \mathbb{Q}, \mathbb{R}, \mathbb{S}, \mathbb{T}, \mathbb{U}, \mathbb{V}, \mathbb{W}, \mathbb{X}, \mathbb{Y}, \mathbb{Z}$

Um diese Symbole nutzen zu können, muss man am Anfang des Dokumentes mit `\usepackage{bbm}` das bbm-Paket laden.

Alternativ nutzt man die Symbole aus `\mathbb{}`, dazu benötigt man z.B. das amssymb-Paket:

$\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}, \mathbb{E}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathbb{I}, \mathbb{J}, \mathbb{K}, \mathbb{L}, \mathbb{M}$
$\mathbb{N}, \mathbb{O}, \mathbb{P}, \mathbb{Q}, \mathbb{R}, \mathbb{S}, \mathbb{T}, \mathbb{U}, \mathbb{V}, \mathbb{W}, \mathbb{X}, \mathbb{Y}, \mathbb{Z}$

Welche einem besser gefallen, muss jeder für sich selber entscheiden.

4.10 Funktionsnamen

Der weltweite Standard in mathematischen Formeln Variablenzeichen in Italic zu setzen, schreibt andererseits vor, Funktionsnamen in Roman zu setzen.

Damit LaTeX einen Funktionsnamen als solchen erkennt, muß ihm ein \ vorangestellt werden.

Beispiel:

arccos	`\arccos`
sin	`\sin`
max	`\max`

4.11 Über- und Unterstreichen von Teilformeln

$$\overline{a - \underline{b + \overline{c}} - d}$$

Mit den Befehlen `\overline{`*Formelteil*`}` bzw. `\underline{`*Formelteil*`}` können Formeln oder Teile von Formeln über- oder unterstrichen werden.

Um statt des Querstrichs horizontale Klammern zu verwenden, benutzt man die Ausdrücke `\overbrace` bzw. `\underbrace`.

$$\overbrace{a - \underbrace{b + \overbrace{c}} - d}$$

4.12 Aufgestocke Symbole

Mit dem Befehl `\stackrel{`*oberes Symbol*`}{`*unteres Symbol*`}` können zwei Symbole zentriert übereinandergestellt werden. Für das obere Symbol wird eine kleinere Zeichengröße gewählt.

$$A \xrightarrow{\mu} B$$

4.13 Summen- und Produktzeichen

Einige mathematische Symbole benötigen Grenzen. Beispiele dafür sind das Summenzeichen (`\sum`) und das Produktzeichen (`\prod`), aber auch Integral (`\int`) und Grenzwert (`\lim`) haben Grenzen. Diese erzeugt man ganz analog zur Hoch- und Tiefstellung mit `^` und `_`. Beispiel:

$$\sum_{i=1}^{n} n = \frac{n(n+1)}{2} \qquad \int_{i=1}^{\infty} \frac{1}{x} dx$$

Die dazugehörigen Quelltexte lauten:
`\sum_{i=1}^{n} n = \frac{n(n+1)}{2}` bzw. `\int_{i=1}^{\infty} \frac{1}{x}`

Wie man sieht, stehen die Grenzen beim Summenzeichen *über bzw. unter* dem Symbol, beim Integral stehen sie rechts daneben. Ersteres bezeichnet man als " limits" -Darstellung, zweiteres als " nolimits" . Auswählen lässt sich dies mit `\sum\nolimits` bzw. `\int\limits`. Bei Textformeln wird ebenfalls standardmäßig die nolimits-Darstellung benutzt.

Mit Hilfe des AMS-Pakets kann man dies auch global für das gesamte Dokument umstellen. Dazu gibt man *amsmath* eine, oder mehrere der folgenden Optionen mit:

- *sumlimits,nosumlimits* Anfangs- und Endgrenzzeichen erscheinen mit *sumlimits* in abgesetzten Formeln unterhalb und oberhalb des $\sum$-Zeichens in limits-Darstellung. Umgekehrt werden diese Grenzangaben mit *nosumlimits* unten und oben hinter dem Summenzeichen angeordnet (nolimits).
- *intlimits,nointlimits* steuert in entsprechender Weise die Grenzangaben des Integralzeichens.
- *namelimits,nonamelimits* steuert in entsprechender Weise die Grenzangaben von `\det \lim \inf` usw..

4.14 Klammern

LaTeX kennt die unterschiedlichsten Klammersymbole:

$$(,),[,],\{,\},\langle,\rangle,\lfloor,\rfloor,\lceil,\rceil$$

Die runden und eckigen Klammern kann man einfach so verwenden, die geschweiften müssen als `\{\}` " maskiert" werden (da sie ja in LaTeX zum Gruppieren verwendet werden), alle anderen haben spezielle Namen:
`\langle, \rangle, \lfloor, \rfloor, \lceil, \rceil`.

Damit sich die Klammern automatisch der Größe der darin enthaltenen Teilformel anpassen, stellt man ihnen `\left` bzw. `\right` voran. Diese beiden müssen immer gepaart auftreten! Will man nur einseitig Klammern, so muss man mit `\left.` bzw. `\right.` um eine leere " Klammer" ergänzen.

Beispiel:

$$\left(\frac{1}{k}\right\rangle$$

Der dazugehörige Quellcode: `\left( \frac{1}{k} \right\rangle`

4.15 Matrizen

4.15.1 Die Array-Umgebung

Mit Hilfe dieser grossen Klammern und der Array-Umgebung lassen sich Matrizen darstellen.

$$\left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$$

lässt sich z.B. durch `\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)` erzeugen. Dabei funktioniert die Array-Umgebung analog zur Tabular-Umgebung im Textmodus: Als Parameter wird angegeben, wie die Spalten ausgerichtet sein sollen (l für links, c für center, r für rechts, | für eine senkrechte Linie zwischen den Spalten). Innerhalb der Umgebung sind die einzelnen Zeilen durch \\ getrennt, die Einträge innerhalb der Zeile durch &.

4.15.2 Die AMS-Methode

Mit Hilfe des AMS-Pakets lässt sich das ganze auch etwas " beschreibender" formulieren. Dort hat man nämlich mit `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` und `Vmatrix` spezielle Umgebungen zur Verfügung, die automatisch für zentrierte Spalten und Klammern $((), [], \{\}, ||, \|\|)$ sorgen. Des weiteren wird der horizontale Platzverbrauch reduziert.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

wird also durch `\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}` erzeugt.

4.15.3 Auslassungszeichen

Auslassungszeichen (die berühmten drei Punkte) gibt es in LaTeX in mehreren Varianten. Mit dem AMS-Paket erhält man die folgenden horizontalen Auslassungszeichen:

- `\dotsc` auf Höhe der Kommata $a_1, a_2, \ldots$
- `\dotsb` für binäre Operatoren, wie z.B. $+ \, a_1 + a_2 + \cdots$
- `\dotsm` für die Multiplikation $a_1 a_2 \cdots$
- `\dotsi` für Integrale $\int_{a_1} \int_{a_2} \cdots$
- `\dotso` für " sonstiges" $foo \ldots bar$

`\dotsc` und `\dotso` sind standardmässig Aliase für `\ldots` (l wie line, also auf der Linie), `\dotsb`, `\dotsm` und `\dotsi` für `\cdots` (c wie center, also in der Mitte der Zeile). Bei Bedarf kann man diese aber schnell umdefinieren.

Daneben gibt es noch `\vdots` (vertikal) und `\ddots` (diagonal von links-oben nach rechts-unten). Bindet man das Paket mathdots ein, so werden `\vdots` und `\ddots` auch vertikal an die Schriftgröße angepasst, normalerweise erfolgt die Anpassung nur horizontal. Des weiteren erhält man dann `\iddots` für diagonale Auslassungszeichen von rechts-oben nach links-unten.

4.15.4 Text innerhalb von Formeln

Um Text innerhalb von mathematischen Formeln gibt es den Befehl `\textnormal` bzw. `\text`.

Kapitel 5

Präsentationen mit LaTeX erstellen

Jörn Spannhacke, Andre Gass

Vortrag vom 28.11.2003

5.1 Das Seminar Package

5.1.1 Allgemein

Das Seminarpaket ist ein Timothy Van Zandt entwickeltes Paket für Latex, dass sich durch ein einfaches Layout auszeichnet. Es bietet nur wenige Features, z.B. fehlt das nach und nach Einblenden von Informationen. Es bildet die Grundlage für das verbreitete Prosper Paket.

5.1.2 Landscape und Portrait Format

Das Seminarpaket unterscheidet zwei grundsächlich Formate von Folien, das Landscape Format (quer) und Portrait Format (hochkant).

Quellcode für eine einfache Landscape Folie:

```
\documentstyle{seminar}
\begin{document}
\begin{slide}
Eine Folie im Landscape Format.
\end{slide}
\end{document}
```

Commandline Befehle (win32), um eine PDF aus dem Quellcode zu erstellen:

```
latex %1
dvips -t landscape %1
epstopdf %1.ps
```

Zu beachten ist, dass unbedingt die Option -t landscape mit angegeben wird.

Quellcode für eine einfache Portrait Folie:

```
\documentstyle[portrait]{seminar}
\begin{document}
\begin{slide*}
Eine Folie im Portrait Format.
\end{slide*}
\end{document}
```

Commandline Befehle (win32), um eine PDF aus dem Quellcode zu erstellen:

```
latex %1
dvips %1
epstopdf %1.ps
```

5.1.3 Die Slide Umgebung

Innerhalb einer `\begin{slide}` und `\end{slide}` Umgebung kann mit `\newslide` erzwungen werden, dass eine neue Folie beginnt, ansonsten fängt LaTeX automatisch nach `\end{slide}` eine neue Folie an.

5.1.4 Schriftgrösse der Folien

Seminar bietet den Befehl `\slidesmag` an, um den Folieninhalt zu vergrössern oder verkleinern.

Ein Beispiel Quellcode dazu:

```
\documentstyle[portrait]{seminar}
\slidesmag{9}
\begin{document}
\begin{slide*}
Text
\end{slide*}
\end{document}
```

Die Grösse wird als 1.2^n berechnet. n muss eine ganze Zahl zwischen -5 und 9 sein.

5.1.5 Grössen der Folien ändern

Das Format der Folie kann mit folgenden 3 Befehlen verändert werden:

`\slidewidth{breite}`

gibt die Breite der Folien an.

`\slideheight{höhe}`

gibt die Höhe der Folien an.

`\begin{slide}[breite,höhe]`

gibt die Grösse jeder Folie einzeln an.

In der Dokumentationen des Pakets sind viele weitere Möglichkeiten die Grössen zu ändern ausgelistet.

Mit `\centerslidestrue` oder `\centerslidesfalse` kann die vertikale Zentrierung der Folien geändert werden. Default ist `\centerslidestrue`.

Ein Beispiel Quellcode dazu:

```
\documentstyle[portrait]{seminar}
\centerslidesfalse
\begin{document}
\begin{slide*}
Dieser Text ist nicht zentriert
\end{slide*}
\end{document}
```

5.1.6 Ränder der Folien gestalten

Der Befehl `\slideframe[options]{style}` ermöglicht es, die Ränder der Folien anders zu gestalten.

Mögliche Stile sind:

- none
- plain

Weite Stile können z.B. mit `\documentstyle [fancybox]{seminar}` eingebunden werden:

- shadow (Folienumrandung mit Schatten)
- double (doppelte Folienumrandung)
- oval (ovale Folienumrandung)
- Oval (dickere ovale Folienumrandung)

Ein Beispiel Quellcode für die double Option.

```
\documentstyle
    [portrait,fancybox]{seminar}
\slideframewidth 0.5cm
\slideframe{double}
\begin{document}
\begin{slide*}
Folie mit dickem, doppeltem Rand.
\end{slide*}
\end{document}
```

5.2 Das Prosper Package

5.2.1 Allgemein

Das Prosper Paket wurde von Frédéric Goualard und Peter Møller Neergaard geschrieben. Es erstellt PowerPoint ähnliche Präsentationen und baut auf dem Seminar Paket auf. Seine Stärken sind die vielen vorgegebene Themes und die deutlich erweiteren Möglichkeiten gegenüber dem Seminar Paket.

Ein Beispiel Quellcode für eine einfache Folie mit Prosper:

```
\documentclass[pdf]{prosper}
\title{prosper}
\subtitle{Vorlage}
\author{J{\"o}rn Spannhacke}
\begin{document}
\maketitle/
\end{document}
```

Mit folgenden Commandline Befehlen (win32), kann eine PDF aus dem Quellcode erstellen werden:

```
latex %1
dvips %1
epstopdf %1.ps
```

Die Überschriften einer Folie kann mit `\begin{slide}{Titel}` angegeben werden.

Ein Beispiel Quellcode dazu:

```
\documentclass[pdf]{prosper}
\begin{document}
\begin{slide}{Titel}
Unser Text.
\end{slide}
\end{document}
```

5.2.2 Foliengestaltung

Die folgenden Beispiele demonstriert, welche Möglichkeiten Prosper bei der Gestaltung der Folien bietet:

Erstes Beispiel:

```
\documentclass[pdf,azure,
    slideColor,colorBG,
    total]{prosper}
\title{prosper}
\subtitle{Vorlage}
\author{J{\"o}rn Spannhacke}
\begin{document}
\maketitle
\end{document}
```

Zweites Beispiel:

```
\documentclass[pdf,gyom,
    slideColor,colorBG,
    total]{prosper}
\title{prosper}
\subtitle{Vorlage}
\author{J{\"o}rn Spannhacke}
\begin{document}
\maketitle
\end{document}
```

5.2.3 Optionen

Prosper bietet einige Optionen, die `\documentclass[...Optionen...]{prosper}` mitgegeben werden können

final	Gibt an, dass es sich um die finale Version handelt
draft	Gibt an, dass es sich um die Entwicklungsversion handelt
total	Zeigt die Seitennummer und Gesamtseitenzahl am Folienrand an
nototal	Zeigt nur die Seitennummer am Folienrand an
slideBW	Erstellt die Folien in schwarz/weiss
slideColor	Der volle Umfang an Farben wird benutzt, geeignet für farbige Präsentationen
colorBG	Benutzt die Farben des aktuellen Styles als Hintergrundfarbe
nocolorBG	Erzwingt, dass die Hintergrundfarbe weiss ist
noaccumulate	Overlays werden beibehalten
accumulate	Overlays werden unterdrückt

5.2.4 Aufzählungen

Die `itemstep` Umgebung erzeugt eine Aufzählung, bei die nach und nach eingeblendet wird.

Beispiel Quellcode:

```
\begin{itemstep}
    \item Punkt 1
    \item Punkt 2
    \item ...
\end{itemstep}
```

5.2.5 Overlays

Prosper bietet die `overlays` Umgebung an, die es ermöglicht, Informationen nach und nach einzublenden oder auch wieder auszublenden.

Quellcodebeispiel:

```
\overlays{n}{
    \begin{slide}{Folientitel}
        ...
    \end{slide}
}
```

n gibt die Anzahl der Überlagerungen an. Innerhalb einer Overlay Umgebung kann mit folgenden Befehlen kontrolliert werden, auf welchen der Inhalt erscheinen soll:

`\fromSlide{p}{Inhalt}`	`Inhalt` erscheint von Folie p bis n
`\onlySlide{p}{Inhalt}`	`Inhalt` erscheint nur auf Folie p
`\untilSlide{p}{Inhalt}`	`Inhalt` erscheint von Folie 1 bis p
`\FromSlide{p}`	Nachfolgendes erscheint von Folie p bis n
`\OnlySlide{p}`	Nachfolgendes erscheint nur auf Folie p
`\UntilSlide{p}`	Nachfolgendes erscheint von Folie 1 bis p

5.2.6 URLs

Prosper bietet den `\href{URL}{Text}` Befehl, um anklickbare Links ins Internet einzubinden.

Quellcodebeispiel:

```
\documentclass[pdf]{prosper}
\begin{document}
\begin{slide}{Suchmaschinen}
\large{ Link zu
\href{http://www.google.de}
    {Google}.}
\end{slide}
\end{document}
```

5.3 PDFscreen

5.3.1 Allgemein

PDFscreen ist ein Package, das von C.V. Radhakrishnan entwickelt wurde. Herunterladen kann man sich PDFscreen unter: `http://river-valley.com/`

Der Schwerpunkt wurde darauf gelegt, vorhandene Dokumente für Präsentationen aufbereiten zu können. PDFscreen ist daher auch nur ein Style und keine eigene Dokumentenklasse. Als Klasse kann man z.B. article oder book verwenden. Ein Vorteil von PDFscreen ist die Möglichkeit, den Präsentationscode zusätzlich zum eigentlichen Code des Dokuments in einer Datei zu verwalten. Durch eine Änderung der Optionen von PDFscreen entsteht hinterher entweder die Druckversion oder eben die Präsentationsversion.

5.3.2 Verwendung von PDFscreen

PDFscreen wird mit `\usepackage[Optionen]{pdfscreen}` eingebunden. Mögliche Optionen sind hierbei:

- article, book o.ä. um pdfscreen mitzuteilen, welche Dokumentenklasse benutzt wird.

- print oder screen um entweder die Druckversion oder die Präsentationsversion zu erstellen.

- panelleft oder panelright um ein PDF-Navigationspanell zu erzeugen, welches dann die Präsentation erleichtert.

5.3.3 Gestaltung der Seite

Vor dem Beginn des eigentlich Textes sollte man noch verschiedene Variablen von PDFscreen einstellen. Mit `\screensize{Höhe}{Breite}` stellt man die Grösse der Folien ein. Wenn man hier relativ geringe Werte wählt, erscheint der Text entsprechend grösser, ohne dass man die Schriftgrösse verändern muss. Sinvolle Werte sind z.B. 12cm für die Höhe und 16cm für die Breite. Es empfiehlt sich, hier die Seitenverhältnisse des Bildschirms (3:4) zu beachten. Mit `\marginsize{}{}{}{}` stellt man den Rand am Rande der Folien ein.

Da PDFscreen zwei Versionen verwalten kann, gibt es eine `\begin{screen} ... \end{screen}` Umgebung, die Text umschliesst, der nur in den Folien erscheint. Analog dazu gibt es noch eine `\begin{print} ... \end{print}` Umgebung, für Text der nicht in den Folien erscheinen soll.

Mit `\begin{slide} ... \end{slide}` kann man explizit einzelne Folien machen. Der Nachteil hierbei ist allerdings, dass die Folien auch in der Druckversion in einem Rahmen dargestellt werden.

5.3.4 Beispiel

Ein Beispiel für eine Präsentation mit PDFscreen:

```
\documentclass[a4paper]{article}
\usepackage[screen,article]{pdfscreen}
\begin{screen}
\screensize{12cm}{16cm}
\marginsize{1cm}{1cm}{1cm}{1cm}
\end{screen}
\author{Name des Autors}
\title{Titel}
\begin{document}
\tableofcontents
\section{section}
\subsection{subsection}
\end{document}
```

5.3.5 Nachteile

Einige Nachteile hat PDFscreen allerdings: Es wird bei grösseren/komplexeren Dokumenten ziemlich kompliziert, zwei Versionen zu verwalten.

Es ist nicht möglich mit PDFscreen eine .dvi oder eine .ps Datei zu erstellen, da PDFscreen nur mit pdflatex funktioniert.

Ausserdem ist es mit PDFscreen fast unmöglich, Overlays zu machen.

5.4 Beamer

5.4.1 Allgemein

Beamer ist im Gegensatz zu PDFscreen eine Dokumentenklasse, die rein darauf ausgelegt ist, ansprechende Präsentationen mit Latex zu produzieren. Beamer befindet sich zur Zeit noch in der Entwicklungsphase, ist allerdings schon sehr gut benutzbar. Der Autor ist Till Tantau.

Herunterladen kann man sich das Paket unter: `http://latex-beamer.sourceforge.net`

5.4.2 Vorteile

Beamer wurde bewusst so entwickelt, dass es einfach zu erlernen ist, es werden also weitestgehend Standardbefehle von Latex verwendet. Teilweise wurden die Befehle für die speziellen Bedürfnisse bei Präsentationen angepasst. Ebenso wie PDFscreen ist es bei Beamer möglich, neben der Folienversion noch eine Druckversion zu verwalten, allerdings ergeben sich hier auch die gleichen Probleme wie bei PDFscreen (zu unübersichtlich/komplex).

5.4.3 Gestaltung der Folien

Das Aussehen der Folien lässt sich mit verschiedenen Themes beeinflussen. Diese lassen sich recht einfach mit `\usepackage{themename}` einbinden. Die mitgelieferten Themes sehen teilweise sehr unterschiedlich aus.

5.4.4 Verwendung von Beamer

`\frame{...text...}` ist der Befehl um ein Frame zu gestalten. Der Text erscheint dabei in einem eigenen Frame. Beamer unterstützt direkt Overlays. Diese werden mit `<n>` benutzt, n ist in diesem Fall eine Aufzählung der Seiten, auf denen der entsprechende Punkt erscheinen soll. Mit `\item<1-3,6>` würde der Text in einer Aufzählung auf den Seiten 1-3 und 6 eines Frames erscheinen. Eine Zahl mit einem - bedeutet, dass dieser

Punkt ab der ensprechenden Seite erscheinen soll. Wie bereits erwähnt, gibt es auch in Beamer die Möglichkeit, zwei Versionen in einer Datei zu verwalten. Dazu dienen die Befehle

```
\presentation
\common
\article
```

Verwendet man den Befehl `\note`, lassen sich Notizen in eine Präsentation einfügen. Diese werden nur in einer speziellen Version angezeigt.

5.4.5 Grafiken in Beamer

Zum Einbinden von Grafiken wird bei Beamer das Paket PGF verwendet. Diese Paket wird ebenfalls von Till Tantau entwickelt. PGF ist ein sehr mächtiges Grafikpaket, auf das hier nicht näher eingegangen werden soll. Mit `\plainframe` kann man eine komplett weisse Folie erzeugen, z.B. um dort ein grosses Bild zu zeigen. `\pgfimage{dateiname}` ist der Befehl zum Einbinden von Bildern in die Präsentation.

5.4.6 Nachteile

Einer der grössten Nachteile von Beamer sind die Probleme, die entstehen, wenn man viele `\begin{verbatim} ... \end{verbatim}` Umgebungen benutzt. Diese lassen sich nicht direkt -oder nur mit Einschränkungen- in die Frames einbinden. Die beiden Möglichkeiten sind:

- `\frame[all:1]{...text...}` wobei die Folien dann nur noch aus einer Seite bestehen (keine Overlays) oder
- Definieren der Verbatims vor dem Frame mittels `\defverb\name!...!` oder `\defverbatim\name{\begin{verbatim}...\end{verbatim}}`

  Dabei müssen die Verbatims hinterher mit `\name` eingebunden werden.

5.4.7 Beispiel

Ein Beispiel für eine Präsentation mit Beamer:

```
\documentclass{beamer}
\usepackage{pgf}
\usepackage{beamerthemetreebars}
\title{Titel der Präsentation}
\author{Name des Autors}
```

```
\begin{document}
\titlepage
\section{section}
 \subsection{subsection}
\end{document}
```

Kapitel 6

Index und Bibliographie

Somayeh Khorram, Rouzbeh Rouhi

Vortrag vom 5.12.2003

6.1 MakeIndex

6.1.1 Introdution

Größere wissenschaftliche Abhandlungen lassen sich oft erst mit Hilfe eines alphabetischen Namens- oder Schlagwortregisters gut erschließen. Es bereitet jedoch viel Mühe, einen qualitativ hochwertigen Index zu erstellen. Umfangreiche Register überfordern bisweilen die gängigen Textverarbeitungen. Das Textsatzsystem LaTeX, das eigens auf das Layout wissenschaftlicher Werke abgestimmt ist, bietet dagegen die Möglichkeit, auch komplexere Indizes zu erstellen. Die vorliegende Anleitung beschreibt den Arbeitsablauf einer Indexerstellung unter LaTeX

6.1.2 Wie verläuft die Indexerstellung?

Die Indexerstellung verläuft in vier Schritten:

1. Im Text werden die Indexbegriffe markiert.
2. Beim Umbruch mit LaTeX werden die markierten Begriffe in einer .idx-Datei gesammelt.
3. Mit „Makeindex“ wird diese Datei eingelesen,sortiert, formatiert und als .ind-Datei ausgegeben.
4. Bei einem erneuten Umbruch mit LaTeX wird die .ind-Datei in den Text eingebunden.

Der Indexprozessor „Makeindex“ bietet folgende Möglichkeiten:

- Makeindex kann Groß- und Kleinbuchstaben, Zahlen und Sonderzeichen sortieren.

