LaTeX statt Word – Professionelle Textverarbeitung für Anthropologen

Martin Dockner

9. September 2014

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Was genau ist LaTeX überhaupt? Wikipedia spuckt zu dem Begriff `Latex` folgende Liste aus:

- den Werkstoff Naturkautschuk
- den Werkstoff Gummi
- den Milchsaft des Gummibaums *Hevea brasiliensis*, des Kautschukbaums
- verallgemeinernd ähnliche Sekrete anderer Pflanzen, siehe Milchsaft
- wässrige Polymerdispersionen, die z. B. durch Emulsionspolymerisation hergestellt wurden
- international wird der Begriff synonym zu Naturkautschuk oder Gummi verwendet, weil diese aus Latexsaft hergestellt werden,
- eine Art von Wandanstrich, siehe Latexfarbe
- ein Bekleidungsmaterial, siehe Latexkleidung
- eine Software, die Makropakete für das Textsatzprogramm TeX bereitstellt, siehe LaTeX
- einen Pornofilm von Michael Ninn, siehe Latex (Film)

Noch etwas weit gefaßt, also suchen wir nach der *exakten* Schreibweise – also `LaTeX` (großes L, großes T und großes X!)

Hier werden wir fündig, und kommen sofort auf den richtigen Wiki-Eintrag. Bei LaTeX handelt es sich also um kein Kautschukerzeugnis, sondern um ein Textsatz- und Drucksatzprogramm. Weitere Hintergrundinformationen finden sich auf der Wiki-Seite oder im WorldWideWeb nahezu überall (Google spuckt bei der suche nach `LaTeX` immerhin 35 100 000 Ergebnisse aus!)

Wie man sieht kommt es hier also sehr wohl auch auf die Schreibweise an, und exaktes schreiben ist in LaTeX etwas, dem wir noch öfter begegnen werden! Aber erstmal sehen wir uns an, *warum* wir uns für LaTeX überhaupt interessieren sollten.

2 warum LaTeX? Ich habe ja Word ...

Wer heutzutage am Computer arbeitet, der ist gewissen Komfort gewohnt, und dazu zählt unter anderem Microsoft®'s Office-Suite mit Word, Excel, Powerpoint, etc. Lange vergessen sind die Zeiten, als man beim Schreiben des Textes noch nicht sehen konnte, wie die fertige Druckfassung einmal aussehen würde. Der 80 Zeichen und 25 Zeilen Textbildschirm ist für die meisten von uns ein Relikt vergangener Zeiten, sofern überhaupt noch jemand etwas mit dem Begriff MS-DOS anfangen kann.

In Microsoft® Word sehen wir also sofort, wie unser Text auf dem Drucker aussehen wird, und wenn wir etwas verändern, dann geschieht das in Echtzeit. Programme dieser Art werden auch als WYSIWYG-Software bezeichnet – **w**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et.

LaTeX hingegen wird in einem einfachen Texteditor geschrieben, und ein `.TEX`-Dokument wirkt beim Betrachten eher wie der Quellcode eines `.HTML`-Dokuments (als simpler Vergleich für alle, die das schon einmal gesehen haben.)

Auf den ersten Blick ist also unklar, warum man sich auf die durchaus komplexere Handhabung von LaTeX einlassen soll, wenn man doch mit Word bereits ein ähnliches Werkzeug in Händen hält? Wie bei allen Werkzeugen gibt es immer wieder Vor- und Nachteile. Die Vorteile von Word liegen im einfachen und intuitiven Programmaufbau, der es Anfängern innerhalb kürzester Zeit und ohne jegliche Vorkenntnis des Programmes ermöglicht, Dokumente zu schreiben. Geht man jedoch in die Tiefe, und arbeitet an komplexeren Texten, erreicht man bald einen Punkt, an dem die intuitive Handhabung plötzlich Ecken und Kanten zeigt, und gar nicht mehr so selbsterklärend ist! Wer schonmal Dokumente mit 20+ Seiten, Inhaltsverzeichnis, Fußnoten, Kopf- und Fußzeilen, Abbildungen und Tabellen erstellt hat, weiß, dass Word manchmal sehr zickig sein kann, vor allem was die exakte Platzierung bestimmter Elemente betrifft, Verschiebung von Objekten, etc. Außerdem kommt man sehr schnell dahinter, dass eine Gliederung des Dokumentes unumgänglich ist, will man später nicht völlig im Gewirr der Überschriften und Formate untergehen. LaTeX zäumt das Pferd sozusagen von hinten her auf. Wer sich nicht von Anfang an mit gewissen Regeln herumschlägt, und ein Grundwissen an Befehlen aneignet, kann wahrscheinlich keine einzige Seite in druckreife Form bringen. Die Stärken von LaTeX liegen nicht in kurzen Texten, sondern in komplexen, langen Artikeln, mit denen jeder angehende Wissenschaftler früher oder später konfrontiert wird (spätestens beim schreiben einer Bakk- oder Master-Arbeit!) LaTeX hat seine Stärken in Automatisierungsroutinen, einfachen und logischen Querverweisen, perfekter Einbindung eines Literaturverzeichnisses und – wohl einer der wichtigsten Punkte – perfektem Layouting für den Druck! LaTeX-Dokumente wirken professionell, selbst wenn man überhaupt keine persönliche Zeit und Mühe in das Layout steckt. Das liegt zuallererst daran, dass LaTeX auf Layouting-Regeln des Buchdruckes basiert, und entsprechende Voreinstellungen mitbringt! Das Programm selbst achtet bereits besonders auf Lesbarkeit und professionelles Erscheinungsbild.

So, jetzt aber erstmal genug davon. Was wir als erstes brauchen, ist das Werkzeug, denn ohne das kann man nicht arbeiten. Sehen wir also erstmal, wo man LaTeX bekommt und wie man es installiert.

3 Installation

Als erstes stellt sich die Frage – muss ich LaTeX überhaupt installieren? Denn ähnlich *Google Documents* gibt es mit der Plattform `http://www.sharelatex.com` eine Art Online-LaTeX-Umgebung im Webbrowser. Nach der Anmeldung hat man Zugriff auf einen Editor, in dessen Hintergrund eine vollständige TeXLive-Installation arbeitet. Man braucht sich nicht mit Installationen abzumühen, sondern kann im Grunde gleich loslegen. Dazu kommt, dass auf `www.ShareLatex.com` auch eine Bibliothek öffentlicher Projekte zu finden ist, aus denen man sich Anregungen, Ideen oder ganze Arbeitszyklen abschauen kann. Die Voraussetzung für die Nutzung dieser Plattform ist jedoch eine stehende Internetverbindung. Ohne Internet gibt es keinen Zugriff auf die Daten. Weiters ist der Editor auf `www.ShareLaTeX.com` recht eingeschränkt und lässt viele Funktionen vermissen. Das macht ihn anderseits dafür wieder sehr übersichtlich!

Wer häufig ohne Netz unterwegs ist, oder nur eine instabile Internetverbindung hat, kann natürlich auch auf die herkömmliche Methode der persönlichen Installation zurückgreifen. Die Vorteile hier sind klarerweise die Möglichkeit, LaTeX auch offline zu nutzen, sowie die freie Wahl aus einer Vielzahl von Editoren. Wie man LaTeX installiert, wird also im folgenden erklärt.

LaTeX ist von Grund auf ein freies Programm (oder eher eine Sammlung von Programmen). Wie alle Programme, deren Quellcode frei zugänglich und von jedermann veränderbar ist, liegen darin auch gewisse Vor- und Nachteile. Die Vorteile sind klarerweise der kostenfreie Zugang über das Internet, keine Lizenzprobleme, Plattformunabhängigkeit, und weltweite Hilfe durch tausende Programmierer und Nutzer. Die Nachteile sind offensichtlich die vielen unterschiedlichen LaTeX-Distributionen und die allzu freien Möglichkeiten in der Zusammenstellung der „persönlichen“ LaTeX-Installation.

Wir werden uns primär mit der *TEX-Live* Distribution befassen. Es handelt sich dabei um ein sehr komplettes Paket, dass als DVD-Abbild im Internet gratis downloadbar ist. *TEX-Live* kommt als Hybrid-Disk, die eine Installation sowohl auf Windows, als auch auf Mac OSX und Linux ermöglicht. Es beinhaltet auch eine Reihe von eigenen LaTeX-Editoren, die – ähnlich HTML-Editoren – sogenanntes „content-highlighting“ unterstützen (das bedeutet, LaTeX-Befehle werden im Text farblich hervorgehoben, um besseren Überblick zu gewähren).

3.1 Installation auf Windows

- Erstmal die DVD herunterladen (ca. 2,4 GB). *TEX-Live* wird auch von der Uni-Wien zur verfügung gestellt, der Link lautet wie folgt:

  `http://ftp.univie.ac.at/packages/tex/systems/texlive/Images/`

- `.ISO`-Datei brennen (Windows verfügt bereits über ein integriertes Brennprogramm, das `.ISO`-Dateien richtig erkennt), und dann ins DVD-Laufwerk einlegen. Die Installation startet automatisch, und begrüßt uns mit dem Setup-Bildschirm aus Abbildung 1.

- Alle Einstellungen sind bereits automatisch optimal gewählt, und wir können einfach in jedem Fenster auf auf `Next` bzw. `TeX-Live installieren` clicken. Die Installation dauert möglicherweise sehr lange (30 Minuten sind nicht selten), also nicht ungeduldig werden!

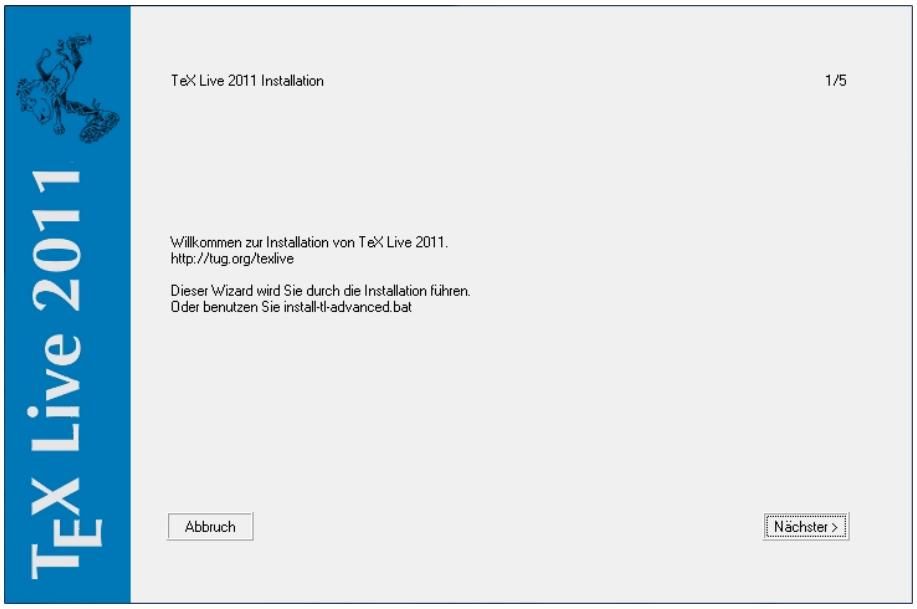


Abbildung 1: Installation auf Windows, Setup-Bildschirm

- Nachdem wir jetzt unser komplettes LaTeX-Paket installiert haben, fehlt uns noch ein geeigneter Editor. Hierfür nutzen wir den Editor *WinShell*. Es befindet sich zwar eine Version auf der *TeX-Live*-DVD jedoch empfehle ich die aktuellste Version herunterzuladen, die man hier beziehen kann:

  `http://www.winshell.org/modules/download/`

- Zuguterletzt installieren wir noch die deutschen Wörterbuchdateien für *WinShell*. Diese erhält man auf der Lernplattform:

  `https://moodle.univie.ac.at/course/view.php?id=16339`

  Die beiden Dateien müssen noch ins *WinShell*-Verzeichnis kopiert werden; üblicherweise lautet der Pfad

  `C:\Programme\WinShell\Dictionaries`

3.2 Installation auf OSX

- Für Mac OSX existiert auch eine sehr einfach zu installierende *TeX-Live* Variante, namens *MacTeX*, die über folgenden Link zu beziehen ist (ca. 2GB):

  `http://ftp.univie.ac.at/packages/tex/systems/mac/mactex/mactex20120701.pkg`

  Die heruntergeladene `.PKG`-Datei kann wie jedes andere Apple-Programm installiert werden.

  Das Paket beinhaltet auch den LaTeX-Editor *TeXshop* sowie das Literaturverwaltungs-Programm *BibDesk*, das als Ersatz zu *JabRef* benutzt werden kann (siehe hierzu Kapitel 15).

3.3 Installation auf Linux

Ich beziehe mich hier und im folgenden Text primär auf die Distribution *Ubuntu Linux*. Wann immer ich im Text also über Linux spreche, meine ich damit Ubuntu!

In Ubuntu Linux läuft die Installation am einfachsten über den Paketmanager *Synaptic*, bzw. über die Kommandozeile. Mit dem Befehl `sudo apt-get install texlive-full` wird das komplette TEX-Live-System aus dem Internet heruntergeladen und installiert. Die Größe des Downloads und der Installation liegt bei ca. 1GB.

Wer dennoch auf die DVD zurückgreifen will, kann auch von dort – der Anleitung folgende – in einem Terminal-Fenster die Installation starten. Dazu geht man folgendermassen vor:

- Einlegen der DVD und öffnen eines Terminalfensters
- Betreten des richtigen Verzeichnisses auf der DVD (hat man nur ein CD/DVD-Lauferk, wird sie üblicherweise automatisch unter `/media/cdrom` eingehängt, auch wenn am Desktop vielleicht etwas anderes angegeben ist!): `cd /media/cdrom`
- Starten der Installation: `sudo ./install-tl.sh`
- Die Standard-Vorgaben für die Installation sind in Ordnung, wir beginnen also mit der Installation, indem wir `I` drücken. Der Installationspfad lautet `/usr/local/texlive/2012`. Hier befindet sich jetzt unsere neue TEX-Live-Umgebung.

Der Installationspfad kann auch frei gewählt werden, was ich allerdings nur erfahrenen Benutzern empfehle.

4 Minimalwissen

Um ein LATEX-Dokument zu schreiben reicht grundsätzlich sehr wenig Wissen über die diversen Befehle aus. Sehen wir uns ein Minimaldokument an (Abbildung 2)[1]

Simpel, oder? Natürlich finden hier noch keine der zahlreichen Sonderfunktionen von LATEX Verwendung, aber der grundsätzliche Aufbau wird klar.

Die Zeile `\documentclass{article}` ist in diesem Dokument die sogenannte „Präambel“ und enthält grundsätzliche Befehle, die für das restliche Dokument gelten. Üblicherweise wächst die Präambel auf mehrere Zeilen an, um diverse Spezialfunktionen realisieren zu können. Aber das kommt später.

Die Zeile `\begin{document}` startet das eigentliche Dokument. Nach dieser Zeile kommt der komplette Inhalt. Wie man im Beispiel gut erkennen kann, werden Absätze einfach durch das Einfügen einer (oder mehrerer!) Leerzeile erzeugt. Ein einfacher Zeilensprung hingegen wird von LATEX ignoriert, und nur als Leerzeichen gewertet!

Schlußendlich steht `\end{document}` am Ende des Dokumentes.

[1] *Hinweis für Mac-User:* den Backslash \ erhält man durch drücken der Tasten Alt+Shift+7; die, für LATEX notwendigen Klammern `{`, `}`, `[` sowie `]` erhält man durch Drücken von Alt+5, Alt+6, Alt+7 sowie Alt+8.

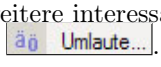
```
\documentclass{article}
\begin{document}

Hier steht der Text des Dokumentes. In LaTeX ist es
sehr simpel einen Absatz zu machen. Man muß einfach
nur eine Leerzeile einfügen!

So wie hier. Dieser Textblock ist jetzt ein neuer
Absatz!

\end{document}
```

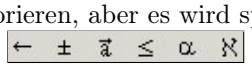
Abbildung 2: Ein Minimaldokument

Da wir bisher nur den Quellcode des Dokumentes haben, muß dieses erst von LaTeX „gerendert“ werden. Zwar besteht LaTeX grundsätzlich aus vielen Kommandozeilen-Programmen, aber unser Editor *WinShell* nimmt uns hier einige Arbeit ab. Um das Dokument also in druckreifer Form zu sehen, clicken wir einfach auf den Knopf  . Der Editor gibt daraufhin einige Meldungen im Ausgabe-Fenster aus, und daraufhin finden wir im Verzeichnis des Testdokumentes ein PDF mit dem selben Namen. Voilá, unser erstes Dokument ist fertig!

Das PDF kann im übrigen durch click auf den  -Knopf direkt angezeigt werden (WinShell nutzt dafür den voreingestellten PDF-Betrachter des Systems; meistens handelt es sich dabei – zumindest auf Windows-Systemen – um den *Adobe Reader*).

Der Editor WinShell hat noch einige weitere Besonderheiten. So verfügt er z.B. über eine integrierte Rechtschreibprüfung, die sich durch einen click auf  öffnet und das Prüffenster mit den bekannten Optionen wie *ersetzen, ignorieren, Wort hinzufügen,* anzeigt.

Eine weitere interessante Funktion findet sich nach einem Click auf *Einstellungen* →  . Hier läßt sich die Kodierung der Umlaute einstellen, was mit dem LaTeX-Befehlssatz zusammenhängt. WinShell fügt automatisch bei Druck auf die entsprechende Umlaut-Taste den hier festgelegten Code ein! Wir können das derzeit einfach ignorieren, aber es wird später noch wichtig.

Die Sonderzeichen-Buttons  eröffnen kleine Menüs mit einer geringen Auswahl möglicher Sonderzeichen, Pfeile, griechischer Buchstaben, etc. Diese Auswahl stellt allerdings nur einen geringen Anteil der Möglichkeiten zur Verfügung, die LaTeX tatsächlich darstellen kann! Mehr dazu findet sich in Pakin (2008).

5 Und jetzt der nächste Schritt

Nachdem der Grundaufbau geklärt ist (und eigentlich gar nicht so kompliziert aussieht) wagen wir uns an etwas mehr heran. Wir werden uns jetzt mit einigen

Spezialbefehle und -funktionen befassen, die in LaTeX üblich sind (Abbildung 3). Die Befehle werden im folgenden erklärt.

```
\documentclass[12pt,a4paper]{article}
\usepackage[ngerman]{babel}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}

\begin{document}
% jetzt kommt die Titelseite
\title{Das zweite Dokument}
\author{Martin Dockner}
\date{\today}
\maketitle

% jetzt kommt das Inhaltsverzeichnis
\tableofcontents

% jetzt startet der Text
\section{Einleitung}
Hier steht die Einleitung zu diesem Dokument.

\section{Unterteilungen}
Dieser Abschnitt wird jetzt noch weiter unterteilt.
\subsection{Die sieben Ebenen}
Es gibt bis zu sieben Ebenen zur Einteilung des
Dokumentes.
Chapter -- Section -- Subsection -- Subsubsection --
Paragraph -- Subparagraph.

\end{document}
```

Abbildung 3: Ein zweites Dokument

Die erste Neuerung gegenüber dem ersten Dokument ist schon in der Präambel sichtbar. Die Zeile `\documentclass{article}` wurde erweitert und lautet jetzt `\documentclass[12pt,a4paper]{article}`. In Eckigen Klammern können zusätzliche Optionen für die Dokuementklasse gesetzt werden. `12pt` setzt die Schriftgröße auf 12 Punkte[2]. `a4paper` definiert als Seitenformat A4[3]. Außerdem befinden sich in der Präambel jetzt zwei Zeilen die mit `\usepackage` beginnen.

