2 Grundlagen

Hier geht es darum, ein Gefühl für die grundsätzliche Funktionsweise von TEX bzw. von LATEX zu bekommen, das ein in TEX geschriebenes Makro–Paket ist. Wir werden im Kurs „nur“ LATEX behandeln. Nach einer grundsätzlichen Beschreibung, des Arbeitens mit LATEX betrachten wir, was beim Eingeben von Text zu beachten ist.

2.1 Die Funktionsweise von LATEX

Wichtig ist sich zunächst klar zu machen, dass LATEX kein Textverarbeitungsprogramm wie etwa M$–Word sondern ein Satzprogramm ist. Damit muss man sich als erstes von der Idee des „What you see is what you get“ (WYSIWYG) lösen, was sich allerdings schnell als ein großer Vorteil erweist: Man erlangt die Freiheit, sich beim Schreiben auf den Inhalt und die logische Struktur des Textes zu konzentrieren, statt dauernd von seiner äußeren Erscheinung abgelenkt zu werden. Dafür, dass das Ganze am Ende gut aussieht (wesentlich besser als mit vielen anderen Programmen), sorgt LATEX schon, oder man kann sich darum kümmern, wenn der eigentliche Text einmal steht.
Während dies die hinter TEX und LATEX stehende Philosophie angeht, interessiert uns hier zunächst mehr die praktische Seite: Das Dokument, das wir bearbeiten ist eine ASCII–Datei, die die Endung `.tex` trägt, sagen wir `beispiel.tex`, und dem Quellcode eines Programms vergleichbar ist. Diese Datei kann mit einem beliebigen Editor bearbeitet werden. Im folgenden werden wir uns hauptsächlich damit befassen, was in dieser Quelldatei stehen kann.
Die Datei `beispiel.tex` zusammen mit anderen, die etwa Informationen zur grundsätzlichen Struktur und dem Layout der vorliegenden Art von Text (z.B. Buch oder Artikel) enthalten, und den LATEX–Makros bilden den Input für TEX, das daraus eine Datei `beispiel.dvi`[1] erstellt, die sämtliche Informationen darüber enthält, wie der Text auf dem Papier oder dem Bildschirm erscheinen soll. Je nach Bedarf wird daraus mittels eines passenden Treibers ein Ausdruck, eine Darstellung auf dem Bildschirm oder etwa eine Postscript–Datei erzeugt. TEX stellt also den Kompilierer dar, der aus unserem Quellcode einen gesetzten Text erstellt.
Gleichzeitig werden beim Durchlauf durch TEX je nach Anwendung noch andere Dateien erzeugt. Unter diesen ist auf jeden Fall eine namens `beispiel.log`, ein Logbuch des TEX–Laufs. Mehr zu den unterschiedlichen Dateien, die für TEX eine Rolle spielen findet man in Abschnitt 1.2 des LATEX Begleiters (Goossens, Mittelbach, und Samarin, 2002).
Während in der `.log`–Datei unter anderem diejenigen während des Kompilierens aufgetretenen Probleme aufgelistet sind, die nicht so schwerwiegend sind, dass der TEX–Lauf abgebrochen werden muss — dazu gehören etwa Zeilen, die ein wenig über den Textrand hinausragen, die berüchtigten „overfull hboxes“ — passiert es gerade am Anfang häufig, dass Fehler dazu führen, dass TEX nicht mehr weiter weiß. In solchen Fällen bricht der TEX–Lauf mit einer Fehlermeldung ab. Diese Fehlermeldungen sind zwar aussagekräftiger als andere in der Softwarebranche übliche, die Fehlersuche gehört aber dennoch zu den mühseligsten Schritten. Da ein

[1] Die Dateieindung `.dvi` steht für **d**e**v**ice **i**ndependent

Fehler viele andere auslösen kann, würden wir generell dazu raten, zunächst den ersten Fehler zu beheben und dann einen neuen TEX–Lauf anzustoßen, statt die Option zu wählen, den Fehler zu ignorieren und weiter zu übersetzen, um dann alle Fehler auf einmal zu verbessern. Die `.dvi`–Datei kann man mit einem passenden Previewer (zu MiKTEX gehört zum Beispiel YAP — **Y**et **a**nother **p**reviewer) am Bildschirm anschauen oder aber ausdrucken. Letzteres setzt voraus, dass ein passender Druckertreiber vorhanden ist, was aber in der Regel kein Problem darstellt.