- Mit Hilfe von Schlüsselwörtern können Sonderzeichen alphabetisch korrekt einsortiert werden.
- Verweise auf gleiche oder aufeinanderfolgende Seiten werden nicht separat aufgeführt sondern zusammengefasst.
- Makeindex gibt die Möglichkeit innerhalb einer Indexerstellung bis drei Eintragsebenen zu erstellen.Die sogennante Untereinträge.
- Seitenangaben können unterschiedlich formatiert werden, beispielsweise fett für einen Verweis auf eine ausführliche Behandlung des Indexbegriffs oder kursiv für eine zugehörige Abbildung.
- Statt besonderer Seitenangaben kann auf einen anderen Indexbegriff verwiesen werden. (Querverweise)
- Mittels einer benutzerdefinierten Stildatei kann auf das Layout des Index Einfluss genommen werden. (Formatierung)

Bevor man überhaupt in einem LaTeX Dokument ein Index erstllen kann sind zuerst zwei Einträge notwendig:

- Vor dem Befehl `\begin{document}` muss das Makropaket makeidx mit dem Befehl `\usepackage{}` geladen werden. Ebenfalls noch vor dem Befehl `\begin{document}` muss der Befehl `\makeindex` stehen, den das Paket makeidx bereitstellt. Im Text werden mit dem Befehl `\index{}` die Begriffe eingefügt, die im Register erscheinen sollen. Der Indexbegriff wird direkt an das entsprechende Wort im Text angefügt.
- Als letzter Schritt muss an der Stelle im Text, an der das Register später eingefügt werden soll,der Befehl `\printindex` eingefügt werden, der vom Paket makeidx bereitgestellt wird.Falls gewünscht, können vor dem Befehl `\printindex` weitere den Index betreffende Befehle eingefügt werden, beispielsweise:

```
\renewcommand{\indexname}{Sachregister}
\addcontentsline{toc}{section}{Sachregister}
```

Statt der Standardüberschrift İndex"wird SSachregister"gewählt und die überschrift ins Inhaltsverzeichnis übernommen. Der Eingangsbefehl `\makeindex` erzeugt die Datei beispiel.idx, in der alle Indexeinträge gesammelt werden. Die Datei beispiel.idx wird nun mit dem Indexprozessor „makeindex“ bearbeitet. Das Programm wird vom DOS-Prompt aus mit dem Befehl makeindx aufgerufen:

```
makeindx -g -s a:\mkidx.ist beispiel.idx
```

Ein Beispiel

```
\begin{theindex}
{\bf H} \indexspace \item Heinrich-Heine-Universit\"at\dotfill 1
\indexspace {\bf N}
\indexspace\nopagebreak%
\item Neurobiologie\dotfill 1
\end{theindex}
```

Index

Zunächst werden die Attribute der Stildatei mkidx.ist gelesen, die bestimmen, wie der Index aussehen soll. Anschließend werden die Indexeinträge aus der Datei beispiel.idx gelesen und sortiert. Das Ergebnis wird unter dem Namen beispiel.ind gespeichert. Die Bildschirmmeldungen werden als Log-Datei unter dem Namen beispiel.ilg abgelegt. Um den Index in das Dokument einzubinden, muss erneut ein Umbruch durchgeführt werden. Die Datei beispiel.ind wird dabei durch den Befehl `\printindex` eingelesen. Nach eventuellen Textänderungen müssen alle Schritte der Indexerstellung wiederholt werden, damit neue Begriffe aufgenommen beziehungsweise die Seitenzahlen aktualisiert werden.

6.1.3 Der Befehl index

Haupt- und Untereinträge

Mit dem Befehl `\index{}` werden im LaTEX-Dokument alle Begriffe gekennzeichnet, die im Index erscheinen sollen.Untereinträge werden mit einem! als Trenner direkt an den Haupteintrag angeschlossen. Untereinträge sind bis zur dritten Ebene möglich:

```
\index{Haupteintrag} \index{Haupteintrag!Untereintrag}
\index{Haupteintrag!Untereintrag!Unteruntereintrag}
```

Da Haupt- und Untereinträge durch ein! voneinander getrennt werden, dürfen sie selbst kein Ausrufezeichen enthalten. Soll dennoch ein! im Register erscheinen, muss es maskiert werden. Nach Bearbeitung der beim Umbruch entstandenen .idx-Datei mit „Makeindex“ entsteht folgende .ind-Datei mit den Indexeinträgen und jeweiligen Seitenzahlen in der theindex-Umgebung:

```
\begin{theindex}
{\bf H} \indexspace \item Haupteintrag\dotfill 1 \subitem
Untereintrag\dotfill 1 \subsubitem Unteruntereintrag\dotfill 1
\end{theindex}
```

Die Umgebung theindex bewirkt eine zweispaltige Formatierung des Index. Außerdem wird die überschrift İndex"über das Register gesetzt. Falls ein Indexeintrag größer ist als die Spaltenbreite, wird eingerückt in der nächsten Zeile weitergeschrieben.

Der Befehl `\indexspace` bewirkt eine Leerzeile im Index.Die Befehle `\item` ,`\subitem` und `\subsubitem` stehen für Haupt-, Unter- beziehungsweise Unteruntereinträge.

Soll das Layout des Index insgesamt gändert werden, muss die Umgebung theindex umdefiniert werden. Die Standarddefinitionen befinden sich in den Klassendateien article, report und book, in denen auch die Befehle `\item` ,`\subitem` und `\subsubitem` definiert werden.

Verweise auf längere Textteile

Häufig soll sich der Indexeintrag auf einen ganzen Absatz oder einen längeren Textteil beziehen. In diesem Fall müssen Anfang und Ende des Textteils in der Form `\index{|(}` und `\index{|)}` gekennzeichnet werden, um eine von/bis-Angabe zu erhalten. Das Verknüpfungszeichen selbst darf nicht als Text im Indexeintrag enthalten sein. Soll dennoch ein | im Register erscheinen, muss es maskiert werden. Steht zwischen von/bis-Angaben ein überflüssiger Einzeleintrag mit gleichem Inhalt,wird er korrekterweise ignoriert.

Einsortierung von Sonderzeichen, Umlauten und ß

Die Sortierung erfolgt entsprechend den Optionen des Programms „Makeindex“ .Es besteht jedoch auch über den Befehl `\index{}` die Möglichkeit,auf die Sortierreihenfolge Einfluss zu nehmen. Das ist insbesondere zur korrekten Einordnung der deutschen Umlaute und der Ligatur ßoder bei mathematischen Sonderzeichen notwendig, falls sie als Sonderzeichen gedruckt, lexikalisch jedoch unter ihrem Namen einsortiert werden sollen. Wenn beispielsweise im Index das griechische Sigma als Summen-Zeichen lexikalisch auch unter Summe einsortiert erscheinen soll, müssen im `\index{}`-Befehl sowohl der Sortierbegriff als auch das zu druckende Zeichen mit einem @ voneinander getrennt angegeben werden:

```
\index{Summe@$\sum$}
```

Entsprechend sollten auch alle Begriffe mit Umlauten oder mit ßeingegeben werden:

```
\index{Universitat@Universit\"at}\index{Grusswort@Gru\sswort}
```

Der Umlaut ä in Universität wird so wie ein a behandelt, das ßin Grußwort wie ss. Der Klammeraffe (at) darf nicht als Text im Indexeintrag enthalten sein. Soll dennoch ein @ im Register erscheinen, muss es maskiert werden.

Hervorhebungen in der Seitenangabe

Sollen bestimmte Seitenangaben im Register hervorgehoben werden, muss ein entsprechender Eintrag im `\index{}`-Befehl erfolgen. Der gewünschte Formatierungsbefehl wird ohne Backslash mit einem Verknüfungszeichen | an den Indexeintrag angehängt.Bei Indexmarkierungen für einen längeren Textteil wird entsprechend die Anfangsmarkierung um den gewünschten Befehl erweitert.Die Schlussmarkierung bleibt unverändert. Zur Realisierung komplexerer Formatierungswünsche muss ein neuer Befehl definiert werden, der die Gestaltung der Hervorhebung beinhaltet, beispielsweise kursiv mit nachfolgendem Stern:

```
\newcommand{\nn}[1]{{\it #1*}}
```

Der neue Befehl `\nn` hat einen Parameter, nämlich die automatisch eingetragene Seitenzahl, die kursiv `{\it }` und mit nachfolgendem Stern * gedruckt werden soll. Statt `\nn` kann jeder beliebige andere Befehlsname gewählt werden, sofern er noch nicht als TEX oder LATEXBefehl existiert. Im `\index{}`-Befehl wird das neue Kommando ohne Backslash \ mit dem Verknüpfungszeichen `|` an den Eintrag angehängt:

```
\index{Neurobiologie|nn}
```

Maskieren von Sonderzeichen

Die Sonderzeichen !, @ und | in den geschweiften Klammern des Befehls interpretiert „Makeindex“ als Anweisungen:

! als Trenner zwischen Haupt, Unter- und Unteruntereintrag @ als Trenner zwischen lexikalischer Einsortierung und eigentlichemIndexeintrag | als Verknüpfung zu einer die Seitennummerierung betreffenden Angabe

Beim Aufruf von „Makeindex“ ohne Option -g (german) wird die Sonderrolle dieser drei Zeichen durch ein vorangestelltes Anführungszeichen äufgehoben. Wird jedoch die Option -g (german) gewählt oder im Dokument das Paket german geladen, muss das Anführungszeichen als Standardmaskierungszeichen quote ”‘ umdefiniert werden, beispielsweise in ein Plus + . Ein Ausrufezeichen erscheint dann im Index durch die vorangestellte Eingabe des Maskierungszeichens + ; entsprechendes gilt für @ und |.

6.1.4 Der Indexprozessor Makeindex

Nach dem Umbruch mit latex müssen nun die entsprechend den -Befehlen automatisch erstellten Einträge in der .idx-Datei mit dem Indexprozessor „Makeindex“ sortiert werden. Der Aufruf für die Datei beispiel.idx lautet: makeindx -g -s `a:\mkidx.ist beispiel.idx`

Das Ergebnis des „Makeindex“-Durchlaufs ist eine Datei namens beispiel.ind, die den sortierten und formatierten Index enthält. Er wird bei einem erneuten Umbruch mit latex an der Stelle eingebunden, in der im Dokument der Befehl `\printindex` steht.

Die Option -c (compress blanks)

Durch Angabe der Option -c werden alle Leerzeichen und Tabulatoren in den `\index{}`-Befehlen bei der Sortierung ignoriert. Normalerweise kann diese Option beim Aufruf von „Makeindex“ angegeben werden, um Sortierfehler aufgrund zusätzlicher Leerzeichen oder Tabulatoren vor oder zwischen Wörtern zu vermeiden. Es kann jedoch sinnvoll sein, die Option nicht anzugeben, wenn das Leerzeichen bewusst zur Sortierung der Indexeinträge genutzt wird.

Die Option -g (german)

Die Option -g bewirkt eine Sortierung des Index entsprechend den Regeln der DIN 5007: Zuerst kommen Symbole, dann Kleinbuchstaben, schließlich Großbuchstaben und Zahlen. Ohne die Option -g wird so sortiert: Symbole, Zahlen, Großbuchstaben, Kleinbuchstaben. Soll die Option -g benutzt werden, muss das Anführungszeichen als Standardmaskierungszeichen quote ”“ umdefiniert werden. Makeindex hat eine Reihe weiterer Optionen, die hier nicht näher erläutert werden.

6.1.5 Makeindex-Fehlermeldungen und Warnungen

„Makeindex“ untersucht eingelesene .idx-Dateien auf syntaktische Fehler. Beim Schreiben auszugebender .ind-Dateien werden gegebenenfalls logische Fehler anzeigt. Alle Bildschirmmeldungen während des „Makeindex“-Durchlaufs werden in einer .ilg-Datei aufgezeichnet. Alle Fehlermeldungen werden mit!! Input index error... eingeleitet. Die häufigsten Fehlermeldungen in der Lesephase der .idx-Datei sind:

```
Extra ‘!’ at position ...
Extra ‘@’ at position ...
Extra ‘|’ at position ...
Illegal Null field
Argument ...  too long (max 1024)
```

„Makeindex“ erkennt eine Reihe weiterer Fehler, die hier nicht näher erläutert werden.

Alle Warnungen in der Schreibphase werden mit `## Warning` eingeleitet. Die häufigsten Warungen in der Schreibphase der auszugebenden .ind-Datei sind:

```
-- Unmatched range opening operator
-- Unmatched range closing operator
-- Extra range opening operator
-- Inconsistent page encapsulator ... within range
-- Conflicting entries
```

6.2 Bibliography

6.2.1 Introduction

Bibtex gibt Regeln für das Zitieren von Literaturstellen und allen sonstigen zitierbaren Quellen, d.h. Titelangaben und zusätzlichen Angaben, die zur Identifizierung von Dokumenten zweckmäßig sind. Bibtex ermöglicht die notwendigen Bestandteile des Zitats, ihre Form sowie ihre Reihenfolge festlegen. Erstellen von Literaturverzeichnissen mit BIBTEX hat viel mehr Vorteile im Vergleich zur manuellen erstellung einer Bibliography.

6.2.2 Manuelle Literaturverzeichnisse

```
Text Text Text \cite{zimmermann:1997}. Text Text
Text \cite{BOnse:1997,Schneider:1997}.
\begin{thebibliography}{Bon97}
\bibitem{zimmermann:1997}
Martin von Zimmermann, Thomas Niem"oller, Jochen R. Schneider.
\emph{X-ray scattering study of charge scattering associated with
stripe order in La$_{1.775}$Sr$_{0.225}$NiO$_4$}, J. Superconduc.
\textbf{10}, 447 (1991)
\bibitem[Bon97]{bonse:1997}
Ulrich Bonse, Felix Beckmann, Markus Bartscher, Theodor Biermann,
Frank Busch, Olaf G"unnewig. \emph{Phase contrast tomography using
synchrotron radiation}. In: Ulrich Bonse (Hg.),
\emph{Developmentsin X-ray tomography}, Proc. SPIE \textbf{3149},
108 (1997)
\bibitem{schneider:1997}
Jochen R. Schneider. \emph{Scattering of high energy photons in
condensed matter}. In: Armin Haase, Georg Landwehr, Erich Umbach
(Hg.). \emph{R"ontgen centennial}. World Scientific Publ. Corp.,
New York, 1997, S. 538
\end{thebibliography}
```

ergibt : Text Text Text [1]. Text Text Text [**?**, **?**].

Literaturverzeichnis

[1] Martin von Zimmermann, Thomas Niemöller, Jochen R. Schneider. *X-ray scattering study of charge scattering associated with stripe order in* $La_{1.775}Sr_{0.225}NiO_4$, J. Superconduc. **10**, 447 (1991)

[Bon97] Ulrich Bonse, Felix Beckmann, Markus Bartscher, Theodor Biermann, Frank Busch, Olaf Günnewig. *Phase contrast tomography using synchrotron radiation.* In: Ulrich Bonse (Hg.), *Developmentsin X-ray tomography*, Proc. SPIE **3149**, 108 (1997)

[2] Jochen R. Schneider. *Scattering of high energy photons in condensed matter.* In: Armin Haase, Georg Landwehr, Erich Umbach (Hg.). *Röntgen centennial.* World Scientific Publ. Corp., New York, 1997, S. 538

Allgemein:

```
\begin{thebibliography}{breitester Eintrag}
...
\bibitem[optionales Label]{Zitierschl"ussel}
...
\end{thebibliography}
```

Vorteile eines manuellen Literaturverzeichnisses lauten dann:

- Eine schnellere Dastellung bei kleinen Literaturverzeichnissen
- Einzelne Einträgen sind dann einfacher zu ändern

und die **Nachteile**:

- Schwierige Wiederverwenden in anderen Dokumenten
- änderungen des Layouts erfordern änderungen in jedem einzelnen Eintrag.

6.2.3 thebibliography-Umgebung

Mit BIBTEX kann die thebibliography-Umgebung automatisch erstellt werden. Hierzu werden benutzt:

- Die .aux-Datei, in die während des LaTeX-Laufs geschrieben wird, welche Zitierschlüssel im Dokument benutzt werden,
- Die .bib-Datei, die Datenbank, die die eigentlichen Literaturstellen enthält und die .bst-Datei, die die Layout-Vorgaben für das Literaturverzeichnis enthält. Dies ist der Trennung zwischen logischer Struktur und dem Layout, das bestimmt, wie diese logische Struktur visuell umgesetzt wird, bei LaTeX vergleichbar. Die BIBTEX-Datenbank entspricht dem LaTeX-Dokument, die BIBTEX-Stildatei enspricht der Dokumentenklasse.

6.2.4 Typischer Ablauf

1. Bei erstem LaTeX-Lauf (latex dokument) wird die .aux-Datei erzeugt, in der die Zitierschlüssel enthalten sind.
2. Bei BIBTEX-Lauf (bibtex dokument) werden die zu den Zitierschlüsseln gehörenden Literaturstellen aus der bib-Datei gelesen, und eine thebibliography-Umgebung erzeugt, die nach den Vorgaben aus der bst-Datei formatiert ist.
3. Bei zweitem LaTeX-Lauf (latex dokument) wird die thebibliography-Umgebung eingelesen.
4. Bei drittem LaTeX-Lauf (latex dokument) werden die Label an den Verweisstellen aufgelöst.

6.2.5 Standard-Stile für BIBTEX

Die Standard-Stile für BibTeX sind:

- **plain** Numerisches Label, Literaturverzeichnis alphabetisch nach Autoren sortiert.
- **unsrt** Numerisches Label, Reihenfolge nach Auftauchen der Verweise
- **alpha** Label aus Autor und Jahr (z. B. [Jon90]), Literaturverzeichnis alphabetisch nach Autoren sortiert
- **abbrv** Wie plain, aber mit abgekürzten Vornamen

6.2.6 Eintragsarten in der BIBTEX-Datenbank

- article Ein Artikel in einer Zeitschrift

  Zwingend: author, title, journal, year;

  optional: volume, number, pages, month, note

- book Ein Buch (mit Verlagsangabe)

  Zwingend: author oder editor, title, publisher, year;

  optional: volume oder number, series, address, edition, month, note

- inbook Ein Teil oder ein Kapitel eines Buches

  Zwingend: author oder editor, title, publisher, year, chapter und/oder pages;

  optional: volume oder number, series, address, edition, month, note

- incollection Ein Teil eines Buches mit eigenem Titel (und evtl. eigenem Autor)

  Zwingend: author, title, booktitle, publisher, year;

  optional: editor, volume oder number, series, type, chapter, address, edition, month, note

- proceedings Ein Konferenzbericht

  Zwingend: title, year;

  optional: editor, volume oder number, series, address, publisher, month, note, organization

- inproceedings Ein Teil eines Konferenzbandes

  Zwingend: author, title, booktitle, year;

  optional: editor, volume oder number, series, type, chapter, address, edition, month, organization, publisher, note

- phdthesis/masterthesis Eine Dissertation bzw. Diplom- oder Magisterarbeit

  Zwingend: author, title, school, year

  optional: type, address, month, note

- misc : Wenn sonst nichts passt

  Zwingend: Nichts

  Optional: author, title, howpublished, month, year, note

6.2.7 Deutsche Anpassungen für BIBTEX

Die Standard-Stildateien für BIBTEX sind für den englischen Sprachraum gedacht.Daher erfolgt die Formatierung des Literaturverzeichnisses nach englischen Gepflogenheiten. Von Prof. Klaus F. Lorenzen, FH Hamburg, wurden Stildateien entwickelt, die sich an der deutschen Zitiernorm DIN 1505 Teil 2 orientieren. Als Ersatz für die Standard-Stildateien stehen alphadin, plaindin, abbrvdin und unsrtdin zur Verfügung.damit wird versucht, einige bibliographische Beschränkungen der Originalstyles zu überwinden. Es läßt sich in fast allen Fällen problemlos eine Original-BIB-Datei (das ist die BibTEX-Datenbank, die die bibliographischen Informationen enthält) wahlweise nach US- oder DIN-Norm verarbeiten.

- **alphadin** : Die Zitate werden alphabetisch nach Verfassern sortiert und sind durch abgekürzte Verfasserbuchstaben plus Erscheinungsjahr in eckigen Klammern gekennzeichnet.
- **plaindin** : Zitate werden alphabetisch nach Verfassern sortiert und fortlaufend in eckigen Klammern gezählt.
- **abbrvdin** : Die Vornamen der Verfasser, Herausgeber, Zeitschriftennamen, Monatsnamen werden abgekürzt. Die Anordnung im Literaturverzeichnis entspricht der von plaindin.bst, also alphabetisch nach Verfassern sortiert und fortlaufend in eckigen Klammern gezählt.
- **unsrtdin** : Die Reihenfolge der Zitate im Literaturverzeichnis entspricht der Zitierung im Text. Form und Gestaltung entsprechen der von plaindin.bst.

Außerdem kann man auch eigene Stildatein mit dem Paket custom-bib erzeugen. Eine komfortable Möglichkeit zum Erzeugen eigener Stildateien für BIBTEX ist custom-bib von Patrick Daly. Custom-bib bietet:

- Verschiedene Zitierstile (numerisch, alpha, Autor-Jahr insbes. zusammen mit natbib vom gleichen Autor)
- Verschiedene Fonts für Autorennamen, Titel etc.
- Verschiedene Namensformate (V. Name; Vorname Nachname; Nachname, Vorname)
- Anpassungen für verschiedene Sprachen
- Vordefinierte Zeitschriftennamen für verschiedene Fachgebiete

Kapitel 7

Metafont – Metapost

Markus Riesenbeck, Sönke Thiesen

Vortrag vom 12.12.2003

7.1 Metafont

7.1.1 Einleitung

Entstehung

METAFONT wurde parallel zu TEX von DONALD E. KNUTH entwickelt. In seiner Dokumentation *The METAFONTbook* wird dabei die Fuktionsweise detailliert beschrieben. METAFONT ist elementarer Bestandteil von TEX und bildet dabei in vielen Aspekten Analogien. Während TEX die Möglichkeit bietet die darzustellenden Inhalte ansprechend zu gestalten, bietet METAFONT diese Möglichkeit auf der Ebene der einzelnen Zeichen.

Anwendungsgebiete

METAFONT kommt in vielen Anwendungsgebieten zum Einsatz. Es wird beispielsweise zum Entwurf und zur Generierung neuer Schriftarten, Zeichen und Grafiken, zum Vergrößern und Verkleinern bestehender TEX Schriftarten, zur Entwicklung grafischer Firmen- und Namenssymbole, sowie zur Bearbeitung, Kombination und Erstellung von Schriftarten und bestehenden Zeichensätzen verwendet. Eine Erstellung von Koordinatensystemen und mathematischen Objekten, bzw. die Generierung beliebiger Objekte auf der Basis von Linien, Kurven und Flächen sind darüber hinaus weitere gebräuchliche Anwendungsfelder bei denen METAFONT zum Einsatz kommt. In bestimmten Fällen kann es auch als Alternative zu PICTEXverwendet werden.

7.1.2 Warum META-?

METAFONT ist eine Programmiersprache zur Definition und Erzeugung von Zeichen und Symbolen. Der Unterschied zu herkömmlichen Schriftarten besteht dabei darin, dass es

problemlos möglich ist durch diverse Parameter das Layout einer Schriftart zu beinflussen. Insgesamt ist es möglich 62 verschiedene Parameter zu definieren, die dann ein Zeichen oder Symbol generieren, dass dabei im Wesentlichen durch das Setzen von Punkten, Generieren von Linien, Kurven und Flächen zwischen diesen Punkten und der Verwendung von sogenannten *Pinseln* eindeutig festgelegt wird. Es ist also möglich durch die einfache Veränderung von Parametern das Aussehen einer Schrift stark zu verändern ohne dabei das grundsätzliche Konzept zu variiren. Die Veränderungen werden allein durch die Beeinflussung der Paramter hervor gerufen.

7.1.3 Unterschiede zu normalen Bitmap-Fonts

Ein wesentlicher Unterschied zu herkömmlichen Bitmap Fonts besteht darin, dass die Zeichen erst beim Ausführen von METAFONT generiert werden, und nicht, wie bei normalen Schriftarten über statisch gespeicherte Bitmaps bereit gestellt werden. Dies bewirkt eine erheblich höhere Flexibilität bezüglich der Skalierbarkeit des Schriftsatzes und eröffnet weitaus mehr Möglichkeiten als lediglich eine Nutzung bezüglich der Darstellung von Buchstaben oder Sonderzeichen. Vielmehr ist es möglich auch komplexere Formen und Objekte zu generieren, wie sie z.B. in der Mathematik benötigt werden. Ein weiterer grosser Vorteil liegt in der Tatsache dass die Symbole erst nach einer Prüfung der übergebenen Parameter erzeugt werden. Dies verhindert unerwünschte Effekte bei der Skalierung der Zeichen und bewirkt somit eine, für den Betrachter angenehmere, Visualisierung der Zeichen. Wie auch TEX ist METAFONT plattformunabhängig und somit universell einsetzbar.

7.1.4 Konzepte von Metafont

Jedes Zeichen wird einzeln über *Punkte* definiert, die im nächsten Schritt mit Hilfe sogenannter *Pinsel* verbunden werden. Alternativ dazu können die definierten Punkte auch eine Fläche bezeichnen, die je nach Belieben mit Farbe gefüllt werden kann. Bezüglich der Anzahl und Anordnung, der auf diese Weise erzeugten Objekte, sind der Kreativität des Anwenders keine Grenzen gesetzt. Es ist zudem möglich, bereits definierte Objekte per Variation der Parameter zu beeinflussen bzw. zu verändern und so wiederum neue Figuren zu erzeugen.