`\usepackage[ngerman]{babel}` fügt die Unterstützung für die neue deutsche Rechtschreibung hinzu, sowie für deutsche Abteilungsregeln (engl. „hyphenation“ – darauf kommen wir später noch zu sprechen).

[2]die möglichen Werte sind `10pt`, `11pt` und `12pt`. Für andere Schriftgrößen gibt es eigene Regeln.

[3]Die weiteren Optionen sind `letterpaper` (11×8.5 in), `legalpaper` (14×8.5 in), `a5paper` (21×14.8 cm), `b5paper` (25×17.6 cm) und `executivepaper` (10.5×7.25 in)

Die Pakete `\usepackage[T1]{fontenc}` und `\usepackage[latin1]{inputenc}` laden erweiterte Zeichensätze, die direkte Unterstützung für deutsche Umlaute enthalten[4]. Da LaTeX ursprünglich aus dem englischsprachigen Raum stammt, war die Unterstützung für Umlaute relativ umständlich, da diese im Englischen nicht zum allgemeinen Repatoire gehören. Das erste Paket bewirkt vor allem, dass über die Tastatur eingegebene Umlaute nicht in LaTeX-üblicher Kodierung angegeben werden müssen, und das zweite Paket tut das selbe für das scharfe s.

Der Umlaut „ä“ wird beispielsweise in (reinem) LaTeX durch Eingabe von `\"{a}` realisiert. Da das nicht nur mühsam zu schreiben ist, sondern somit der LaTeX-Quelltext auch nur umständlich zu lesen ist, bietet das Paket `ngerman` bereits die Möglichkeit, für ä einfach `"a` zu schreiben. Das macht alles zwar einfacher, aber ein Problem bleibt nach wie vor bestehen – und zwar die automatische Rechtschreibprüfung unseres Editors! Hier wird diese Kodierung auch nicht erkannt. Und das ist der Grund, warum das paket `fontenc` geladen wurde, denn hiermit können wir Umlaute einfach schreiben, und LaTeX setzt sie richtig in das resultierende `.PDF`-File um!

Dennoch sollte man die offizellen LaTeX-Regeln für Umlaute beherzigen, da hiermit auch völlig andere Sonderzeichen geschrieben werden können! Tabelle 1 zeigt einige Beispiele.

Tabelle 1: Akzente, Sonderzeichen und Symbole in LaTeX.

á	`\'{a}`	ő	`\H{o}`	§	`\S`
à	`\‘{a}`	ė	`\.{e}`	£	`\pounds`
â	`\^{a}`	ṃ	`\d{m}`	©	`\copyright`
č	`\v{c}`	ş	`\c{s}`	TM	`\texttrademark`
ã	`\~{a}`	ø	`{\o}`	®	`\textregistered`
ä	`\"{a}`	ł	`{\l}`	†	`\dag`
å	`\r{a}`	å	`{\aa}`	‡	`\ddag`
ā	`\={a}`	ß	`{\ss}`	¶	`\P`
k̲	`\b{k}`	æ	`{\ae}`	¡	`!‘`
ă	`\u{a}`	œ	`{\oe}`	¿	`?‘`

Durch diesen codierten Aufbau ist es in LaTeXalso ohne großen Aufwand möglich, sogar völlig abstruse Sonderzeichen zu erzielen! So ist s̈ genauso möglich wie ḫ, x̃, z̄ oder K̇, da alle in der Tabelle angegebenen Regeln[5] grundsätzlich auch auf sämtliche Buchstaben des Alphabetes angewendet werden können!

Gehen wir jetzt in unserem Dokument weiter. Die Zeile `% jetzt kommt die Titelseite` ist ein Kommentar. Wird das %-Zeichen im Dokument verwendet, wird der Rest der entsprechenden Zeile nicht im fertigen Dokument ausgedruckt. Man kann hiermit Anmerkungen oder Kommentare setzen, oder auch einfach gewisse Befehle oder Dokumenteigenschaften aussetzen. Das %-Zeichen kann auch in der Präambel eingesetzt werden! Um das %-Zeichen im Text abzubilden, muß ein \ vorangesetzt werden. Das bedeutet, man „maskiert“ das Codezeichen.

[4] *Hinweis für Linux- bzw. Mac-Benutzer:* Viele Linux-Systeme arbeiten nicht mit der *Latin*-Textkodierung, sondern mit *UTF-8*. In diesem Fall ist auf Linux die Zeile `\usepackage[utf8]{inputenc}` zu verwenden. Auf Apple OSX hingegen empfiehlt sich `\usepackage[applemac]{inputenc}`.

[5] jedenfalls alle, die mit einem \ beginnen.

Das gilt im übrigen auch für einige andere Zeichen, die in LaTeX als Codezeichen eingesetzt werden. Tabelle 2 gibt über diese Ausnahmen Auskunft.

Tabelle 2: Codezeichen in LaTeX.

\	`\textbackslash`
$	`\$`
%	`\%`
&	`\&`
#	`\#`
_	`\_`
{	`\{`
}	`\}`

Die nächste Zeile startet mit der `\title`-Funktion, die den Dokument-Titel definiert.

Ebenso wird mittels `\author` der Name des Authors festgelegt.

`\date` hat keine Zahl, sondern die angabe `\today` in geschwungener Klammer stehen! Hiermit greift LaTeX auf die Systemzeit des Computers zu, und findet damit den aktuellen Tag heraus.

Sind diese drei Definitionen (oder auch nur eine oder zwei davon) vorgenommen, wird mittels `\maketitle` die Titelseite im fertigen Dokument sichtbar gemacht. Zwar können die Befehle `\title`, `\author` und `\date` sowohl im Textbereich als auch in der Präambel stehen, die sichere Lösung ist aber, wie im Beispiel auch gezeigt, sie im Textbereich (also innerhalb der `\begin{document}` – `\end{document}` Umgebung) zu setzen.

Nach einem weiteren Kommentar haben wir jetzt den Befehl `\tableofcontents`, der – wie der Name schon sagt – ein Inhaltsverzeichnis an dieser Stelle einfügt. Das Inhaltsverzeichnis wird automatisch aus dem Dokument generiert, je nachdem welche *Chapters* und *Sections* eingetragen wurden. Eine genaue Erklärung dazu kommt gleich.

Unser Haupttext beginnt jetzt nämlich mit dem Befehl `\section`. Statt Überschriften (wie in Word üblich) zu definieren, arbeitet LaTeX mit den tatsächlichen Bereichen, die ein Dokument üblicherweise aufweist. In Büchern gibt es somit auch *Parts* und *Chapters*, in Briefen wiederum gar keine Unterteilung. Tabelle 3 gibt über die verschienen Aufteilungsebenen der diversen Dokumentklassen Auskunft.

Tabelle 3: Dokumentebenen in LaTeX.

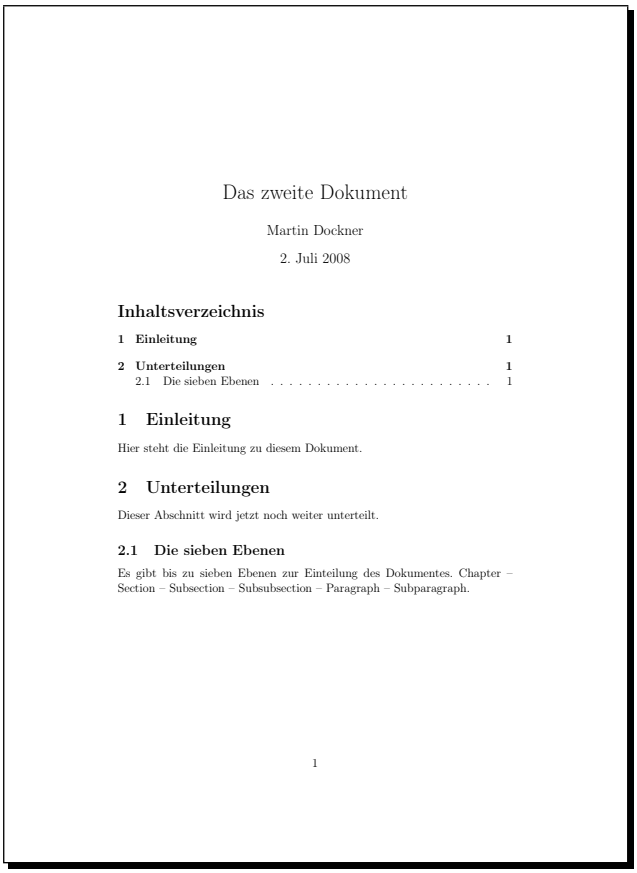
Kommando	verfügbar in diesen Dokumentklassen	Nummerierung
`\part`	book, report	römisch Ziffern
`\chapter`	book, report	arabische Ziffern
`\section`	book, report, article	arabische Ziffern
`\subsection`	book, report, article	arabische Ziffern
`\subsubsection`	book, report, article	arabische Ziffern
`\paragraph`	book, report, article	arabische Ziffern
`\subparagraph`	book, report, article	arabische Ziffern

Da es sich bei unserem zweiten Dokument um einen *article* handelt, benutzen

wir auch nur die Einteilung in *Sections* und *Subsections*. Aber keine Sorge, wir sehen uns später auch noch *Book* und *Report* an.

Haben wir alles richtig getippt, lassen wir LaTeX zwei oder drei mal durchlaufen (indem wir auf den Knopf  clicken). Das ist notwendig, weil LaTeX für die Erstellung des Inhaltsverzeichnisses einige Temporärdaten braucht, die durch den ersten Durchlauf erstellt werden. Um auf diese Daten dann zugreifen zu können, muß man einen zweiten Durchlauf starten. Am besten man läßt das Programm so oft durchlaufen, bis sich die Fehler- und Warnmeldungen im Informationsbereich von *WinShell* nicht mehr verändern.

Danach sollte das fertige Dokument in etwa aussehen, wie in Abbildung 4.



Das zweite Dokument

Martin Dockner

2. Juli 2008

Inhaltsverzeichnis

1 Einleitung

Hier steht die Einleitung zu diesem Dokument.

2 Unterteilungen

Dieser Abschnitt wird jetzt noch weiter unterteilt.

2.1 Die sieben Ebenen

Es gibt bis zu sieben Ebenen zur Einteilung des Dokumentes. Chapter – Section – Subsection – Subsubsection – Paragraph – Subparagraph.

1

Abbildung 4: Das fertige zweite Dokument

6 Abbildungsverzeichnis und Tabellenverzeichnis

Wie gerade erklärt, erzeugt der Befehl `\tableofcontents` an der Stelle, wo er eingefügt wird, ein Inhaltsverzeichnis. Die selbe Möglichkeit existiert natürlich auch, für Abbildungsverzeichnisse und Tabellenverzeichnisse! Die beiden Befehle hierfür lauten `\listoffigures` bzw. `\listoftables` an der Stelle im Dokument, wo sie eingefügt werden sollen.

7 Dokumentklassen und ihre Unterschiede

Bisher haben wir uns nur mit der Dokumentklasse *article* beschäftigt, aber LaTeX bietet hier ja noch einiges mehr. Die vier Grunddokumentklassen von LaTeX bestehen aus *book, report, article* und *letter*.

Dazu kommen noch einige Klassen, die im TeX-Live-Paket ebenfalls mitgeliefert werden und dem sogenannten *KOMA-Script-Paket* anghören. Es handelt sich hierbei um die Klassen *scrbook, scrreprt, scrartcl* und *scrlttr2*[6]. Das KOMA-Script-Paket einthält eine sehr umfangreiche Erweiterung der normalen LaTeX-Dokumentklassen, die sich bei oberflächlicher Betrachtung vor allem in der Darstellung der Dokumente wiederspiegelt. Es wurden andere Schriftarten und ein anderes Seitenlayout gewählt, und noch einige weitere Zusätze eingeführt. Für uns ist vorerst die optische Komponente interessant. Denn mit den vier Dokumentklassen des KOMA-Script-Paketes stehen uns nun insgesamt schon acht verschiedene Dokumentstile für Druckdokumente zur Verfügung.

Die Klassen *amsart* bzw. *amsbook* sind ihrerseits Ersatzklassen für *article* bzw. *book* mit verändertem Layout. Grundsätzlich wurden in den beiden ams-Klassen die größen und Stile von Überschriften und Titelseite verändert.

Die Klasse *elsart* wurde vom Elsevier-Publishing Verlag erstellt und dient insbesondere zur Erstellung von wissenschaftlichen Artikeln. Allerdings muß man bei dieser Klasse einige Spezialitäten bezüglich des Titel-Bereiches beachten. Alles wissenswerte hierzu enthält Pepping (2006).

Wer Artikel im Design und Layout der *American Psychological Association* schreibt, sollte hingegen zu der Dokumentklasse *apa* greifen. Allerdings kommen auch hier, ähnlich der Klasse *elsart*, einige Spezialoptionen zum Einsatz, die auf der Homepage `http://www.ilsp.gr/homepages/protopapas/apacls.HTML` erklärt werden.

Wenn man das zusammen betrachtet, hat man bereits zwölf Dokumentklassen mit leicht variierendem Aussehen. Je nachdem, welche Art von Dokument man schreiben möchte, bieten sie alle Vor- und Nachteile. Am einfachsten ist es, man probiert sie selbst aus, und schlägt bei Warnungen oder Fehlermeldungen des Editors einfach im Internet nach. Wir werden uns vorerst mit drei der vier Hauptklassen, *article, book* und *report* befassen, da sie für unsere Dokumente am ehesten in Frage kommen.

Worin liegen jetzt die groben Unterschiede der Klassen?

- Die Klasse *article* ist in 10pt Schrift gehalten und einseitig. Die Titelseite und das Inhaltsverzeichnis erhalten keine eigenen Seiten. Es gibt keine Kapitel, und Sections werden fortlaufend im Text abgedruckt. Diese Klasse eignet sich gut für Magazin- oder Journal-Artikel, Reviews, normale Texte, etc.

- Die Klasse *book* ist in 10pt Schrift gehalten und doppelseitig. Die Titelseite erhält ein eigenes Blatt, das Inhaltsverzeichnis erhält ebenfalls eine/mehrere eigene Seite(n). Jedes Kapitel beginnt mit einem neuen, rechtsseitigen, Blatt; Sections sind fortlaufend. Kapitel erhalten die volle Beschriftung „Kapitel 1“, „Kapitel 2“, etc. anstatt nur der Nummern.

[6] Die Klasse *scrlettr* existiert ebenfalls, ist aber mittlerweile veraltet und sollte desshalb nicht mehr benutzt werden!

- Die Klasse *report* ist in 10pt Schrift gehalten und einseitig (jede Seite hat ihr eigenes Blatt). Die Titelseite erhält ein eigenes Blatt, das Inhaltsverzeichnis erhält ebenfalls eine/mehrere eigene Seite(n). Jedes Kapitel beginnt mit einem neuen Blatt. Diese Klasse eignet sich für jegliche Reports, aber auch für Bakkalaureats- und Diplomarbeiten sowie Dissertationen[7]

Um diese Vorgaben zu variieren, kann man jetzt mehrere *Options* einsetzen. Optionen zur Dokuementklasse werden immer in eine eckige Klammer geschrieben, und durch Kommata voneinander getrennt (siehe dazu auch schon den Beispielartikel aus Abbildung 3). Folgende Variationen sind möglich.

Schriftgröße – die 10pt-Standardgröße kann durch Angabe der Option `[11pt]` oder `[12pt]` geändert werden. Diese Angabe bezieht sich auf den *normalen* Dokumenttext. Überschriften, Fußnoten, Kommentare, Abbildungsbeschriftungen, etc. werden in Relation zu dieser Ausgangsgröße gesetzt.

Papiergröße – wie bereits in Kapitel 5 beschrieben, kann eine der folgenden Seitengrößen angegeben werden:

- `[a4paper]` (29.7×21 cm)
- `[letterpaper]` (11×8.5 in)
- `[legalpaper]` (14×8.5 in)
- `[a5paper]` (21×14.8 cm)
- `[b5paper]` (25×17.6 cm)
- `[executivepaper]` (10.5×7.25 in).

Druckseiten – die Vorgabe einseitig oder zweiseitig kann durch die Option `[oneside]` oder `[twoside]` entsprechend geändert werden.

Titelseite – mit der Option `[titlepage]` wird ein eigenes Blatt für die Titelseite reserviert, im Gegenzug dazu gibt es `[notitlepage]` um die eigenständige Titelseite zu unterbinden.

Fehlerkommentare – mit der Option `[draft]` werden alle Zeilen, in denen LaTeX einen Fehler oder eine Warnung ausspuckt mit einem schwarzen Rechteck am rechten Rand markiert. Außerdem werden in diesem Modus keine Grafiken eingebunden, was zu einer schnelleren Verarbeitung des Dokumentes durch LaTeX führt. Anstelle der Grafiken sind nur Platzhalter mit dem Namen der Grafik zu sehen. Diese Option ist nützlich, wenn man auf Fehlersuche ist, aber sehr viele Grafiken eingebunden hat, und die pdfLaTeX-Funktion sehr lange für die Verarbeitung braucht.

Die Option `[landscape]` setzt den kompletten Text in Querformat, was jedoch nur selten gewünscht wird. Falls man tatsächlich eine oder mehrere Seiten in Querformat setzen muß (man denke z.B. an ausladende Tabellen) gibt es für diesen Fall ein eigenes Paket namens `pdflscape`, das es ermöglicht einzelne Bereiche des Dokumentes ins Querformat zu bringen. Darauf gehe ich später aber noch gesondert ein.

[7] Nach den neuen Abgaberegeln müssen diese Arbeiten jedoch doppelseitig sein, siehe dazu die *Druckseiten*-Option!

Ähnliches gilt für die Option `[twocolumn]`, die das gesamte Dokument in zweispaltiges Format setzt. Auch hier gibt es ein eigenes Paket (mit Namen `multicol`) mit dessen Hilfe Teilbereiche des Dokumentes in mehrspaltigen Satz gebracht werden können. `multicol` ermöglicht außerdem (im Gegensatz zu der documentclass-Option) eine freie Anzahl an Spalten. Auch dieses Paket werde ich später noch gesondert besprechen.

Weitere Optionen für die Dokumentklasse sind `openright, openany, openbib, fleqn,` und `leqno`, auf die ich hier jedoch nicht näher eingehe.

8 Wichtige Pakete

Nachdem sich das letzte Kapitel mit der Dokumentklasse beschäftigt hat, gehen wir jetzt auf den nächsten wichtigen Bereich der Präambel ein. Pakete sind sozusagen die Lego-Steine des Baukastens LaTeX. Ich versuche die wichtigsten Pakete, die man als Anfänger nutzen wird, kurz zu beschreiben. In den folgenden Kapiteln werde ich dann, wenn ich neue Pakete einführe, ebenfalls eine Kurzbeschreibung dazu abgeben.