2.1.1 Postscript– und PDF–Output

Schon zum Ausdrucken kann es hilfreich sein, statt der einer `.dvi`–Datei eine Postscript–Version oder eine PDF–Version zu erzeugen. Dies gilt um so mehr, wenn man seine Werke anderen — insbesondere auch Personen die (noch) keine TEX–User sind — zukommen lassen möchte; zum Beispiel per Email oder indem man sie ins Netz stellt.
Um aus der Datei `beispiel.dvi` eine Postscript–Datei zu erzeugen, kann man das Programm `dvips`[2] verwenden, das Thomas Rokicki geschrieben hat. Das Ergebnis, die Datei `beispiel.ps` kann man dann mit dem Postscript–Previewer GSView ansehen.
Um eine PDF–Version zu erhalten gibt es drei Möglichkeiten. Wir beschreiben sie hier alle drei kurz, werden aber im folgenden, so lange nicht explizit etwas anderes gesagt wird stets davon ausgehen, dass Ihr den dritten Weg beschreitet.

1. Wenn man die LATEX–Quelldatei `beispiel.tex` statt mit LATEX mit pdfLATEX übersetzt, erhält man direkt eine PDF–Datei `beispiel.pdf`. Bei diesem Vorgehen wird keine `.dvi`–Datei `beipiel.dvi` erzeugt.

2. Man kann eine PDF–Datei erzeugen, indem man die `.dvi`–Datei `beipiel.dvi` mit dem Programm `dvipdfm`[3] von Mark A. Wicks bearbeitet.

3. Schließlich kann man eine PDF–Datei auch aus der Postscript–Datei erzeugen. Dazu kann man falls vorhanden natürlich Adobes Acrobat Distiller verwenden. Es geht aber auch mit Ghostscript. Dazu öffnet man die Datei `beispiel.ps` mit GSView, wählt dann das „File“-Menue, dort „Convert“ und darin den `pdfwriter`. Sinnvollerweise nennt man die Ausgabedatei wohl `beispiel.pdf`.

2.2 Wesentliche Elemente eines LATEX–Dokuments

Jedes LATEX–Dokument besteht aus zwei Teilen, dem Deklarationsteil und dem Textteil. Die meiste Zeit werden wir uns mit dem befassen, was im Textteil möglich ist. Hier werden wir zunächst zum Deklarationsteil nur die wesentlichen Elemente kurz ansprechen. Später lernen wir weitere Elemente kennen, die den Deklarationsteil betreffen. Insbesondere sollte das Finetuning des Layouts im Wesentlichen durch den Deklarationsteil erreicht werden.

[2] Dokumentation unter http://tug.org/texinfohtml/dvips.html.
[3] Siehe die Dokumentation auf `CTAN` http://www.ctan.org/tex-archive/dviware/dvipdfm/dvipdfm.pdf sowie die Homepage http://gaspra.kettering.edu/dvipdfm/. Zusammen mit Jin-Hwan Cho und Shunsaku Hirata hat Mark A. Wicks auch eine Erweiterung `dvipdfmf` geschrieben, siehe http://www.ctan.org/tex-archive/help/Catalogue/entries/dvipdfmx.html.

2.2.1 Deklarationsteil

Dieser Teil entspricht dem `<HEAD>` eines HTML–Dokuments und enthält wesentliche generelle Deklarationen für das gesamte LaTeX–Dokument.
Dazu gehören insbesondere

- Die **Dokumentenklasse**: Mit dem Befehl `\documentclass` und seinen Argumenten die Festlegung der **Klasse** von Dokument, das erstellt werden soll, etwa ein Buch, ein Artikel oder ein Brief.
- Die benutzten **Pakete**, in denen zusätzliche Funktionen bereitgestellt werden.
- Stilangaben für dieses Dokument, etwa zu Seitennummerierung, Kopf- und Fußzeilen etc.
- Eigene (Um-)Definitionen von Befehlen und Umgebungen.
- Metaangaben zum Dokument, wie Autor, Titel etc.