Ein Beispiel für einen Quellcode sieht wie folgt aus:

```
mode_setup;
u#:=8pt#;
define_pixels(u);
beginchar("A", u#, u#, 0);
pickup pencircle scaled .04u;
z.center=(.5w, .5h);
z.eye.left=(.3w, .5h);
```

```
z.eye.right=(.7w, .5h);
z.nose=(.5h, .25h);
draw fullcircle scaled u shifted z.center;
draw fullcircle scaled .13u shifted z.eye.left;
draw fullcircle scaled .13u shifted z.eye.right;
filldraw fullcircle scaled .175u shifted z.nose;
mouthlength=.3w;
draw (.5w-.5 mouthlength ,.14h)-- (.5w+.5 mouthlength ,.14h);
z0=(0, .5h);
z1=(w,.5h);
filldraw fullcircle scaled .175u shifted z0;
filldraw fullcircle scaled .175u shifted z1;
endchar;
bye.
```

Das oben aufgelistete Prgramm erzeugt eine gesichtsähnliche Figur. In den ersten Zeilen werden die Stiftstärke sowie die Art des Pinsels gewählt. Im Weiteren wird festgelegt dass das zu erzeugende Zeichen auf der Position des *A* in der Zeichentabelle abgelegt werden soll. Die nun folgenden Zeilen definieren die relevanten Punkte der Figur, bzw. Zeichnen die entsprechenden Teile des Gesichtes, basierend auf gerade diesen Punkten. Bei der Definition der Punkte ist es möglich die Variablennamen frei zu wählen. Die Koordinaten werden dann in Form von Tupeln festgelegt. Nachdem die Variablen mit den entsprechenden Cursorpositionen belegt wurden ist es möglich an diesen Punkten, bzw. auf diesen Punkten, entsprechende Objekte zu zeichnen. Dies wird durch den Befehl *draw* realisiert.

7.1.5 Bearbeitungsmodi

METAFONT arbeitet mit drei Bearbeitungsmodi. Zur Entwicklung einer Grafik oder eines Symbols empfiehlt sich die Verwendung des *proof*-Modus. Hierbei wird jedes Zeichen stark vergrößert und jeweils auf einer gesonderten Seite dargestellt. Die später schwarzen Flächen werden grau dargestellt und es werden sogenannte Kontrollpunkte angezeigt, über die das visualisierte Objekt definiert wurde. Um dem User eine Hilfestellung bei der Arbeit zu liefern wird zudem ein Gitternetz aus feinen Linien im Hintergrund aufgespannt, die es dem Anwender ermöglichen, notwendige Änderungen präziser und einfacher umzusetzen. Neben diesem Modus gibt es bei METAFONT die Möglichkeit den *smoke*-Modus zu verwenden. Hierbei wird jedes Zeichen ebenfalls vergrößert und auf einer gesonderten Seite dargestellt, allerdings nicht transparent, sondern vielmehr in der als Ausgabe zu erwartenden Darstellungsform. Als weitere Hilfestellung werden die Abmessungen des Symbols in Form von kleinen Eckmakierungen veranschaulicht. Im Modus *lowres* wird ein fiktives Ausgabegerät simuliert, wobei die Auflösung auf 200 Pixel / Zoll

festgelegt wird. Der letzte, und für die endülgite Ausgabe der Zeichen vorgesehene Modus, ist der *localfont*-Modus. Er formatiert den Zeichensatz mit den für den Hauptdrucker angegebenen Konfigurationen und setzt unter anderem die entsprechende Auflösung. Mit dem METAFONT-befehl

```
\mode=localfont; mag=2.5; input font
```

lasst sich eine Schriftart beispielsweise mit einem beliebigen Faktor skalieren (hier wurde 2.5 gewählt). Die dabei erzeugte Schriftart wird entsprechend der Auflösung benannt (z.B. font.750gf)

7.1.6 Von der Implementierung zur Nutzung in Latex

Zunächst ist es notwendig eine Datei font.mf zu erzeugen. Hierfür kann ein beliebiger Editor (z.B. Notepad) verwendet werden. In dieser Datei wird der für METAFONT notwendige Quellcode gespeichert. Nun wird die METAFONT -Umgebung mit `mf font.mf` gestartet. Im Verzeichnis, in dem sich die font.mf Datei befindet, wird nun eine Datei font.750gf erzeugt, wobei die in der Dateiendung enthaltene Zahl für die Rasterung steht. (Es wird die Auflösung des Standartdruckers verwendet). Nun wird mit dem Befehlt `gftodvi font.750gf` das Programm gftodvi gestartet. Dies erzeugt eine DVI Datei font.dvi die mit jedem DVI-Viewer betrachtet werden kann.

7.1.7 Generierung von einfachen Objekten

Ein Objekt wird durch mehrere Punkte definiert. Die Reihenfolge in der die Punkte verbunden werden ist dabei variabel. Werden die einzelnen Punkte nun lediglich hintereinander aufgezählt - beispielsweise durch

```
draw z1..z2..z5..z3..z4
draw z1..z2..z3..z4..z5..z1
```

werden diese automatisch mit einer optimierten Kurve verbunden. Für den Fall das dies nicht gewünscht ist, gibt es die Möglichkeit den Befehl `cycle` anzuhängen, was bewirkt, dass die Kurve nicht am Endpunkt endet, sondern als gezerrter Kreis über die Punkte gelegt wird:

```
draw z1..z2..z3..z4..z5..cycle
```

Durch die Angabe zusätzlicher Parameter kann das Ergebnis darüber hinaus noch weiter beeinflusst werden. Der Befehl `dir` bewirkt beispielsweise, daß eine Kurve zwischen zwei Punkten nicht optimiert erzeugt wird:

```
draw z1{dir 60}..z2
```

verändert die verbindende Kurve derart, daß die Linie beim Punkt z1 in einem Winkel von 60 Grad beginnt.

```
draw z1{dir 60}..{dir -90}z2
```

bewirkt, daß die Linie beim Punkt z1 in einem Winkel von 60 Grad beginnt und im Punkt z5 mit einer sekrechten Tangente nach unten endet. Auf diese Weise lassen sich nun diverse geometrische Objekte definieren und erstellen.

7.1.8 Generierung komplexer Objekte und Formen

Mit METAFONT lassen sich jedoch nicht nur exakt definierte Gebilde erstellen. Es ist ebenso möglich Figuren über eine Vorschrift zu definieren. Mit Hilfe von Variablen lassen sich Punkte flexibel Positionieren und ermöglichen Implementierungen von Vorschriften zum Zeichnen komplexer Grafiken. Die hierbei verwendeten Algorithmen müssen natürlich den in der Informatik üblichen Anforderungen entsprechen. Die in diesem Abschnitt aufgezeigten Möglichkeiten und Beispiele decken natürlich nur einen sehr kleinen Teil der Möglichkeiten und der Vielseitigkeit dieser Programmiersprache ab. Es gibt darüber hinaus noch zahlreiche weitere Nutzungsmöglichkeiten, die es dem Nutzer ermöglichen METAFONT kreativ und problemorientiert zu verwenden.

7.1.9 Quellenangaben

- [Hoe98a]
- [Kop97]
- [Knu86a]
- [Knu86b]
- [Knu79]
- [Gün02]
- http://www.ctan.org
- http://homepage.mac.com/farwer/iblog/latex/index.html
- http://www.dante.de/dante/dante-faq.html

7.2 MetaPost

7.2.1 Entstehung

METAPOST wurde von John Hobby Anfang der 90er Jahre entwickelt. John Hobby war bereits an der Entwicklung von METAFONT beteiligt und entwickelte nach dessen Fertigstellung seine eigene weiterentwickelte Form: METAPOST. Es ist METAFONT sehr ähnlich, bietet aber statt der systemabhängigen Bitmap-Ausgabe eine universelle PostScript-Ausgabe.

Im August 1989 stellt er auf einer TEX -User-Tagung erstmals seine Idee unter dem Titel *A METAFONT-like System with Postscript Output*. Im März 1992 wird METAPOST für alle akademischen Institute zugänglich die eine Haftungsklausel unterschreiben. Für die Öffentlichkeit ist METAPOST ab Dezember 1994 verfügbar.

7.2.2 Was ist eigentlich MetaPost ?

METAPOST ist eine Makrosprache zur Erstellung von Grafiken, basierend auf METAFONT. Neben dem Ausgabeformat unterscheidet es sich auch durch die Möglichkeit, Text in die erstellten Grafiken zu setzten, von seinem Vorfahr METAFONT. Leider hat sich in die aktuelle Version ein Bug eingeschlichen, der für Grafiken mit gesetzten Texten eine fehlerhafte PostScript-Datei ausgibt.

7.2.3 Funktionsweise

Datentypen

Die wichtigsten Datentypen in METAPOST sind:

numeric sind Skalare (Vielfache von $\frac{1}{65536}$). Wenn sie für Längenangaben benutzt werden, ist die Standarteinheit Postscriptpunkte ($\frac{1}{72}$Inch).

pair wird zur Angabe von Koordinaten verwendet (in der Form *(0, 50)*).

color gibt eine Farbe als (Rot,Grün, Blau)-Tupel an (in der Form *.8red+ .3green+ .9blue*).

transform ist ein Tupel, das eine zweidimensionale Transformation eines *pair* beschreibt. Die Transformationen werden meist sehr einfach durch *shifted, rotated, xscaled, yscaled oder xyscaled* ausgedrückt.

path ist eine gezeichnete Figur, z.B. eine Linie die drei Punkte miteinander verbindet.

pen ist ein geschlossener Pfad, der eine konvexe Figur beschreibt, mit der man dann einen *path* malen kann.

Ein weiterer Datentyp ist **picture** mit dem ganze Bilder gespeichert werden können. Eine wichtige Variable dieses Typs ist **currentpicture**. Ein gutes Anwendungsbeispiel ist die Ausschnittsvergrösserung. Hier kann man das gemalte Bild mit currentpicture in eine andere Variable übergeben und danach verschieben, vergrössern und entsprechend zurechtschneiden.

Mit Hilfe des **lable** Befehls können Zeichenketten(Strings) an gemalte Figuren angetragen werden. *label.top('texttext', point X.XX of* path*);*

Vorgehensweise

Um eine METAPOST-Grafik richtig zu erstellen sollte man einige Dinge beachten. Nachdem man eine **.mp* Datei erzeugt hat sollte man diese mit syntaktisch korrektem Inhalt füllen. Näheres zu der Syntax wird später anhand von Beispielen erleutert. Mit dem Programm *mpost* kann man diese Datei dann kompilieren. Ein Aufruf könnte wie folgt aussehen: *mpost beispiel.mp*. Wenn keine Fehler auftreten gibt mpost zwei Dateien aus, beispiel.N und beispiel.log, wobei N der Nummer der Figur im Quellcode entspricht. Wenn man mehrere Figuren berechnen lässt erhält man für jede Figur eine weitere Datei. Die Datei beispiel.N ist eine PostScript-Datei und kann z.B. mit gsview angesehen werden und natürlich auch in LaTeX eingebunden werden.

Um die Grafik in ein LaTeXFile einzubinden gibt es mehrere Möglichkeiten:

- mit

  ```
  \includegraphics{beispiel.N}
  ```

- wenn man pdf-LaTeX verwendet muss man beispiel.N in beispielN.mps umbenennen
- oder man wandelt die PostScript-Dateien mit *epstopdf* in PDF's um (epstopdf beispiel.N) und fügt das entstandene beispiel.pdf dann mit

  ```
  \pgfimage[width=Xcm]{beispiel}
  ```

  ein.

7.2.4 Beispiele

Quadrat

Code

pair a, b, c, d;	Die Variablen a, b, c, d werden als pair deklariert.
a := (50, 0);	
b := (50,50);	Hier werden den pair Variablen
c := (0,50);	Tupel zugewiesen, so dass
d := (0, 0);	sie die Ecken eines Quadrates beschreiben.
beginfig(1);	Die zu zeichnende Figur erhält ihre Nummer, die sie Später auch als Dateiendung erhält.
draw a - - b - - c - - d - - a;	Die Ecken werden mit durchgezogenen Linien verbunden.
endfig;	Figur 1 wird beendet.
end;	EOF

Resultat

pgflastimage

Spirale

Code

beginfig(2);	Figur 2 wird gestartet.
pickup pencircle scaled 3;	Es wird ein neuer Stift ausgewählt, der 3 mal so dick ist wie der normal Stift.
draw (0,0) for i=1 upto 8: ..(0,(i-.5)*10)..(0,-(i+.0)*10) endfor withcolor .8blue;	Mit hilfe einer Schleife wird eine Spirale gezeichnet und zwar in einem Blauton. Die .. geben an, dass die einzelnen Punkte mit Kurven verbunden werden sollen.
endfig;	Die Figur 2 wirds beendet.
end;	EOF

Resultat

pgflastimage

Pentagramm

Code

beginfig(3);	Figur 3 wird gestartet.
pickup pencircle xscaled 10 yscaled 5;	Es wird ein ovaler Stift ausgewählt.
draw for i=0 upto 4: (50, 0)	Ein Pentagramm wird gemalt.
rotated (i*144) - - endfor cycle;	
endfig;	Figur 3 wird beendet.
beginfig(4);	Figur 4 wird gestartet.
pickup makepen (for i=0 upto 4:	Es wird ein Pentagramm
(5, 0) rotated (i*72) - - endfor cycle);	als Stift ausgewählt.
draw for i=0 upto 4: (50, 0)	Ein Pentagramm wird gemalt.
rotated (i*144) - - endfor cycle;	
endfig;	Figur 4 wird beendet.
end;	EOF

Resultat

7.3 MetaPost Quellen und Handbücher

- [Hob91]
- [Sia02]
- [Hus02]

Kapitel 8

Collaborative Writing of Documents

Tobias Rathjen, Sascha Steinbiss

Vortrag vom 19.12.2003

You will notice that this paper is not strictly LaTeX related, you can use most of the presented tools and hints in a wide range of word processors or even offline. This is necessary due to the relation between LaTeX and collaborative writing, you can use LaTeX as a part of a collaborative writing project but collaborative writing is not a feature of LaTeX.
While most of the presentations we've seen during the proseminar "wissenschaftliches Arbeiten mit TeX und LaTeX" were focused on technical aspects of the subjects, our presentation was divided into a social/theoretical part and a technical part. The structure of this document is quite similar to our presentation but varies in style, as we tried to be more didactic, especially in the first part.

8.1 How To Handle Collaborative Writing

8.1.1 What Is Collaborative Writing?

Two Definitions Of Collaborative Writing

What collaborative writing actually is, is a simple question with a few simple answers. The problem of a useful definition is the range of existing definitions. Ede and Lunsford [EL90] for example defined in their study that collaborative writing is anything that leads to a written document while Couture and Rymer [CR91] stated a year later that collaborative writing is given when two or more authors are writing a text.
The first definition caused some contradictionary results; While 87% of the analysed texts were written by at least two authors, 82% of the primary authors claimed that they wrote the text alone. Despite of these contradictionary results, this definition is the definition of my choice. If you write a text on your own and let someone proofread this text, it was written collaboratively. The sense of this definition shows up if you write a longer text and someone edits the beginning of your text while you to continue writing the text, you will need a collaborative writing tool.

Text/Work Partitioning

Once there is more than one author writing a document, there has to be a kind of partitioning of the document. Sharples et al. [SGB+91] defined three different kinds of partitioning:

- sequential or longitudal partitioning
  is used when the different "authors" have different tasks during the project, those tasks have a chronological order like "writing the text" → "edit the text" → "layout the text" → "print the text".
- parallel partitioning
  describes the actual parallel writing of a text; The authors may write different chapters of the book at the same time.
- reciprocal partitioning
  is the anarchistic way of partitioning. Each author contributes a part of the later text, layout or editing whenever and whereever he likes to.

In the publishing reality you will notice that the forms explained above are rarely found in their pure form. You'll most likely find hybrid forms of parallel and sequential partitioning. One of the common hybrids is found when the writing of the text is parallelly partitioned while the other stages are partitioned sequentially.

Forms Of Collaborative Writing

In this section I would like to introduce you to a few forms of collaborative writing according to Ede's and Lunsford's definition, without a claim of completeness (and accuracy).

The Author/Publisher relation is probably the most common form among the different collaborative writing styles. It's very similar to our familiar student/teacher relation. The main difference between those relations is the context.
The project is partitioned sequentially, while the author does the writing, sometimes the layout and even the printing, the publisher is responsible for the quality of the text and finally for the release.

Congress Proceedings The special needs of congress proceedings, i.e. a quick completition, are suggesting a collaborative writing style which allows a parallel partitioned text creation. The publishing on the other hand is a job for the congress organisation. A few participants of the congress or special recorders write down the results on their topic of the congress. All the text are later assembled by the congress organisation to a proceeding.

The Interactive Paper Project was created by Prof. James Levin and graduate student James Buell at the University of Illinois at Urbana-Champaign. Unlike the previous forms of collaborative writing the IPP does not necessary aim for a publication. The IPP provides a webbased access to a given document and allows the user to comment it and to comment already existing comments.
Since it's possible to write the text as comments. The IPP is a reciprocal partitioning "collaborative writing word processor", comments and text can be contributed, whenever a participant likes to do so.
The chapters or paragraphs of the text are divided into records in Filemaker, a common data base application, and linked to each other. Likewise the added comments are saved as records. If you want to know more about the Interactive Paper Project you may visit their homepage[1].

8.1.2 Why Should I Write Collaboratively (Or Not)?

There are some pros and cons regarding collaborative writing. A strong pro is that collaboratively written texts often reflect different points of view. One single author tends to concentrate on his own point of view while he neglects other points of view or even entire topics.
The costs, on the other hand, are often critical while publishing a text. The more people are working on a single text, the more time and money it takes to write this text. There is often a deadline for a text. If this deadline is missed the work was in vain or at least it's a loss of money in case of a publication.
However we don't need a product consisting of errors[2], a text written by a single author doesn't have to be faulty but discretion is the better part of valor. A co-author that checks content, style and for misspellings can easily reduce error appearances in a text.
Although a correct text is important, on a challenging text it's desirable to have just one writing style. It's often hard enough to get into one style, thus to change the reception of the text chapterwise is an unnecessary experience.
Taking the arguments into account, it's easy to formulate the imperative "Use as much collaborative writing as affordable".

8.1.3 Advices

I am not a collaborative writing pro, anyway I want to provide you some wisdom regarding collaborative writing. So I decided to leech the wisdom gathered by Sylvie Noël and Jean-Marc Robert[NJM01]. They conducted a study about collaborative writing, particularly about the preferences of the authors and co-authors[3]. It was a relatively small study,

[1] Interactive Paper Project Homepage: `http://lrsdb2.ed.uiuc.edu:591/ipp/`
[2] Unless we attend to the current T1 lecture.
[3] There is a funny anecdote attached to this study, although there were 35 Microsoft Word users among the participants (and Word does not support collaborative writing except a crappy version control), 36 of the participants thought that the software they used supported collaborative writing well.

as only 41 subjects participated. Out of this contemplative round I have extracted the following hints and advices.

- create and use drafts
  Quite obvious, if there isn't a draft, the authors are predeterminated to write incoherently.
- establish a good plan/schedule
  Have you ever tried to print a document without all text parts?
- have someone in charge for project, document editing schedule and/or style
  The larger the project is, the more people have different pictures of the project. As a result they'll work against each other, not together.
- carefully select the members
  Sometimes the group members are just too different, in their character, style and or aims, so that they can't work together.
- have good version control
  A good version control is *the* tool to prevent your mates from working on your last month version of chapter 5.
- have good communication
  This is not just derived from the group composition, it is also essential to have contact with your co-authors on a regular basis to "stay tuned" about the project.
- agree on appropriate tools
  Have you ever tried to load an Apple Document on an IBM Machine? Or fax a Compact Disc?

8.2 Collaboration in Practice

Since it generally is a good idea to write collaboratively when possible, it is important to decide on how to put the intended way of collaboration into practice.
Though there are several possibilities of doing this, not all of them are equally appropriate when looking at a distinct aspect of a given writing situation, such as the number of co-authors or their physical locations. In this section, three different approaches to realizing collaboration will be presented. Furthermore, their advantages and disadvantages will be discussed, as well as the situations they can be used in most effectively.

8.2.1 Source Splitting

The most simple way of allowing several people to edit a big LaTeX document is, of course, to split it into several smaller LaTeX source files which can be edited separately. This

subdivision should preferably done by logical units. For example, `\parts` or `\chapters` become single files that can be worked on by the author who is responsible for that part and submitted later to an editor to be merged into the final document.

Advantages

Although this is not a complicated thing to do, using this technique already has a few important advantages:

- The document source becomes much more structured because e.g. the directory structure the files are stored in could be designed to reflect the logical structure of the document
- The source becomes a lot more readable since authors need not scroll unnecessarily within a huge document in their editor software to reach the parts they are editing; they simply load the file containing their parts
- Filesystem-level security features such as simple access control features can be used to restrict access to document areas that are e.g. finished and need no longer be altered

Parts of a larger document that have been split into separate files can be reincluded into a main ßkeleton"document using the `\include` or `\input` commands. A single source file containing all the content is created and fed to the typesetter which creates the final document. Consequently, the single files need not be valid LaTeX documents, but the

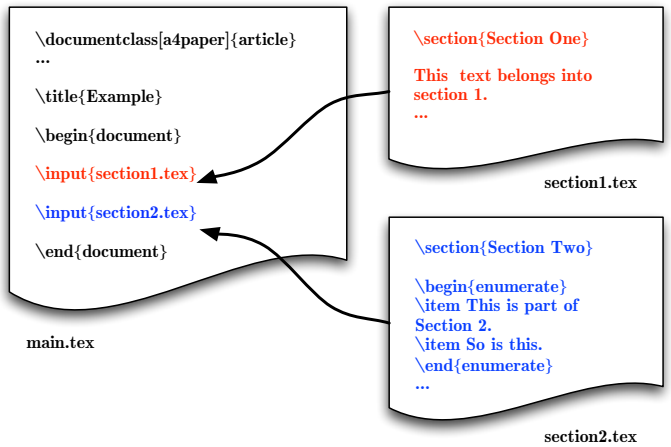


Abbildung 8.1: Splitting a single document into two separate files

resulting source file must be. So if one of the authors leaves an error in his source, the typesetting of the whole document will fail.

Disadvantages

This is already one of the disadvantages inherent to this approach to collaboration. Others can be found easily if you think a bit further:
If several authors with the same access rights have to edit the same part (=file) of a document, consistency problems will occur. Whoever saves the file last will overwrite all the changes all the others have made in the meantime! A possible solution to this problem would be to "lock"the file while it is in use, preventing other people from accessing it. But in everyday use this would unneccessarily slow down work, especially when someone forgets to remove the lock from a file after editing it and cannot be reached.
Another big disadvantage is the lack of a proper documentation feature. Of course, text files can be used to store change logs, but they would be subject to the same issues as the documents themselves, such as the saving problems etc.
In fact, this approach gives the users no possibility of any version control whatsoever. Only one version of the source tree at a time exists whereas previous versions cannot be restored in case of a "disaster".
If these issues can be neglected, which would be the case if there is only a small number of authors each of which exclusively edits his own chapter, source splitting will work for a group without too many problems. But the version control problem remains.

8.2.2 Using the Concurrent Versions System

So what can be done to resolve this issue and make changes to a source tree more controllable?
That is what software tools such as versioning systems are for. Versioning systems are used to track changes to a directory tree containing a set of files, documenting the changes that are made to them by different users. They provide some kind of interface layer between user and document.
One – and probably the most popular one – of them is called *CVS (Concurrent Versions System)*. In use since 1986, this free software project[4] implements a very flexible version control system that can be used to maintain big sets of data of all sorts (software source code, WWW pages, LaTeX documents, even binaries like images etc.). But the most impressive features of this powerful software will work on data that is based on text files.

Advantages

Using CVS as a versioning system helps us get rid of most of the problems encountered in the previous section.
People like to use CVS because:

- multiple users can edit the same set of files ßimultaneously"

[4]More information can be found on the CVS home page at `http://www.cvshome.org`

- no one can accidentally overwrite the work of others
- file locking is not necessary and therefore does not slow down actual work
- changes must be documented
- any older version of a single file can be retrieved
- people can easily collaborate over a network
- read and write access to project files can be controlled (independent of the kind of filesystem the source files are stored on)

CVS Architecture

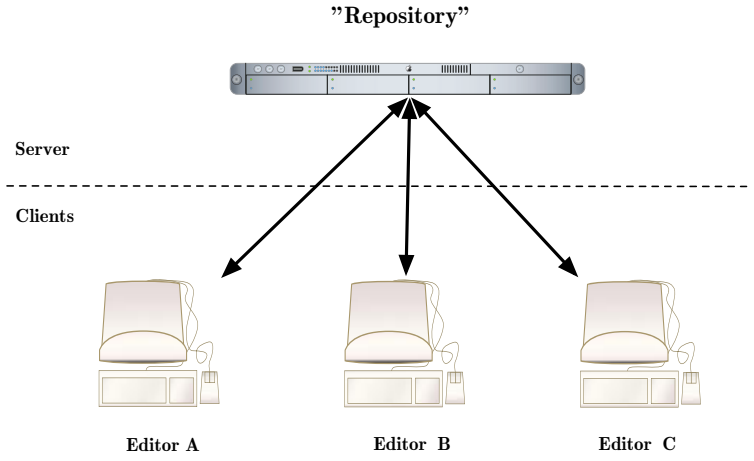


Abbildung 8.2: The CVS client-server architecture

A CVS system consists of a centralized stock of files related to a certain project and some meta-information associated with those files. All of them reside in something that is called the *Repository*, which is basically a directory tree in a filesystem on a server. In this directory tree every project that is managed by the CVS server is represented by a directory of its own. No user (in our case, this means no author) can access the files in the repository directly, he or she has to use the CVS interface. Furthermore, the authors do not modify the files themselves, but have to download their own local copies, modify them locally, and send them back to the server which processes them before they are incorporated back into the repository. This way no user can wreak havoc on the whole project by simply clicking a button.
The authors can access the files in a repository via two access methods:

- local → the repository is located on the same machine the user works on

- remote → the user accesses files on a remote server
  This way CVS can also be used to distribute files over a network or to allow authors from different locations to collaborate on a central server.