Pakete werden, wie auch schon in Abbildung 3 ersichtlich, immer mittles des Befehls `\usepackage[`*Paketoption(en)*`]{`*Paketname*`}` in die Präambel eingebunden. Pakete, die keine Optionsangaben erfordern, können mittels Kommata getrennt werden (z.B. `\usepackage{ngerman,graphicx,natbib}`). Pakete, die Optionen erfordern, müssen immer in einen eigenen `\usepackage`-Bereich gestellt werden. Manche Pakete können mit verschiedenen Optionen genauer definiert werden, benötigen diese jedoch nicht zwingend. Für solche Fälle gebe ich in meiner Auflistung *keine* Optionen an! Für manche Pakete ist es schlußendlich sogar erforderlich, an welcher Position in der Präambel sie stehen, da die einzelnen Pakete und ihre Reihenfolge sich gegenseitig beeinflußen können.

`\usepackage{ngerman}` – dieses Paket ist sicher eines der wichtigsten für deutsche Texte. Es versetzt LaTeX in die Lage, Wörter nach den neuen deutschen Rechtschreibregeln zu setzen und abzuteilen (diese Paket hat jedoch *keinen* Einfluß auf die Rechtschreibprüfung des Editors!) Es sind keine weiteren Optionen notwendig. *Hinweis: Die gleiche Funktion erfüllt auch das Paket `\usepackage[ngerman]{babel}`. Wir verwenden jedoch weiterhin das zuvor genannte!*

`\usepackage{fontenc}` – das zweitwichtigste Paket, wenn man in deutscher Sprache schreibt. Mittels `fontenc` erkennt LaTeX die direkte Eingabe von Umlauten im Editor, und setzt sie richtig im Ausgabefile um. Somit muß man sich bei den drei Umlauten ä, ö und ü keine Gedanken über ihre LaTeX-Codes machen (siehe dazu auch Kapitel 5). Um diese Funktion zu aktivieren ist es zwingend notwendig die Option `[T1]` zu setzen. Die Zeile in der Präambel muß also lauten: `\usepackage[T1]{fontenc}`.

`\usepackage{inputenc}` – wie auch das Paket `fontenc`, hilft auch `inputenc` mit erweitertem Zeichensatz, deutsche Texte leichter zu bearbeiten. Es ermöglicht den Buchstaben „ß“ ohne umständliche LaTeX-Codes zu schreiben. Wie auch bei `fontenc` ist hier die Angabe einer Option erforderlich. Die Zeile in der Präambel muß also auf Windows-Systemen lauten: `\usepackage[latin1]{inputenc}`. Auf Mac OSX muß die Zeile fol-

gendermassen aussehen: `\usepackage[applemac]{inputenc}`. Auf Linux-Systemen sieht sie, aufgrund des wieder anderen Sonderzeichensatzes, so aus: `\usepackage[utf8]{inputenc}`.

`\usepackage{graphicx}` – dieses Paket ermöglicht es, Grafiken in LaTeX-Dokumente einzubinden. Es ermöglicht auch verschachtelte Grafiken (d.h. mehrere Bilder mit gemeinsamer Bildbeschriftung, bzw. Subbeschriftungen, etc.) zu erstellen. Eine genaue Erklärung dazu findet sich in Kapitel 9. Für dieses Paket sind keine Optionen notwendig.

`\usepackage{url}` – dieses Paket ermöglicht es, Internet-Adressen innerhalb des Textes mit dem Befehl `\url{`*Adresse*`}` anzugeben, ohne dabei LaTeX-Fehler wegen Verwendung von Codezeichen zu erzeugen. Das Paket braucht keine weiteren Optionen, es ist aber hilfreich, parallel dazu das Paket `hyperref` zu verwenden!

`\usepackage{hyperref}` – mittels dieses Paketes können Internetlinks in PDF-Dateien zu echten, clickbaren, Hyperlinks umgewandelt werden. Desweiteren erzeugt das Paket `hyperref` auch Links innerhalb des LaTeX-Dokumentes, z.B. im Inhaltsverzeichnis, bei Abbildungs- oder Tabellenreferenzangaben, oder bei Literaturzitaten. Als gutes Beispiel hierfür dient dieses Dokument. Das Paket `hyperref` sollte immer als *letztes* Paket in der Präambel geladen werden! Andernfalls kann es zu Kollisionen mit anderen Paketen kommen (wie z.B. mit dem Paket `lineno`). Es werden zwar für `hyperref` keine weiteren Optionen benötigt, jedoch sind einige Zusatzangaben über eine eigene Eingabezeile in der Präambel möglich. Diese zusätzliche Konfiguration wird durch die Zeile `\hypersetup{`*option1, option2, option3, ...*`}` erreicht. Die wichtigste Option hierfür ist `breaklinks=true`, die einen Zeilenumbruch für Internet-links ermöglicht. Der gesamte Eintrag in der Präambel sollte also wie folgt aussehen:

```
\usepackage{hyperref}
\hypersetup{breaklinks=true}
```

Weitere Konfigurationsmöglichkeiten finden sich im Internet, z.B. auf `http://en.wikibooks.org/wiki/LaTeX/Packages/Hyperref`.

`\usepackage{natbib}` – dieses Paket werde ich etwas später in Kapitel 15 genau besprechen. Es handelt sich hierbei um eine erweiterte Unterstützung für das Literaturverzeichnis und entsprechende Literaturverweise.

`\usepackage{bibgerm}` – auch dieses Paket ist für die Literaturlisten wichtig, und stellt mehr oder weniger die Voraussetzung für deutsche Literaturzitate und Literaturverzeichnisse dar. Wie *natbib* wird auch *bibgerm* näher im Kapitel 15 beschrieben.

`\usepackage{lineno}` – dieses Paket hat den alleinigen Zweck, alle Zeilen eines Dokumentes durchzunummerieren! Der Einsatz dieser Funktion ist besonders für Reviewer nützlich, die Fehler in einem Artikel anstreichen sollen (im Falle einer Bakk- oder Masterarbeit ist es also für den Betreuer sehr hilfreich). Mit durchnummerierten Zeilen ist eine exakte Lokalisierung eines Textabschnittes möglich. Das Paket benötigt keine Optionen, um es jedoch im Dokument einzusetzen muß an der Stelle, ab der die Zeilen

nummeriert werden sollen, der Befehl `\linenumbers` eingesetzt werden. Will man, dass die Zeilennummern auf jeder Seite des fertigen Dokumentes wieder mit der Ziffer 1 beginnen, benutzt man stattdessen den Befehl `\pagewiselinenumbers`.

`\usepackage{setspace}` – mithilfe von *setspace* können eineinhalbfache bzw. doppelte Zeilenabstände realisiert werden. Dabei achtet LaTeX automatisch darauf, dass *ausschließlich* der Haupttext verändert wird, jedoch Tabellen und Fußnoten den einfachen (normalen) Zeilenabstand beibehalten! Um eineinhalbfachen Zeilenabstand zu aktivieren, wird in die Präambel (also oberhalb von `\begin{document}`) das Kommando `\onehalfspacing` eingefügt. Um einen doppelten Zeilenabstand zu erhalten, benutzt man stattdessen das Kommando `\doublespacing`.

`\usepackage{fullpage}` – Wie der Name schon sagt, Werden die Seitenränder bei den Standard-Dokumentklassen *Article, Book* und *Report* etwas kleiner gesetzt, so dass mehr Text auf einer Seite Platz findet.

In den folgenden Kapiteln werden wir uns nun mit einigen dieser Pakete näher befassen.

9 Grafiken einbinden

Grafiken sind heutzugate aus keiner Publikation wegzudenken, seien es jetzt Tabellen, Diagramme, Fototafeln, Zeichnungen, etc.

Das Paket `graphicx` bietet die Möglichkeit, in LaTeX-Dokumenten Grafiken jeder Art einzubinden. Hierbei werden die Grafiken nicht – wie man es von Word oder Powerpoint gewohnt ist – direkt in das Dokument eingearbeitet, sondern in der `.TEX`-Datei wird nur der Pfad und der Dateiname angegeben. Erst bei Erstellung der `.PDF`-Datei wird die Grafik – in Originalqualität – eingebunden. Das hat zweierlei Vorteile:

Zum einen bleibt die LaTeX-Datei „schlank“ und kann weiterhin in sekundenschnelle geöffnet und bearbeitet werden.

Zum anderen können Grafiken mit denen man nicht zufrieden ist sehr einfach ausgetauscht werden. Dazu muß lediglich die Grafikdatei auf der Festplatte ersetzt werden, und beim nächsten pdfLaTeX-Durchlauf ist das fertige Dokument auf dem aktuellen Stand.

Abbildung 5 zeigt den LaTeX-Code, der für das Einfügen einer Grafik notwendig ist.

Der Code wird folgendermassen aufgeschlüsselt:

Die Zeile `\begin{figure}[ht]` startet eine Subumgebung, in die nun eine Abbildung eingefügt werden kann. Diese Subumgebung wird dann – klar ersichtlich – mit dem Befehl `\end{figure}` wieder beendet. Interessant fällt bei der `\begin`-Zeile die Option `[ht]` auf! Diese ist grundsätzlich optional, und dient der Platzierung der Grafik im Text.

LaTeX platziert Grafiken üblicherweise nicht immer dort, wo man sie als Benutzer eingetragen hat, sondern sucht einen geeigneten Bereich auf dem die Grafik sich gut in den Fließtext einpaßt. Das führt zwar einerseits dazu, dass Abbildungen manchmal mehrere Seiten nach hinten wandern, anderseits verhindert LaTeX damit auch, dass Grafiken an unvorteilhaften Positionen platziert

```
\begin{figure}[ht]
\centering
\includegraphics[width=9.8cm]{bilder/testbild}
\caption[Kurzbeschriftung]{Lange Beschriftung}
\label{testbildkey}
\end{figure}
```

Abbildung 5: Einfügen einer Grafik

werden. Die Optionsangabe für den `\begin{figure}`-Befehl hilft jetzt mit vier möglichen Angaben, die Platzierung grob zu beeinflußen. Die Möglichkeiten sind:

`[h]` – mit dieser Option (Abkürzung für *here*) teilt man LaTeX mit, dass die Abbildung wenn möglich exakt hier platziert werden soll.

`[t]` – mit dieser Option (Abkürzung für *top*) teilt man LaTeX mit, dass die Abbildung im oberen Drittel auf dieser, oder einer der folgenden Seiten platziert werden soll.

`[b]` – mit dieser Option (Abkürzung für *bottom*) teilt man LaTeX mit, dass die Abbildung im unteren Drittel auf dieser, oder einer der folgenden Seiten platziert werden soll.

`[p]` – mit dieser Option (Abkürzung für *page*) teilt man LaTeX mit, dass die Abbildung eine eigene Seite erhalten soll. Diese Seite wird möglichst nah an den Grafikaufruf gesetzt, landet jedoch meist am Ende des Kapitels oder am Ende des Dokuments.

Die verschiedenen Optionen können auch in Kombination gesetzt werden, und werden von LaTeX üblicherweise in der Reihenfolge `htbp` (*here – top – bottom – page*) interpretiert. Setzt man also alle vier Optionen versucht LaTeX die Grafik erst an der gewünschten Stelle einzubinden. Ist das nicht möglich (weil die Grafik sonst über den unteren Seitenrand hinausstehen würde, versucht LaTeX eine geeignete Stelle am Top der Folgeseite(n) zu finden. Ist auch das nicht möglich (was sehr selten vorkommt) wird als nächstes versucht eine passende Stelle im unteren Bereich der Folgeseite(n) zu finden, und wenn selbst das scheitert, erstellt LaTeX eine komplett neue Seite für die Grafik. Üblicherweise reichen eine oder zwei Angaben (wie in meinem Beispiel `[ht]`) völlig aus, um zufriedenstellende Ergebnisse zu bringen. LaTeX gibt bei Problemen auch eine entsprechende Warnung aus (in unserem Fall im Ausgabe-Fenster von WinShell).

Die Zeile `\centering` ist optional, und ihre einzige Funktion ist, den weiteren Inhalt der Figure-Umgebung zu zentrieren.

Die Zeile `\includegraphics{bilder/testbild}` sagt LaTeX jetzt, wo sich die einzubindende Bilddatei befindet. In unserem Beispiel als im Unterverzeichnis „bilder“ die Datei „testbild“. Die Angabe eines Unterordners dient nur zur besseren Übersicht. Man kann die Bilddateien auch direkt in das Verzeichnis legen, in dem sich schon die `.TEX`-Datei befindet, allerdings erstellt LaTeX dort ebenfalls

schon einige Files, die es für seine Arbeit benötigt, und somit kann eine weitere Gliederung nicht schaden. Unsere Bilddatei mit Namen „testbild“ kann jetzt die Formate `.JPG`, `.PNG` oder `.PDF` haben. LaTeX benötigt dabei keine Angabe einer Dateiendung. Dabei sollte darauf geachtet werden, dass .PNG ein verlustfreies Format darstellt, also besonders für Grafiken geeignet ist, während .JPG eine starke Kompression hat, die im Detail zu Artefakten im Bild führen kann, und desshalb besonders für Fotos geeignet ist (wo sowieso viele Farbübergänge vorhanden sind). Ein .PDF-Dokument kann auf diesem Weg ebenfalls eingebunden werden (jedoch scheint bei mehrseitigen Dokumenten dann klarerweise nur die erste Seite auf!)

Besonders nützlich ist diese Funktion, wenn man Vektorgrafiken benutzt. Programme wie *Adobe Illustrator*, *Corel DRAW* oder *InkScape* erstellen vektorisierte Grafiken, die grundsätzlich unendlich vergrößerbar sind, ohne unscharf zu werden (im Gegensatz zu Pixelgrafiken, wie z.B. Fotos). Eine Vektorgrafik kann dann üblicherweise über einen PDF-Drucker in eine `.PDF`-Datei umgewandelt werden, ohne die Vektoren zu verlieren. Eine, auf diese Art, in LaTeX eingebundene Vektorzeichnung hat im fertigen Dokument die gleiche hohe Qualität, wie im Ausgangsprogramm!

Die Optionsangabe `[width=9.8cm]` in der Zeile `\includegraphics[width=9.8cm]{bilder/testbild}` kann verwendet werden, um dem eingebundenen Bild eine bestimmte Größe zuzuweisen. LaTeX liest aus den Bilddateien üblicherweise ihre Größe und Auflösung (DPI-Anzahl) aus, sofern diese in der Datei gespeichert sind, und bindet sie in der entsprechenden Größe ein (in *Adobe Photoshop* bearbeitete Bilder haben diese Angaben beispielsweise mit abgespeichert). Falls man hier jedoch nachträglich noch eine Änderung durchführen will, ist das mit dieser Option direkt in LaTeX möglich. Natürlich kann auch `[heigth=5in]` oder ähnliches angegeben werden. LaTeX versteht hierbei (und auch überall anders, wo die Angabe einer Entfernung notwendig wird!) folgende Angaben:

- `pt` (*points*; 1pt ≈ 0.35mm)
- `pc` (*Picas*; 1pc ≈ 12pt ≈ 4.2mm),
- `em` (die Breite des Buchstaben „M“ in der ausgewählten Schriftart)
- `ex` (die Höhe des Buchstaben „x“ in der ausgewählten Schriftart)
- `in` (*inch*)
- `mm` (*Millimeter*)
- `cm` (*Zentimeter*)

Natürlich ist das für unsere Verhältnisse ein Overkill. Für den normalen Mitteleuropäer reicht also sicher die Angabe `cm` oder `mm`. Will man Kommastellen angeben (also z.B. `3.5cm` muß, wie gezeigt, für das Komma-Zeichen ein Punkt verwendet werden!

Gibt man nur die Höhe *oder* die Breite an, errechnet LaTeX den jeweils anderen Wert automatisch und skaliert die Grafik korrekt. Gibt man *beide* Werte an (also z.B. `[width=9.8cm,heigth=2cm]`, kann man damit die Grafik auch absichtlich verzerren.

Als nächstes kommt die Funktionszeile `\caption[Kurzbeschriftung]{Lange Beschriftung}`, die – leicht zu Erraten – die Bildbeschriftung trägt. In

eckigen Klammern steht hier eine Kurzbeschriftung, die dann in einem Abbildungsverzeichnis aufscheint, während man in geschwungenen Klammern eine lange Beschriftung angeben kann, die für das Abbildungsverzeichnis zu ausführlich wäre. Die Angabe der Kurzbeschriftung ist jedoch völlig Optional, und kann auch weggelassen werden! In diesem Fall scheint die lange Beschriftung im Abbildungsverzeichnis auf.

Eine sehr wichtige Zeile ist jetzt der Befehl `\label{`*`Bildreferenz`*`}`, der eine Verweismarke zu der Abbildung enthält! Hier kommen wir den wahren Qualitäten von LaTeX jetzt schon nahe, denn mittels dieser Verweismarke (die beliebig gewählt werden darf) kann man innerhalb des Dokumentes dynamisch auf die entsprechende Abbildung verweisen. In unserem Beispiel lautet die Zeile `\label{testbildkey}`, und somit ist für diese Abbildung die Verweismarke „testbildkey“ reserviert[8]. Auf diese Abbildung kann jetzt aus dem normalen Text mittels des Befehles `\ref{testbildkey}` verwiesen werden, wobei LaTeX eigenständig die entsprechende Abbildungsnummer errechnet. In einem Dokument könnte das also ca. so aussehen wie in Abbildung 6, und das Endergebnis sieht man dann in Abbildung 7.

```
Wie auch in Abbildug \ref{Abb:mausbild} zu erkennen, hat
die Apple-Maus mittlerweile auch mehr als nur einen
Mausbutton,sowie ein Scrollrad auf der Oberseite.

\begin{figure}[ht]
\centering
\includegraphics[width=6cm]{bilder/maus}
\caption[Bild einer Mighty Mouse]{Bild einer neuen
Apple Macintosh Mighty Mouse}
\label{Abb:mausbild}
\end{figure}
```

Abbildung 6: Einfügen einer Grafik, Beispiel

9.1 Querverweise in Dokumenten

Querverweise können nicht nur auf Grafiken oder Tabellen zeigen! Fügt man beispielsweise die Zeile `\label{`*Verweismarke*`}` unterhalb eines `\chapter`-Befehles ein, kann mittels `\ref{`*Verweismarke*`}` auf dieses Kapitel verwiesen werden. Die richtige Kapitelnummer wird dabei dynamisch erzeugt. Selbiges gilt natürlich auch für Sections, Fußnoten (siehe Kapitel 14), etc.

Also noch einmal zusammenfassend:

`\label{`***Verweismarke***`}` — definiert innerhalb des Fließtextes das Kapitel/die Section/etc., in der/dem der Befehl steht; innerhalb einer `\figure-`

[8] Verweise müssen *einzigartig* innerhalb eines Dokumentes sein! Sonst meldet LaTeX Fehler in der Dokumentverarbeitung!