2.2.2 Textteil

Der Hauptteil des Dokuments, in dem der eigentliche Textkörper steht, also das was dem `<BODY>` eines HTML–Dokuments entspricht, beginnt mit `\begin{document}` und endet mit `\end{document}`. Was danach im Quellcode folgt wird von TeX ignoriert.
Durch ein Prozentzeichen `%` wird Kommentar gekennzeichnet, d. h., alles, was in einer Zeile hinter dem `%` steht wird von TeX ignoriert. Es empfiehlt sich, mit Kommentaren nicht zu sparsam umzugehen. So kann es hilfreich sein, im Deklarationsteil kurz zu kommentieren, zu welchem Zweck einzelne Pakete geladen werden, was man sich bei möglichen Optionen zu diesen Paketen gedacht hat etc. Auch im Text kann man sich selbst und möglichen Koautoren durch Kommentare gelegentlich das Leben erleichtern; spätestens dann, wenn man sein eigenes Dokument nach längerer Zeit noch einmal bearbeitet. Dies gilt verstärkt für kompliziertere Strukturen, wie mathematische Formeln (siehe Kapitel 4), Tabellen (siehe Kapitel 7) oder Grafiken (siehe Kapitel 8).
Im Textteil ist es wesentlich zu unterscheiden, ob man sich gerade im Text–Modus oder aber im Mathematik–Modus befindet. Standard ist der Text–Modus. Das Umschalten in den Mathematik–Modus erfolgt durch Dollarzeichen für Formeln im Text, z. B. `$e = m c^2$`, was $e = mc^2$ ergibt, oder durch einschließen in `\[` und `\]` für abgesetzte Formeln, etwa

$$e = mc^2.$$

Mehr zu mathematischen Formeln in Kapitel 4.

2.2.3 Befehle

Befehle in LaTeX beginnen stets mit einem Backslash \. Sie können Argumente haben, bei denen man optionale Argumente, die in eckigen Klammern geschrieben werden, und in geschweifte Klammern gesetzte notwendige Argumente unterscheidet. Ein Befehl ohne Argumente ist etwa `\LaTeX`, durch den das LaTeX–Symbol erzeugt wird. Zu beachten ist dabei, dass LaTeX merkt, dass ein Befehl beendet ist, wenn ein Leerzeichen oder ein Sonderzeichen folgt. Daraus

ergibt sich das Problem, dass tatsächlich gewollte Leerzeichen nach einem Befehl „aufgefressen“ werden: Die Eingabe `\LaTeX begeistert` ergibt LaTeXbegeistert, was weniger begeisternd ist. Der Ausweg besteht darin, `\LaTeX{} begeistert` einzugeben, was wie gewünscht LaTeX begeistert ergibt. Ein Beispiel für Befehle mit einem Argument sind Umstellungen der Schriftart, etwa `\textit{`*Text*`}`. Durch diesen Befehl wird der Text im Argument *kursiv* (englisch *italics*) gesetzt.
Befehle können auch selbst definiert werden (vgl. Abschnitt 2.2.3). Sinnvollerweise tut man dies im Deklarationsteil. Es empfiehlt sich, dabei einigermaßen planvoll vorzugehen, das heißt insbesondere, Befehlsdefinitionen sinnvoll zu gruppieren und sie gegebenenfalls zu kommentieren.

2.2.4 Umgebungen

Umgebungen werden begonnen mit dem Befehl `\begin{`*umgebungsname*`}` und beendet mit `\end{`*umgebungsname*`}`. Im Prinzip stellt der gesamte Textteil eine Umgebung dar. Weitere werden wir in Abschnitt 2.3.4 noch kennenlernen. Umgebungen können ineinander verschachtelt werden.
Der Inhalt einer Umgebung bildet eine **Gruppe**. Ebenso Argumente eines Befehls. Gruppen können auch durch das Einschließen in geschweifte Klammern beliebig gebildet werden. Es ist wichtig die Idee der Gruppierung zu verstehen, da es Befehle gibt, die nur innerhalb einer Gruppe wirken. So stellt der Befehl `\large` etwa auf eine größere Schriftgröße um, und zwar von der Stelle an, an dem er auftaucht, bis zum Ende der jeweiligen Gruppe. Möchte man also ein Wort größer schreiben, so erreicht man dies durch die Eingabe `{\large Wort}`. Dies ergibt den erwünschten Effekt, dass nur das eine {\large Wort} größer erscheint. Ohne die geschweiften Klammern würde der gesamte folgende Text groß geschrieben, bis eine Gruppe endet.
Auch eigene Umgebungen können definiert werden (vgl. Abschnitt 2.3.4.4), wobei sich wiederum ein planvolles Vorgehen empfiehlt.