Basics of CVS Usage

The authors interact with the CVS system using the `cvs` executable. `cvs` is a CLI (command line interface) based tool which accepts a lot of parameters most of which will be presented in this section. A more detailed list of commands can be found in the ünofficial"CVS manual[Ced93].
For those who prefer a GUI (graphical user interface), several frontends are available. Windows users will certainly feel at home with *WinCVS*[5], which is being ported to other platforms and also available for Macintosh users and Unix users (as long as the gtk+ toolkit is available). An alternative is *TkCVS*[6] , which only requires a Tcl/Tk interpreter and the CVS client to work. Both frontends provide a user-friendly way of browsing and manipulating files inside the repository.

The most important commands when using a CVS system are `import`, `checkout`, `update`, `commit`, `diff`, `log` and `status`. I will provide a short overview of what these commands do when invoked from the command line:

- import
  - Syntax: `cvs import -m "description"projectname`
  - This command parameter imports the files from the current working directory into the repository (including subdirectories). This must be done before a project can be accessed via CVS.
- checkout
  - Syntax: `cvs checkout [-D "date"] projectname`
  - This command parameter creates a local working copy of the source tree in the repository on the editor's disk that can be worked on. The additional parameter `-D` allows time-based checkout:
    `cvs checkout -D "2003-12-11 13:42"projectname`
    `cvs checkout -D "2 weeks ago"projectname`
- commit
  - Syntax: `cvs commit [-m "log comment"] filename`

[5] `http://www.wincvs.orghttp://www.wincvs.org`
[6] `http://www.twobarleycorns.net/tkcvs.html`

  - If possible, this command puts the changed working copy back into repository and increments the version counter. If an older version is about to be committed ("version conflict"), `update` must be run before commit is allowed. The log comment is mandatory, so no change can be saved without being properly documented.

- update
  - Syntax: `cvs update filename`
  - This command compares the local working copy of *filename* with the current file in the repository and checks for differences. If there are differences, the CVS system tries to automatically update the local file. If a conflict occurs and cannot be resolved automatically, the conflicting parts are marked in the source, so the editor knows where to re-edit file before he can commit.

- log
  - Syntax: `cvs log [-r versions] filename`
  - This command shows a file's version history (version numbers and log comments).
    The result set can be narrowed down using the `-r` parameter:
    `cvs log -r1.3,1.4 filename.tex`

- diff
  - Syntax: `cvs diff -c -r version1 -r version2 filename`
  - This command prints the differences between two given versions of a file. The differences are marked in the output using different symbols:
    `+` : line has been inserted
    `-` : line has been deleted
    `!` : line has been changed

An Example Session

Now that we know the CVS commands and what they do we can look at a typical situation when multiple authors edit one single file, the problems that could occur and how using the CVS system can help them avoid those problems. The authors will be named A, B and C and some of them have a slightly different on-the-job morale.

1. Editors A and B check out Version 1.2 of the same file (e.g. *file.tex*).
2. A starts writing immediately and commits his changes. Repository version: 1.3.
3. B watches TV.
4. A continues writing and again commits his changes. Repository version: 1.4.

5. B goes to the cinema.

6. C joins them and checks out version 1.4 containing A's changes.

7. A, having worked all night, finishes his writing session and commits his changes. Repository version: 1.5.

8. B decides to start working and uses his old 1.2 version as a basis for his changes. When finished, he does a `cvs commit`.

9. CVS denies commit to B, since version 1.5 is already available in the repository, which is newer than B's 1.2 working copy. It therefore forces B to do a `cvs update` of *file.tex* so A's changes will not be lost but reflected in the resulting document.

Now there are two possible solutions depending on the part of the original version that B worked on. If A and B changed different parts of the file, the two versions will be merged into one file and the file can be committed properly. No further intervention by an editor will be necessary, since CVS can automatically identify changed parts (see fig. 8.3).
If the two editors worked on the same part of the text or the changed parts are over-

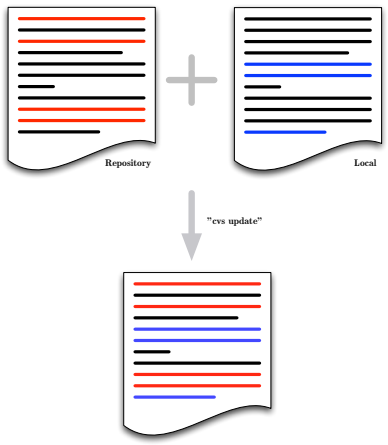


Abbildung 8.3: Version conflict with non-overlapping changes

lapping, this problem cannot be solved by CVS alone. This is because the CVS software cannot say anything about the file's content or "meaning", or what can be done to unify the information in these two versions in a sensible way. So B cannot commit his file unless he checks the output of `cvs update` and adds modifications to his changes to match the existing version. If the parts in question do not overlap any longer or even match each other, `cvs update` again merges the two versions into a new version and allows `cvs commit` to B, putting the resulting file back into the repository (see fig. 8.4).
Both ways, the version conflict has been resolved without any unnoticed data loss.

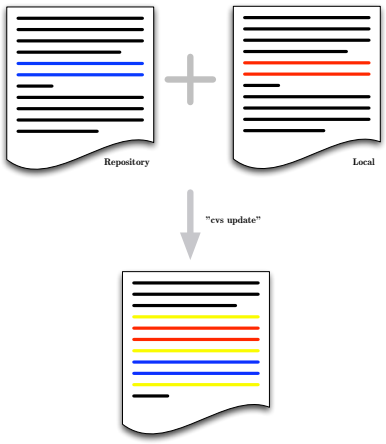


Abbildung 8.4: Version conflict with overlapping changes

8.2.3 Extreme Programming and Concurrent Live Editing

Besides the approaches covered in this paper so far, there is another interesting concept of collaborative work that is worth being mentioned here. This methodology is called *Extreme Programming* or abbr. *XP*.

What does XP mean?

Created in 1996 in a DaimlerChrysler corporate project by Kent Beck, Extreme Programming has proved to be an approach to software development in general that made every thing seem simple and more efficient. Beck analyzed what exactly made software simple to create and what made it difficult and used this newly gained knowledge to come up with a much more flexible way to improve creative work in the software industry.
Since writing a LaTeX document is not that different from traditional software development, it is not very far-fetched to adapt some key ideas from XP to LaTeX authoring. One of these key ideas is the idea of doing *pair programming*.

Pair Programming

Pair Programming means two people working together at a single computer to increase communication and thus productivity in developer/author groups. According to the XP home page[7], "[...] it is counter intuitive, but 2 people working at a single computer will add as much functionality as two working separately except that it will be much higher in quality. With increased quality comes big savings later in the project. The best way to pair program is to just sit side by side in front of the monitor. Slide the key board and

[7] `http://www.extremeprogramming.org/rules/pair.html`

mouse back and forth. One person types and thinks tactically about the method being created, while the other thinks strategically about how that method fits into the class. It takes time to get used to pair programming so don't worry if it feels awkward at first."
So much for the traditional version. Today there are software tools available that support and even extend the "pair programmingïdea, making it possible to share documents instantly among a more or less unlimited group of people, each and every one in front of their own computer, connected via a computer network.
The best known and most advanced are *iStorm*[8](not covered here) and *SubEthaEdit*[9] (formerly known as *Hydra*), both of which are excellent collaborative text editors for the Apple Macintosh platform.

SubEthaEdit - A Collaborative Editor

SubEthaEdit used to be called Hydra, but due to legal issues the developer team had to give it a new name. It was written by a group of students of the Munich Technical University who call themselves the Coding Monkeys. Initially, the project's goal was to create a network protocol for online collaboration, but the first application of this protocol developed a life of its own and finally won the Apple Design Awards and the Mac OS X Innovators Contest[10].
SubEthaEdit is a Cocoa application for Mac OS X, which means that, sadly, there is little chance of ever seeing a Windows or *nix version of the software. The most prominent feature of the text editor is the possibility of working "liveön one document with several people at once. To accomplish this, SubEthaEdit acts as a network ßerver"hosting the file to other client SubEthaEdit instances on a network, which can be a Local Area Network such as a corporate or academic workgroup, or even an intercontinental Wide Area Network. All users can connect to the server instance and type in their changes in real-time. This makes the setup and use of this application very simple and therefore increases the general public acceptance of this writing tool, even when it comes to users who are rather inexperienced with computers. This is especially the case when used in combination with the Apple Rendezvous technology. Every user can see each other's changes in real-time, easily distinguishable because every user gets his individual color that his changes are displayed in.
But even without using the advanced network collaboration features, SubEthaEdit is a formidable and feature-rich text editor, supporting e.g. syntax highlighting, unlimited undo levels, ßmartïndentation etc.

SubEthaEdit is best used in a situation involving few authors of few frequently changed documents who have to communicate a lot. The simple concept of collaboration that SubEthaEdit supports removes the work overhead that exists when, for example, CVS is used (checkout, commit, connect, disconnect) due to the complexity of the CVS com-

[8] http://www.mathgamehouse.com/istorm/
[9] http://www.codingmonkeys.de/subethaedit/
[10] http://www.macdevcenter.com/pub/a/mac/developer/2003/07/10/innovators.html

mands compared to the simplicity of the SubEthaEdit interface.
But if the number of authors increases and exceeds a certain level, the advantages of a proper versioning system wil surely outweigh is disadvantages.

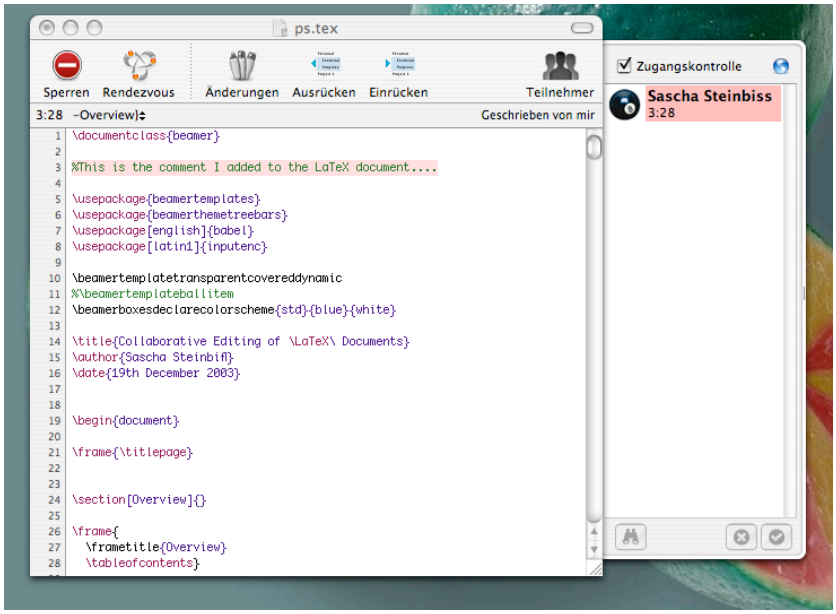


Abbildung 8.5: SubEthaEdit

Kapitel 9

LaTeX-Dokumente und das Internet

Christian Mein, Michael Spork

Vortrag vom 9.1.2004

Das heutige Zeitalter (Informationszeitalter) zeichnet sich dadurch aus, daß Informationen schnell zugänglich sind. Das beinhaltet, daß diese Informationen schnell zur Verfügung gestellt werden müssen. Ein Mittel, um diese Vorgehensweise zu entsprechen, ist das Übertragungsmedium World Wide Web [1]. Um Informationen durch dieses Medium darzustellen, wird die Auszeichnungsprache HTML, die im Jahr 1991 im Kernforschungszentrum CERN[2] ins Leben gerufen worden ist, genutzt.

Um nun LaTeX Informationsdokumente für das WWW zu transformieren, damit die Senke diese Informationen erhalten kann, sollen zunächst folgende Tabellen veranschaulichen, daß Ähnlichkeiten zwischen beiden Sprachen bestehen.

Die Tabellen auf Seite 80 verdeutlichen, daß die logische Struktur und die damit verbunden Tiefe (siehe Tabelle 9.1), die Hervorhebung von Textelementen (siehe Tabelle 9.2) oder Listen (siehe Tabelle 9.3) sowohl in der Sprache LaTeX als auch in HTML vorzufinden ist.

Rein strukturell sollten somit keine Probleme bestehen, die es verhindern könnten, LaTeX Dokumente in die Sprache HTML zu konvertieren. Jedoch haben Browser[3], die dafür verantwortlich sind, der Senke über das Übertragungsmedium die Informationen anzuzeigen, Defizite in der Wiedergabe von Tabellen, Bildern oder mathematischen Formeln. Um diese Defizite zu verringern, gibt es verschiedene Möglichkeiten, das jeweilige LaTeX Dokument zu konvertieren.

Im folgenden werden die Transformationsmöglichkeiten PDF mit Hypertextinformationen (siehe Abschnitt 9.1), LaTeX2HTML (siehe Abschnitt 9.2) und tex4HT (siehe Abschnitt 9.3) vorgestellt, analysiert und abschließend gegenübergestellt.

[1] WWW
[2] European Organisation for Nuclear Research, Genf
[3] z.B. : Safari, Konquerer, Netscape, Opera

Tabelle 9.1: Logische Struktur

Überschriften	HTML	LaTeX
Ebene 1	`<H1>text</H1>`	`\chapter{text}`
Ebene 2	`<H2>text</H2>`	`\section{text}`
Ebene 3	`<H3>text</H3>`	`\subsection{text}`
Ebene 4	`<H4>text</H4>`	`\subsubsection{text}`
Ebene 5	`<H5>text</H5>`	`\paragraph{text}`
Ebene 6	`<H6>text</H6>`	`\subparagraph{text}`
Ebene 7	`<P>`	`\par`

Tabelle 9.2: Hervorhebung

Hervorhebung	HTML	LaTeX
Betonung	`<EM>text</EM>`	`\emph{text}`
Fettdruck	`<B>text</B>`	`\textbf{text}`
		`\mathbf{text}`
Schreibmaschine	`<TT>text</TT>`	`\texttt{text}`
		`\mathtt{text}`

Tabelle 9.3: Listen

Listen	HTML	LaTeX
Geordnete	`<OL>...</OL>`	`\begin{enumerate}...`
		`\end{enumerate}`
nicht geordnete	`<UL>...</UL>`	`\begin{itemize}...`
		`\end{itemize}`
Listenpunkte	`<LI>text`	`\item{text}`
Beschreibung	`<DL>...</DL>`	`\begin{description}...`
		`\end{description}`
Schlüsselbegriff	`<DT>text`	`\item[term]`

9.1 Portable Document Format

Damit Hypertext Informationen in ein PDF Dokument eingebunden werden können, wird das LaTeX Paket `hyperref`, welches von Heiko Oberdiek und Sebastian Rahtz entworfen worden ist, benötigt. Dieses baut auf dem HyperTeX Projekt auf. Es erweitert die Funktionalität vorhandener Verweis-Befehle von LaTeX und fügt neue Befehle hinzu, um z.B. auf externe Dokumente zu verweisen.

Das Paket wird über `\usepackage{hyperref}` eingebunden und muß als letztes Paket im jeweiligen LaTeX Dokument deklariert werden, da es viele LaTeX-Befehle neu definiert und somit Befehle anderer Pakete überschreiben könnte.

Im folgenden wird darauf eingegangen welche Optionen das Paket bietet und wie Hyperlinks in das jeweilige LaTeX Dokument eingefügt werden können.

9.1.1 Paket Optionen

Die optionalen Argumente können zu dem `hyperref` Paket angegeben oder über das `\hypersetup` Makro geladen werden. Wenn das Makro genutzt wird, so werden die jeweiligen Paket-Optionen aus der Datei `hyperref.cfg` geladen, sofern diese vorhanden ist.

Folgende Optionen sind z.B. in diesem Paket standardisiert.

- `a4papaer` : setzt die Papiergröße auf 210mm x 297mm
- `bookmarks` : Es wird ein Verzeichnisbaum erstellt, welcher durch PDF-Viewern[4] angezeigt werden kann.
- Links, die farblich gekennzeichnet sind.

Normale Links	rot
Literaturverzeichnisseintäge	grün
Links, die Dateien öffnen	magenta
⋮	

Die weiteren, folgenden Optionen ermöglichen es dem Benutzer sein Dokument zusätzlich zu optimieren.

- `colorlinks = true`
  Wenn ein PDF - Dokument mir `hyperref` - Paket erstellt wird, so erhalten die jeweiligen Links eine farbige Umrahmung. Durch die genannte Option wird diese Variante deaktiviert und die Links werden farbig dargestellt.

[4] z.B. Acrobat Reader

- `citecolor = true`
  Durch diese Option wird die Farbe der Literaturverzeichniseinträge in die angegebene, neue Farbe geändert. Analog zu dieser Option kann die Farbe der Verknüpfungen zu anderen Seiten durch `pagecolor = true` verändert werden.

- `bookmarksopen = true`

  Der Verzeichnisbaum wird beim öffnen des Dokuments, sofern ein PDF-Viewer genutzt wird, bis zur kompletten Tiefe geöffnet. Dies erleichtert die Übersicht.

9.1.2 Hyperlinks im PDF

Es gibt verschiedene Arten von Hyperlinks, die verwendet werden können.

- `\href{URL}{Text}`
  Die klassische Variante, um einen Hyperlink zu erstellen, kann durch diesen Befehl entworfen werden. Die benötigte Information *Text* wird im Dokument angezeigt und ist ein Link zu der externen, angegebenen *URL*.

- `\hyperimage{URL}`
  Wenn dieser Befehl in das LaTeX Dokument aufgerufen, so wird das externe Bild von der angegebenen *URL* in das Dokument geladen.

- `\hypertarget{Name}{Text}`
  `\hyperlink{Name}{Text}`
  Diese Befehle können genutzt werden, damit ein Sprungpunkt innerhalb des Dokuments entstehen kann. Der Startpunkt bzw. Zielpunkt, welcher jeweils durch den `Text` im LaTeX Dokument wiedergegeben wird, kennzeichnet sich durch `hyperlink` bzw. `hypertarget` aus. Damit die Verknüpfung beider Elemente erstellt werden kann, muß der `Name` bei beiden Befehlen derselbe sein.

9.2 LaTeX2HTML

Das Konvertierungstool LaTeX2HTML ist von dem Erfinder Nikos Drakos 1995 über die "LaTeX2HTML" - Mailing - Liste vorgestellt und 1996 von der Universität Bayreuth veröffentlicht worden. Da sich das WWW und die Auszeichnungssprache HTML ständig weiter entwickeln, wird dieses Tool weiterhin aktualisiert und von Ross Moore koordiniert.

Im folgenden wird darauf eingegangen, wie zunächst das Konvertierungswerkzeug LaTeX2HTML genutzt werden kann, welche Optionen zur Optimierung zur Verfügung stehen und was das LaTeX - Paket `html`, welches mit der Installation des genannten Werkzeugs zur Verfügung steht, beinhaltet.

9.2.1 Erforderliche Software

Folgende Software Produkte müßen auf dem jeweiligen Betriebssystem[5] installiert sein, damit das Tool LaTeX2HTML genutzt werden kann.

1. TeXbzw. LaTeX
2. Perl[Per03]
3. dvips[Rad03]
4. Ghostscript[Gnu03]
5. netpbm[Hen04]
6. ImageMagick[LLC04]

Um die bereits angesprochenen Defizite der jeweiligen Browser in der Wiedergabe von Bildern und mathematischen Formeln auszugleichen, verwendet LaTeX2HTML die genannten Software - Produkte, um zunächst Bilder (bzw. mathematische Formeln) in der kompilierten DVI - Ausgabe in PostScript - Dateien, diese dann in ein Bitmap - Format (Gif oder PNG) und abschließend durch Grafikprogramme in ein Web akzeptables Bildformat zu konvertieren.

9.2.2 Benutzung

Um das erstellte LaTeX- Dokument durch LaTeX2HTML zu bearbeiten, kann folgender Befehl genutzt werden.

```
latex2html <Dateiname>[.tex]
```

Der Suffix `.tex` kann optional angegeben werden

Im folgenden werden einige, wichtige und hilfreiche Optionen zu dem Befehl `latex2html` vorgestellt, die es ermöglichen, eine individuelle Darstellung, durch die Sprache HTML erstellte, Web - Site zu entwerfen.

[5] Unix, Macintosh, Windows

Optionen

Verzeichnis Wenn durch den Aufruf `latex2html <Dateiname>[.tex]` keine weiteren Optionen genannt worden sind, erstellt LaTeX2HTML einen neuen Ordner, welcher den Namen der jeweiligen zu konvertierenden Datei erhält, indem die neu, von LaTeX2HTML erstellten Dateien, abgelegt werden. Wenn diese standardisierte Vorgehensweise nicht erwünscht ist, kann diese durch die Option `-dir Ordnername` unterbunden werden.

Der Befehl

```
latex2html -dir Ordnerneme <Dateiname>[.tex]
```

bewirkt somit, daß die konvertierten Dateien, in den Ordner `Ordnername` abgelegt werden.

Wenn der Benutzer kein Verzeichnis möchte, so kann er dies dem Tool durch folgende Option mitteilen. Die neu erzeugten Dateien werden dann in dem aktuellen Verzeichnis, indem LaTeX2HTML aktiviert wird, abgelegt.

```
latex2html -no_subdir <Dateiname>[.tex]
```

logische Hierarchie Ein LaTeX Dokument kann durch die Struktur - Befehle (siehe Tabelle 9.1) eine Tiefe mit sieben Ebenen erhalten. Das Tool LaTeX2HTML konvertiert das angegeben Dokument standardisiert in sieben verschiedene HTML- und zusätzlich einer Index - Datei.

Wenn diese Eigenschaft nicht erwünscht ist, sei es aus Platzgründen oder damit das erstellte HTML Dokument im WWW schneller erreicht werden soll, dann kann durch die folgende zusätzliche Option, die gewünschte Tiefe `AnzahlDerGewuenschtenTiefe` (es ist eine Angabe bis zu acht möglich) eingestellt werden, welches die Auswirkung hat, dass LaTeX2HTML nur so viele Dateien erstellt, wie angegeben sind.

```
latex2html -split AnzahlDerGewuenschtenTiefe <Dateiname>[.tex]
```

Kapitelnummer Nachdem der Konvertierungsvorgang abgeschloßen ist, kann in den erstellten HTML - Dateien, die durch einen Browser geöffnet werden können, erkannt werden, daß die jeweiligen Kapitel nicht nummeriert worden sind. Dies kann durch die folgende Option aufgehoben werden.

```
latex2html -show_section_numbers <Dateiname>[.tex]
```

In der neu, erstellten Datei `index.html` werden die einzelnen Kapitel mit der jeweiligen Kapitelnummer versehen.

HTML - Version LaTeX2HTML bietet zusätzlich die Option an, die HTML - Version auszusuchen, mit der das jeweilige LaTeX Dokument konvertiert werden soll. Derzeit sind die Versionen 2.0, 3.0, 3.2 und 4.0 vorhanden, wobei die Version 3.2 der Standard von LaTeX2HTML ist. Durch folgende Option können die Wünsche des Nutzers optimiert werden.

```
latex2html -html_version (2.0|3.0|3.2|4.0) <Dateiname>[.tex]
```

Navigationsleiste Die erstellten HTML - Dateien erhalten eine Navigationsleiste, welche in folgender Abbildung 9.1 gezeigt wird, die auf Wunsch deaktiviert werden kann.

Abbildung 9.1: Navigation

Die benötigte Option, um die Navigation zu deaktivieren, lautet :

```
latex2html -no_navigation <Dateiname>[.tex]
```

9.2.3 LaTeX - Paket

Nachdem das Tool LaTeX2HTML installiert worden ist, steht dem LaTeX System das zuätzliche Paket `html` zur Verfügung. Dieses Paket ermöglicht dem Benutzer weitere Befehle, das zu konvertierende LaTeX Dokument für die Konvertierung zu optimieren. Im folgenden werden die einzelnen, neuen Bereiche Hyperlinks zu externen Websiten, Umgebungen und Spezialeffekte, des genannten Pakets vorgestellt.

Hyperlinks zu externen Websiten

externer Link Desöfteren werden Informationen von anderen Quellen zitiert, genutzt oder explizit darauf hingewiesen. Durch das Medium WWW besteht die Möglichkeit auf diese Information durch ein URL[6] - Link aufmerksam zu machen. LaTeX2HTML stellt dem LaTeX - System zwei verschiedene Möglichkeiten bereit, die es ermöglichen die beschriebene Option in Anspruch zu nehmen.