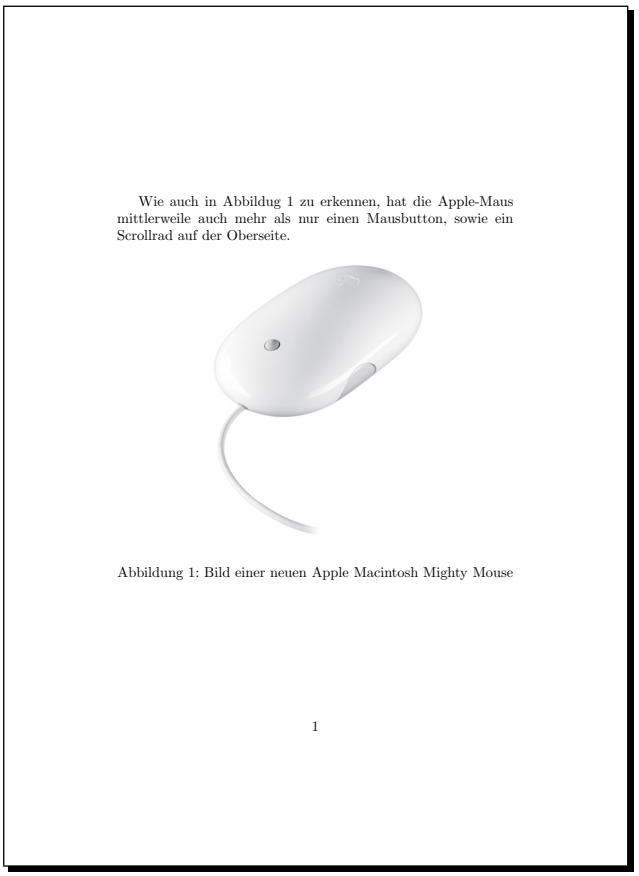


Abbildung 7: Einfügen einer Grafik, Ergebnis des Beispiels

Umgebung die Abbildungsnummer; innerhalb einer `\table`-Umgebung die Tabellennnummer.

`\ref{`***Verweismarke***`}` — Das ist der dazugehörige Querverweisbefehl. Die *Verweismarke* muß exakt angegeben werden! Groß-/Kleinschreibung ist hier ebenfalls zu beachten!

Nochmal zur Erinnerung (weil es so wichtig ist): Die *Verweismarke* in einem `\label` muss im *ganzen Dokument* einzigartig sein! Hingegen `\ref`-Befehle dürfen so oft eingesetzt werden, wie man will, wenn man z.B. an mehreren Stellen auf die selbe Abbildung oder das selbe Kapitel verweisen will!

10 Tabellen

Tabellen sind – so ehrlich muß man einfach sein – leider ein Krampf in LaTeX. Es ist natürlich klar, dass es in einem Programm, das mit reiner Texteingabe arbeitet, kompliziert ist, Tabellen zu erstellen. Ich werde mich also erstmal mit der händischen Erstellung von Tabellen in LaTeX befassen, und dann noch einige alternativen Aufzeigen. Diese jedoch ohne das Handwerkszeug zu nutzen,

ist nahezu unmöglich. Die Grundkenntnisse in den Tabellen-Codes sind dafür essentiell.

Sehen wir uns mal an, wie eine einfache Beispieltabelle (Tabelle 4) in LaTeX aussehen kann.

Tabelle 4: Beispieltabelle

Buchstabe	Nummer im Alphabet
A	1
C	3
M	13
Z	26

Den LaTeX-Code zu dieser Tabelle kann man in Abbildung 8 sehen. Wir erkennen bereits bekannte Befehle wie z.B. den `\begin`- und `\end`-Bereich, oder den Befehl `\centering`, aber sehen wir uns wieder alle Zeilen genau an.

Der Tabellen-Bereich beginnt mit dem einleitenden Befehl `\begin{table}` und endet ganz unten mit `\end{table}`, genau wie der Figure-Bereich, den wir in Kapitel 9 besprochen hatten. Zwischen diesen beiden Zeilen steht unser Tabellen-Code. Auch hier ist die Option für die Platzierung möglich, die wir ebenfalls schon bei der Beschreibung der Figure-Umgebung kennengelernt haben.

Die nächste Zeile, `\centering`, steht wieder für die zentrierte Ausrichtung unserer Tabelle.

Da Tabellen üblicherweise die Beschriftung am oberen Rand haben, kommt jetzt unsere `\caption[`*Kurzbeschriftung*`]{`*Langbeschriftung*`}`.

Mit `\begin{tabular}` fangen wir einen neuen Bereich innerhalb unserer Table-Umgebung an, der jetzt den tatsächlichen Tabelleninhalt enthält. Um die Anzahl der Spalten, und die jeweilige Orientierung der einzelnen Spalten zu definieren, kommt nach `\begin{tabular}` noch ein `{c|c}`. Die beiden Buchstaben stehen jeweils für eine zentrierte Spalte, und der Strich bedeutet, dass LaTeX zwischen den Spalten eine vertikale Linie einfügen soll. Wollen wir z.B. fünf Spalten, wobei die erste zentriert sein soll, und alle weiteren linksbündig, würden wir `\begin{tabular}{clllll}` schreiben. Wollen wir nach der ersten und vor der letzten Spalte eine vertikale Trennlinie, dann ändern wir das ganze in `\begin{tabular}{c|lll|l}`. Wollen wir unsere gesamte Tabelle und alle Spalten mit vertikalen Linien eingrenzen, sähe das so aus: `\begin{tabular}{|c|l|l|l|l|}`. Doppelte Linien sind ebenfalls möglich, indem zwei vertikale Striche nebeneinander gesetzt werden, wie hier zwischen erster und zweiter Spalte: `\begin{tabular}{c||llll}`.

Folgende Positionen für die Inhalte der einzelnen Spalten sind möglich:

`{c}` – zentriert; die Spalte wird zentriert dargestellt.

`{l}` – linksbündig; die Spalte wird linksbüdnig dargestellt.

`{r}` – rechtsbündig; die Spalte wird rechtsbündig dargestellt.

`{p{`***Breite***`}}` – Blocksatz; die Spalte wird im Blocksatz dargestellt, und in der definierte Breite fixiert. Längere Inhalte werden mehrzeilig dargestellt. Die Breite kann mit den üblichen Angaben definiert werden, also z.B. `{p{3cm}}`.

```
\begin{table}[ht]
\centering
\caption[Beispieltabelle]{Beispieltabelle}
\begin{tabular}{c|c}
Buchstabe  &  Nummer im Alphabet  \\
\hline
A       &       1       \\
C       &       3       \\
M       &       13      \\
Z       &       26      \\
\end{tabular}
\label{Tab:Beispieltabelle
\end{table}
```

Abbildung 8: Beispieltabelle, Quellcode

Damit ist unsere Tabular-Umgebung definiert, und wir können mit dem Tabelleninhalt beginnen. Jede Zeile einer Tabelle endet hier mit einem doppelten Backslash \\. Davor werden die Inhalte jeder Spalte durch ein &-Zeichen getrennt. Dadurch ergibt sich unsere erste Tabellenzeile `Buchstabe & Nummer im Alphabet \\`.

Eine horizontale Linie wird durch den Befehl `\hline` eingefügt.

Darunter kommen unsere weiteren Tabelleninhalte. Jede Spalte getrennt durch ein &-Zeichen, und jede Zeile beendet mit einem doppelten Backslash \\. Um besseren Überblick zu behalten, können dazwischen jedoch beliebig viele Tabulatoren oder Leerzeichen gesetzt werden. LaTeX nimmt diese Platzhalter nicht mit in die Tabelle auf!

Nachdem wir alle Inhalte eingetragen haben, beenden wir die Tabular-Umgebung mit `\end{tabular}`.

Als nächstes kommt unsere Verweisreferenz mit `\label{`*Verweisname*`}`.

Zuguterletzt schließen wir auch die Table-Umgebung mit `\end{table}`.

Wie man bereits erkennen kann, ist es sehr simpel kurze Tabellen mit wenig Inhalt zu erstellen, aber sobald die Komplexität der Tabelle zunimmt, wird es mühsam. Und als Biologen haben wir üblicherweise mit größeren Datensätzen und somit komplexeren Tabellen zu tun! Welche Alternativen hat man also, abgesehen von der händischen Codeeingabe?

Für Microsoft Excel sowie für OpenOffice Calc existieren bereits entsprechende Plugins, um Tabellen in LaTeX-Code umzuwandeln. Diese umgewandelten Tabellen bedürfen meist kleiner Nachbearbeitungen, aber grundsätzlich lassen sich damit sehr leicht große, komplexe Datensätze in LaTeX übernehmen.

Alternativ existiert auch noch der *Tabellenasistent* in WinShell. Über *Ausführen* → *Tabellenasistent* öffnet sich ein kleiner Assistent, der eine simple Möglichkeit bietet, Tabelleninhalte zu schreiben. Auch hier gilt, dass händisches Nachbearbeitung meist notwendig ist!

Weder die Plugins für Excel und Calc, noch der Tabellenassistent, wissen

üblicherweise wo man gerne Trennlinien einsetzt, bzw. ob Spalten zentriert oder links-/rechtsbündig sein sollen.

10.1 schönere Tabellen mittels `booktabs`

Die Standardtabellen von LaTeX sind simpel und zweckmässig. Wer jedoch Tabellen erstellen will, wie sie auch in wissenschaftlichen Publikationen eingesetzt werden, hat auch diese Möglichkeit.

Dafür ist ein neues Paket notwendig, welches wir in der Präambel unseres Dokumentes mit dem Befehl `\usepackage{booktabs}` einbinden.

Jetzt bieten sich folgende zusätzliche Befehle an, die unsere Tabelle optisch aufpeppen. Diese Befehle sind nur innerhalb der `\begin{tabular}` ... `\end{tabular}`-Umgebung erlaubt!

addlinespace — Hiermit wird ein kleiner Abstand zwischen der Tabellenbeschriftung und der eigentlichen Tabelle geschaffen. Der Befehl gehört direkt unterhalb des `\begin{tabular}`-Befehls.

toprule — Eine etwas dickere Linie, die am oberen Rand der Tabelle eingesetzt werden sollte.

midrule — Eine dünnere Linie, die in etwa der üblichen Linie des Befehls `\hline` entspricht, jedoch mehr Abstand zum Tabellentext darüber und darunter einhält.

bottomrule — Eine etwas dickere Linie, die am unteren Rand der Tabelle eingesetzt werden sollte.

Ein wichtiger Hinweis: durch die größeren Abstände zum Tabellentext, die bei den eben genannten horizontalen Linien eingesetzt werden, sind vertikale Spaltentrennlinien nicht mehr möglich! Wer also seine Spalten mit vertikalen Linien trennen will, sollte weiterhin den Befehl `\hline` für seine Tabellenlinien verwenden.

Den Quelltext unseres vorigen Tabellenbeispiels sieht man in Abbildung 9, und die zugehörige Tabelle ist in Tabelle 5 abgebildet

Tabelle 5: Beispieltabelle

Buchstabe	Nummer im Alphabet
A	1
C	3
M	13
Z	26

```
\begin{table}[ht]
\centering
\caption[Beispieltabelle]{Beispieltabelle}
\begin{tabular}{cc}
\addlinespace
\toprule
Buchstabe  &  Nummer im Alphabet  \\
\midrule
A       &       1       \\
C       &       3       \\
M       &       13      \\
Z       &       26      \\
\bottomrule
\end{tabular}
\label{Tab:Booktabs-Tabelle}
\end{table}
```

Abbildung 9: Beispieltabelle mit `\booktabs`, Quellcode

11 Auflistungen und Beschreibungen

LaTeX kennt drei Arten von Listen.

- Nummerierste Listen.
- Durch Auflistungszeichen definierte Listen (wie diese Auflistung).
- Beschreibungen.

Alle drei Listentypen werden mit einem `\begin{`*Listentyp*`}` eingeleitet, und mit einem `\end{`*Listentyp*`}` beendet. Dazwischen werden die einzelnen Zeilen mit `\item` voneinander getrennt. Der LaTeX-Code zur Auflistung am Beginn dieses Kapitel sieht beispielsweise so aus, wie in Abbildung 10 gezeigt.

Die LaTeX-Namen der drei Auflistungstypen lauten

- Für nummerierste Listen `\begin{enumerate}` ... `\end{enumerate}`
- Für unstrukturierte Listen `\begin{itemize}` ... `\end{itemize}`
- Für Beschreibungen `\begin{description}` ... `\end{description}`

Jeder einzelne Auflistungspunkt wird dabei durch ein `\item` eingeleitet (wie im Beispiel ersichtlich).

Eine Ausnahme bildet hierbei der Typ *description*, dessen Zeilen mit `\item [`*Schlüsselwort*`]` beginnt, wobei das Schlüsselwort das zu beschreibende Element ist! Das Beispiel in Abbildung 11 verdeutlicht das.

Natürlich ist es auch möglich, mehrschichtige Aufzählungen, bzw. unterschiedliche verschachtelte Listen zu erstellen. Um so etwas zu erreichen, beginnt

```
\begin{itemize}
\item Nummerierste Listen.

\item Durch Auflistungszeichen definierte Listen
(wie diese Auflistung).

\item Beschreibungen.

\end{itemize}
```

Abbildung 10: Auflistungstyp *itemize*

```
\begin{description}
\item[MS Office] Office-Paket von Microsoft, mit
diversen Büroprogrammen.

\item[OpenOffice] Freies Office-Paket von Sun
Microsystems, dass die meisten Funktionen von MS Office
ebenfalls beinhaltet.

\item[StarOffice] Komerzielle Adaption von OpenOffice,
ebefnfalls von Sun Microsystems.

\end{description}
```

Abbildung 11: Auflistungstyp *description*

man einfach ein weiteres `\begin{`*Listentyp*`}` ... `\end{`*Listentyp*`}` mitten in einer Auflistung.

Die Aufzählungszeichen lassen sich auch relativ einfach anpassen. Am einfachsten ist das bei den nummerierten Listen. Hier reicht es, volgendes Paket in die Präambel einzubinden – `\usepackage{enumerate}` – und den Listeneintrag um eine Kleinigkeit zu erweitern:

`\begin{enumerate}[I.]` für römische Ziffern als Aufzählungszeichen (I., II., III., IV., ...).

`\begin{enumerate}[a.]` für Kleinbuchstaben als Aufzählungszeichen (a., b., c., d., ...).

`\begin{enumerate}[A.]` für Großbuchstaben als Aufzählungszeichen (A., B., C., D., ...).

Um die Aufzählungszeichen bei unstrukturierten Listen zu ändern, können in der Präambel einige Befehle gesetzt werden:

`\renewcommand{\labelitemi}{`*gewünschtes Symbol*`}` für die erste Ebene

`\renewcommand{\labelitemii}{`*gewünschtes Symbol*`}` für die zweite Ebene

`\renewcommand{\labelitemiii}{`*gewünschtes Symbol*`}` für die dritte Ebene

`\renewcommand{\labelitemiv}{`*gewünschtes Symbol*`}` für die vierte Ebene

`\renewcommand{\labelitemv}{`*gewünschtes Symbol*`}` für die fünfte Ebene

Je nach Listentiefe müssen nur die Befehle geändert werden, die auch tatsächlich genutzt werden! Wer also maximal drei verschachtelte Ebenen in einer Liste nutzt (und auch in jeder Ebene ein eigenes Symbol definieren will!), braucht auch nur die ersten drei Befehle in die Präambel aufzunehmen.

Für Ebenen, für die das Listenzeichen nicht neu definiert wurde, bleiben die LaTeXStandardzeichen aktiv.

12 Hervorheben von Texten

Wie euch sicher bereits beim Lesen dieses Dokumentes aufgefallen ist, kann man Wörter in LaTeX auch Fett oder Kursiv hervorheben, wie man es von Word gewohnt ist.

In LaTeX arbeiten wir hier wieder mit einer speziellen Codeeingabe für jede dieser Möglichkeiten. Die folgende Auflistung zeigt, was alles möglich ist.

Hervorgehoben Um Wörter oder Textteile *hervorzuheben* nutzt man den Befehl `\emph{`*hervorzuhebender Text*`}`. Der Text wird hierbei Kursiv dargestellt. Setzt man weiteren Text *innerhalb* von bereits hervorgehobenem Text wieder in eine `\emph{}`-Umgebung, wird dieser wieder normal dargestellt.

Kursiv Um Text *kursiv* darzustellen benutzt man den Befehl `\textit{`*kursiver Text*`}`.

Fett Um Text **fett** darzustellen kommt der Befehl `\textbf{`*fetter Text*`}` zum Einsatz.

Unterstriche Um Text zu unterstreichen, nutzt man den Befehl `\underline {`*unterstrichener Text*`}`.

Schräggestellt Im Gegensatz zu kursivem Text ändert sich beim *schräggestellten* Text nicht die Darstellung der Zeichen (man merkt das besonders beim Buchstaben a - kursiv *a* - schräggestellt *a*). Erreicht wird das mit dem Befehl `\textsl{`*schräggestellter Text*`}`.

Kapitälchen Um Text in Kapitälchen darzustellen, gibt es den Befehl `\textsc{`*Kapitälchen*`}`.

Schreibmaschinenschrift Es kann manchmal nützlich sein – besonders wenn man Angaben über Computercode machen möchte – eine `Schreibmaschinenschrift` zu verwenden. Hierfür gibt es in LaTeX den Befehl `\texttt{`*Schreibmaschinenschrift*`}`. Leider kommt durch den Schriftwechsel

jedoch der Zeilenumbruch oft durcheinander, Überstehende Worte können die Folge sein! Man muß also sehr genau darauf achten, ob LaTeX keine Fehler im fertigen Dokument produziert, bzw. möglicherweise händische Wortabteilungen einfügen (wie das funktioniert, steht in Kapitel 18).

Sans Serif Um Schrift sans Serif darzustellen gibt es den Befehl `\textsf{`*sans Serif*`}`.

Grundsätzlich lassen sich alle dieser Befehle auch kombinieren, wobei nicht alle Kombinationen auch miteinander funktionieren! Fette Kapitälchen sind z.B. möglich, aber fette, kursive Kapitälchen werden nur als fette Kapitälchen angezeigt! Fett und Kursiv geht auch, und fett, kursiv und sans Serif funktioniert ebenfalls. Man muß einfach ein wenig experimentieren.

Auch die Schriftgröße kann mit den entsprechenden Befehlen angepaßt werden, wobei eher davon abzusehen ist, Schriftgrößenanpassung im Text einfach so vorzunehmen. Die einzelnen Teile des Dokuments erhalten von LaTeX selbst bereits die richtigen Angaben, und händische Eingriffe verbessern da nur selten das Schriftbild.

Die möglichen Größen können folgendermassen angewählt werden:

tiny – Sehr kleine Schrift; wird mit dem Befehl `\tiny` oder mit der Umgebung `\begin{tiny}` ... `\end{tiny}` ausgelöst. Die Schrift hat <small>diese Größe</small>.

footnotesize – Kleine Schrift, die auch für Fußnoten benutzt wird; wird mit dem Befehl `\footnotesize` oder mit der Umgebung `\begin{footnotesize}` ... `\end{footnotesize}` ausgelöst. Die Schrift hat <small>diese Größe</small>.

small – Beinahe so groß wie die normale Schrift; wird mit dem Befehl `\small` oder mit der Umgebung `\begin{small}` ... `\end{small}` ausgelöst. Die Schrift hat diese Größe.

normalsize – Diese Schriftgröße entspricht genau der Dokumentschriftgröße.

large – Etwas größere Schrift als normal; wird mit dem Befehl `\large` oder mit der Umgebung `\begin{large}` ... `\end{large}` ausgelöst. Die Schrift hat diese Größe.