2.3 Texteingabe

2.3.1 Hilfreiche Pakete zur Sprachanpassung

TeX bzw. LaTeX sind ursprünglich in den USA entwickelt worden und daher auf die englische Sprache und amerikanische Vorstellungen über Schriftsatz ausgerichtet. Inzwischen gibt es aber komfortable Möglichkeiten, LaTeX auch an diverse andere Sprachen anzupassen. Hier beschreiben wir drei Pakete, die dazu verwendet werden können (die ersten beiden sind eigentlich sogar unverzichtbar).

2.3.1.1 Das `inputenc`–Paket

Das Paket `inputenc` von Alan Jeffrey und Frank Mittelbach, das es ermöglicht, LaTeX mitzuteilen, welche Kodierung des Inputs im vorliegenden Dokument verwendet wird (es ist dokumentiert in Jeffrey und Mittelbach (2004). Während der ASCII–Kode für normale Groß- und Kleinbuchstaben, Ziffern und Satzzeichen, der im Bereich von 32 bis 127 festgelegt ist, für englischsprachige Texte ausreicht, gibt es etwa für die Umlaute und das „ß“ im Deutschen je nach Betriebssystem unterschiedliche Kodierungen. Gibt man zum Beispiel ein „Ö“ ein, wird es unter Windows ab Version 3.1 intern mit 214 kodiert, während es zum Beispiel auf einem

Mac als 133 kodiert wird (214 ist auf dem Mac die Kodierung für ÷ im Mathematik–Modus bzw. in der Apple Central European Codepage für ÷ im Textmodus; 133 unter Windows steht für ...). Ohne die Verwendung des `inputenc`–Pakets kann LaTeX daher Eingaben, die nicht im Bereich zwischen 32 und 127 kodiert werden nicht interpretieren und ignoriert sie daher. Das heißt, dass somit sämtliche Umlaute fehlen würden.
Somit sollte man für deutsche Texte das Paket `inputenc` stets laden und dabei die richtige Kodierung angeben. Für Windows–Nutzer heißt das, dass der Deklarationsteil die Zeile `\usepackage[ansinew]{inputenc}` enthalten sollte, während europäische Mac-Benutzer `\usepackage[macce]{inputenc}` und andere `\usepackage[applemac]{inputenc}` verwenden.

2.3.1.2 Das `babel`–Paket

Das Paket `babel` von Johannes Braams implementiert die Unterstützung vieler verschiedener Sprachen, von denen auch mehrere innerhalb eines Dokuments verwendet werden können. Sein Gebrauch wird in Braams (2005b) beschrieben.[4] Für die deutsche Sprache existieren zwei Optionen, nämlich `german` für die alte sowie `ngerman` für die neue deutsche Rechtschreibung. Gibt man im Deklarationsteil je nach Präferenz `\usepackage[german]{babel}` beziehungsweise `\usepackage[ngerman]{babel}` ein, so verwendet LaTeX die korrekten Trennmuster für die deutsche Sprache und stellt eine Reihe weiterer Möglichkeiten bereit, auf die wir im im folgenden an verschiedenen Stellen eingehen werden.
Im Prinzip bewirkt das Paket `babel` mit der Option `german` das selbe wie das Paket `german` (siehe Raichle, 1998), das von Hubert Partl entwickelt und aktuell von Bernd Raichle gepflegt wird. Entsprechendes gilt für die `babel`–Option `ngerman` und das Paket `ngerman`.
Wir werden im folgenden lediglich von der `german`–Option sprechen, wenn wir auf Funktionen eingehen, die nur bei Verwendung von `babel` mit einer der Optionen `german` oder `ngerman` oder aber einem der Pakete `german` oder `ngerman` zur Verfügung stehen.