Im klassischen Fall wird mit dem Befehl

```
\htmladdnormallink[Name]{URL - Text}{URL}
```

[6] Uniform Ressource Locator

im LaTeX Dokument darauf hingewiesen, daß im konvertierten HTML - Dokument, nachdem es mit einem Browser aktiviert worden ist, auf eine externe Information zurückgegegriffen wird. Das bedeutet für den Browser, daß dieser, wenn diese Zeile (`URL - Text`) durch einen Mausklick aktiviert wird, die entsprechende Site (`URL`) versucht zu kontaktieren und dementsprechend anzuzeigen.

Durch die Option `[Name]` kann im späteren Verlauf des LaTeX - Dokuments auf diese Zeile verwiesen werden.

Jedoch bietet dieser Befehl ein Defizit, denn in der kompilierten DVI- oder PDF-Ausgabe werden nur die Angaben von dem `URL - Text` wiedergegeben. Es befinden sich keine zusätzlichen Informationen über die `URL`.

Dieses Defizit hebt der folgende Befehl auf :

```
\htmladdnormallinkfoot[Name]{URL - Text}{URL}
```

Dieser bietet dieselbe Funktionaltität wir der Befehl `\htmladdnormallink[Name]{URL - Text}{URL}`, fügt jedoch dem entsprechenden DVI- oder PDF - Dokument eine Fußnote hinzu mit der Information über die `URL`.

externes Bild Um externe Bilder, sofern die angegebenen Verknüpfungen (`URL`) im WWW vorhanden sind, in das entstehende HTML - Dokument einzubinden, kann folgender Befehl genutzt werden.

```
\htmladdimg[Atrribute]{URL}
```

Die Option `[Attribute]` bezieht sich auf die HTML - Sprache, die z.B. die Befehle `width`[7], `height`[8] oder `align`[9] beinhaltet, die wiederum für die Positionierung des entsprechenden Bildes verantwortlich sind, und dort benutzt werden zu können.

Dieser Befehl wird von der kompilierten DVI - Ausgabe ignoriert.

Umgebungen

Wenn das LaTeX Paket `html` in das jeweilige LaTeX Dokument eingebunden worden ist, steht dem Benutzer zusätzlich die Möglichkeit dem jeweiligen Konvertierungs- bzw. Kompilierungsprogramm mitzuteilen, welche Zeilen interpretiert bzw. überspringen werden sollen.

[7] Breite
[8] Höhe
[9] Ausrichtung

HTML - Umgebung Die folgende Umgebung

```
\begin{htmlonly} Text \end{htmlonly}
```

veranlaßt den LaTeX Compiler den Text, der durch den Befehl `htmlonly` eingeschloßen ist, zu ignorieren. Analog wird dem Konvertierungstool LaTeX2HTML mitgeteilt, gerade diese Zeilen zu übersetzen.

Soll genau das Gegenteil bezweckt werden, so kann dies durch die Umgebung

```
\begin{latexonly} Text \end{latexonly}
```

veranlaßt werden.

Ebenfalls stellt das Paket eine Hybridumgebung bereit, die durch folgende Umgebung genutzt werden kann.

```
\latexhtml{Dier steht der Latex Text}{Und hier die HTML Text}
```

Wie bereits zu erkennen ist, interpretiert der LaTeX Kompilier nur die Informationen der ersten geschweiften Klammer, das Werkzeug LaTeX2HTML hingegen die zweite geschweifte Klammer.

HTML "pur" Das Paket bietet aber noch einen weiteren Zusatz.

Durch die Umgebung

```
\begin{rawhtml} HTML - Befehle \end{rawhtml}
```

können dem LaTeX Dokument alle Befehle aus der Sprache HTML mitgeteilt werden, die anschlie§end von dem LaTeX 2HTML Werkzeug interpretiert werden. Der LaTeX Compiler ignoriert diese Umgebung.

Spezialeffekte

Die Sprache HTML bietet eine Vielfalt an Spezialeffekten, um die jeweilige Site übersichtlicher, gestalterischer oder attraktiver zu gestalten.

So kann z.B. ein Wort oder Textzeile in einem HTML Dokument explizit durchgestrichen werden. Diese Option kann in dem jeweiligen LaTeX Dokument, welches anschließend in das Format HTML konvertiert wird, durch die Befehlszeile

```
\strikeout{Text}
```

erweckt werden.

Desweiteren kann eine horizontale Linie, zur besseren Übersicht, hinzugefügt werden. Der dazu benötigte Befehel lautet :

```
\htmlrule[Attribute]
```

Erneut bezieht sich die Option `[Attribute]` auf die Befehle der HTML - Sprache, die hier z.B. durch die Befehle `noshade`[10], `size`[11] oder `width` zur Geltung kommen können.

Ein weitere große Option, die viele HTML Benutzer nicht vermißen möchten, ist die Angabe, die sich auf alle folgenden, erstellen HTML - Dokumenten bezieht.

```
bodytext[Attribute]
```

Durch die optionalen Angabe der `Attribute` kann dem HTML - Dokument z.B. mitgeteilt werden, daß die Farbe der Schrift die Farbe "Gelb" (`text = yellow`) oder die Hintergrundfarbe die Farbe "schwarz" (`bgcolor = black`) erhalten soll.

9.3 TeX4ht

Das Paket TeX4ht ist eine weitere Möglichkeit die Vorteile des digitalen Mediums WWW, für die Publizierung von LaTeX-Dokumenten, zu nutzen. Es erzeugt aus einem TeX-Dokument `html` Seiten, die mit Hilfe von Links und Navigationsleisten durchsucht werden können., wobei Abschnitte bis zu einer bestimmten Tiefe auf getrennten html-Seiten angezeigt werden.

9.3.1 Erforderliche Software

TeX4ht kann zwar einfache Formeln komplett aus dem vorhandenen Zeichensatz erstellen, da aber auch TeX4ht nicht selbst Bilder erzeugen kann, müssen die folgenden Programme zusätzlich zum Paket installiert sein, um komplexere Formeln in eine Bilddatei zu konvertieren.

1. GhostScript[Gnu03]
2. ImageMagick[LLC04]

[10] Kein Schatten
[11] Grösse

9.3.2 Benutzung

Damit ein html-Dokument aus einer TEX-Datei erzeugt werden kann, muß das Paket über `\usepackage{tex4ht}` eingebunden sein. Um nun die Konvertierung zu starten wird folgender Befehl genutzt :

```
ht latex <Dateiname>.tex
```

in der Kommandozeile eingegeben.

9.3.3 Paket Optionen

Bei den Paket-Optionen ist wichtig zu beachten, dass die erste Option immer eine Konfigurationsdatei sein sollte. Wenn keine eigene Konfigurationsdatei erstellt worden ist, so kann optional das Attribut `html` angegeben werden.

```
\usepackage[html]{tex4ht}
```

Wird weder die erstellte Konfigurationsdatei noch das zuletzt genannte Argument benutzt, so wird nur eine DVI-Ausgabe erzeugt.

Hierarchische Tiefe

TEX4ht erzeugt standardisiert eine hierarchische Tiefe bis zur vierten Ebene, danach werden tiefere Abschnitte auf der selben html Seite angezeigt. Dies ist gleichzeitig die maximale Tiefe die TEX4ht erstellen kann.

Es ist aber möglich, die Tiefe bis zu der die Abschnitte auf getrennten html-Seiten angezeigt werden, noch zu verringern. Dies ist bei kleineren Dokumenten, die aber mehrere Unterabschnitte haben sinnvoll, da dies zur Übersichtlichkeit beiträgt. Es folgt eine Beispiel-Option, bei der ab einer Tiefe von *3*, die Unterabschnitte auf der selben Seite, wie der darüber liegende Abschnitt angezeigt werden.

```
\usepackage[html, 3]{tex4ht}
```

Es ist möglich die Tiefe durch Angabe der Zahlen 1-4 einzustellen.

Inhaltsverzeichnis Links

Um die Navigation durch das html-Dokument zu erleichtern, kann die Option `sections+` verwendet werden. Sie bewirkt eine direkte Verknüpfung vom Inhaltsverzeichnis in jeden Abschnitt und von den Abschnitten direkt wieder zurück in das Inhaltsverzeichnis. Somit muß der jeweilige Benutzer nicht die Navigationsleiste nutzen, um die Informationen der jeweiligen, erstellten `html` zu erhalten.

Folgender Befehl soll die eben beschriebene Vorgehensweise veranschaulichen.

```
\usepackage[html, 3, sections+]{tex4ht}
```

Vor-Zurück Button

Durch die Option `next` wird zusätzlich zur Navigationsleiste ein Vor-und Zurück Button auf jeder Seite eingefügt. Sia haben die gleiche Funktion, wie die Vor-und Zurück Buttons von Web-Browsern und ermöglichen es, zurück zur vorher betrachteten Seite zu kommen.

```
\usepackage[html, 3, sections+, next]{tex4ht}
```

HTML-Version

Standardmässig wird von TeX4ht die HTML-Version 4.0 verwendet. Es ist aber noch möglich, die HTML-Version 3.2, durch Angabe der jeweiligen Versionsnummer, zu verwenden.

```
\usepackage[html, 3, sections+, next, 3.2]{tex4ht}
```

HTML-Dateiname

Zur kompatibilität mit dem Betriebsystem Dos[12], ist es möglich, daß beim Erzeugen der html-Dateien der Dateiname auf 8 Zeichen und die Endung auf 3 Zeichen (`*.htm`), zu beschränken.

Hierzu wird die Option `html` durch die Option `htm` ersetzt.

```
\usepackage[htm, 3, sections+, next, 3.2]{tex4ht}
```

[12] Microsoft

9.3.4 Mathematische Formeln

Das besondere bei TEX4ht ist, dass nicht alle Formeln als Bild dargestellt werden, sondern, daß eingebettete Formeln, wenn möglich, aus dem vorhandenem Zeichensatz des jeweiligen Browsers erstellt werden. Dies bietet die Möglichkeit die Formeln aus dem fertigem Dokument heraus zu kopieren und zu bearbeiten. Dies funktioniert bei ganz einfachen Formeln noch gut. Sobald aber in den Formeln gleichzeitig Zeichen und Bilder zur Darstellung einer Formel benötigt werden, ist die Darstellung fehlerhaft.

Dies sieht in etwa so aus:

LATEX-Code: `$H = \sum_{i=1}^{n} p_i h_i$`

Darstellung im Browser[13]:

$$H=\sum_{i=1}^{n} p_i h_i$$

Abbildung 9.2: fehlerhafte Formel

Hier ist nur das Summenzeichen ein Bild, der Rest sind reguläre Zeichen. Es kann erkannt werden, daß die Bereichsindikatoren des Summenzeichens nicht über und unter dem Zeichen liegen, sondern weit daneben.

Abgesetzte Formeln, die dem LATEX System durch die Befehle

```
\begin{mathdisplay} Formel \end{mathdisplay}
```

zu signalisieren sind, werden von TEX4ht zu einem Bild konvertiert. Diese Darstellung sollte bevorzugt werden, da sie eine fehlerfreie Anzeige bietet. Die Formeln sehen dann im Browser und im PDF gleich aus:

$$H = \sum_{i=1}^{n} p_i h_i$$

9.3.5 Hypertext

Dieser Abschnitt beschäftigt sich damit, wie die Vorteile von html Seiten für die Konvertierung von LATEX-Dokumenten nach html genutzt werden können.

[13] Für diese Fehlerbeschreibung ist der Browser Konquerer genutzt worden.

Bilder einfügen

Mit diesem Paket ist es nicht nur möglich Bilder, die lokal bei der Konvertierung vorhanden sind, in die html-Seiten einzubauen, sondern es ist mit dem folgendem Befehl auch möglich, Bilder die auf externen Seiten liegen, anzeigen zu lassen.

```
\Picture[Name]{URL oder lokaler Pfad}
```

Das optionale Attribut `Name` wird als Bildunterschrift angezeigt, der `Pfad` kann lokal sein, wobei dann das Bild immer angezeigt wird sobald das Dokument geöffnet wird. Wird eine URL angegeben, muß dagegen eine Internet-Verbindung bestehen, damit das Bild angezeigt werden kann.

Text als Bild

Mit diesem Befehl ist es möglich, daß ein normaler Text zu einem Bild konvertiert werden kann. Dies kann ein direktes kopieren des Textes aus dem Dokument erschweren.

```
\Picture+[Name]{}...text...\EndPicture
```

Das Attribut `Name` ist die Bildunterschrift. Der `...text...` wird bis zu dem Befehl `\EndPicture` zu einem Bild konvertiert, er kann somit hinterher nicht mehr bearbeitet werden.

HTML-Code "pur"

Da es logischerweise nicht für jeden html-Befehl äquivalente LaTeX-Befehle gibt, ist es möglich im LaTeX-Code mit dem Befehl `\HCode{html-code}` den gewüschten html-Code einzugeben.

Dies kann z.B. so aussehen:

```
\HCode{<Strong>}
```

Durch diesen Befehl wird somit der gesamte folgende Text **fett** geschrieben.

Im html-Code muss jeder Befehl explizit beendet werden. Dies geschieht durch die Eingabe von :

```
\HCode{</Strong>}
```

Der danach folgende Text wird nicht mehr fett geschrieben.

extra HTML-Seite

Mit dem Befehl

```
\HPage{Linkname} text \ExitHPage{Exitname} \EndHCode{}
```

wird an der angegebenen Stelle ein Link erstellt, der den Text *Linkname* besitzt. Dieser führt zu einer neuen html-Seite, der beliebigen *text* enthält.

Mit dem Befehl `\ExitHPage{Exitname}` kann man von der neuen html-Seite auf die vorige Seite zurück kehren. Der Befehl `\EndHCode{}` gibt an, bis wohin der `text` auf der neuen html-Seite angezeigt werden soll.

Hyperlinks

Um Links zu externen Seiten zu setzen, kann der Befehl

```
\Link[URL]{}{} LinkName \EndLink
```

benutzt werden.

Der `LinkName` erscheint als Link im geschriebenem Text, dieser führt dann zu der angegebenen `URL`.

9.3.6 Features

Das TEX4ht Paket erlaubt noch viele weiter Optimierungen des html-Dokuments. Im folgenden werden einige Beispiele vorgestellt.

Inhaltsverzeichnis

Um von einer Seite des html-Dokuments in das Inhaltsverzeichnis zu gelangen, kann der Befehl

```
\tableofcontents
```

genutzt werden

Dieser erzeugt einen Link zu einer html-Seite mit dem komplettem Inhaltsverzeichnis.

[next] [prev] [prev-tail] [tail] [up]

Abbildung 9.3: Navigationsleiste tex4ht

Layout der Navigationsleiste

Folgende Abbildung 9.3, daß TeX4HT standardisierend in Textform auf jeder html-Seite, am oberen und am unterem Rand, eine Navigationsleiste erzeugt.

Diese kann durch den Befehl `\configure{crosslinks}{1}{2}...{8}` nach belieben umkonfiguriert werden. Die Optionen `{1}-{8}` sind dabei folgendermaßen zu verstehen:

1 := Linke Begrenzung
2 := Rechte Begrenzung
3 := Naechste Seite
4 := Vorherige Seite
5 := Ende der vorherigen Seite
6 := Anfang der Seite
7 := Ende der Seite
8 := Aufwärts

9.4 Fazit

Der folgende Abschnitt soll dem Anwender, der sein LaTeX Dokument und somit seine Publikation über das Informationsmedium WWW zur Verfügung stellen möchte, verdeutlichen, welches Werkzeug für seinen individuellen Anspruch zu empfehlen ist. Desweiteren werden hinzufügend die Schwierigkeiten der jeweiligen Konvertierungssoftware, die sowohl in der Benutzung als auch bei der Installation entstehen können, beschrieben.

Wenn das LaTeX Dokument in das PDF Format übertragen werden soll, so werden keine weiteren Software - Produkte auf dem jeweiligen Betriebssystem benötigt. LaTeX2HTML und TeX4ht hingegen verwenden weitere Produkte, um Ihrer Funktionalität nachzukommen. So müssen für Bildverabeitungen oder in der Wiedergabe von mathematischen Formeln diverse Programme installiert sein (siehe Abschnitt 9.2.1 und 9.3.1). Obwohl zusätzliche Tools zwingend notwendig sind, kann es aber speziell bei dem TeX4ht System und mathematischen Formeln zu Problemen kommen (siehe Abschnitt 9.3.4). Diese Engpässe kommen hingegen in dem LaTeX2HTML und PDF - System nicht vor, da diese, die jeweiligen Formeln direkt durch Grafikprogramme konvertieren bzw. von dem LaTeX System übernehmen.

Da das Informationsmedium WWW u.a. die Sprache HTML verwendet, um über einen Browser die jeweiligen Information zur Verfügung zu stellen, bieten die Systeme TeX4ht und LaTeX4HTML die geschickteren Varianten, da die jeweiligen LaTeX Dokumente direkt

in die benötigte HTML Sprache konvertiert werden. Die erstellten PDF Dateien können hingegen nur mit einem entsprechenden PDF - Viewer angezeigt werden. Wenn die Datei in einem Browser und somit in eine HTML - Site eingebunden werden sollen, kann dies nur funktionieren, wenn der Browser einen entsprechenden Viewer implementiert hat. Die aktuellen Browser bieten mittlerweile diese Option. Jedoch sollte es im Interesse des Publizisten sein und somit einer Beachtung verdienen, daß die Senke oft nicht die benötigten Voraussetzungen, die notwendig sind, um an die jeweilige Information zu gelangen, erfüllen kann.

Kapitel 10

Fonts und virtual Fonts in TEX/LATEX

Henrik Paulini, Nils Semmelrock

Vortrag vom 16.1.2004

10.1 Fonts

10.1.1 Einleitung

Die in TEX-Systemen voreingestellten Schriftarten sind im Grunde wohl überlegt und darauf bedacht das menschliche Auge beim Lesen längerer Texte nicht zu belasten. Trotzdem macht es manchmal Sinn andere Schriftarten zu verwenden um einen Text einen gewünschten Ausdruck zu verleihen oder einfach nur um Textstellen vom normalen Fließtext abzuheben.

Das Einbinden anderer Schriftarten ist abgesehen von dem Erlernen der Syntax und deren Benutzung häufig nicht trivial. Schwierigkeiten ergeben sich abhängig von dem Format und vorliegenden Installationshilfen der gewünschten Schriftart. So ist es einfacher auf TEX-Systeme zugeschnittene Pakete einzubinden als Schriftarten zu verwenden, die in einem inkompatiblen Format vorliegen und erst entsprechend bearbeitet werden müssen.

Ich habe mir folgende Fragen gestellt, die mir zu Anfang meiner Recherchen als Leitfaden gedient haben und nach denen ich auch die Struktur dieser Ausarbeitung aufbaue:

- Welche Schriftarten und welche Formate können in TEX-Systemen verwendet werden?
- Wie können neue Schriftarten in ein bestehendes System installiert werden und was passiert eigentlich im Hintergrund?
- Wie können neue Schriftarten in Texten eingesetzt werden?

10.1.2 Schriftformate und Schriftarten

Formate

Verschiedene Schriftformate können grob in zwei Gruppen eingeteilt werden, „Bitmap"-Schriften und „Scaleable Outline"-Schriften. Bei „Bitmap"-Schriften wird für jedes Zeichen in jeder Auflösung eine Bitmap erstellt, was diese Schriftgruppe sehr speicherintensiv macht, aber dafür auch eine ästhetische, gründliche Modellierung zulässt. Bei „Scaleable Outline"-Schriften wird mathematisch durch sogenannte "Bézier"-Kurven ein Umriss der einzelnen Zeichen definiert. Durch Skalierung ist es hier möglich aus einer Schriftdefinition alle Schriftgrößen abzuleiten.

In TEX-Systeme lassen sich zwei Arten von Schriftformaten einbinden, einmal die METAFONT Schriften, welche für die Benutzung in TEX/LATEX in Bitmap-Schriften umgewandelt werden und die sogenannten Typ-1 Schriften, welche als Scalable-Outline-Schriften vorliegen. Es lassen sich natürlich auch andere Schriftformate einbinden, hier ist aber eine Umformatierung in Typ-1 Schriften nötig, welche selbst mit zu TEX-Systemen gehörenden Tools nicht gerade einfach ist.

Auf die Erstellung und Benutzung von METAFONT Schriften ist bereits in einem früheren Beitrag eingegangen worden, deshalb werde ich mich hier nur mit den übrigen Schriftformaten beschäftigen.

Damit eine Schrift in TEX/LATEX-Systeme eingebunden werden kann muss eine Datei mit den Metriken der Schrift vorliegen und eine, die das Aussehen der einzelnen Zeichen beschreibt. Das Aussehen der Schrift kann entweder als Bitmap oder als Angabe der Außenlinien in Form von den genannten Bézier-Kurven (Scalable Outline) vorliegen.

Typ-1

Typ-1 ist ein von Adobe entwickeltes Schriftformat, das die Schriftbeschreibung in Form von PostScript-Befehlen abspeichert, deshalb werden Typ-1 Schriften auch häufig als „PostScript-Fonts" bezeichent. Eine Typ-1 Schrift liegt in der Regel wie von LATEXverlangt in zwei Dateien vor. Die Metriken sind in einer `.afm`- oder `.pfm`-Datei (*Adobe Font Metrik* bzw. *Printer Font Metrik*) abgelegt, die Beschreibung der Aussenlinien in einer `.pfa`- (lesbar) oder einer `.pfb`-Datei (binär kodiert) abgelegt. Wie diese Dateien bearbeitet werden müssen, werde ich im nächsten Kapitel – der Installation – beschreiben.

andere Formate

Zu anderen Schriftformaten möchte ich hier nur die so genannten TrueType Schriften nennen, welche von Microsoft als Antwort auf Adobes Typ-1 Schriften entwickelt wurden. Auch dieses Schriftformat lässt sich nach einer konvertierung nach Typ-1 in TEX/LATEX-Systemen verwenden, welche aber sehr aufwendig ist. Hierzu möchte ich nur einen Verweis auf eine Anleitung geben (die mir schlaflose Nächte bereitet hat):

`http://www.radamir.com/tex/ttf-tex.htm`

Pakete

Es gibt auch Schriftpakete, die vorkonfiguriert in Systeme eingebunden werden können. Es ist hier keine Konvergierung von Formaten nötig oder umständliche Einbindung in das Dateisystem.

Ich möchte hier nur einige Beispiele nennen, es gibt natürlich viele solcher Schriftpakete, aber alle würden den Rahmen sprengen, deshalb folgen einige Pakte, welche bekannt sind und auch öfters einmal von Nutzen sein können:

`cm` : Das Paket `cm` wird bei der Installation eines TEX/LATEX-Systems automatisch mit installiert, es enthält die in der Einleitung angesprochenen voreingestellten *Computer Modern* Schriften. Die Schriften dieses Paketes werden in `lrefintro (Tabelle 10.1)` noch im Detail vorgestellt.

`cm-super` : Dieses Paket ist eine Erweiterung zu dem *Computer Modern* Schriften aus dem Paket `cm`. Die Schriften werden lediglich geliefert, aber nicht explizit installiert. Um diese Schriften zu benutzen ist Kapitel 2 zu beachten.

`comicsans` : Dieses Paket beinhaltet die bekannten Comic SansSchriften von Microsoft.

`palatino` : Hier finden sich Adobes Palatino Schriften. Dieses Paket wird bei der LATEX-Erweiterung *PSNFSS*, was in der Regel bei einem TEX/LATEX-System dabei sein sollte mit installiert und ist so von Anfang an einsatzbereit. Einige Beispiele zu Adobes Palatino finden sich in Abscnitt 10.1.3 (Tabelle 10.3).

10.1.3 Installation und Hintergrund

Berry-Schema

Um Schriftarten auf einem System zu verwalten bietet es sich aus mehren Gründen an eine einheitliche Namensgebung der Schriftdateien vorzunehmen. Wenn diese Namensgebung alle wesentlichen Daten einer Schriftart enthält lassen sich so installierte Schriften schnell finden und besser ablegen. Das Paket *fontinst*, welches bei der Installation von Schriftarten durch verschiedene Macros hilft, verlangt sogar eine bestimmte Namensgebung.

Karl Berry[1] schlägt eine einheitliche Namensgebung vor, die in TEX-Systemen Einzug gefunden hat und von vielen Tools benutzt wird.

Da dieses Namensschema Plattform unabhängig sein soll wird auf Grund der Einschränkungen von DOS-Systemen ein maximal acht Zeichen Dateiname verlangt, der folgende Form hat :

[1] Mitarbeiter im Department of Computer Science der University of Massachusetts at Boston

```
FNNW[S][V]DE
```

Hier stehen die einzelnen Zeichen für :

`FNN` : `FNN` steht für die Schriftfamilie (*font family*), wobei `F` für den Hersteller steht und `NN` als Abkürzung für den Namen der Schriftart. (z.B. `phv` für *Adobes Helvetica*)

`W` : `W` bezeichnet das Gewicht (*weight*) der Schriftart. (z.B. `r` für *regular* oder `b` für *bold*)

`[S]` : `[S]` ist optional und bezeichnet eine spezielle Gestalt (*shape*) der Schriftart, wenn sie vorliegt. (z.B. `i` für *italic* (kursiv) oder `c` für *small caps*)

`[V]` : `[V]` ist ebenfalls optional und bezeichnet die Variante einer Schriftart. (z.B. `p` für *ornament* (verziehrt) oder `9` für *oldstyle digits* (alte Ziffern))

`DE` : `DE` steht für die Kodierung der Schriftart, wobei `D` die Anzahl der Bits angiebt, wie viele Zeichen möglich sind (normal 7 oder 8) und `E` abkürzend für den Namen der Kodierung. (z.B. `7t` für *original TEXencoding* oder `8a` für das *Adobe standard encoding*)

Es ergeben sich so zum Beispiel folgende Dateinamen :

`padri8a` : Adobes Garamond Schriftart mit regular weight, italic shape und Adobes standard encoding

`pads8a` : Adobes Garamond Schriftart mit semibold weight und Adobes standard encoding

`mbvbi8a` : Monotypes Baskerville Schriftart mit bold weight, italic shape und Adobes standard encoding

Eine detaillerte Beschreibung und eine Auflistung vieler möglicher Attribute findet sich unter :

ftp://ftp.dante.de/tex-archive/info/fontname/fontname.pdf

Neue Schriftarten installieren

Die Installation von neuen Schriftarten und was TEX/LATEX-Systeme im Hintergrund damit tut möchte ich mit der Grafik 1 beschreiben.