Large – Große Schrift, die auch für Sektion-Überschriften verwendet wird; man beachte die Groß-/Kleinschreibung! Wird mit dem Befehl `\Large` oder mit der Umgebung `\begin{Large}` ... `\end{Large}` ausgelöst. Die Schrift hat diese Größe.

huge – Sehr große Schrift, die auch für den Dokumenttitel verwendet wird; wird mit dem Befehl `\huge` oder mit der Umgebung `\begin{huge}` ... `\end{huge}` ausgelöst. Die Schrift hat diese Größe.

Huge – Riesige Schrift; man beachte die Groß-/Kleinschreibung! Wird mit dem Befehl `\Huge` oder mit der Umgebung `\begin{Huge}` ... `\end{Huge}` ausgelöst. Die Schrift hat diese Größe.

Die Schriftgrößen stehen in Abhängigkeit zur anfangs gewählten Dokumentschriftgröße (siehe dazu Kapitel 7). Sie werden sozusagen prozentuell in ihrer Größe verändert, wenn die allgemeine Dokumentschrift größer oder kleiner eingestellt wird.

13 Abkürzungen für häufige Begriffe definieren

Wenn man eine Diplom- oder Doktorarbeit schreibt, so wird man häufig bestimmte Begriffe verwenden, die mit dem Thema der Arbeit zusammenhängen. Nehmen wir an, wir schreiben über Haushunde, und brauchen den Begriff *Canis lupus familiaris* häufig in der Arbeit. Dieser Begriff ist lang, ungewohnt zu tippen, und soll außerdem jedesmal kursiv hervorgehoben werden – ein durchaus mühsames Unterfangen.

Für genau diesen Fall gibt es in LaTeX die Möglichkeit, häufig benutze Begriffe in der Präambel mit kurzen Referenznamen zu belegen! Im Dokument muß dann jeweils nur die Referenz getippt werden, und LaTeX ersetzt diese dann durch die definierte Parallelbezeichnung.

Im Beispielfall von *Canis lupus familiaris* könnte das ganze z.B. so aussehen, wie in Abbildung 12 gezeigt.

```
\newcommand{\clf}{\emph{Canis lupus familiaris}}
```

Abbildung 12: Definition von Textkürzeln

Wichtig ist dabei, dass der Befehl `\newcommand{\`*Kürzel*`}{`*referenzierter Text*`}` immer *vor* der Zeile `\begin{Document}` in der Präambel vorkommt. Das Kürzel muß immer einen Backslash vorangestellt haben, damit es in LaTeX eingesetzt werden kann. Der referenzierte Text kann die üblichen LaTeX-Definitionen zum hervorheben von Texten enthalten. Zahlen sind im Kürzel leider nicht erlaubt, da sie von LaTeX falsch interpretiert werden!

Nachdem wir jetzt also die Zeichenfolge `\clf` für *Canis lupus familiaris* definiert haben, reicht es, im Dokument einfach `\clf` zu schreiben, und LaTeX ersetzt das Kürzel mit dem langen Referneznamen. Es gilt jedoch zu beachten, dass LaTeX üblicherweise *kein* Leerzeichen hinter eine derartige Referenz setzt. Damit die Worte also nicht aneinanderkleben, empfiehlt es sich, einen weiteren Backslash hinter das Kürzel zu setzen, wenn es innerhalb eines Satzes eingefügt wird, jedoch keinen Backslash dahinter zu setzen, wenn es vor einem Satzzeichen (Punkt, Komma, etc.) verwendet wird! Die beiden Möglichkeiten sehen also wie folgt aus:

- das Kürzel innerhalb eines Satzes wird mit `\clf\` aufgerufen.
- vor einem Satzzeichen wird es mit `\clf` aufgerufen, damit kein Leerzeichen zwischen dem ersetzten Text und dem Satzzeichen auftritt.

Der Befehl `\newcommand` ermöglicht innerhalb von LaTeX noch einige andere Dinge, und wenn man sich gut genug auskennt, kann man damit das komplette Layout des Dokumentes ändern. Das erfordert jedoch sehr tiefgreifendes Wissen rund um die programmtechnischen LaTeX-Befehle, welches man üblicherweise nicht benötigt (auch ich habe mich nie damit beschäftigt).

14 Fußnoten

Diese Kapitel wird sehr kurz ausfallen, denn Fußnoten sind wohl das simpelste, was es in LaTeX gibt.

Um eine Fußnote zu einem Wort einzufügen, reicht es, hinter das entsprechende Wort den Befehl `\footnote{`*Text der Fußnote*`}` zu setzen, und LaTeX übernimmt alles weitere. Die Fußnote wird automatisch nummeriert und an das Ende der Seite gesetzt. LaTeX versucht dabei auch, optisch attraktiv vorzugehen – besonders, wenn das Wort zu dem die Fußnote gehört, sich schon am untersten Rand der Seite befindet! Probleme könnte es hier mit sehr langen Fußnoten, bzw. mit mehreren Fußnoten auf einer Seite geben, aber Üblicherweise kann man sich auf das Typesetting von LaTeX verlassen, dass hier von selbst ordentlich arbeitet.

Fußnoten dürfen leider nicht in Kapitelüberschriften, Abbildungsbeschriftungen oder Tabellenbeschriftungen vorkommen. Hier spuckt LaTeX Fehlermeldungen aus.

15 Literaturverzeichnis und Literaturverweise

Dieser Abschnitt wird etwas ausführlicher, da wir uns mit einigen Spezialitäten von LaTeX sowie mit einem weiteren Programm namens *JabRef* auseinandersetzen werden.

Aber erst einmal zur Theorie einer LaTeX-Literaturliste:

1. Als erstes erstellen wir eine Literaturdatenbank, die im Grunde einer langen Liste mit allen Artikeln und Büchern entspricht, die wir für unsere Arbeit gelesen haben, und zitieren wollen (oder auch nicht). In dieser Literaturliste wird jedem Dokument ein *einmaliges* Schlüsselwort zugewiesen, das diesen Eintrag symbolisiert.

2. In unserem LaTeX-Dokument verweisen wir durch spezielle Befehle auf die Literaturdatenbank, indem wir Querverweise zu den Schlüsselwörtern einfügen (ähnlich den bereits beschriebenen Querverweisen zu Abbildungen oder Tabellen).

3. Am Ende unseres LaTeX-Dokumentes fügen wir zwei Befehle ein, wobei der erste den Stil der Literaturverweise definiert, während der zweite den Ort (sprich: den Dateinamen) unserer Literaturliste bezeichnet.

Haben wir all das ordentlich gemacht, erhalten wir eine wunderbar einfache Literaturliste, die sich mit nahezu keinem Zeitaufwand in jedweden Stil umstellen läßt. Egal, ob ein Journal seine Literaturverweise lieber im Format *Autor(Datum)* hat, oder Zahlen in eckigen Klammern [1] bevorzugt, mit LaTeX kann man den Literatur-Stil schnell und sauber ändern.

Für die Literaturliste erstellen wir jetzt eine neue Datei mit der Endung `.BIB`. In dieser `.BIB`-Datei werden der Reihe nach alle Literaturzitate nach einem bestimmten Schema eingefügt. Die Reihenfolge der Zitate ist hierbei völlig irrelevant, da LaTeX die Reihung abhängig vom verwendeten Zitatstil selbstständig vornimmt (also alphabetisch, nach auftreten im Text, nach Datum, etc.). Kommt ein neuer Artikel hinzu, fügen wir ihn einfach ganz unten in unserer `.BIB`-Datei an. Eine solche `.BIB`-Datei sieht jetzt etwa wie in Abbildung 13 gezeigt, aus.

```
@book{WS91,
author = {Lars Werdelin and Nikos Solounias},
title = {The {H}yaenidae: taxonomy, systematics and
evolution},
publisher = {Universitetsforlaget},
series = {Fossils and Strata},
number = {30},
address = {Oslo},
year = {1991}
}

@article{Roh05,
author = {Nadin Rohland and Joshua L. Pollack and
Doris Nagel and C\'{e}dric Beauval and Jean Airvaux
and Svante P\"{a}\"{a}bo and Michael Hofreiter},
title = {The Population History of Extant and Extinct
Hyenas},
publisher = {Oxford University Press},
journal = {Molecular Biology and Evolution},
number = {12},
volume = {22},
pages = {2435--2443},
year = {2005}
}
```

Abbildung 13: Beispiel einer `.BIB`-Datei

Damit wir uns nicht zu sehr mit den, durchaus auf den ersten Blick unverständlichen, Einträgen herumquälen müssen, nehmen wir jetzt ein weiteres Programm zu hilfe, das uns eine solche `.BIB`-Datei sehr komfortabel erstellt. Das Programm *JabRef* (derzeit in der Version 2.8.1) ist Plattformunabhängig[9] und bietet uns eine hübsche grafische Oberfläche, in der wir unsere Literatureinträge einfügen können. Es bietet weiters eine Literatursuch-Funktion sowie Exportfunktionen für andere Literaturdatenbank-Programme und Dokumentformate. Vor der Nutzung ist es essentiell, das *Java Runtime Environment* von *SUN Microsystems* bereits installiert zu haben (zu finden auf `http://www.java.com/de/download`)

JabRef läßt sich über die Website `http://jabref.sourceforge.net/` beziehen. Es stehen Downloads für Windows, Mac OSX sowie als reine .JAR-Datei (=Java-Datei) für Linux zur Verfügung.

Nachdem wir das Programm gestartet haben, können wir eine neue Datenbank anlegen, und über den +-Knopf neue Einträge hinzufügen. *JabRef* empfielt jeweils, abhängig von der Art des Eintrags (Artikel, Book, Techreport, etc.), wichtige und weniger wichtige Informationen zum jeweiligen Eintrag. Siehe hierzu auch einen Screenshot aus meiner Literaturdatei für dieses Dokument, in Abbildung 14.

[9] d.h. es wurde für Windows, Mac OSX und Linux programmiert

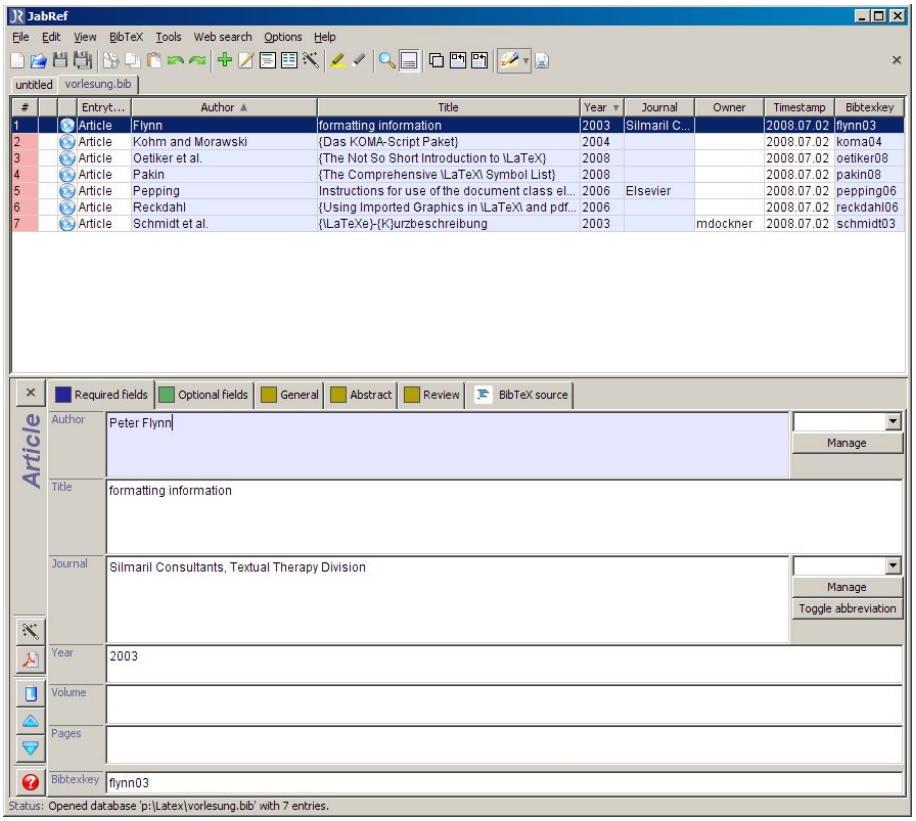


Abbildung 14: Screenshot aus *JabRef*

Grunsätzlich kann man jedoch sagen, die *absolut essenziellen* Felder für jeden Eintrag, sind **Author, Title, Year** und **Bibtexkey**. Die ersten drei Angaben sind klar, denn ohne diese läßt sich kein ordentlicher Literaturverweis erstellen. Und die vierte Angabe definiert jetzt unser Schlüsselwort für *exakt diesen Literatureintrag*. Der Bibtexkey muß einzigartig sein, denn über diesen Schlüssel läuft die Referenzierung in LaTeX.

Natürlich sind weitere Angaben in der Literaturliste ebenfalls wichtig, aber ich überlasse es anderen Vorlesungen, das zu klären. Uns reichen vorerst die minimalen Angaben zur Veranschaulichung.

Wenn man übrigends bei einem Literatureintrag auf den Knopf *BibTeX source* clickt, erhält man den tatsächlichen Code, der von *JabRef* erstellt wird, und unserem Beispiel aus Abbildung 13 sehr ähnelt.

Sehr wichtig ist jetzt, dass *JabRef* bzw. die .BIB-Datei, keinerlei Länderanpassung kennt. Somit *müssen* Umlaute hier in der LaTeX-Codierung wie in Kapitel 5 und Tabelle 1 beschrieben, eingetragen werden.

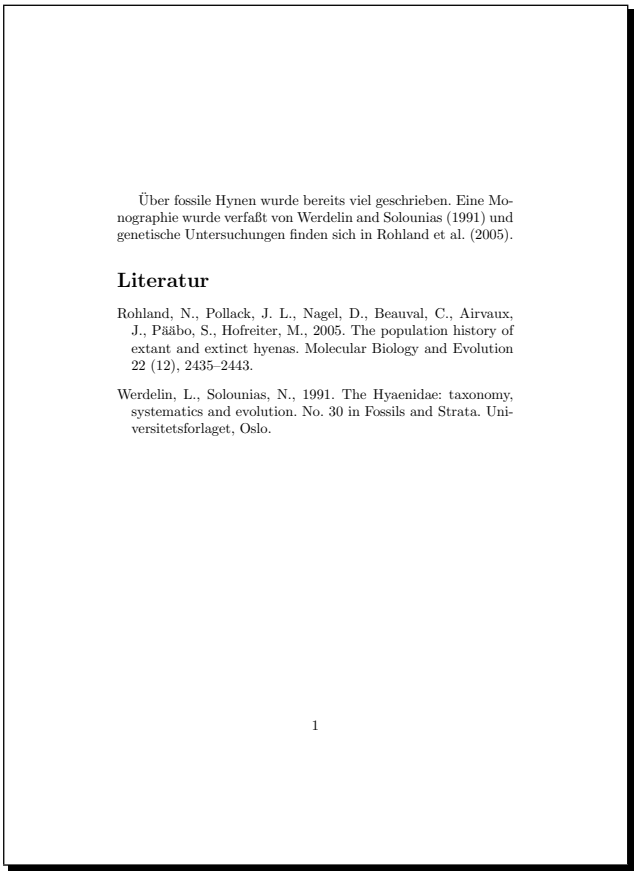
Bei der Angabe des Autors bzw. der Authoren werden alle Authoren nach Möglichkeit mit ihrem vollständigen Namen angegeben, und zwar in der Reihung *Vorname Nachname* (ohne Komma dazwischen!) und mit einem *and* verknüpft. Egal wie viele Authoren man angibt, LaTeX formatiert die Literaturliste nach günstigen Gesichtspunkten, bzw. fügt in der Text-Referenz des Zitates automa-

tisch ein „et al“ nach dem ersten Author ein.

Außerdem ist es nützlich, Sonderzeichen in den Namen mit den entsprechenden LaTeX-Codes zu schreiben (siehe Tabelle 1), anstatt auf Formatierungspakete zu vertrauen. Dadurch wird der Einsatz der Literaturdatenbank in allen Dokumenten erleichtert, egal in welcher Sprache man das Dokument verfasst.

Auch zu beachten ist, dass in der **Title**-Zeile entweder der gesamte Titel oder zumindest die Großbuchstaben im Titel in geschwungene Klammern zu setzen sind! LaTeX formatiert den Titel sonst üblicherweise in Kleinbuchstaben, die geschwungenen Klammern erzwingen jedoch die Ausgabe in Großbuchstaben.

Um das zu Verdeutlichen sind in Abbildung 15 die beiden Einträge aus Abbildung 13 nach Harvard-Zitierregeln abgebildet.


Über fossile Hynen wurde bereits viel geschrieben. Eine Monographie wurde verfaßt von Werdelin and Solounias (1991) und genetische Untersuchungen finden sich in Rohland et al. (2005).

Literatur

Rohland, N., Pollack, J. L., Nagel, D., Beauval, C., Airvaux, J., Pääbo, S., Hofreiter, M., 2005. The population history of extant and extinct hyenas. Molecular Biology and Evolution 22 (12), 2435–2443.

Werdelin, L., Solounias, N., 1991. The Hyaenidae: taxonomy, systematics and evolution. No. 30 in Fossils and Strata. Universitetsforlaget, Oslo.

1

Abbildung 15: Ein Zitat nach Harvard-Zitierregeln

Ist unsere Literaturdatei fertig, speichern wir sie im selben Verzeichnis, in dem sich auch unsere `.TEX`-Datei befindet. Jetzt fehlen nur noch die zusätzlichen Befehle, die LaTeX sagen, wie es unsere Literaturliste einbinden soll, und natürlich die Literaturverweise im Dokumenttext! Als erstes muß (zumindest bei den meisten Zitierstilen) ein bestimmtes Paket (das je nach Zitierstil ein anderes sein kann) in die Präambel eingebunden werden. Näheres findet sich bei der Auflistung der Zitierstile.

Als weiteres werden noch die Verweise auf unsere Literaturdatei, sowie

die Wahl des Zitierstiles benötigt. Beispiele zu den beiden Befehlen finden sich in Abbildung 16 und müssen sich innerhalb der `\begin{document}` ... `\end{document}`-Umgebung befinden (bevorzugt ganz am Ende des Dokumentes, also genau oberhalb der `\end{document}`-Zeile).

```
\bibliographystyle{plain}
\bibliography{literatur}
```

Abbildung 16: Beispiel der Literatur-Befehle

Die erste Zeile `\bibliographystyle{`*Stil*`}` definiert den gewünschten Zitierstil. LaTeX kennt viele Zitierstile, und ich werde versuchen erstmal die wichtigsten für englischsprachige Dokumente aufzulisten[10].