Ob eine Bitmap oder eine scalable Outline Schrift eingebunden werden soll, es muss auf jedem Fall eine Datei vorhanden sein, welche die Metriken der Schrift beschreibt. In einer Metrikdatei stehen Informationen zu den Maßen der einzelnen Zeichen wie Höhe, Breite und Bezug zur Grundlinie, außerdem Zeilen- und Wortabstände, sowie weitere Daten, die benötigt werden um Zeichen auf einem Dokument zu positionieren. TEXbenötigt

ein eigenes Metrikformat, eine `.tfm` Datei, das in der Regel aber noch aus gelieferten Metrikdateien erstellt werden muss.

Die eigentliche Installation geschieht jetzt in 6 Schritten :

1. Als Erstes müssen die vorliegenden Dateien nach dem Berry-Schema entsprechend umbenannt werden.

2. Liegt die Metrikdatei am Anfang der Installation, wie in Grafik 1 gezeigt als `.afm` (*Adobe font-metric file*), dann kann diese mithilfe des Makro-Pakets *fontinst*[2] bearbeitet werden. Es werden drei Dateien erzeugt, eine `.fd` Datei, eine `.pl` Datei und eine `.vpl` Datei, die wie folgt zu verstehen sind :

   `.fd` : (*Font Definition*) - Angaben, über welche Abkürzungen die Schriftart und ihre Variationen in TEX/LATEX-Syntax eingebunden werden können.

   `.pl` : (*Property List*) - Lesbare Form einer Metrikendatei als Listen.

   `.vpl` : (*Virtual Property List*) - Da meist nicht alle Zeichen in einer neuen Schriftart vorhanden sind, wird sie bei der Installation mit anderen Zeichen aufgestockt. Die `.vpl`-Datei ist daher eine lesbare Form einer virtuellen Schrift (dazu mehr im zweiten Teil).

3. Mithilfe der Programme *pltotf*[3] und *vptotf*[4], welche in der Regel in TEX-Systemen mit installiert sind, lassen sich nun aus den `.pl`- und `.vpl`-Dateie die TEX- Metrikdateien (`.tfm`) erstellen. *vptotf* erstellt auch gleich aus den `.vpl` Dateien die dazugehörigen virtuellen Schriftarten (dazu mehr im zweiten Teil).

4. Durch die Konvergierung derAdobe Metrik Datei ist nun eine *font divinition* (`.fd`), eine *TEXfont metric* (`.tfm`) und eine *virtual font* (`.vf`) Datei erstellt worden, alles andere ist nun nicht mehr von Gebrauch. Nun müssen die Dateien noch im TEX-System positioniert werden. Dies ist nicht sehr schwer, wenn man weiß, wo Schriftarten im System untergebracht sind. Es wird jeder Dateityp wie foglt verteilt :

   texmf/fonts/⟨*type*⟩/⟨*supplier*⟩/⟨*typface*⟩

   Wobei ⟨*type*⟩ für die verschiedenen Dateiendungen steht, ⟨*supplier*⟩ für den Hersteller der Schriftart (z.B. Adobe, Monotype, etc.) und ⟨*typface*⟩ als Abkürzung für den Namen der Schriftart.

   Jetzt müssen noch für Bitmap Schriften die *packet bitmap files* (`.pk`) oder für Scalable Outline Schriften die Outline Beschreibungen in Form von `.pfa-` (lesbar) oder `pfb`-Dateien (binär) nach dem gleichen Schema im Dateisystem untergebracht werden.

[2] Beispiel zu einer Schriftinstallation und *fontinst*-Syntax gibt es unter `http://www.qno.de/computer/latex/fonts/tutorial.html`

[3] Beispiel zu Syntax unter `http://www.qno.de/computer/latex/fonts/tutorial.html`

[4] Beispiel zu Syntax unter `http://www.qno.de/computer/latex/fonts/tutorial.html`

5. Nun muss noch *dvips*, *pdftex* sowie *dvipdfm* konfiguriert werden, damit die neuen Schriftarten auch im System bekannt sind und beim Erstellen eines Dokuments gefunden werden. Hierfür werden zwei *font-map*-Dateien erstellt und in die Konfigurationsdateien eingetragen.

   Beispiel : In eine neu erstellte Datei `schrift1.map` wird die Zeile:

   ⟨`Dateiname`⟩ <⟨`Kodierungsdatei`⟩ <⟨`Dateiname`⟩`.pfa`
   (oder entsprechend `.pfb`)

   eingetragen und in eine neu erstellte Datei `schrift2.map` die Zeile:

   ⟨`Dateiname`⟩ ⟨`Kodierungsdatei`⟩ ⟨`Dateiname`⟩`.pfa`
   (oder entsprechend `.pfb`)

   Die Datei `schrift1.map` wird nun in den *dvips* und *pdftex* Konfigurationsordner kopiert. Die Datei `schrift2.map` in den Konfigurationsordner von *dvipdfm*.Für *dvips* wird in die letzte Zeile der `config.ps` die Zeile

   ```
   p +schrift1.map
   ```

   eingefügt. Für *pdftex* in die letzte Zeile der `pdftex.cfg` die Zeile

   ```
   map +schrift1.map
   ```

   eingefügt und schließlich für *dvipdfm* in die letzte Zeile der `config` die Zeile

   ```
   f schrift2.map
   ```

   eingefügt.

6. Im letzten Schritt muss nur noch das TEX/LATEXDateisystem Aktualisiert werden (*file name databse*) und die neuen Schriftarten können benutzt werden.

Ausführliche Beschreibungen sind leicht über eine Suchmaschine zu finden oder direkt bei `http://www.qno.de/computer/latex/fonts/tutorial.html`.

Hintergrund

Auch die Abläufe im Hintergrund bei der Erstellung eines Dokumentes ist in Grafik 1 gezeigt. TEX/LATEXbenötigt bei der Verarbeitung einer `.tex`-Datei in das `dvi`-Format lediglich die Metrikdateien (`.tfm`) der eingesetzten Schriftarten um abzumessen, wie der Text positioniert werden muss. Um eine Schriftart im TEX/LATEX-Syntax anzusprechen ist noch die `.fd`-Datei (*font difinition*) nötig, da hier eine Abkürzung definiert ist, über die eine Schriftart eingebunden werden kann.

dvips erstellt nun im nächsten Schritt aus der `.dvi`-Datei das fertige Dokument (in Grafik 1 eine `.ps`-Datei). Nun ist es nötig das Aussehen der einzelnen Zeichen zu kennen, dafür kommt hier die `.pfa`- oder `.pfb`-Datei für Scalable Outline Schriften bzw. die `.pk`-Datei für die Bitmap-Schriften zum Einsatz. Eventuell ist es nötig dem System Zuordnungen zu anderen Schriftarten bekannt zu machen, falls eine virtuelle Schriftart benutzt wurde (mehr dazu im zweiten Teil). *dvips* benötigt also :

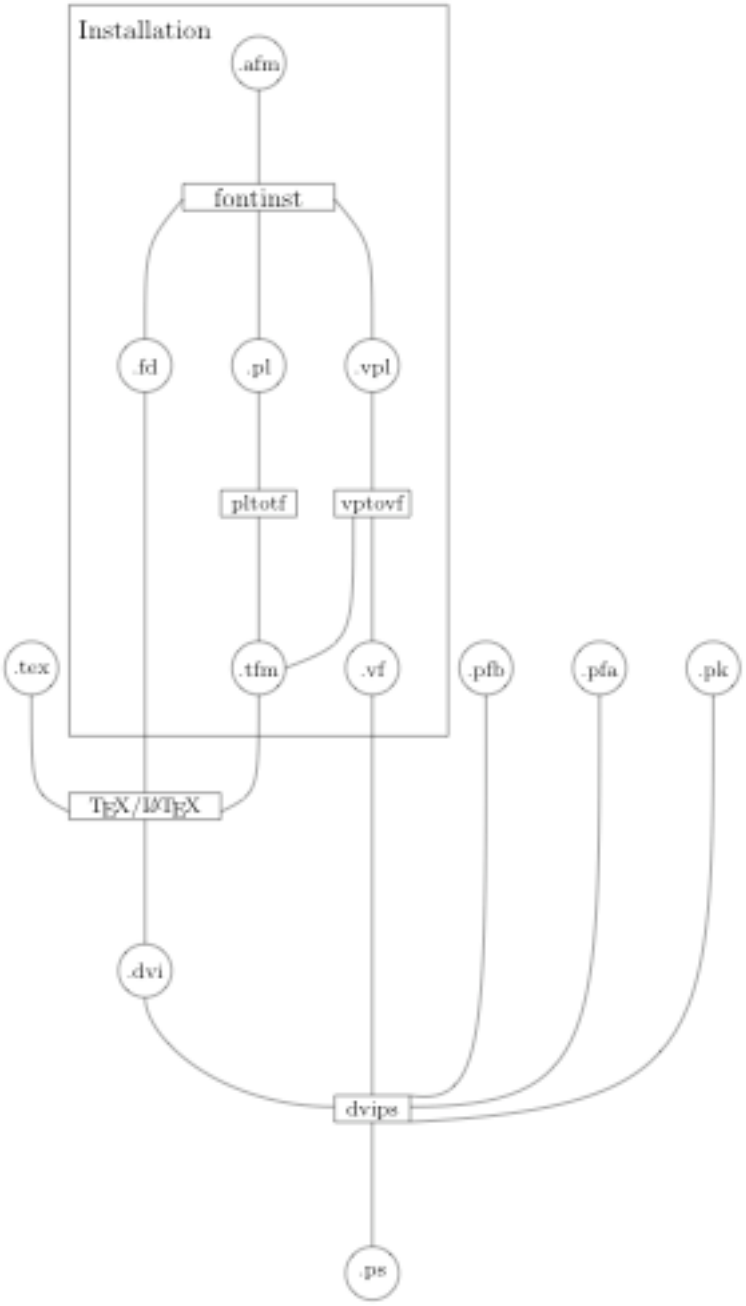


Abbildung 10.1: Benutzung von Schriftarten in TEX/LATEX-Systemen

- Eine `.dvi`-Datei mit dem Text des Dokumentes und der Abmessung der einzelnen Schriftzeichen, Zeilen- und Wortabstände
- Eine `.vf`-Datei (*virtual font*)
- Für Scalable Outline Schriften eine `.pfa`- oder `.pfb`-Datei
- Für Bitmap Schriften eine `.pk`-Datei

Heraus kommt eine PostScript Datei, die den gesetzten Text beinhaltet. Manche Systeme überspringen auch die Erstellung einer `.dvi`-Datei und übersetzen direkt in ein PDF- oder PS-Format, hier wird das Textsetzen durch die Metriken und das Erstellen eines Dokumentes mit dem Aussehen der Schriftarten von einem Programm übernommen, die einzelnen Schritte an sich bleiben jedoch vom Prinzip her gleich.

10.1.4 Benutzung neuer Schriften (NFSS)

NFSS

NFSS steht für *New Font Selection Scheme* und ist eine Erweiterung von LaTeX, mit der es möglich ist Schrifte an Hand von fünf Atributen strukturiert auszuwählen. Entwickelt wurde NFSS von Frank Mittelbach und Rainer Schöpf, was durch seinem Komfort zu einem "neuen" LaTeXführte. Das alte System zur Schriftauswahl in LaTeXwar durch mehrere Unschönheiten geprägt, so war es zum Beispiel nicht möglich verschiedene Attribute wie "fett" (*bold*) und kursiv (*italic*) zu schachteln, ohne dass automatisch auf Standardattribute zurückgestellt wurde.

Die fünf Attribute des NFSS für eine Schrift sind *Kodierung*, *Familie*, *Serie*, *Gestalt* und *Grösse*. Diese Begriffe erklären sich wie folgt :

Kodierung Die Kodierung ist die Anordnung der einzelnen Zeichen und die Größe des Zeichensatzes an sich. Die lateinischen Buchstaben und die Ziffern sind in der Regel gleich geordnet, bei den Sonderzeichen gibt es aber oft Unterschiede. Die Größe des Zeichnsatzes beträgt meist $2^7 = 128$ oder $2^8 = 256$ Zeichen.

Familie Als Familie werden alle Schriften bezeichnet, die zu einer Gruppe gehören, welche gleiche visuelle Charakteristiken aufweisen. Im Grunde unterscheiden sich die Schriften einer Familie durch verschiedene anwendbare Serien und Gestalten (siehe unten).

Serie Die Serie ist eine Zusammenfassung von Eigenschaften, die im Berry-Schema als Gewicht und Variante beschrieben wurden.

Gestalt Schriften gleichen sich in ihrer Gestalt, wenn sie ähnliche geometrische Attribute haben. Auch dieser Punkt wurde schon im Berry-Schema beschrieben.

Tabelle 10.1: Computer Modern Schriften, Encoding: OT1,T1

Familie	Serie	Gestalt	Beispiel
cmr	m	n, it, sl, sc, u	Computer Modern Roman
cmr	b	n	**Computer Modern Roman bold**
cmr	bx	n, it, sl	***Computer Modern bold extended italic***
cmss	m	n, sl	*Computer Modern SansSerif slanted*
cmss	bx	n	**Computer Modern SansSerif bold extended**
cmss	sbc	n	**Computer Modern SansSerif semibold condensed**
cmtt	m	n, it, sl, sc	Computer Modern Typewriter Small Caps
cmfib	m	n	Computer Modern Fibonacci
cmfr	m	n, it	Computer Modern Funny Roman
cmdh	m	n	Computer Modern Dunhill

Tabelle 10.2: Euler Schriften, Encoding: U

Familie	Serie	Gestalt	Beispiel
eur	m	n	Euler Roman
eur	b	n	**Euler Roman bold**
eus	m	n	EULER SCRIPT (nur Grossbuchstaben)
euf	m	n	Euler Fraktur

Größe Die Größe der Schriftzeichen. Das Erscheinungsbild der Größe kann von Schrift zu Schrift ein unterschiedliches Erscheinungsbild haben, so hat das folgende Beispiel die selbe Schriftgrösse (10pt):

CM Fibonacci Euler Roman → A A B B C C

Es ist also gelegentlich nötig unterschiedliche Schriftgrößen zu wählen, damit ein Text bündig aussieht.

Beispiele für die ersten vier Attribute bei zwei verschiedenen Schriften finden sich in Tabelle 10.1 und 10.2. Es ist zu beachten, dass nicht bei jeder Familie alle Serien und Gestalten vorhanden sind. Eine der umfangreichten Schriften ist die *Computer Modern* Schrift, die in Tabelle 10.1 gezeigt ist und deren *Roman*-Familie in TEX/LATEX-Systemen als Standardschrift eingestellt ist. Eine weniger umfangreiche Schrift ist die *Euler* Schrift, die durch das Paket `euler` nachinstalliert werden kann und in Tabelle 10.2 gezeigt ist.

Tabelle 10.3 zeigt noch ein Beispiel für Schriften, die nicht besonders umfangreich in ihrer Anzahl an Familien ist. Adobes Palatino ist wie in `1.2.4` erwähnt eine durch *PSNFSS* im System vorhandene Schrift.

Um das NFSS in LATEXzu benutzen ist ein bestimmter Syntax nötig, deren Befehle in drei Kategorien eingeteilt werden:

1. *High-level commands*

Tabelle 10.3: Adobes Palatino Schriften, Encoding: OT1

Familie	**Serie**	**Gestalt**	**Beispiel**
ppl	m	n, sl, sc, it, ui	Palatino Normal
ppl	b	n, sl, sc, it, ui	**Palatino Bold Small Caps**
ppl	bx	n, sl, sc, it, ui	***Palatino expert Bold italic***

2. *Mid-level commands*

3. *Low-level commands*

NFSS *High-level commands*

Die *High-level commands* sind die gebräuchlichsten und lassen eine einfache Auswahl zwischen verschiedenen Familien, Serien und Gestalten sowie der Größe zu. Ein Nachteil ist, dass nur zwischen Familien einer Schriftart gewählt werden kann. Es ist höchstens möglich durch eine globale Umdefinierung der Makros für die voreingestellten Attribute der *High-level commands* andere Schriften anzusprechen. Tabelle 10.4 zeigt die wichtigsten *High-level commands*.

Die wichtigsten *High-level commands* liegen in einer aktiven Befehlsform und in einer Deklarations- Form vor. Bei aktiven Befehlen wird ein Argument in Form eines Textes gegeben, welches sie verändern. Eine Deklaration verändert alles, was hinter ihr steht, entsprechend. Es bietet sich daher an eine Deklaration in geschwungene Klammern zu setzen, um so einen Textabschnitt zu bearbeiten.

Die Befehle sind eigentlich selbsterklärend und lassen sich wie angegeben in Texte und mathematische Umgebungen einbauen. Um andere Schriften und nicht nur andere Familien anzuwählen werden die *Mid-level commands* benötigt, oder man legt eine globale Definition an, wobei dann aber alle Familienkommandos, die man benutzen möchte, mit verändert werden müssen. Tabelle 10.5 zeigt eine Auswahl von vorgegebenen Schriftattributen, welche verändert werden können (siehe anschliessendes Beispiel).

Beispiel :

```
\renewcommand\familydefault{⟨Familie⟩}
\renewcommand\rmdefault{⟨Familie⟩}
\renewcommand\ttdefault{⟨Familie⟩}
...
```

Jetzt ist es für die global definierte Schrift und die Familien möglich sie in Dokumente über die *High-level commands* einzubinden.

Tabelle 10.4: NFSS *High-level commands*

active Befehle	Declarationen	Beispiel
`\textnormal{...}`	`{\normalfont...}`	class default
`\emph{...}`	`{\em...}`	*emphasis*
	Familie	
`\textrm{...}`	`{\rmfamily...}`	roman family
`\textsf{...}`	`{\sffamily...}`	sans serif family
`\texttt{...}`	`{\ttfamily...}`	`typewriter family`
	Serie	
`\textmd{...}`	`{\mdseries...}`	upright, medium series
`\textbf{...}`	`{\bfseries...}`	**bold-face series**
	Gestalt	
`\textup{...}`	`{\upshape...}`	normal, upright shape
`\textit{...}`	`{\itshape...}`	*italic shape*
`\textsl{...}`	`{\slshape...}`	*slanted shape*
`\textsc{...}`	`{\scshape...}`	Small Caps Shape
	Mathe-Befehle	
`\mathcal{...}`, `\mathrm{...}`, `\mathbf{...}`, `\mathsf{...}`, `\mathtt{...}`, `\mathnormal{...}`, `\mathit{...}`		
	Größe-Deklarationen	
`\tiny`, `\scriptsize`, `\footnotesize`, `\small`, `\normalsize`, `\large`, `\Large`, `\LARGE`, `\huge`, `\Huge`		

NFSS *Mid-level commands*

Mit den *Mid-level commands* lassen sich Schriften und ihre Attribute detaillierter auswählen. Es wird hier von einer Grundeinstellung ausgegangen. Bei dieser werden nur bestimmte Attribute verändert. Nach einer Auswahl von gewünschten Attributen wird die Änderung durch einen Startbefehl (`\selectfont`) aktiv. Da es sich hier ausschließlich um deklarierende Befehle handelt bietet es sich wieder an den zu veränderten Textteil in geschweifte Klammern zu setzen, da sonst der komplette nachfolgende Text betroffen ist. Die *Mid-level commands* finden sich in Tabelle 10.6.

Mit den ersten fünf Befehlen aus Tabbel 10.6 lassen sich die fünf NFSS Attribute einer Schrift auswählen, die Auswahl wird dann mit `\selectfont` aktiv. Der Befehl `\usefont` fasst die ersten fünf Befehle zusammen und verlangt als Argumente alle fünf Attribute, ein `\selectfont` ist hier nicht nötig, die Auswahl wird sofort aktiv.

NFSS *Low-level commands*

Der "normale" LaTeX-Benutzer bleibt in der Regel weitgehend von den NFSS *Low-level commands* verschont, da diese Kommandos im Quelltext keine Verwendung finden. Die

Tabelle 10.5: Vorgegebene Schriftatttribute

Befehle	**Standard**	**Setzt**
`\encodingdefault`	`OT1`	`\normalfont, \textnormal{...}`
`\familydefault`	`\rmdefault`	`\normalfont, \textnormal{...}`
`\rmdefault`	`cmr`	`\rmfamily, \textrm{...}`
`\ttdefault`	`cmtt`	`\ttfamily, \texttt{...}`
`\sfdefault`	`cmss`	`\sffamily, \textsf{...}`
`\seriesdefault`	`m`	`\normalfont, \textnormal{...}`
`\mddefault`	`m`	`\mdseries, \textmd{...}`
`\bfdefault`	`bx`	`\bfseries, \textbf{...}`
`\shapedefault`	`n`	`\normalfont, \textnormal{...}`
`\updefault`	`n`	`\upshape, \textup{...}`
`\itdefault`	`it`	`\itshape, \textit{...}`
`\scdefault`	`sc`	`\scshape, \textsc{...}`
`\sldefault`	`sl`	`\slshape, \textsl{...}`

Tabelle 10.6: NFSS *Mid-level commands*

Befehle	**Beispiele**
`\fontencoding`	`\fontentcoding{OT1}`
`\fontfamily`	`\fontfamily{cmr}`
`\fontseries`	`\fontseries{b}`
`\fontshape`	`\fontshape{it}`
`\fontsize`	`\fontsize{12}{14}`
`\usefont`	`\usefont{U}{eur}{b}{n}`
`\selectfont`	`\selectfont`

`.fd`-Dateien (*font definition*) haben die Kommandos dieser Ebene als Inhalt, da hierduch eine Schrift mit ihren fünf Attributen für das NFSS bekannt gemacht wird. Die verschiedenen Metrikendateien bekommen ihren abkürzenden Familiennamen, Serie und Gestalt zugeteilt, welche dann über die *High-level-* und *Mid-level commands* angesprochen werden können. Da die `.fd`-Dateien automatisch von *fontinst* erstellt werden, wird hier auch nicht weiter auf den Syntax der *Low-level commands* eingegangen.

10.2 virtual Fonts

10.2.1 Entstehung

Die Einführung von Virtual Fonts in TEX und damit auch LATEX geht auf einige Anpassungen David Fuchs' am in Stanford vorhandenen DVI-Prozessor zurück. Dieser ermöglichte es zwar, für die Ausgabe von TEX-Dokumenten auf den dortigen Postscript-Druckern entsprechende Schriften zu benutzen, das Encoding dieser Schriften war aber

ein völlig anderes als der übliche Standard für LaTeX-Schriftarten.

Da die Foundries natürlich keine Schriften für TEX anboten, war dieses Problem zunächst oft umgangen worden, indem man die Dokumente gleich mit für den Font spezifischem Encoding verfasste, oder mit Makros zumindest dieses ausgab.

Die von Fuchs verwendete Methode dagegen erlaubte es, das Wissen über die Zuordnung von TEX-Zeichen zu den tatsächlichen Zeichen erst im letzten Schritt, der Umwandlung von DVI in das Druckerformat (in seinem Fall Adobe Postscript) zu nutzen, indem der Konverter von Zuordnungstabellen gebrauch macht.

Dieses Verfahren schlug sich generalisiert im Begriff der virtuellen Schriften nieder:

> All we need is a good way to specify a mapping from TEX's notion of a font character to a device's capabilities for printing. Such a mapping was called a "virtual font" by the AMS speakers at the TUG meetings this past August.[5]

Das virtual font format, um diese Zuordnung zu spezifizieren, wurde dann von Knuth festgelegt, mit der Aufforderung an die Entwickler der DVI-Konverter, die Unterstützung dieses Verfahrens auch in ihre Programme einfliessen zu lassen.

10.2.2 Möglichkeiten

Die Möglichkeiten dieses Formats gehen weit über die der 1:1-Übersetzung von Zeichen hinaus – dies wird dadurch möglich, dass das Zuordnungsziel eines einzelnen Zeichens nicht nur ein anderes Zeichen sein kann, sondern alles, was auch zur Zusammenstellung von Seiten in DVI-Dateien erlaubt ist, also auch mehrere Zeichen, primitive Linien und Ausgabeformatspezifische `\special`s.

10.2.3 Anwendungen

Diese erweiterten Möglichkeiten haben neben der ursprünglichen Verwendung, der Anpassung von Schriften mit fremden Encodings für die Benutzung mit DVI-Prozessoren[6] auch noch zu einigen anderen Anwendungen geführt.

Eine beschreibt Knuth auch gleich in dem oben zitierten TUGboat-Artikel: Die Erstellung von „Preview-Schriftarten", also Schriftarten, die zwar die gleichen Metriken (Größenangaben zu Zeichen und Abständen) verwenden, wie z.B. eine kommerzielle Postscript-Schriftart, aber nur auf den Vorrat der vorhandenen Schriftarten zurückgreifen – so kann man auf Anzeigegeräten oder Druckern „probedrucken", die gar nicht die Möglichkeiten des Zielsystems aufwiesen.

[5] [Knu90]

[6] pdfLATEX enthält auch einen DVI-Prozessor, dieser ist nur als Programmteil und nicht als eigenständiges Programm implementiert

Eine weitere Verbesserung, die Knuth in seinem Artikel anführt ist die Möglichkeit, Zeichen mit Akzenten aus dem Vorrat der normalen Zeichen und der Akzente zu bilden.

Einige weitere Anwendungsmöglichkeiten:

- Mischen von Schriftarten (häufig verwendet, um eine „Experten"-Schriftart benutzen zu können)
- beliebige Anpassung von Kerning- und Metrikinformationen
- Größenanpassung an andere Fonts
- Preview-Schriftarten/„dreckige" Tricks
  - „poor man's bold": Nachbildung von Fettschrift durch versetztes „Überdrucken"
  - „poor man's oldstyle figures": Mediävalziffern (0123456789), die keine Expertenschriftart benötigen[Cha99]
  - zusätzlicher Schnitt für unter-/durchgestrichen (z.B. mit den primitiven Linien oder mit Postscript-`\specials`)
  - uvm.
- besondere visuelle Effekte

Die meisten dieser Anwendungsmöglichkeiten werden umfassend in [Hoe98a] beschrieben.

10.3 Ein einfaches Beispiel für virtuelle Fonts mit `fontinst`

10.3.1 Vorgehen/Technischer Überblick

Da die wesentlichen Züge der Fontinstallation und die verwendeten Dateien schon im ersten Teil behandelt werden, bleibt nur, erneut darauf hinzuweisen, dass mit der Verfügbarkeit des `fontinst`-Pakets bzw. Programms eigentlich keine Notwendigkeit mehr besteht, Dateien zu editieren, die keine LATEX-ähnliche Syntax aufweisen, auch die meisten Konvertierungsschritte werden automatisch vorgenommen.

Für eigene Experimente ist es auf Multiuserbetriebssystemen wie Linux ratsam, eine eigene `texmf`-Hierarchie z.B. in seinem `home`-Verzeichnis anzulegen. Bei der teTEX-Distribution ist z.B. das Verzeichnis `$HOME/texmf` schon eingestellt. Bei einigen Distributionen muss der Befehl `texhash` ausgeführt werden, damit dort abgelegte Dateien gefunden werden. Meist werden die von diesem Befehl erzeugten Dateilisten nur zur schnelleren Suche genutzt, wenn sie vorhanden sind, sollten dann allerdings auch richtig sein.

Ziel ist es, aus einer existierenden Metrikbeschreibung und einer „high level"-Beschreibung der durchzuführenden Aktion eine Metrikbeschreibung des virtuellen Fonts für LaTeX

und eine „low level“ Beschreibung (ähnlich DVI, aber noch lesbar) der einzelnen Zeichen des virtuellen Fonts für den DVI-Treiber herzustellen.

Dazu wird dem fontinst-Programm (das eigentlich nur ein LaTeX ist, welches fontinst.sty automatisch vorlädt) eine Installationsanweisung gegeben, nach der ggf. (also in den Fällen, in denen nicht z.B. nur eine Konvertierung von einem bekannten Encoding in das andere durchgeführt werden muss) weitere mtx-Dateien geladen werden, die die Bildung des neuen Fonts beschreiben.

Von allen verwendeten realen Fonts muss ferner eine pl-Datei verfügbar sein, meist sind im System nur die Metriken verfügbar, die zunächst mit tftopl konvertiert werden müssen.

Nach dem fontinst-Lauf müssen noch eine eigene tfm- und eine vf-Datei erzeugt werden, diese werden dann entsprechend dem *TDS* in die texmf-Hierarchie kopiert.

10.3.2 Beispielcode

```
% shaded.fontinst.tex
\input fontinst.sty % Nur nötig, wenn Aufruf mit
                    % latex shaded.fontinst.tex
                    % möglich sein soll
\installfonts
\installfamily{OT1}{shaded}{}
\installfont{shaded}
            {cmr10,shaded}%
            {OT1}{OT1}{shaded}{m}{n}{}
\endinstallfonts
\bye

\relax
% shaded.mtx
Based on a Usenet Article by Vincent Zoonekynd <zoonek@math.jussieu.fr>
Message-ID: <yisaehisu4w.fsf@borel2.institut.math.jussieu.fr>
Modified by Henrik Paulini <3paulini@informatik.uni-hamburg.de>
-----------------------------------------------------------------------
Chaque caractère est d'abord tracé legèrement décalé, épaissis, en
gris : cela donne un effet d'ombre.

% Alles vor \metrics ist ein Kommentar
\metrics
% Wir definieren hier die Funktion "doit"
\def\doit#1{%
  \ifisglyph{#1}\then
% (\resetglyph...\endsetglyph) Definition eines neuen Glyphen
```

```
    \resetglyph{#1}
% (\push...\pop) Zeichenposition und -breite erhalten
      \push
% Verwendung von Postscript-Befehlen
        \glyphspecial{ps:
          gsave
          10 10 translate
          .8 .8 .8 setrgbcolor }
%       \glyphspecial{PDF:.8 .8 .8 1 k}
% mir sind nicht alle benötigten PDF-Specials bekannt
        \push\moveup{0}\movert{0}\glyph{#1}{1000}\pop
        \push\moveup{30}\movert{0}\glyph{#1}{1000}\pop
        \push\moveup{30}\movert{30}\glyph{#1}{1000}\pop
        \push\moveup{0}\movert{0}\glyph{#1}{1000}\pop
        \push\moveup{0}\movert{30}\glyph{#1}{1000}\pop
                      % Verschieben des Zeichens
        \glyphspecial{ps: grestore }
      \pop
      \glyph{#1}{1000}
                      % Originalzeichen zeichnen
    \endsetglyph
  \fi
}
% Wir führen "doit" für jedes Zeichen aus
\input glyphes2
```

Zusätzlich benötigen wir noch die in `shaded.mtx` einzubindende glyphes2.tex, die einen Aufruf von `\doit` für alle zu veränderten Glyphen enthält. Diese kann z.B. auf einem teTeX-System so erstellt werden:

```
sed -n 's/^\\\(unfakable\|showglyph\)/
       \\doit/p' `kpsewhich latin.mtx` >glyphes2.tex
```

Die Datei sieht dann so aus:

```
\doit{Gamma}
\doit{Delta}
\doit{Theta}

⋮

\doit{germandbls}
\doit{lslashslash}
```

10.3.3 Ergebnis

Der installierte Font kann folgendermaßen verwendet werden:

```
\usefont{OT1}{xshaded}{m}{n}\selectfont
\huge{Schatten}
\normalfont
```

Kapitel 11

Diagramme mit dem xy–pic-Paket

Julam Do

Vortrag vom 23.1.2004

11.1 Einleitung

Als typischen Vertreter der vielen in Latex integrierten Sprachen wird in diesem Kapitel das Paket von Kristoffer H. Rose und Ross Moore genauer betrachtet. Das xy-pic Paket ist eine eigene mit Tex-Makros implementierte Sprache zum beschreiben von Bildern. Diese Bildbeschreibungssprache hat einen Objekt orientierten Charakter. Das Makropaket von xy-pic ist modular aufgebaut, ermöglicht also das Einbinden von sogenannten Features wie curve, line oder frame. Weitere Features sind cmtip, tips, rotate, color matrix, arrow oder graph. Diese Features erweitern die Bildbeschreibungssprache um zusätzliche Befehle die man benutzen kann für spezielle Anwendungen.

11.2 Nachteile

xy-pic ist dabei ein Paket, dass sehr komfortabel ist, aber nicht ganz trivial. Darstellen lassen sich Diagramme jeglicher Art, wobei vieles nur für denjenigen interessant ist, für den es sich lohnt viel Mühe in das erlernen der Syntax zu investieren. Die Bildbeschreibungen von xy-pic übertreffen in Hinsicht auf Kryptik, die schlimmsten C- und Assemblerprogramme, da die Syntax fast nur aus Sonderzeichen besteht. Dies erfordert somit viel Einarbeitungszeit und bedarf auch nach längerer Beschäftigung das gelegentliche nachschlagen von Befehlen. Die Sprache des xy-pic Paketes hat sich in der Vergangenheit leider oft sehr stark verändert, so dass ältere Bildbeschreibungen nicht mehr verarbeitet werden konnten.

11.3 Konzept

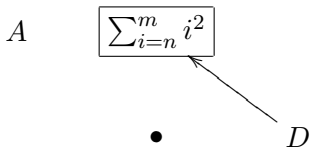
Eine Integration der Grafik in das DVI-Dokument erfolgt mittels spezieller Zeichensätze, die Linien in verschiedenen Steigungen und Symbole wie Pfeile in verschiedenen Ausfüh-

rungen enthalten. Somit können nicht alle, sondern nur einige diskrete Winkel dargestellt werden, welches beim Ausdruck dann zu störenden Treppen werden kann. [Kor98]

11.4 Anwendung

Die Erweiterungen von xy-pic stellen kleine Spezialsprachen für besondere Anwendungen bereit und ermöglichen somit sehr kurze Bildbeschreibungen. Es gibt leider nicht für jede Anwendung ein spezielles Feature, so zum Beispiel gibt es keine Erweiterung, die speziell zum darstellen von Bäumen benutzt werden kann. Der Anwender muss alle Knoten in selber anordnen und dann umständlich die passenden Kanten dazu zeichnen. Dies ist mithilfe von xy-matrix jedoch sehr leicht hinzukriegen. Die xy-matrix wird speziell für Darstellungen verwendet die Matrix ähnliche Strukturen vorweisen. Sehr leicht lassen sich zum Beispiel Baum-, Automatendiagramme darstellen, welche in der theoretischen Informatik benutzt werden. Ein gutes Beispiel ist das bilden von Resolventen in dem F-Zyklus. Hier ein einfaches Baumdiagramm:

```
\xymatrix{
A &*+[F]{\sum_{i=n}^m {i^2}} \\
  &{\bullet} & D \ar[ul] }
```



[Ros99]

An der Anordnung derSymbole ist die Matrix ähnliche Struktur zu erkennen.

Eingebunden wird das xy-pic Paket mit dem Befehl `\usepackage{xy}` und zwar am Anfang des Dokumentes.

Die xy-matrix Umgebung wird dann mit dem Befehl `\xymatrix{...}` gestartet. [Ros99]

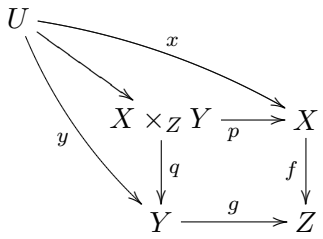
11.5 Positionen

Einträge in einer Reihe werden mit dem `&` Zeichen getrennt. Ganze Zeilen werden dabei mithilfe des `\\` getrennt.

Um nun die Anordnung der Objekte innerhalb der Matrix richtig darzustellen und Ziele festzulegen, braucht man Positionsangaben. Wie werden diese gemacht? Es gibt verschiedene Möglichkeiten Positionen in xy-matrix darzustellen. Eine dieser Möglichkeiten ist mithilfe des [hop] Formats. In den eckigen Klammern kommt die Positionsangabe, welche durch eine Sequenz der Buchstaben u, d, l und r. Diese stehen für up, down, left

und right. Ein Sprung nach unten, oben, links oder rechts entspricht dabei jeweils einen Zeilen,- oder Spalten Abstand innerhalb der Matrix.

```
U \ar@/_/[ddr]_y \ar[dr] \ar@/^/[drr]^x \\
& X \times_Z Y \ar{d}^q \ar[r]_p
& X \ar[d]_f \\
& Y \ar[r]^g & Z }
```



[RM99]

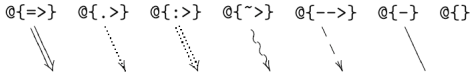
11.6 Alternativen

Eine andere Möglichkeit für die Positionierung von Objekten in einer Matrix ist das [r,c] Format. Die Buchstaben r und c sind in diesem Falle integer Zahlen und stehen für row und column. Dieses Format ist vergleichbar mit dem [hop] Format. zum Beispiel entspricht [1,2] im [r,c] Format der Positionsangabe [ddr] im [hop] Format. Positive Zahlen für r bedeuten im [hop] Format down, dementsprechend negative Zahlen versetzen die Position nach oben. [-2,0] entspricht also [uu] im [hop] Format. [Ros99]

11.7 Pfeile

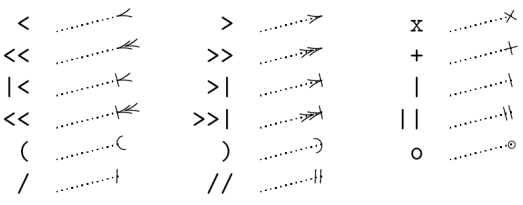
Ein pfeil in einem xy-pic Diagramm ist ein Term für eine gezeichnete Dekoration zwischen zweier Matrix Strukturen. Dekorationen sind Objekte die nicht an den aufeinander folgenden Positionsbeschreibungen der Matrix maschen liegen. Jeder Pfeil wird immer von ihrem Startpunkt aus definiert. Dies nennt man base entry. Die Ziele der Pfeile werden als target entry bezeichnet. [RM99]

Mit dem Befehl `\ar@style` kann man die Art des Pfeilrumpfes abändern. Hier sind die die gebräuchlichsten:



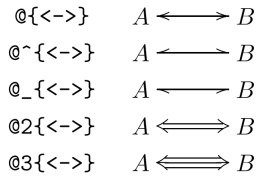
11.8 Pfeilspitzen

Genau wie bei den Pfeilrümpfen kann man auch die Pfeilspitzen abändern,indem man die gewünschte Syntax für die Spitze benutzt. Hier sind die gebräuchlichsten aufgelistet.



[Ros99]

Die Darstellung von Doppelpfeilen

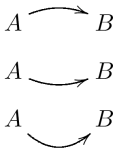


[Ros99]

Manchmal ist es erwünscht einen Pfeil zu biegen, um einem anderem Eintrag auszuweichen. Dies tut man mithilfe folgender Befehle.

- `@/^/`
- `@/_/`
- `@/_1pc/`

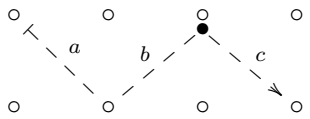
Das Hochzeichen biegt den Pfeil nach oben. Der Unterstrich biegt den Pfeil nach unten. Im drittem Beispiel ist eine Grössenangabe angegeben, die die Stärke der biegung bestimmt. Dies sieht dann folgenderma/ssen aus



[Ros99]

Pfeile können auch über mehrere Ziele gehen. Um somit das Zeichnen von mehr als nur einen Pfeil ohne die Pfeilspitze zu vermeiden kann man mithilfe von dem Hochkomma ’ Pfeile über mehrere Ziele hin verbunden werden. Das Quote Zeichen gefolgt von einem Ziel steht für jeweils einen Zwischenpunkt des Pfeils. Das endgültige Ziel hat kein Quote Vorzeichen.

```
\xymatrix{
  {\circ}
\ar@{|-->} '[dr]^a
     '[rr]+D*{\bullet}^b
    [drrr]^c
& {\circ} & {\circ} & {\circ} \\
{\circ} & {\circ} & {\circ} & {\circ} }
```



[Ros99]

11.9 Labels

Labels werden immer senkrecht angeordnet und wird bei der Beschriftung der Pfeile unterschieden in oberhalb des Pfeiles und unterhalb des pfeiles. Festgelegt wird dieses mithilfe vom `^` zeichen und vom Unterstrich `_`. Von der Syntax her, also genau wie beim biegen der Pfeile.

```
$\xymatrix@1{
X\ar[r]^a_b & Y & Z\ar[l]^A_B }$
```

$$X \underset{b}{\overset{a}{\longrightarrow}} Y \underset{A}{\overset{B}{\longleftarrow}} Z$$

Das `@1` ist ein spezieller code zur Optimierung der Platzierung in einer Linie [Ros99]

Wenn die Grösse eines Diagramms stark variiert, gibt es Optionen für die Positionierung der Labels. Mithilfe des Minus Zeichens - wird die Label Positionierung auf die Mitte des Pfeils verlegt.

Ohne Minus Zeichen:

$$A \times B \times C \times D \overset{+}{\longrightarrow} B$$

```
$\xymatrix@1{
A\times B\times C\times D \ar[r]^-{+} & B}$
```

$A \times B \times C \times D \xrightarrow{+} B$ [Ros99]

11.10 Pfeile brechen

Um Pfeile zu brechen und Labels einzufügen benutzt man das | Symbol.

```
$\xymatrix@1{A\ar[r]|f&B}$
```

ergibt folgendes:

$A \xrightarrow{f} B$

Wenn man nur einen leeren Bruch haben will, benutzt man `\hole`

```
$\xymatrix@1{A\ar[r]|\hole&B}$
```

$A \longrightarrow B$ [Ros99]

Um ein Label ausserhalb der normalen Matrizen Maschen zu setzen, kann man auch den Trick eines nicht sichtbaren Pfeilrumpfes nutzen indem man einen nicht sichtbaren Pfeil bricht und in dessen Mitte ein Label einfügt. Wobei Labels nicht nur Buchstaben oder Ziffern sein können, sondern auch komplexere Formeln beinhalten können.

```
\xymatrix{\ar @{} [dr] |{=}
A \ar[d] \ar[r] & B \ar[d] \\
B \ar[r] & C }
```

$$\begin{array}{ccc} A & \longrightarrow & B \\ \downarrow & = & \downarrow \\ B & \longrightarrow & C \end{array}$$

[Ros99]

11.11 Optimierung

Es kann sehr lange dauern einige Diagramme zu kompilieren, aufgrund der vielen Zeichen-Operationen die LaTeX machen muss. Dies kann verhindert werden, indem man `\CompileMatrices` an den Anfang des Dokumentes implementiert. Dies erzeugt temporäre Dateien die angelegt werden, welche die vorkompilierten Diagramme enthalten. Bei einer Änderung, werden diese automatisch neu kompiliert. Ein Problem bei dem vorkompilieren gibt es aber. Es kann dazu führen, dass einige Diagramme nicht funktionieren oder falsch dargestellt werden. Man kann für das jeweilige Diagramm mit dem Befehl `\xymatrixnocompile` anstatt von `\xymatrix` die Vor-kompilierung aufheben.

11.12 Vergleich

	Abstraktionsgrad der grafischen Elemente	Definition eigener Abstraktionen	Realisierung
Xy-pic	mittel	nicht möglich	LaTeX Markosammlung
METAPOST	mittel	knifflig	externes Programm
PIC	hoch	exterm kompliziert	externes Programm
functional PostScript	mittel	sehr einfach	eingebettete Sprache
mlPictTEX	sehr hoch	sehr einfach	eingebettete Sprache
TkGofer	sehr hoch/speziell	sehr einfach	eingebettete Sprache
Pictures	hoch	sehr einfach	eingebettete Sprache

- Wo bekomme ich xy-pic?
  - `ftp://ftp.diku.dk/diku/users/kris/tex`
  - ftp://ftp.mpce.mq.edu.au/pub/maths/tex
- weitere Informationen auf der xy-pic Homepage
  - http://www.brics.dk/ krisrose/Xy-pic.html
- User guide von Kristoffer Høgsbro Rose
  - http://www.iti.cs.tu-bs.de/TI-INFO/velebil/teaching/SEP/xyguide.ps

Kapitel 12

Figures and Float placements

Sebastian Meiser, Andreas Schacht

Vortrag vom 30.1.2004

12.1 Einleitung

Normalerweise bricht LaTeX Texte, um sie günstig auf Seiten zu verteilen, so dass kein unnötiger Freiraum entsteht. Bei Bildern oder Tabellen wäre ein solches Verhalten unerwünscht, da man diese möglichst als ganzes betrachten möchte. Um dies zu erreichen, lässt LaTeX Abbildungen oder Tabellen, die nicht mehr auf eine Seite passen, z.B. an den Anfang oder das Ende der nächsten Seite (oder noch weiter) gleiten um den so entstandenen Freiraum mit Text zu füllen. Dazu stellt es die beiden gleitenden (floating) Umgebungen `figure` und `table` bereit.

Für Benutzer herkömmlicher Textverarbeitungen ist dieses Verhalten ungewohnt, da sie normalerweise die Bilder selbst platzieren. Um so aber ein schönes Seitenlayout ohne große Freiflächen auf den Seiten zu erreichen müssen die Benutzer häufig die Abbildungen umplatzieren, was gerade bei größeren Dokumenten mit vielen Abbildungen schnell lästig werden kann.

Gerade diese Benutzer werden schnell nervös, wenn eine Abbildung nicht genau dort erscheint, wo sie eingebunden wurde. Man sollte sich aber daran gewöhnen, dass das Platzieren von Abbildungen die Aufgabe von LaTeX ist. Um dieses zu unterstützen, sollte man Formulierungen wie „Diese Abbildung ...“ oder “Folgende Abbildung ...“ vermeiden und stattdessen im Text auf die Abbildung verweisen, z.B. mit „In Abbildung 11 sieht man ...“, da man nicht sicher sein kann, dass die Abbildung auch wirklich dort erscheint wo man sie eingebunden hat.

12.2 Grafik in LaTeX

12.2.1 Einbinden von Grafik

Um Grafiken in sein LaTeX–Dokument einzubinden, gibt es mehrere Möglichkeiten. Wir stellen hier die Grafikeinbindung durch das `graphicx`–Paket vor.

Mit dem Befehl

```
\usepackage{graphicx}
```

wird das Paket im Vorspann eingebunden. Nun kann man mittels

```
\includegraphics[option]{Dateiname}
```

ein Bild im eps–Format eingefügen. Einige oft benutzte Optionen werden in Tabelle 12.1 dargestellt.

Tabelle 12.1: Optionen von includegraphics

Option	Beschreibung
height	Höhe der Grafik ab der Baseline in beliebiger LaTeX–typischer Längenangabe.
totalheight	Gesamthöhe der Grafik in beliebiger LaTeX–typischer Längenangabe.
width	Breite der Grafik in beliebiger LaTeX–typischer Längenangabe.
scale	Faktor der Skalierung der Grafik.
angle	Drehen der Grafik in Grad gegen den Uhrzeigersinn.
origin	Gibt den Ausgangspunkt der Drehung an.

12.2.2 Drehung

Wird eine Grafik rotiert, können die in Abbildung 12.1 dargestellten Punkte als Ausgangspunkte angegeben werden.

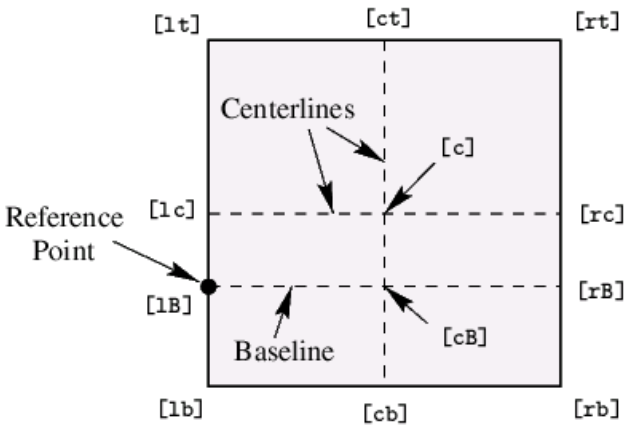


Abbildung 12.1: Ausgangspunkte der Drehung

Werden zusätzlich noch Angaben zur Größe der Grafik gemacht, muss darauf geachtet werden, das die Parameter von links nach rechts ausgewertet werden. Es macht also einen Unterschied, ob man erst die Grafik dreht und dann z.B. die Höhe der Grafik angibt, oder umgekehrt.

12.3 Die Figure- und Table-Umgebung

12.3.1 Einbindung

Um ein Gleitobjekt zu erstellen, wird es in eine Figure- bzw. einer Table–Umgebung eingebunden. Wie die Namen schon vermuten lassen, sind diese Umgebungen insbesondere für Abbildungen bzw. Tabellen ausgelegt. Es lassen sich damit aber beliebige Objekte einbinden. Zum Beispiel wurde die Abbildung 12.1 wie folgt in das Dokument eingebunden:

```
\begin{figure}[htbp]
\centering
\includegraphics{bildorigin}
\caption{Ausgangspunkte der Drehung}
\label{grfrot}
\end{figure}
```

Die Optionen werden im Abschnitt 12.3.6 erläutert. Das `\centering`–Kommando bewirkt, dass die Grafik zentriert dargestellt wird. Der Befehl `\caption` enthält die Bildunterschrift.