`\bibliographystyle{unsrt}` — ein Zitierstil, in dem die Zitate mit einer Zahl in eckiger Klammer durchnummeriert werden, und die Literaturliste nach Reihenfolge der Zitate im Text sortiert ist.

Dieser Stil benötigt kein spezielles Paket in der Präambel.

`\bibliographystyle{plain}` — ähnlich *unsrt*, sind die Zitate mit einer Zahl in eckiger Klammer, aber die Literaturliste wird nach den Erstautoren sortiert. Vornamen der Autoren werden in der Literaturliste voll ausgeschrieben.

Dieser Stil benötigt kein spezielles Paket in der Präambel.

`\bibliographystyle{abbrv}` — exakt gleich wie *plain*, aber die Vornamen der Autoren werden in der Literaturliste abgekürzt.

Dieser Stil benötigt kein spezielles Paket in der Präambel.

`\bibliographystyle{apa}` — ein Stil, der den Vorgaben der *American Psychology Association* entspricht, und den typischen Zitierstil in der wissenschaftlichen Literatur wiedergibt. Zitate werden mit *Autor(Datum)* im Text abgedruckt, und die Literaturliste ist nach Erstautor sortiert. Vornamen der Autoren werden in der Literaturliste abgekürzt. Leider unterstützt der Stil nur die 5. Edition der APA-Zitierregeln, und wurde nie auf Stand der 6. Edition gebracht!

Dieser Stil benötigt das Paket `\usepackage{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{apalike}` — eine Variante des *apa*-Stiles.

Dieser Stil benötigt das Paket `\usepackage{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen. Auch dieser Stil unterstützt nur die 5. Edition der APA-Zitierregeln!

[10] am auffälligsten dadurch, dass bei mehreren Autorennamen als Verbindungswort „and“ statt „und“ verwendet wird!

`\bibliographystyle{apalike2}` — eine weitere Variante des *apa*-Stiles. Dieser Stil ist multilingual, da das Verbindungswort „and“ durch ein Ampersand („&“) ersetzt wurde. Auch dieser Stil unterstützt nur die 5. Edition der APA-Zitierregeln!

Dieser Stil benötigt das Paket `\usepackage{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{elsart-harv}` — ein Stil der den Harvard-Vorgaben für wissenschaftliche Ziate entspricht. Auch hier wird, wie bei *apa*, im Text mit *Autor(Datum)* zitiert, und auch hier sind die Zitate alphabetisch geordnet. Einige kleinere stilistische Unterschiede heben *elseart-harv* jedoch vom *apa*-Stil ab.

Dieser Stil benötigt das Paket `\usepackage{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{chicago}` — auch dieser Stil ähnelt dem *apa*-Stil. Zitate scheinen als *Autor(Datum)* im Text auf, und die Literaturliste ist nach Erstautoren sortiert.

Dieser Stil benötigt das Paket `\usepackage{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{naturemag}` — der Zitierstil der Zeitschrift *Nature* hat Zitate mit kleinen, hochgestellten Zahlen, und die Literaturliste nach Reihung der Zitate im Text durchnummeriert.

Dieser Stil benötigt das Paket `\usepackage[super]{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen. Der hier genutzte Zusatzbefehl [super] erzeugt die von *Nature* geforderten hochgestellten Zahlen bei den Zitaten.

Hinweis: Es existiert außerdem noch ein Stil namens nature*, der jedoch veraltet ist, und nicht mehr verwendet werden sollte!*

Die deutschen Äquivalente (sofern vorhanden) zu diesen Stilen haben ähnliche Namen.

`\bibliographystyle{gerunsrt}` — die deutsche Version des Stiles *unsrt*.

Dieser Stil benötigt das Paket `\usepackage{bibgerm}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{gerplain}` — die deutsche Version des Stiles *plain*.

Dieser Stil benötigt das Paket `\usepackage{bibgerm}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{gerabbrv}` — die deutsche Version des Stiles *abbrv*.

Dieser Stil benötigt das Paket `\usepackage{bibgerm}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

`\bibliographystyle{gerapalike}` — eine deutsche Version des *apa*-Stiles, die allerdings nicht im Internet zu bekommen ist, sondern von mir angepaßt wurde. Die Datei steht in meiner Lehrmaterialsammlung zum Download

bereit. Da ich den `apa`-Stil nur angepasst habe, werden auch hier nur die Vorgaben der 5. Edition der APA-Zitierregeln genutzt!

Dieser Stil benötigt das Paket `\usepackage{natbib}` in der Präambel, um die gewünschten Ergebnisse zu erzielen.

Zu dem englischen Stilen *elsart-harv*, *chicago* und *nature* gibt es kein deutsches Äquivalent. Eine Alternative zu *apa* bietet der Stil *apalike2*, der mehrsprachig ist, weil das Bindewort „and“ durch ein Ampersand („&“) ersetzt wurde.

Es existieren noch viele weitere Zitierstile, wie beispielsweise im Dokument *Stildateien.PDF* (vom Informatik-Institut der Humboldt-Universität, Berlin) gezeigt. Allerdings wollen wir uns nicht von all diesen Möglichkeiten verwirren lassen, denn der Stil, der für Naturwissenschaftler im deutschsprchigen Raum am häufigsten verwendet wird, ist sicher die Zitierung nach *Autor(Datum)*.

Der Befehl `\bibliography{`*Literaturdatei*`}` gibt an, wo sich unsere Literaturdatenbank befindet. *Literaturdatei* entspricht also dem Namen unserer `.BIB`-Datei (die Angabe erfolgt wieder ohne die Dateiendung `.BIB`!) bzw. – falls sich die Literaturdatenbank nicht im selben Verzeichnis wie unser LaTeX-Dokument befindet – dem vollständigen Pfad zu unserer Literaturdatei. Es ist sogar möglich, mehrere Literaturdatenbanken gleichzeitig einzubinden, indem man einfach mehrere `\bibliography`-Befehle mit den verschiedenen Literaturdateien untereinander setzt.

Hinweis: LaTeX hat Probleme mit Pfadangaben die Leerzeichen enthalten! somit würde ein Literaturverweis auf

```
\bibliography{c:/MeinVerzeichnis/Literaturdatei}
```

funktionieren, jedoch

```
\bibliography{c:/Mein Verzeichnis/Literaturdatei}
```

zu einen Fehler führen! Das selbe gilt auch für Dateinamen, die Leerzeichen enthalten. Es ist jedoch schon erlaubt Punkte (.), Bindestriche (-) oder Unterstriche (_) innerhalb von Dateinamen oder Ordnernamen zu verwenden.

Wie benutzt man jetzt die Zitate in seinem LaTeX-Dokument? Dafür existiert der Befehl `\cite{`*Schlüsselwort*`}`. Als *Schlüsselwort* wird einfach der Zitatschlüssel eingefügt, und LaTeX sucht sich den entsprechenden Eintrag aus der Literaturdatenbank heraus. In unserem Beispiel aus Abbildung 15 würde der entsprechende LaTeX-Code in etwa folgendermaßen aussehen, wenn der Zitatschlüssel *WS91* lautet: `\cite{WS91}`. Und für *Roh05* folgendermassen: `\cite{Roh05}`. Der vollständige Quelltext für das Beispiel (Abbildung 15) ist in Abbildung 17 zu sehen.

Da es bei Autor/Datum-basierten Zitaten jedoch auch die Darstellmöglichkeit *(Autor, Datum)* gibt, bietet das Paket *natbib* noch einen weiteren Zitierbefehl namens `\citep{`*Schlüsselwort*`}` an, mit dem das gewünschte Format erreicht werden kann.

Will man mehrere Zitate gleichzeitig angeben, so funktionert das übrigends einfach, indem die Schlüsselwörter mit Kommata getrennt werden; also `\cite{`*Schlüsselwort1,Schlüsselwort2,Schlüsselwort3*`}`!

Hat man alles fertig (also sowohl die Literaturliste als auch das LaTeX-Dokument) startet man BibTeX in *WinShell* über den Button  . Dadurch

```
\documentclass[10pt,a5paper]{article}
\usepackage{german,graphicx,natbib}
\begin{document}

Über fossile Hyänen wurde bereits viel geschrieben.
Eine Monographie wurde verfaßt von \cite{WS91}
und genetische Untersuchungen finden sich in
\cite{Roh05}.

\bibliographystyle{elsart-harv}
\bibliography{meineliteratur}
\bibliography{c:/sonstiges/weitere_literatur}
\end{document}
```

Abbildung 17: Quelltext zum Literaturbefehle-Beispiel aus Abbildung 15.

wird – definiert durch den Zitierstil – eine Datei mit der Endung .BBL angelegt, die LaTeX jetzt in das fertige Dokument einbaut. Um das zu tun, starten wir noch zwei weitere Durchläufe von pdfLaTeX (den ersten um die Literaturdatei einzubinden, den zweiten um die Zitate-Verweise richtig einzutragen), und sehen uns dann unser fertiges Dokument an.

Mit dieser Methode ist es also möglich, dynamisch auf die Literaturliste zuzugreifen, und durch Änderung der Parameter `\bibliographystyle` sowie dem dazugehörigen Paket in der Präambel (gefolgt von einem Durchlauf von BibTeX sowie zwei finalen Durchläufen von pdfLaTeX) einen beliebigen Zitierstil auf das komplette Dokument anzuwenden!

> *Hinweis:* Durch die Änderung des Literaturstiles kann es zu Problemen mit den temporären Dateien kommen, die LaTeX bei den pdfLaTeX-Durchläufen anlegt. Deshalb empfiehlt es sich nach einer Änderung des Literaturstiles zuerst alle temporären Dateien im Arbeitsverzeichnis zu löschen, und dann per  →  →  →  nochmal die `.PDF`-Datei komplett neu zu erstellen. Mehr Informationen dazu finden sich in Kapitel 16.

16 Warnungen, Fehlermeldungen, und was dabei zu beachten ist

Jetzt ist es an der Zeit, einen wichtigen Punkt anzusprechen. Wie wir bereits in unserem Dokument-Verzeichnis sehen, legt LaTeX für seine Arbeit sehr viele verschiedene Dateien an! All diese Dateien, die Endungen wie .AUX, .BBL, .BLG, .LOF, .LOG, .LOT, .TOC, .OUT, und .IDX tragen können, sind für die dynamischen Vorgänge der Dokumentgenerierung wichtig. Allerdings kann es teilweise vorkommen, dass eine Veränderung (und hierbei vor allem eine Veränderung

bei der verwendeten Literaturstil-Vorlage) zu Problemen in der Verarbeitung führt! LaTeX beendet dann beispielsweise seinen Durchlauf unvorhergesehen, oder meldet viele unerwartete Fehler. Sollte so etwas auftreten, ist es hilfreich, das Dokumentverzeichnis aufzuräumen, indem man alle Zusatzdateien löscht. Die wichtigen Dateien tragen *ausschließlich* die Endungen .TEX, .BIB und .WSP[11], wohingegen alle weiteren Dateien problemlos entfernt werden können.

Es ist also *essenziell* einen Überblick über die eigenen Dateien zu haben, sowie Dateien aufgrund ihrer dreistelligen Endungen identifizieren zu können! Vor allem Windows blendet die Endungen gerne aus, wenn eine Datei mit einem bestimmten Programm verknüpft ist (z.B. sieht man bei Word-Dokumenten die Endung .DOC nicht, oder bei Vielen Bilddateien nicht, ob es sich um `.JPG`, .BMP oder was auch immer handelt).

Dieses „Feature“, das oft genug für Verwirrung sorgt (ich denke dabei an die jüngste Entwicklung mit Office 2007, wo die neuen .DOCX-Dokumente in Windows nicht von dem alten .DOC-Format zu unterscheiden sind, jedoch für Benutzer älterer Office-Pakete unleserlich sind!), kann aber auch deaktiviert werden. Ich gehe darauf im Anhang noch näher ein.

Als Entwarnung kann aber gesagt werden: oft liegen die Probleme in der .AUX oder der .BBL-Datei, und es reicht meistens, diese beiden zu löschen.

WinShell meldet üblicherweise nach einem erfolgreichen pdfLaTeX-Durchlauf im Ausgabefenster, wieviele Fehler, Warnungen und Overfull- bzw. Underfull Boxen produziert wurden. Aber was genau bedeuten diese Angaben?

- Treten *Fehler* auf, bedeuted das grobe Schwierigkeiten bei der Verarbeitung, und es wird üblicherweise keine `.PDF`-Datei produziert. Fehler können verschiedene Ursachen haben, *WinShell* gibt aber meist die Zeile des Fehlers an (dazu muß im Ausgabefenster einfach nach oben gescrollt werden, bis man eine rote Zeile findet, die den Fehler meldet. Die Zahl in Klammer sagt uns, welche Zeile im Dokument den Fehler enthält.)

  Häufig werden Fehler durch fehlende Voraussetzungen verursacht. Verwendet man z.B. einen LaTeX-Befehl, der ein bestimmtes Paket voraussetzt, hat dieses Paket aber in der Präambel nicht eingebunden, erkennt LaTeX den Befehl nicht, und meldet einen Fehler. Auch kann es vorkommen, dass man ein LaTeX-Paket in der Präambel einbinden will, das nicht installiert ist, was ebenfalls zu einem Fehler führt. Selbiges gilt für das Einbinden von Literaturstilen.

- *Warnungen* sind weniger schlimm, und normalerweise produziert pdfLaTeX auch eine hübsche `.PDF`-Datei. Dennoch sollten Warnungen überprüft werden, weil die erstellte Datei meist nicht korrekt aussieht, wenn Warnungen ausgegeben wurden. Ein hübsches Beispiel wäre die Verwendung des Befehls `\tableofcontents`: der erste pdfLaTeX-Durchlauf produziert sofort eine Warnung, da beim ersten Durchlauf erst die .TOC-Datei erstellt wird, die das Inhaltsverzeichnis enthält. Ein weiterer Durchlauf von pdfLaTeX hat jetzt keine Warnung mehr – LaTeX konnte auf die .TOC-Datei zugreifen, und das Inhaltsverzeichnis ordnungsgemäß einbinden. Somit gibt es auch keine Warnung mehr.

[11] .TEX- und .BIB-Dateien sind uns ja bereits bekannt; .WSP-Dateien stellen WinShell-Projektdateien dar, auf die ich in Kapitel 22.1 noch eingehen werde.

Oft kommt es auch zu Warnungen beim Einsatz von `figure`-Umgebungen. Wenn die Platzierungsregeln hier zu streng eingeschränkt sind, und LaTeX sie nicht einhalten kann, gibt der Editor eine Warnung aus, die darauf hinweist, dass die Einschränkung gelockert wurde. *Beispiel:* ein Eintrag von `\begin{figure}[h]` kann von LaTeX durchaus auf `\begin{figure}[ht]` erweitert werden. Siehe dazu die Platzierungsregeln für Abbildungen und Tabellen in Kapitel 9.

- Eine *Overfull Box* tritt immer dort auf, wo LaTeX keinen perfekten Zeilenwechsel durchführen konnte. Da LaTeX auf Kriterien des Buchdrucks zurückgreit, achtet es nicht nur auf korrekte Worttrennung, und Buchstabenabstände, sondern im Blocksatz auch besonders auf die Laufweite der Leerzeichen! Bevor LaTeX ein zu großes Leerzeichen in einen Textteil einfügt (das die Lesbarkeit sowie das Schriftbild negativ beeinflußt), versucht es die Worte am Zeilenende entsprechend abzuteilen. Da das nicht immer perfekt funktioniert ragen manchmal Buchstaben geringfügig über den Zeilenrand hinaus. Überprüft man die Zeilen, die Overfull Boxes produziert haben, merkt man, dass es hier weitere Angaben gibt. LaTeX gibt jeweils in Klammer an, um wie viele Punkte die Zeile zu lang ist, was in etwa „(2.76556pt too wide)“ sein kann. Angaben kleiner als 1 können hier üblicherweise ignoriert werden – man sieht sie nicht. Zeilen, die einen Overfull-Wert zwischen 1 und 2 aufweisen, sollte man einmal überprüfen, vielleicht teilt LaTeX ein Wort falsch ab, oder kennt keine richtige Abteilung (das kann besonders bei Fachbegriffen ein Problem sein). Angaben größer als 2 sollten auf jeden Fall überprüft werden! Um Zeilen mit Overfull Boxen im Text leichter zu finden, kann die Dokumentklassen-Option `draft` aktiviert werden, die in jeder problematischen Zeile eine schwarze, rechteckige, Box abdruckt!

- Das logische Gegenstück zur Overfull Box ist die *Underfull Box*, die Zeilen markiert, die eine zu geringe Laufweite aufweisen! So etwas kommt eigentliche nie vor, da LaTeX hier sehr genau vorgeht. Es kann jedoch in Tabellen vorkommen, oder bei falscher Nutzung des Zeilensprung-Befehls[12]. Underfull Boxen werden *nicht* bei Einsatz der `draft`-Option markiert!

Ich fasse einige Abhängigkeiten in Tabelle 6 zusammen. Natürlich sind das bei weitem nicht alle, aber sie geben einen guten Überblick, welche Abhängigkeiten existieren können.

17 Schriftarten wählen

LaTeX ist weitaus strikter und eingeschränkter mit verschiedenen Schriftarten, als beispielsweise Word. Man kann nicht einfach eine andere Schriftart per Menü auswählen, und irgendeiner Textstelle zuweisen. Man muß, wie wir bereits anhand vieler Beispiele gesehen haben, auch Schriftarten per LaTeX-Paket(en) einbinden. Abgesehen davon sind auch in verschiedene Dokumentklassen andere Schriftarten fix eingebunden (z.B. in alle KOMA-Script Klassen).

Eine gute Auflistung diverser Schriftarten findet sich im *LaTeX Font Catalogue* im Internet: http://www.tug.dk/FontCatalogue

[12] Siehe Kapitel 18.

Tabelle 6: Abhängigkeiten von Befehlen

LaTeX-Befehl	zugehörige Abhängigkeit
`\usepackage{...}`; Paket in der Präambel	Paket muß installiert sein!
`\bibliographystyle{...}`; Stil des Literaturverzeichnisses	Stil muß installiert sein!
Zur richtigen Darstellung von Umlauten und scharfem ß	`\usepackage{fontenc}`, `\usepackage{inputenc}`
`\includegraphics{...}`; Bilder einfügen	`\usepackage{graphicx}`
`\url{...}`; zum Einfügen von Internetadressen	`\usepackage{url}`
`\href{...}`; zum Erstellen von clickbaren Links innerhalb des `.PDF`-Dokumentes	`\usepackage{hyperref}`
`\cite{foo}`; Zitierbefehl	Zitatname `foo` muß in Literaturliste enthalten sein!
`\citep{...}`; Zitierbefehl	`\usepackage{natbib}`
`\linenumbers, \pagewiselinenumbers`; Zeilennummerierung im `.PDF`-Dokument	`\usepackage{lineno}`
`\onehalfspacing, \doublespacing`; für größere Zeilenabstände im Dokument	`\usepackage{setspace}`
`\ref{foo}`; Referenzlink innerhalb der Datei	`\label{foo}` muß existieren!

Die meisten der dort gelisteten Schriftarten müssen müssen händisch nachinstalliert werden (siehe dazu Kapitel 23) und funktionieren teilweise auch nur eingeschränkt mit Text- und Mathematik-Sonderzeichen. Hier hilft nur ausprobieren.

Einige, bereits in unserer Installation enthaltene, Schriftarten werde ich hier inklusive der notwendigen Paketnamen auflisten (außerdem benötigt jede Schriftart noch das Paket `\usepackage[T1]{fontenc}`):

Times – eine *Times New Roman* ähnliche, Serifen-Schrift. Das zugehörige Paket wird mit `\usepackage{mathptmx}` geladen.

Helvetica – eine serifenlose Schriftart, die über das Paket `\usepackage[scaled]{helvet}` sowie den zusätzlichen Eintrag `\renewcommand*\familydefault{\sfdefault}` geladen wird.

Latin Modern Sans – eine andere serifenlose Schriftart, die über das Paket `\usepackage{lmodern}` sowie den zusätzlichen Eintrag `\renewcommand* \familydefault{\sfdefault}` geladen wird.

Courier – eine Schreibmaschinenschriftart mit fixer Buchstaben-Laufweite, die über das Paket `\usepackage{courier}` sowie den zusätzlichen Eintrag `\renewcommand*\familydefault{\ttdefault}` geladen wird.

Bera Mono – eine serifenlose Schreibmaschinenschriftart mit fixer Buchstaben-Laufweite, die über das Paket `\usepackage[scaled]{beramono}` sowie den zusätzlichen Eintrag `\renewcommand*\familydefault{\ttdefault}` geladen wird.

Calligra – eine Kalligraphie-Schrift, die über das Paket `\usepackage{calligra}` geladen wird, und im Text durch `\begin{calligra}...\end{calligra}` aktiviert wird.

Zapf Chancery – eine Kalligraphie-Schrift, die über das Paket `\usepackage{chancery}` geladen wird.

Die angeführten Schriften sollen als Beispiele dienen, was für Möglichkeiten in LaTeX bestehen, und mit welchen Problemen man sich herumschlagen muß. Leider ist das kein triviales Thema, und bedarf viel Herumexperimentierens.

Dennoch ist es besser, Schriftarten nicht willkürlich zu wechseln. Manche Dokumentklassen (wie z.B. *apa* oder *elsart*) geben aus gutem Grund bestimmte Schriftarten vor! Hier etwas zu ändern zerstört den Grundgedanken, den der Hersteller der Dokumentklasse im Kopf hatte.

18 Besondere Textzeichen, Ligaturen, Abteilungsregeln, mehrere Dokumente

Mir war nicht ganz klar, wie ich dieses Kapitel zusammenfassend nennen sollte. Ich werde ein paar besondere Textzeichen und -möglichkeiten vorstellen, eigene Abteilungsvorgaben für Worte die LaTeX nicht kennt beschreiben, das Verknüpfen mehrerer Dokumente erklären, und noch auf ein paar weitere Dinge verweisen.

Beginnen wir erst mit der Wortabteilung (engl. „hyphenation“). LaTeX hat durch das Paket `ngerman` bereits die deutschen Abteilungsregeln gelernt. Aber besonders in der Wissenschaft kommen oft Fachbegriffe bzw. Eigennamen vor, die nicht den üblichen Regeln entsprechen, und oft von LaTeX ignoriert, oder falsch abgeteilt werden. Es ist aber möglich, diesen Worten händisch Abteilungspunkte anzugeben! Dazu zwei Möglichkeiten:

- Das Wort kommt nur einmal vor, also machen wir die Abteilung direkt im Text. Dazu wird an jeder Stelle, an der eine Abteilung möglich ist einfach ein `\-` eingefügt! „Donaudampfschiff“ würde also so aussehen: `Donau\-dampf\-schiff`. Man kann damit LaTeX auch zwingen, z.B. „Helikopter“ nach seinem griechischen Namensursprung so abzuteilen: `He\-li\-ko\-pter`.

- Das Wort kommt häufig im Text vor, und wir wollen die Abteilungsregeln nicht jedesmal angeben, dann läßt sich einfach im Dokument eine Befehlszeile namens `\hyphenation{...}` definieren, in der alle Worte mit ihren Abteilungsregeln eingetragen werden (jeweils getrennt durch Leerzeichen). Innerhalb des hyphenation-Befehls darf jedoch nur ein Bindestrich „-“ an die Abteilungspositionen gesetzt werden. Unsere Beispiele würden also so aussehen: `\hyphenation{Donau-dampf-schiff He-li-ko-pter}`.

  LaTeX verwendet diese Abteilungsregeln ab der Stelle im Dokument, an der der hyphenation-Befehl aufscheint. Meistens will man die Regeln im gesamten Dokument wirksam haben, also empfiehlt es sich, `\hyphenation{...}` gleich unterhalb von `\begin{document}` zu setzen.

Die Laufweite innerhalb diverser Worte ist ebenfalls ein Punkt, auf den man in LaTeX achten kann (aber es gibt keinen Zwang, sondern es bleibt jedem

selbst überlassen!). Üblicherweise werden im Buchdruck manche Buchstaben näher aneinander gesetzt, um die Lesbarkeit bestimmter Worte zu verbessern. Ebenso ist es manchmal nützlich, vor allem bei zusammengesetzten Worten, diese Ligaturen aufzubrechen. Beispielsweise Schifffahrt und Schifffahrt (zu Beachten ist die leicht größere Laufweite zwischen dem zweiten und dritten f). Solche Ligaturen lassen sich durch einfügen der beiden Zeichen \/ an der gewünschten Stelle im Wort erreichen.

Leerzeichen die keinen Zeilensprung verursachen, sind auch eine wichtige Sache. Manchmal gibt man Titel zu Personen an, will aber nicht, dass exakt nach dem Titel ein Zeilensprung kommt. Oder man will bei Längenangaben ein Leerzeichen aber keinen Absatz! Für soche Fälle gibt es in LaTeX das „~“ Symbol. Man schreibt dann also z.B. `Hr. Dr.~Mayer` (für Hr. Dr. Mayer), oder `100~km` (für 100 km). LaTeX wird an dieser Stelle nie die Zeile umbrechen.

Ein etwas schwieriger Punkt sind in LaTeX die Anführungszeichen. Grundsätzlich kennt das Programm englische, deutsche und französische Anführungszeichen (die Bezeichnung bezieht sich auf den typischen häufigen Einsatz in normalen Texten). Leider kann man jedoch nicht einfach das übliche Zeichen benutzen, da LaTeX auch zwischen einleitendem und abschließendem Anführungszeichen unterscheidet! Darauf sollte man als Autor achten, wenn man auf ordentliche Anführungszeichen wert legt.

Das “englische” Anführungszeichen sieht aus wie zwei kleine 66 bzw. 99 und befindet sich immer am oberen Zeilenrand. In LaTeX erreicht man das einleitende durch `‘‘` und das abschließende durch `’’`.

Das „deutsche“ Anführungszeichen ist umgekehrt, 99 und 66, wobei das einleitende 99 am Fuß der Zeile steht, während das abschließende 66 am oberen Rand der Zeile steht. Erreicht wird das über die Zeichen `"‘` und `"’`.

Das »französische« Anführungszeichen erreicht man über die Eingabe von `">` und `"<`.

Leider hat unser Editor *WinShell* keine Automatische Erkennung für einleitende und abschließende Anführungszeichen, und somit müssen wir diese entweder jedesmal händisch angeben, oder in ein Makro erstellen und entsprechend aufrufen[13].

Um drei Fortsetzungspunkte (. . .) zu drucken, kann man in einer Schriftart mit variabler Laufweite nicht einfach drei Punkte setzen (der Unterschied sieht in etwa so aus: ... zu . . .). In LaTeX lautet der Befehl für Fortsetzungspunkte einfach `\dots`.

Gedankenstriche in Texten sind üblicherweise länger, als das normale Minus-Zeichen. LaTeX kennt drei verschiedene Längen für Gedankenstriche.

- Das einfach Minus-Zeichen (`-`) findet üblicherweise wirklich nur als mathematisches Zeichen Verwendung.
- Das doppelte Minus-Zeichen (`--`) wird in LaTeX als kurzer Gedankenstrich dargestellt. In deutschen Dokumenten ist es üblich noch ein Leerzeichen vor und hinter den Gedankenstrich zu setzen. *Beispiel:* Dieser Text zeigt – wie eben beschrieben – den Einsatz des kurzen Gedankenstriches.
- Das dreifache Minus-Zeichen (`---`) wird in LaTeX als langer Gedankenstrich dargestellt. Üblicherweise wird der lange Gedankenstrich in englischen

[13] wie das funktioniert, erkläre ich im nächsten Kapitel.

Texten verwendet, und ohne Leerzeichen eingesetzt. *Beispiel:* This example shows the use of the dash—as detailed above—in english language texts.

Hochgestellter und Tiefgestellter Text kann manchmal notwendig sein, will man beispielsweise Zahnformeln oder chemische Elemente angeben. Um Text hochgestellt darzustellen, fügt man ihn in eine `\textsuperscript{...}`-Umgebung ein, was z.B. für den dritten Oberkiefermolaren M^3 so aussieht: `M\textsuperscript{3}`.

Für tiefgestellten Text müssen wir erst das Paket `subscript` in die Präambel einbinden[14]. Danach kann man mittels `\textsubscript{...}`-Umgebung Text tiefgestellt darstellen. Für H_2O schreibt man dann z.B. `H\textsubscript{2}O`.

Zeilensprünge, Seitenwechsel und Leerzeilen können auch in LaTeX erzwungen werden, obwohl man damit sehr vorsichtig umgehen sollte. Das Textlayout kann durch zu häufigen Einsatz solcher händischer Variationen leiden. Um einen Zeilenwechsel zu erzwingen, nutzt man den Befehl `\newline`. Damit wird auch der Blocksatz aufgebrochen, und die neue Zeile sofort begonnen. Um einen Seitenwechsel zu erzwingen existiert der Befehl `\newpage`. Und um eine Leerzeile beliebiger Größe einzufügen, kann man `\vspace{`*Größe*`}` angeben. Als Größenangabe ist wieder nahezu alles möglich, wie in Kapitel 9 erklärt. Im folgenden ist ein 1.5 cm-Abstand mittels `\vspace{1.5cm}` eingefügt:

LaTeX bietet außerdem die Möglichkeit, mehrere Dokumente zusammenzuhängen. Das ist nützlich, wenn man die Reihenfolge der Kapitel umstellen will, oder nicht sicher ist, ob man bestimmte Teile im Text haben will. Man legt einfach ein Hauptdokument an, in dem die Präambel sowie Titelseite, etc. definiert sind, und verweist dann nur mit dem Befehl `\include{`*Dokumentname*`}` auf weitere .TEX-Dateien. Ein Beispiel für so ein Hauptdokument findet sich in Abbildung 18.

Der `\include`-Befehl fügt jedes Dokument beginnend mit einer neuen Seite ein. Will man jedoch keinen Seitenvorschub, kann man alternativ auch den `\input`-Befehl verwenden.

19 Mathematische Formeln

Der Hintergrund von LaTeX beruht eigentlich auf dem Setzen mathematischer Formeln. Für genau diesen Zweck wurde die Benutzerumgebung TeX ursprünglich entwickelt, die später mit mehreren Erweiterungen zu LaTeX wurde.

Obwohl viele Mathematik-Befehle nativ in LaTeX verfügbar sind, empfielt es sich das Paket `amsmath` zu laden, um auch für besondere Fälle Vorsorge zu treffen.

Da es für das setzen mathematischer Formeln sehr viele ausführliche Dokumente (siehe z.B. (Peters, 2005)) und Homepages gibt, werde ich mich hier nur sehr kurz halten, und nur einige rudimentäre Funktionen erläutern.

Um eine mathematische Umgebung in LaTeX einzuleiten, setzt man einfach das Zeichen `$`. Um die mathematische Umgebung wieder zu beenden, kommt

[14] Das Paket `subscript` ist möglicherweise nicht in der Installation enthalten, und muß nachinstalliert werden! Mehr dazu steht in Kapitel 23.

```
\documentclass[10pt,a4paper]{article}
\usepackage{ngerman}

\begin{document}

\title{Mein Dokument}
\author{Martin Dockner}
\date{\today}
\maketitle
\tableofcontents

\include{einleitung}

\include{hauptteil}

\include{diskussion}

\include{danksagung}

\end{document}
```

Abbildung 18: Ein Hauptdokument mit `\include`-Befehlen.

ebenfalls ein `$`-Zeichen. Will man eine größere Formel, oder mehrere Zeilen mathematischer Formeln angeben, empfiehlt sich eine vom Text abgesetzte Mathematikumgebung, die man mit `\begin{displaymath}` einleiten und mit `\end{displaymath}` beenden kann[15]. Im Beispiel sieht das ganze dann so aus:

- Einzelnes `$`-Zeichen: Die Formel steht im Text, also hier $a^2 + b^2 = c^2$.
- Abgesetzte Mathematikumgebung mittels `\begin{displaymath}`...`\end{displaymath}`: Die Formel steht abgehoben vom Text.

$$a^2 + b^2 = c^2$$

  Danach wird der Text einfach weitergeführt.

Innerhalb der `$`-Umgebungen gelten völlig eigene Befehlssätze und Regeln, von denen ich ein paar in den folgenden Beispielen kurz vorstellen will.

- Einfache Additionen oder Subtraktionen lassen sich mit den üblichen Zeichen schreiben:
  `$a + b = c$` wird zu $a + b = c$.
- Multiplikationen lauten folgendermassen:
  `$a \times b = c$` wird zu $a \times b = c$

[15] Alternativ kann man eine Mathematikumgebung auch mit `\[` beginnen und mit `\]` beenden. Die (ebenfalls mögliche) Nutzung des doppelten Dollarzeichens `$$` sollte vermieden werden, da sie mittlerweile veraltet ist!

- Divisionen können entweder mit dem Divisionszeichen geschrieben werden: `$a \div b = c$` wird zu $a \div b = c$, oder aber als Brüche: `$\frac{a}{b} = c$` wird zu $\frac{a}{b} = c$.

- Exponenten zu schreiben ist einfach: `$a^{2} + b^{2} = c^{2}$` wird zu $a^2 + b^2 = c^2$

- Auch tiefgestellte Zahlen sind nicht schwer: `$a_{1} + a_{2} + \ldots + a_{x} = a_{n}$` wird zu $a_1 + a_2 + \ldots + a_x = a_n$

- Das letzte Beispiel läßt sich (etwas verändert) auch mit Summenzeichen schreiben: `$a_{1} + a_{2} + \ldots + a_{\infty} = \sum_{i=1}^{\infty} a_{i}$` wird zu $a_1 + a_2 + \ldots + a_\infty = \sum_{i=1}^{\infty} a_i$. Es sieht jedoch schöner in seiner eigenen Zeile aus (also in einer `\begin{displaymath}...\end{displaymath}`-Umgebung):

$$a_1 + a_2 + \ldots + a_\infty = \sum_{i=1}^{\infty} a_i$$

- Wurzelausdrücke werden wie folgt geschrieben: `$\sqrt{x+y}$` wird zu $\sqrt{x+y}$

  Für andere als Quadratwurzeln sieht es so aus: `$\sqrt[n]{x+y}$` wird zu $\sqrt[n]{x+y}$

LaTeX unterscheidet beim Satz von Formeln sehr wohl zwischen inline und vom Text abgesetzten Bereichen (wie im Summenformel-Beispiel zu sehen). Je nach Anwendungsgebiet steht es einem völlig offen die jeweils passende Darstellungsform zu wählen.

Uns Biologen nützen die mathematischen Formeln nämlich auch anders. Chemische Symbole wie z.B. ${}^{12}_{6}\mathrm{C}$ oder ${}^{14}_{6}\mathrm{C}$ schreibt man folgendermassen: `$^{12}_{6}\text{C}$`. Auch Zahnformeln haben nie hübscher ausgesehen als unter Zuhilfenahme der Bruch-Funktion: „*... die Zahnformel von Canis lupus lautet* $\frac{3143}{3142}$...“. Man schreibt in diesem Fall `$\frac{3143}{3142}$`.

Mit den kompletten Mathematik-Befehlen, die in LaTeX enthalten sind, könnte man noch viele Seiten füllen, aber hier ist es sinnvoller auf auf bereits existierende Arbeiten zu verweisen, wie Oetiker et al. (2008), Peters (2005) und American Mathematical Society (2002) sowie Pakin (2008) für viele mathematische Sonderzeichen und griechische Buchstaben.

20 Index-Erstellung

Ein Index kann in großen Dokumenten die Suche nach bestimmten Begriffen erleichtern. Ähnlich dem Inhaltsverzeichnis bietet es dem Leser die Möglichkeit gezielt Themenbereiche zu finden. LaTeX unterstützt auch die Implementierung eines Index (allerdings nicht so reibungslos wie beim Inhaltsverzeichnis).

Die Vorgehensweise für die Indexerstellung ist folgende:

- Als erstes muß das Paket `makeidx` in der Präambel eingebunden werden. Außerdem wird oberhalb `\begin{document}` – also in der Präambel – die Zeile `\makeindex` eingetragen.
- Jeder Begriff, der in den Index soll muß gekennzeichnet werden. Das funktioniert, indem vor oder nach dem entsprechenden Begriff ein `\index{Begriff}` gesetzt wird.
- An der Stelle, wo der Index im Dokument angezeigt werden soll, muß jetzt noch der Befehl `\printindex` eingetragen werden (üblicherweise am Ende des Dokumentes, knapp vor oder nach dem Literaturverzeichnis).

Da es für `makeindex` üblicherweise keinen Ausführungs-Knopf gibt, müssen wir uns selbst einen in *WinShell* anlegen. Das funktioniert über *Einstellungen* → *Benutzerprogramme*. Wir suchen uns das Programm *makeindex.exe* auf unserer Festplatte (üblicherweise sollte es sich im Verzeichnis `C:\Programme\TeXLive 2011\bin\win32\makeindex.exe` befinden) und legen es als neuen Befehl an. Siehe dazu auch Abbildung 19.

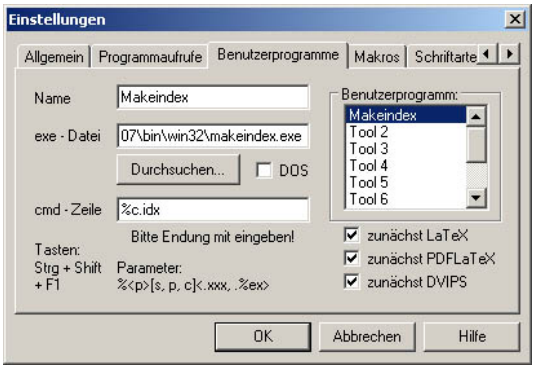


Abbildung 19: Benutzerdefinierten Befehl für Indexerstellung anlegen

Mit diesem Befehl können wir jetzt – ähnlich dem bereits bekannten Button für BibTeX – unseren Index generieren. Dazu starten wir `Makeindex`, und danach lassen wir noch einmal LaTeX durchlaufen. Jetzt sollte sich am Ende unseres Dokumentes (oder wo auch immer unser `\printindex`-Befehl steht) ein Index mit all unseren indizierten Worten befinden.