12.3.2 Referenzieren

Das optionale `\label{Marke}`–Kommando, dass direkt nach dem `\caption`–Kommando folgen muss, dient zum Referenzieren der Grafik bzw. Tabelle. So lässt sich einfach mit dem Befehl `\ref{Marke}` die Nummer und mit `\pageref{Marke}` die Seitennummer der Grafik/Tabelle einfügen. Will man also Bezug auf eine Tabelle nehmen, könnte dies wie folgt aussehen:

```
...Tabelle \ref{Marke} auf Seite \pageref{Marke}...
```

12.3.3 Verzeichnisse

Soll ein Verzeichnis der im Dokument enthaltenen Grafiken erstellt werden, so erreicht man das mit dem Kommando

```
\listoffigures
```

und bei Tabellen

```
\listoftables
```

Die Grafiken bzw. Tabellen werden mit dem Text ins Verzeichnis aufgenommen, der im jeweiligen `caption`–Kommando steht. Wenn im Verzeichnis ein anderer Text stehen soll, kann man dem `\caption`–Kommando den Verzeichnistext als Option mitliefern.

```
\caption[Verzeichnistext]{Text im Dokument}
```

12.3.4 Table vs. Figure

Ein Unterschied beider Umgebungen ist, dass sie eine unterschiedliche Beschriftung haben. So wird einer Bildunterschrift in der Figure–Umgebung z.B. das Label „Abbildung X:“ vorangestellt und bei der Table–Umgebung beispielsweise „Tabelle X:“.

Desweiteren ist die Nummerierung der Objekte in diesen Umgebungen unabhängig voneinander. So erhält beispielsweise eine Tabelle, die nach 3 Bildern eingebunden wird nicht die Nummer 4, sondern die Nummer 1 (falls es die erste Tabelle im Dokument ist).

Ausserdem haben beide Umgebungen jeweils eigene Verzeichnisse, wie vorher schon erwähnt.

Im Gegensatz zu den Bildern können Tabellen auch umgebrochen werden.

12.3.5 unprocessed Floats

Kann ein Gleitobjekt nicht an der gewünschten Stelle eingefügt werden, kommt es solange in einen Queue. Man spricht von einem „unprocessed Float“. Das Objekt bleibt solange im Queue, bis eine geeignete Stelle im Dokument gefunden wurde. Gleitobjekte, die später im Dokument vorkommen, können nicht gesetzt werden, solange der Queue noch Objekte enthält. Damit bleibt die Reihenfolge der Objekte erhalten. Diese Objekte werden dann auch in den Queue, der bis zu 18 Objekte beinhalten kann. Werden mehr als 18 Objekte in den Queue geschrieben, bricht die Kompilierung mit der Fehlermeldung „too many unprocessed Floats“ ab.

Damit das Objekt nicht an eine Stelle gesetzt wird, die zu weit vom darauf beziehenden Text entfernt ist, kann man das Kommando `\clearpage` einfügen. Der Befehl bewirkt, dass alle Objekte, die noch im Queue sind, sofort gesetzt werden und es wird eine neue Seite begonnen.

Da dies nicht immer schöne Ergebnisse erzielt, ist es meist besser, das Kommando `\FloatBarrier` zu benutzen. Es bewirkt das gleiche wie `\clearpage`, beginnt jedoch keine neue Seite, so dass keine großen Lücken entstehen.

Meistens will man die Objekte im dem darauf beziehenden Abschnitt haben. Mit

```
\usepackage[section]{placeins}
```

wird das `\section`–Kommando redefiniert, so dass das `\FloatBarrier`–Kommando vor jedem neuen Abschnitt eingefügt wird.

12.3.6 Platzieren einer Figure–Umgebung

Um die Platzierung anzupassen, hat die `figure` Umgebung einen optionalen Parameter. Diesem kann man die Folgenden Werte zuweisen :

- h : *here* Es wird versucht die Abbildung dort zu patzieren wo der Befehl im Text steht. Kann nur ausgeführt werden, wenn an dieser Stelle genug Platz vorhanden ist.
- t : *top* Platziert die Abbildung im Kopf der Seite.
- b : *bottom* Platziert die Abbildung unten auf der Seite.
- p : *float page* Platziert die Abbildung auf einer Seite die nur gleitende Abbildungen enthält.

Diese Werte können auch kombiniert angegeben werden, wobei die Reihenfolge in der sie angegeben werden keine Rolle spielt, da LaTeX sie immer in der Reihenfolge h-t-b-p verarbeitet. Wenn kein optionaler Parameter angegeben wird, wird per Voreinstellung `[tbp]` als Platzierungsparameter angenommen. Desto mehr Werte man mitgibt, desto besser kann LaTeX die Abbildung platzieren, da man weniger Einschränkungen vorgibt. Wenn man nur einen einzelnen Wert angibt, kann es zu Problemen kommen (siehe 12.3.5).

12.3.7 Anpassen der Umgebung

Mit den folgenden Befehlen lässt sich das Verhalten der Umgebung beeinflussen :

- `topnumber / bottomnumber`, gibt die maximale Anzahl von Gleitobjekten oben bzw. unten auf einer Seite an.

  Beispiel : `{setcounter{topnumber}{3}}`
- `totalnumber`, gibt die maximale Anzahl von Gleitobjekten auf der ganzen Seite an (lässt sich auch mit `\setcounter` setzen).
- `\topfraction / \bottomfraction`, gibt den Bruchteil einer Seite an, bis zu dem Gleitobjekte oben bzw. unten angeordnet werden sollen.

  Beispiel : `\renewcommand{\topfraction}{0.95}`

- `\textfraction`, gibt den Bruchteil einer Seite an, der mit Text belegt werden können muss.

  Beispiel : `\renewcommand{\textfraction}{0.05}`

- `\floatpagefraction`, gibt den Bruchteil einer Seite, die für Gleitobjekte benutzt wird, an, der erreicht werden muss, bevor eine neue Seite angefangen wird. Lässt sich auch mit `\renewcommand` setzen.
  (Die Voreinstellung für diesen Wert liegt bei 50%, sodass schon bei einem Objekt mit 51% der Seitengröße eine neue Seite angefangen wird, wobei viel unschöner Freiraum entsteht. Es lohnt sich also diesen Wert zu erhöhen.)

- `\floatsep / \textfloatsep`, gibt den vertikalen Abstand zwischen zwei Gleitobjekten bzw. zwischen einem Gleitobjekt und dem Text an.

  Beispiel : `\setlength{\floatsep}{1cm}`

- `\intextsep`, gibt den Abstand zwischen dem Text und Gleitobjekten, die mit der Option `[h]` platziert wurden, an.

- `\topfigrule / \botfigrule`, erzeugen eine horizontale Linie vor bzw. nach einem Gleitobjekt. Die Befehle sind Standardmäßig leer, lassen sich aber mit `\renewcommand` neu definieren, sodass sie eine Linie erzeugen.

- `\suppressfloats[<pos>]`, verhindert das erscheinen von Gleitobjekten auf einer Seite. Der Parameter **pos** gibt an, welche Gleitobjekte (oben oder unten) auf der Seite unterdrückt werden sollen. Ohne Parameter wird das nachfolgende Gleitobjekt auf der Seite unterdrückt, wobei Gleitobjekte, die vorher definiert, aber noch nicht verarbeitet worden sind, trotzdem erscheinen können.

- Mit dem Paket `flafter` kann man verhindern, dass Gleitobjekte vor der Stelle erscheinen wo sie definiert wurden. Dazu muss man das Paket einfach nur einbinden.

12.3.8 Captions

Wird der Befehl `\caption` vor dem Objekt in der Umgebung eingefügt, so steht die Bildunterschrift über dem Objekt. Mit den Befehlen

```
\setlength{\abovecaptionskip}{Abstand}
```

und

```
\setlength{\belowcaptionskip}{Abstand}
```

kann der Abstand über und unter der Bildunterschrift verändert werden. Der Abstand kann in jede von LaTeX akzeptierte Längenangabe angegeben werden werden.

Mit dem Befehl

```
\renewcommand{\figurename}{Label}
```

kann das Label, dass vor der Bildunterschrift steht, angepasst werden.

Bindet man das Paket `caption2` ein, so stehen einem eine Vielzahl von gestaltungsmöglichkeiten der Bildunterschrift zur verfügung. Das Paket wird mittels

```
\usepackage[options]{caption2}
```

eingefügt. Einige mögliche Optionen werden in der Tabelle 12.2 dargestellt.

Tabelle 12.2: Optionen von caption2

Stil	normal, center, flushleft, flushright, centerlast, hang, indent	Ändert die Ausrichtung.
Schriftgrösse	scriptsize, footnotesize, small, normalsize, large, Large	Ändert die Schriftgrösse.
Form	up, it, sl, sc	Ändert die Form (upright, italic, slanted, small caps). Betrifft nur das Label.
Schriftserie	md, bf	Ändert die Serie (medium, boldface) Betrifft nur das Label.
Schriftfamilie	rm, sf, tt	Ändert die Schriftfamilie (roman, sans serif, typewriter). Betrifft nur das Label.

Soll die Bildunterschrift eine bestimmte Breite haben, so kann man diese mit

```
\setcaption{Breite}
```

bestimmen. Die Breitenangabe kann in den von LaTeX akzeptierten Einheiten gemacht werden. Dies ist sinnvoll, wenn die Bildunterschrift genauso breit sein soll wie das Bild.

Eine weitere Möglichkeit ist der Befehl

```
\setcaptionmargin{Abstand}
```

Hiermit wird ein Abstand zum Seitenrand auf beiden Seiten gemacht.

Auch das Trennzeichen zwischen dem Label und der Bildunterschrift kann man Anpassen. Der Befehl

```
\renewcommand{\captionlabeldelim}{Zeichen}
```

setzt als Trennzeichen das im Befehl angegeben Zeichen.

12.4 Weitere Pakete

12.4.1 nofloat

Das Paket `nofloat` stellt eine neue Umgebung bereit, mit der man Objekte mit einer Beschriftung und Nummerierung genau wie bei einer Gleitumgebung versehen kann, die aber an der Stelle platziert werden an der sie definiert wurden (also keine Gleitobjekte sind). Dabei wird die Nummerierung einer anderen Umgebung fortgesetzt (z.B. `figure`).

Allgemeiner Aufruf :

```
\usepackage{nofloat}

...

\begin{nofloat}{< Umgebung deren Nummerierung fortgesetzt wird >}
        %...
\end{nofloat}
```

So wird im folgenden Beispiel die Nummerierung der `figure` Umgebung fortgesetzt :

```
\begin{nofloat}{figure}
  \includegraphics{bild}
  \caption{Ein Bild in einer nofloat Umgebung}
\end{nofloat}
```

12.4.2 float

Mit dem Paket `float` lassen sich eigene gleitende Umgebungen definieren. Diese Verhalten sich genauso wie `figure` und `table`. Man kann dem Paket folgende optionale Parameter zur Beeinflussung des Stils von gleitenden Umgebungen mitgeben :

- **plain** , ist der Standard-LaTeX-Stil mit der Beschriftung unter dem Objekt. (Voreinstellung)
- **plaintop** , wie `plain`, aber mit der Beschriftung über dem Objekt.
- **boxed** Um das Objekt wird eine Box gezeichnet, die Beschriftung wird unter das Objekt gesetzt.
- **ruled** Die Umgebung wird oben und unten von horizontalen Linien begrenzt, die Beschriftung wird über das Objekt gesetzt.

Neue Umgebungen erstellt man mit dem Befehl `newfloat` :

```
\newfloat{<Name>}{<pos>}{<Dateiendung>}[Gliederungsebene]
```

Der erste Parameter git den Namen der neuen Umgebung an, wobei dieser auch für die Objektbeschriftumg verwendet wird. Der Parameter <pos> gibt die Voreinstellung für die Positionierung an. Hier kann man die bekannten Werte zur Positionierung angeben. Mit <Dateiendung> legt man fest, wo die Objektlisten gespeichert werden sollen. Der Dateiname setzt sich zusammen aus dem Namen des Hauptdokuments und der angegebenen Dateiendung. Der optionale Parameter gibt die Gliederungsebene an, auf der die Nummerierung durchgeführt werden soll (z.B. `chapter` für die Nummerierung 1.1 bei Kapiteln).

Ausserdem stellt `float` noch folgende Befehle bereit :

- `\floatname{<Name>}{<Label>}`
- `\floatplacement{<Name>}{<pos>}`
- `\floatstyle{<Stil>}`
- `\restylefloat{<Umgebung>}`

Mit `\floatname` wird die Beschriftung der als **Name** angegebenen Umgebung in **Label** geändert. `\floatplacement` ändert nachträglich die Standardpositionierung. Mit `\floatstyle` kann man den Stil einer Gleitumgebung ändern, wobei die selben Parameter wie beim Einbinden des Pakets zur Verfügung stehen. Der so geänderte Stil gilt für alle nachfolgend neu definierten Umgebungen. Mit `\restylefloat` kann man den Stil einer schon definierten Umgebung auf den vorher mit `floatstyle` definierten Wert ändern.

Das folgende Beispiel definiert eine Listing Umgebung mit dem Stil **ruled** und gibt ein Programmlisting in einer gleitenden Umgebung aus:

```
\floatstyle{ruled}
\newfloat{Listing}{htp}{lol}[chapter]
\begin{Listing}
 < Hier darzustellender Quellcode in verbatim-Umgebung >
\caption{Ein Programmlisting in eigener Umgebung}
\end{Listing}
```

```
#include<iostream.h>
 int main ()
 {
   cout<<''Hello World'';
   return ;
 }
```

Listing 12.1: Ein Programmlisting in eigener Umgebung

Ausserdem stellt das Paket den Platzierungsparameter **[H]** bereit, durch den ein gleitendes Objekt genau an der Stelle platziert wird, an der es definiert wurde (es gleitet also eigentlich nicht mehr). Sollte an der Stelle nicht genug Raum frei sein, wird das Objekt an den Anfang der nächsten Seite verschoben und Leerraum eingefügt.

12.4.3 subfigure

Mit dem Paket `subfigure` kann man innerhalb einer `figure` oder `table` Umgebung mehrere Bilder oder Tabellen einbinden und mit einer Unternummerierung versehen (a,b,c ...).

Dazu stellt es die Befehle

```
\subfigure[<Listeneintrag>][<Caption>]{<Bild>}
\subtable[<Listeneintrag>][<Caption>]{<Bild>}
```

bereit.

Der erste optionale Parameter legt den Eintrag in die Objektliste fest. Mit dem zweiten kann man die Beschriftung angeben. Möchte man auf dieses Objekt verweisen, so muss der `\label`-Befehl in der Beschriftung benutzt werden. Man kann normal mit `\ref` auf das Objekt verweisen oder mit dem neuen Befehl `\subref`, der nur die Unternummerierung zurückgibt. Ausserdem gibt es noch den Befehl `\Subref`, der genauso arbeitet wie `\subref`, aber den Font `subcaplabelfont` verwendet.

Dem Paket kann man viele optionale Parameter zur Gestaltung der Beschriftung mitgeben, die denen des `caption2`-Pakets sehr ähnlich sind. Genauere Informationen findet man in der Dokumentation. Sollte das `caption2`-Paket nach dem `subfigure`-Paket geladen werden, so gelten die dort vorgenommenen Einstellungen auch für das `subfigure`-Paket.

Das folgende Beispiel setzt zwei Bilder nebeneinander mit eigener Unternummerierung :

```
\begin{figure}
 \centering
```

```
    \subfigure[Grafik1]{\includegraphics[width=4cm]{bild1}}
  \hspace{1cm}
    \subfigure[Grafik2]{\includegraphics[width=4cm]{bild2}}
  \caption{Zwei Grafiken in einer Umgebung}
\end{figure}
```

(a) Grafik1

(b) Grafik2

Abbildung 12.2: Zwei Grafiken in einer Umgebung

12.4.4 Auswahl an weiteren Paketen

Die nachfolgend aufgezählten Pakete werden hier nicht besprochen sondern nur ihr Aufgabengebiet kuz vorgestellt. Weitere Informationen findet man dann in den jeweiligen Paketdokumentationen :

- `eso-pic`
  Mit diesem Paket kann man auf jede Seite ein Hintergrundbild setzen.
- `rotating`
  Das Paket stellt eine Umgebung bereit, mit der sich Texte und Bilder drehen lassen.
- `subfloat`
  Stellt ähnlich wie `subfigure` eine Möglichkeit dar, Bilder und Tabellen mit einer Unternummerierung zu versehen. Allerdings werden hier die normalen `figure` und `table` Umgebungen innerhalb einer `subfloat` bzw. `subtables` Umgebung verwendet.
- `endfloat`
  Mit diesem Paket lassen sich gleitende Objekte an das Ende eines Kapitels verschieben.

Die vier folgenden Pakete dienen alle dazu, Objekte von Text umfliessen zu lassen :

- `picins`
- `picinpar`
- `wrapfig`
- `floatflt`

Weitere Informationen in den Paketdokumentationen.

Literaturverzeichnis

[Ber03] Berry, Karl: *Fontname: Filenames for TEX fonts*, 2003. – `http://www.ctan.org/tex-archive/info/fontname/fontname.pdf`

[Bir03] Birkenbach, B. *Einführung in LATEX*. 2003

[Car96] Carlisle, D.: A LATEX Tour, part 2: the Tools and Graphics distributions. In: *TUGboat* 17 (1996), Nr. 3, S. 321–326

[Ced93] Cederqvist, P.: *Version Management with CVS*, 1993. – `http://www.cvshome.org/docs/manual/`

[Cha99] Chang, Felix: *Poor man's oldstyle figures*. 1999. – `http://www.tug.org/archives/tex-fonts/msg00154.html`

[CR91] Couture, B. ; Rymer, J.: Discourse interaction between writer and supervisor: A primary collaboration in workplace writing. In: Lay, M.M. (Hrsg.) ; Karis, W.M. (Hrsg.): *Collaborative Writing in Industry: Investigations in Theory and Practice*. Baywood Publishing Company, 1991, S. 87–108

[Dra99] Drakos, Nikos: *The Latex 2HTML Tanslator*, 1999. – `http://www-texdev.ics.mq.edu.au/l2h/docs/manual/`

[EL90] Ede, L. ; Lunsford, A.: *Singular Texts/Plural Authors: Perspectives on Collaborative Writing*. Southern Illinois University Press, 1990

[GMS99] Goossens, Michael ; Mittelbach, Frank ; Samarin, Alexander: *The LATEX Companion*. Addison-Wesley, 1999

[Gnu99] Gnu: *TeX4ht: LaTeX and TeX for Hypertext*, 1999. – `http://www.tug.org/applications/tex4ht/mn.html`

[Gnu03] Gnu: *Download : GhostScript*. 2003. – `http://www.cs.wisc.edu/~ghost/`

[GR00] Goossens, Michael ; Rahtz, Sebastian: *Mit LATEX ins Web*. Addison-Wesley. ISBN 3-8273-1629-4, 2000

[Gün02] Günther, Karsten: *LATEX*. Mitp Verlag, 2002

[Hen04] Henderson, Bryan: *Download : netpbm*. 2004. – `http://sourceforge.net/projects/netpbm`

[Hob91] HOBBY, John: *A Users's Manual for MetaPost*, 1991

[Hoe98a] HOENIG, Alan: *TEX Unbound, LATEX & TEX Strategies for Fonts, Graphics, & More*. Oxford University Press, 1998

[Hoe98b] In: HOENIG, Alan: *TEX Unbound, LATEX & TEX Strategies for Fonts, Graphics, & More*. Oxford University Press, 1998, S. 130 – 256

[Hus02] HUSING, Carl. *Geschichte TEX und Metafont Postscript Vor- und Nachteile*. 2002

[Jac96] JACOBSEN, Dana: *BibTeX*. 1996. – `http://www.ecst.csuchico.edu/~jacobsd/bib/formats/bibtex.html`

[JMV98] JEFFREY, Alan ; MCDONNELL, Rowland ; VIETH, Ulrik: *Fontinst: Font installation software for TEX*, 1998. – `http://www.ctan.org/tex-archive/fonts/utilities/fontinst/doc/manual/`

[Jür95] JÜRGENS, Manuela: *LATEX – Eine Einführung und ein bißchen mehr ... (Chapter 29)*, 1995

[KM03] KOHM, M. ; MORAWSKI, J.-U.: *KOMA-Script - Eine Sammlung von Klassen und Paketen für LATEX 2ε*. Lehmanns, 2003

[Knu79] KNUTH, Donald E.: *TEX and Metafont*. Digital Press, American Mathematical Society, 1979

[Knu86a] KNUTH, Donald E.: *The Metafont Book*. Addison-Wesley, 1986

[Knu86b] KNUTH, Donald E.: *Metafont: The Programm*. Addison-Wesley, 1986

[Knu90] KNUTH, Donald: Virtual Fonts: More Fun for Grand Wizards. In: *TUGboat* 11 (1990), Apr, Nr. 1, S. 13–23. – `http://www.tug.org/tex-archive/info/virtual-fonts.knuth`

[Kop97] KOPKA, Helmut: *LATEXErgänzungen – Band 2*. Addison-Wesley, 1997

[Kop00] KOPKA, Helmut: *LATEX – Einführung Band 1*. dritte. Addison-Wesley, 2000

[Kor98] KORITTKY, Joachim: *Beschreibungssprachen für Grafiken*. Rheinische Friedrich-Wilhelms-Universität Bonn, 1998

[Kuh02] KUHN, Christian H.: *Die Anpassung von PostScript-Type-1-Schriften an den Gebrauch unter LATEX*, 2002

[Lam98] LAMPORT, Leslie: *LATEX - A Document Preparation System*. Addison-Wesley, 1998

[LLC04] LLC, ImageMagick S.: *Download : ImageMagick*. 2004. – `http://www.imagemagick.org/`

[Lor03] LORENZEN, Klaus F.: *BibTeX Styles*. 2003. – `http://www.haw-hamburg.de/pers/Lorenzen/bibtex/`

[MIh94] *How to use Bibtex*. 1994. – `http://cmtw.harvard.edu/Documentation/TeX/Bibtex/Example.html`

[Mös98] MÖSGEN, Peter: *Makeindex (Sachregister erstellen mit LaTeX)*. 1998. – `http://www1.ku-eichstaett.de/urz/schriften/makeidx.pdf`

[NJM01] NOEL, S. ; J.-M., Robert: Empirical Study on Collaborative Writing: What do co-authors do, use, and like? (2001)

[NN03] NIEDERMAIR, Elke ; NIEDERMAIR, Michael: *LaTeX - Das Praxisbuch*. Franzis' Verlag, 2003

[Pah03] PAHL, Martin: *TeX/LaTeX*. 2003. – `http://www.tnt.uni-hannover.de/soft/desk/word/tex/`

[Pak03] PAKIN, Scott: *The Comprehensive LaTeXSymbol List*. 2003. – http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf

[Per03] PERL: *Download : Perl*. 2003. – `http://www.perl.com`

[Rad03] RADICALEYE: *Download : dvips*. 2003. – `http://radicaleye.com`

[Rak] RAKITYANSKY, Damir: *Using True Type fonts with TeX(LaTeX) and pdfTeX(pdfLaTeX)*

[Reb00] REBBECHI, Donovan: *Font HOWTO (Chapter 9)*, 2000

[Rec97] RECKDAHL, Keith. *Using Imported Graphics in LaTeX2e*. 1997

[RM99] ROSE, Kristoffer H. ; MOORE, Ross. *xy-pic Reference Manual Version 3.7*. 1999

[RO03] RAHTZ, Sebastian ; OBERDIEK, Heiko: *Hypertext marks in LATEX: a manual for hyperref*, 2003. – `http://www.tug.org/applications/hyperref/manual.html`

[Ros99] ROSE, Kristoffer H. *xy-pic Users Guide Version 3.7*. 1999

[Sch02] SCHMIDT, Walter: *Schriften verwenden mit - LaTeX Teil 1: Grundlagen*, 2002

[SGB+91] SHARPLES, M. ; GOODLET, J. ; BECK, E. ; WOOD, C. ; EASTERBROOK, S. ; PLOWMAN, L. ; EVANS, W.: A Framework for the Study of Computer Supported Collaborative Writing / School of Cognitive and Computing Sciences, University of Sussex. 1991. – Forschungsbericht

[Sia02] SIART, Uwe: Die texnische Komödie. (2002)

[SKPH99] SCHMIDT, Walter ; KNAPPEN, Jörg ; PARTL, Hubert ; HYNA, Irene. *LaTeX 2e Kurzbeschreibung*. 1999

[SKPH03] SCHMIDT, W. ; KNAPPEN, J. ; PARTL, H. ; HYNA, I. *LaTeX 2ε - Kurzbeschreibung*. 2003

[Soc02] SOCIETY, American M.: *User's Guide for the amsmath Package (Version 2.0)*. 2002. – ftp://ftp.ams.org/pub/tex/doc/amsmath/amsldoc.pdf

[TDSTT03] ON A TEX DIRECTORY STRUCTURE (TWG-TDS), TUG Working G.: *A Directory Structure for TeX Files*. 2003. – `http://www.tug.org/tds/`

[Wri98] WRIGHT, David: *TeX Font Guide*, 1998

[Zoo] ZOONEKYND, Vincent: *Re: shadow font?*. – `news:yisaehisu4w.fsf@borel2.institut.math.jussieu.fr`, `http://groups.google.com/groups?selm=yisaehisu4w.fsf%40borel2.institut.math.jussieu.fr`