Für weitere Informationen über `makeindex`, verschachtelte Indizes, etc. empfehle ich die Lektüre von Görlach (2004) oder Mösgen (1998)

21 Titelseite gestalten

Die Titelseite muß vor allem bei Diplomarbeiten und Dissertationen einem vorgegebenen Standard entsprechen. Leider bietet LaTeX da keine große Hilfe, weder im Grundumfang der Software, noch über irgendwelche Pakete. Um eine Titelseite, die den individuellen Anforderungen der Universität entspricht, zu gestalten, müssen wir also leider ein wenig pfuschen.

Hierfür verwenden wir die folgenden Befehle, die ich (teilweise) bereits in diesem Skriptum erwähnt habe:

- Zeilen beenden und neue Zeilen beginnen. Dazu nutzen wir \\.
- Horizontale Abstände händisch einfügen. Dafür nutzen wir den Befehl `\vspace{...}`.
- Textorientierung von Zentriert auf Links-/Rechtsbündig ändern. Dafür nutzen wir die Befehle `\raggedright` bzw. `\raggedleft`.

All diese Befehle lassen sich in der Befehlszeile `\title{...}` einsetzen. Ein Beispiel findet sich in Abbildung 21.

```
\title{
\raggedleft
\includegraphics[width=7cm]{uni_wien_logo}\\
\vspace{2cm}
\centering
Diplomarbeit\\
\Huge LaTeX und WORD im Vergleich\\
\vspace{4cm}
\large angestrebter akademischer Grad\\
\Large Magister der Naturwissenschaften (Mag.rer.nat)\\
\vspace{2cm}
\raggedright
\large
Verfasser: Martin Dockner\\
Matrikelnummer: 9601633\\
Studienrichtung: Anthropologie\\
\vspace{2cm}
Wien, am \today \\
}
\author{}
\date{}
```

Abbildung 20: Erstellen einer Titelseite, Beispiel

Der dazugehörige LaTeX-Code ist in Abbildung 20 wiedergegeben. Es wurde hier auch das Logo der Universität Wien (Dateiname `uni_wien_logo.JPG`) als Bild eingefügt, das sich entsprechend im gleichen Verzeichnis befinden muß. Zu finden ist das Logo im Internet unter der Adresse http://learnserver.csd.univie.ac.at/statistics/Module/Graphiken/uni_wien_logo.jpg

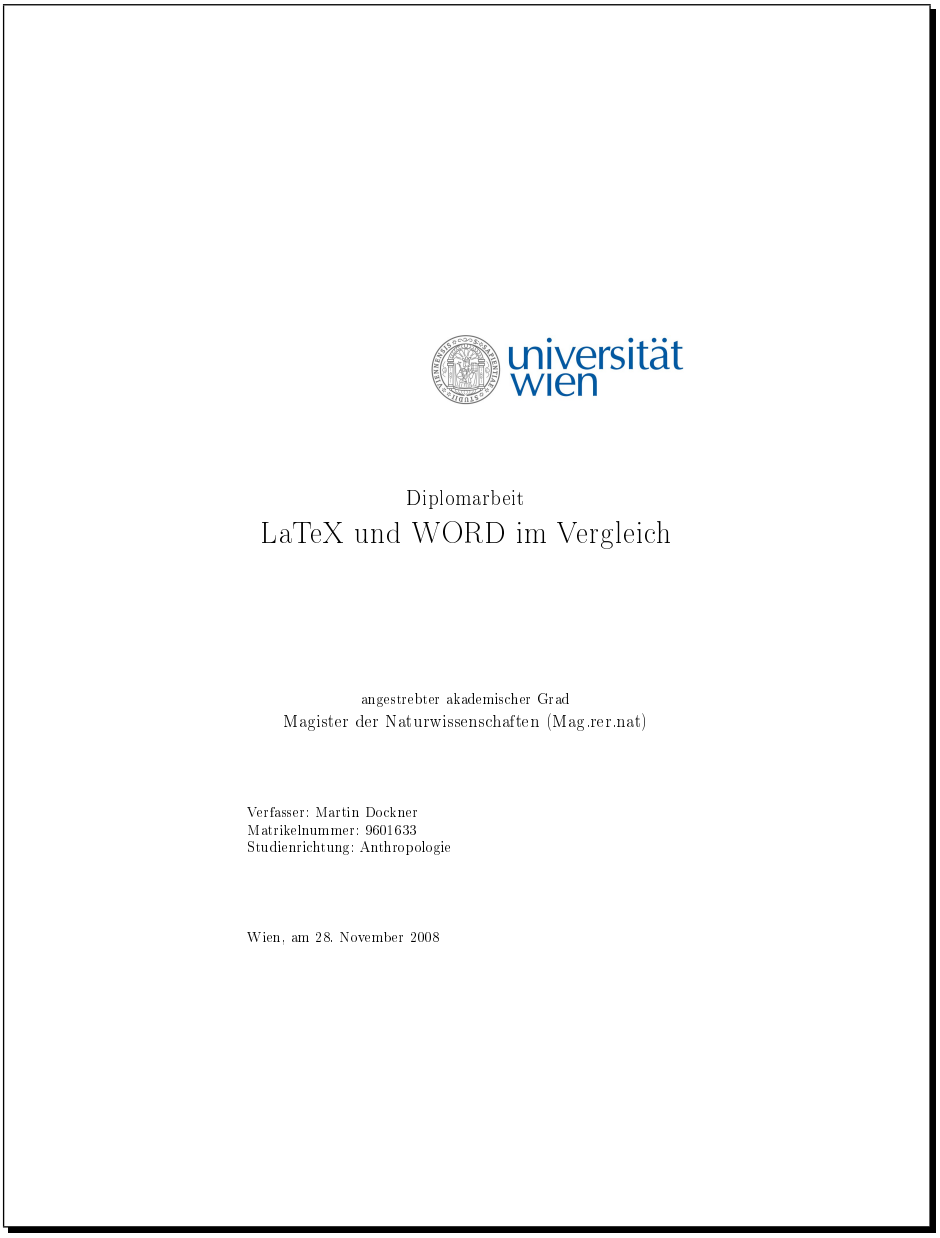


Abbildung 21: Beispiel einer Titelseite

22 Der Editor WinShell

Wer in LaTeX schreibt, hat grundsätzlich eine große Auswahl, welchen Editor er benutzen will. Von textbasiertem Urgestein wie *vi* oder *emacs* (für die jeweils LaTeX-Befehlshighlighting existiert) über diverse grafisch ansprechendere Editoren wie *TeXnic Center*, *WinEdt*, *LaTeX Editor* (und eben *WinShell*) bis hin zu nahezu-WYSIWYG[16]-Editoren wie *OpenOffice* Write (wo es eine .TEX-Export-Funktion gibt!) oder *BaKoMa Tex* (wo der LaTeX-Code und das finale Ergebnis gleichzeitig dargestellt werden).

WinShell ist in dieser Liste sicher nicht der komfortabelste, aber einer der zugänglichsten. Die Menüs und Optionen sind überschaubar, und das Fehler-Ausgabefenster am unteren Bildrand ist gut strukturiert. Desweiteren ist es möglich benutzerdefinierte Programmaufrufe zu starten, und Makros zu definieren.

22.1 Projekte

WinShell bietet weiters die Möglichkeit Projekte zu erstellen. Das erleichtert die Arbeit ungemein, wenn man mehrere Dokumente bearbeitet, und gibt einen zusätzlichen Überblick über Teile des Dokumentes. Um ein Projekt zu erstellen, clicken wir in der Befehlszeile auf *Projekt* → *Neu* und erstellen eine Projektdatei (deren Dateiendung .WSP lautet). Schon erhalten wir rechts, im Projekte-Fenster eine neue Zeile mit unserem Projekt. Ein doppelclick darauf, zeigt uns fünf Bereiche, namens *Dateien, Inhaltsverzeichnis, Abbildungen, Tabellen* und *Literaturverzeichnis*. Mittels Rechtsclick auf *Dateien* können wir neue `.TEX`-Dateien zu unserem Projekt hinzufügen. Als *Haupt-TeX-Dokument* geben wir die `.TEX`-Datei an, die auch die Präambel enthält (falls wir mehrere Dateien durch den vorhin erwähnten `\include`-Befehl verknüpfen) und alle weiteren `.TEX`-Dateien als *TeX-Dokumente*.

Unser Projektname ist jetzt fett gedruckt, was bedeutet, dass dieses Projekt gerade „aktiv“ ist. Wenn ein Projekt als aktiv gekennzeichnet ist, wird immer das Haupt-TeX-Dokument aus diesem Projekt von LaTeX verarbeitet, wenn man auf pdfLaTeX oder BibTeX clickt! Hat man mehrere Projekte, muß man entsprechend das aktuell verwendete Projekt aktivieren.

Sobald wir unsere ersten pdfLaTeX-Durchläufe gestartet haben, werden auch die weiteren Projektpunkte *Inhaltsverzeichnis, Abbildungen, Tabellen* und *Literaturverzeichnis* interessant, denn dort scheinen jetzt unsere Kapitel, Abbildungen, Tabellen bzw. verwendeten Literaturzitate auf. Mittels Doppelclick auf einen dieser Einträge springt *WinShell* sofort an die entsprechende Stelle im `.TEX`-Dokument.

22.2 Makros

Makros sind nichts anderes, als Kürzel für bestimmte Befehlsfolgen. *WinShell* erlaubt uns zehn Makros zu definieren, die dann mit den Tasten shift+F1 bis shift+F10 aufgerufen werden können. Über die Menüzeile kommen wir mit einem click auf *Einstellungen* → *Makros* in das entsprechende Menü. Wie schon in Kapitel 18 erwähnt, können wir hier z.B. die Anführungszeichen definieren.

[16] **W**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et

Wir wählen also das erste Makro aus, und tippen im Textfeld (rechts) die beiden Zeichen für ein einleitendes Anführungszeichen: "‘

Dann wählen wir das zweite Makro, und tippen die Zeichen für das abschließende Anführungszeichen: "’

Nach einem click auf OK sind die Makros gespeichert, und SHIFT+F1 produziert uns jetzt die Zeichen "‘ während SHIFT+F2 die Zeichen "’ in den Text einfügt.

Natürlich können Makros auch weitaus komplexere Befehlsfolgen enthalten. Beispielsweise ein Standard-Dokument mit vordefinierter Präambel oder die Befehlszeilen für das Einfügen einer Grafik oder einer Tabelle.

23 Die Pflege der LaTeX-Installation

Wie bereits mehrfach angesprochen, kann es vorkommen, dass manche Pakete oder Literaturstile nicht in der TeX-Live Installation enthalten sind. Grundsätzlich ist es aber nicht schwierig, neue Pakete einzubinden.

Die simpelste und einfachste Methode besteht darin, das benötigte Paket einfach ins Arbeitsverzeichnis der derzeitigen LaTeX-Datei zu kopieren. Das ist insofern hilfreich, wenn man das Paket wirklich nur für dieses eine Projekt benötigt (z.B. ein exotischer Literaturstil). LaTeX geht grundsätzlich bei der Suche nach Paketen hierarchisch vor, wobei das aktuelle Verzeichnis die höchste Priorität hat, und erst danach die Verzeichnisse der LaTeX-Installation überprüft werden.

Benötigt man ein bestimmtes Paket jedoch öfter, lohnt es sich schon, die entsprechenden Stil- und Paket-Dateien dauerhaft in die LaTeX-Installation einzubinden:

23.1 LaTeX unter Windows pflegen

Das typische Installationsverzeichnis für unsere TeX-Live Distribution lautet `C:\texlive\2012`. Innerhalb dieses Ordner finden sich unmengen an Unterordnern und Dateien, die für uns nur mässig interessant sind. Es gibt nur zwei Unterordner die für uns wichtig sind, und zwar der Ordner, in dem sich die Pakete befinden, die per `\usepackage` eingebunden werden, sowie der Ordner der die Literaturstile enthält, die per `\bibliographystyle` eingebunden werden.

Der Ordner für die Pakete lautet in der Standard-Installation `C:\texlive\2012\texmf-dist\tex\latex`. Wenn wir ein neues Paket installieren wollen, legen wir hier einfach einen Unterordner an (z.B. `mypackages`) und kopieren unsere Paketdateie(en) dort hinein[17]. In den vorigen Kapiteln habe ich z.B. die Pakete `nature` oder `subscript` erwähnt. Die zugehörigen Dateien heißen `nature.sty` bzw. `subscript.sty`, und können über Google schnell gefunden werden.

Ähnlich verhält es sich mit den Literaturstil-Dateien, die sich in der Standard-Installation in `C:\texlive\2012\texmf-dist\bibtex\bst` befinden. Auch dort legen wir einfach einen Unterordner `mystyles` an, und kopieren alle Stildateien dort hinein[18]. Erwähnt habe ich z.B. die Stile `nature` oder `gerapalike`, die wir jetzt in unseren neuen Ordner kopieren.

[17] LaTeX-Pakete tragen meist die Endung `.STY`

[18] Literaturstil-Dateien tragen die Endung `.BST`

Damit LaTeX die neuen Dateien auch findet, starten wir jetzt den *TeX Live Manager* über *Start* → *Programme* → *TeX Live 2012*, und clicken auf den Menüpunkt *Aktionen*. Dort findet sich der Punkt *Neuerstellung der Dateilisten (ls-R)*, den wir jetzt einfach ausführen. Nach kurzer Zeit spuckt das Programm die Meldung „fertig“ aus, und zeigt uns damit an, dass die neuen Pakete und Stile eingebunden wurden. Jetzt können wir sie in unseren .TEX-Dokumenten verwenden.

23.2 LaTeX unter Mac OSX pflegen

Das typische Installationsverzeichnis für die *MacTeX*-Distribution lautet `/Library/TeX`. Innerhalb dieses Ordners finden sich Unmengen an Unterverzeichnissen und Dateien, die für uns nur mässig interessant sind. Für uns sind nur zwei Ordner wichtig, und zwar einerseits das Verzeichnis, in dem sich die Pakete befinden, die per `\usepackage` eingebunden werden, sowie anderseits das Verzeichnis, das die Literaturstile enthält, die per `\bibliographystyle` eingebunden werden.

Der Ordner für die Pakete lautet `/Library/TeX/Root/texmf-dist/tex/latex`. Wenn wir ein neues Paket installieren wollen, legen wir hier einfach einen Unterordner an (z.B. `mypackages`) und kopieren unsere Paketdateie(en) dort hinein[19]. In den vorigen Kapiteln habe ich z.B. die Pakete `nature` oder `subscript` erwähnt. Die zugehörigen Dateien heißen `nature.sty` bzw. `subscript.sty`, und können über Google schnell gefunden werden.

Ähnlich verhält es sich mit den Literaturstil-Dateien, die sich in `/Library/TeX/Root/texmf-dist/bibtex/bst` befinden. Auch dort legen wir einfach einen Unterordner `mystyles` an, und kopieren alle Stildateien dort hinein[20]. Erwähnt habe ich z.B. die Stile `nature` oder `gerapalike`, die wir jetzt in unseren neuen Ordner kopieren.

Damit LaTeX die neuen Dateien auch findet, müssen die Pfade zu den Dateien neu eingetragen werden. Das ist leider etwas umständlich, da es nur über einen Konsolenbefehl funktioniert! Wir öffnen also ein Terminal-Fenster über *Programme* → *Dienstprogramme* → *Terminal* und geben folgende Zeile ein:

```
/Library/TeX/Root/bin/i386-darwin/sudo mktexlsr
```

Es kommt eine Passwortabfrage, an der wir unser Passwort eintragen müssen. Die Voraussetzung dafür ist natürlich, dass wir auch administrative Rechte im OSX besitzen! Nachdem `mktexlsr` die Dateien upgedated hat, können wir das Terminal-Fenster wieder schließen, und die neuen Pakete stehen jetzt in LaTeX zur Verfügung.

23.3 LaTeX unter Linux pflegen

Hat man TexLive über den Paketmanager installiert, befindet sich die LaTeX-Installation aufgeteilt auf verschiedene Verzeichnisse. Die wichtigsten davon lauten `/usr/share/texmf-texlive/bibtex/bst` bzw. `/usr/share/texmf-texlive/tex/latex` wo sich jeweils die Literaturstile bzw. die eingebundenen Pakete befinden.

[19] LaTeX-Pakete tragen die Endung .STY
[20] Literaturstil-Dateien tragen die Endung .BST

Will man also einen neuen Literaturstil in unsere LaTeX-System einbinden, muß man erst ein Verzeichnis im Ordner `/usr/share/texmf-texlive/bibtex/bst` erstellen, und dann die entsprechenden .BST-Datei(en) dort ablegen. Will man hingegen ein neues `\usepackage`-Paket einbinden, erstellt man ein Verzeichnis in `/usr/share/texmf-texlive/tex/latex` und kopiert die entsprechenden .STY-Dateien dorthin[21].

Hat man alle neuen Dateien an ihrem Platz, muß man noch den Befehl

```
sudo mktexlsr
```

ausführen, der die neuen Dateien in das LaTeX-System einbindet.

Sollten die Pfade für die .BST- und .STY-Dateien doch anders lauten, kann man beide sehr einfach herausfinden. Um den Pfad zu den Literaturstil-Dateien zu finden, reicht die Eingabe von `locate bst | grep /bst` in einem Terminal. Es werden dann alle Pfade, in denen „`/bst`“ vorkommt angezeigt. Ähnliches gilt für die Paketdateien, wobei hier der Terminalbefehl `locate latex | grep .sty` zum Erfolg führt.

Wurde die Installation von der TexLive-CD gestartet, lauten die Pfade zu den Literaturstilen und Paketen hingegen `/usr/local/texlive/2011/texmf-dist/bibtex/bst` (für die Literaturstile) bzw. `/usr/local/texlive/2011/texmf-dist/tex/latex` (für alle weiteren Pakete)! Der Befehl für die Aktualisierung der neuen Dateien lautet jetzt auch `sudo /usr/local/texlive/2011/bin/i386-linux/mktexlsr`, da der CD-Installer (im Gegensatz zu *Synaptic* oder *apt-get*) keine Anpassung des Pfades für ausführbare Dateien vornimmt. Siehe hierzu auch Kapitel 3.3.

Literatur

American Mathematical Society (2002). User's Guide for the amsmath Package (Version 2.0).

Görlach, K. (2004). Wissenschaftlich publizieren in LaTeX - Indexerstellung.

Mösgen, P. (1998). Makeindex - Sachregister erstellen mit LaTeX. *Katholische Universität Eichstätt, Schriftenreihe des Universitätsrechenzentrums*, 14.

Oetiker, T., Partl, H., Hyna, I., and Schlegl, E. (2008). The Not So Short Introduction to LaTeX.

Pakin, S. (2008). The Comprehensive LaTeX Symbol List.

Pepping, S. (2006). *Instructions for use of the document class elsart.*

Peters, J. (2005). Eine Einführung - Mathematik mit LaTeX.

[21] Nicht vergessen, unter Linux müssen diese Aufgaben mit root-Rechten ausgeführt werden! Ich setze voraus, dass jeder soweit mit seiner Linux-Distribution vertraut ist.