2.3.1.3 Das KOMA–Script–Bündel

KOMA–Script wurde in der Hauptsache von Frank Neukam, Markus Kohm und Axel Kielhorn entwickelt unter anderem mit dem Ziel einer besseren Anpassung von LaTeX an europäische Gepflogenheiten. Es enthält eigene Dokumentklassen, die jeweils den LaTeX–Standardklassen entsprechen aber andere Voreinstellungen und eine größere Flexibilität aufweisen: `scrartcl` korrespondiert zur Standardklasse `article`, `scrbook` zu `book` und `scrreprt` zu `report`; auch zur Standarddokumentklasse `letter`, die für Briefe gedacht ist, gibt es mit `scrlttr2` ein Pendant in KOMA–Script. Eine sehr lesenswerte ausführliche Anleitung bietet Kohm und Morawski (2004).
Wir werden auf KOMA–Script noch zurückkommen, insbesondere auch im Kapitel 10.

2.3.2 Diverse Kleinigkeiten

Zunächst sprechen wir ein paar Dinge an, die insbesondere in deutschen Texten auftreten.

[4] Eine ausführlichere Beschreibung auch technischer Details findet man in Braams (2005a).

2.3.2.1 Umlaute

Dank des `inputenc`–Pakets können wir Umlaute normal eintippen: ä, Ä, ö, Ö, ü, Ü, ß.
Es geht aber auch anders: Mit der `german`–Option kann man auch `"a`, `"A`, `"o`, `"O`, `"u`, `"U` und `"s` verwenden. Generell funktioniert auch `\"a` oder `\"{a}`, für das „ß“ auch `{\ss}`.[5]
Man muss dabei allerdings auf die entstehenden Probleme mit den Zwischenräumen achten, wie das folgende Beispiel zeigt. Die Eingabe

```
Er ging die Stra\ss e entlang zu Fu\ss  nach Hause.

Er ging die Stra\ss e entlang zu Fu\ss\  nach Hause.

Er ging die Stra{\ss}e entlang zu Fu{\ss}  nach Hause.

Er ging die Straße entlang zu Fuß nach Hause.
```

ergibt folgende Ausgabe:
Er ging die Straße entlang zu Fußnach Hause.
Er ging die Straße entlang zu Fuß nach Hause.
Er ging die Straße entlang zu Fuß nach Hause.
Er ging die Straße entlang zu Fuß nach Hause.
Bei den Buchstaben „i“ und „j“ ergaben Akzente wegen des Punkts ein etwas seltsames Bild, daher fand man die Empfehlung `\'{\i}` zu schreiben, um í zu erhalten. Dies ist allerdings inzwischen in diesem Falle unnötig, wohl aber beim „j“: `\'{j}` sieht blöd aus, nämlich so j́, `\'{\j}` besser, nämlich so ǰ.
Nach ähnlichem Muster erreichen wir auch Buchstaben mit diversen Akzenten: `\'{a}`→ á, `\‘{e}`→ è, `\^{a}`→ â `\~{a}`→ ã `\c{c}`→ ç `\v{s}`→ š etc.
Speziell für Skandinavien Fans interessant sind: `\ae`→ æ, `\AE`→ Æ, `\aa`→ å, `\AA`→ Å, `\oe`→ œ und `\OE`→ Œ.

2.3.2.2 Ligaturen

Besondere Zeichen sind sogenannte Ligaturen, die LaTeX in einigen Kombinationen mit dem Buchstaben f erzeugt. Will man sie abschalten, muss man die entsprechenden Buchstaben durch `\/` trennen, mit der `german`–Option ist es empfehlenswert, stattdessen `"|` zu verwenden.
ff ff fff fl fl fi fi ffi ffi ffi ffl ffl ffl.
Zum Beispiel „Schaffen“ (`Schaffen`), „Schaffell“ (`Schaf"|fell`), „Hafflinger“ (`Hafflinger`), „Sauerstoffflasche“ (`Sauerstoffflasche`), „Stoffliege“ (`Stoff"|liege`) und „Schlafflasche“ (`Schlaf"|flasche`).

2.3.2.3 Anführungszeichen

Wir haben hier schon erste Anführungszeichen oder „Gänsefüßchen“ gesehen. Im englischen gibt es “Quotes” (`‘‘Quotes’’`) und deren einfache Form ‘Single quotes’ (`‘Single quotes’`). Die entsprechende deutsche Form kann durch ,,Gänsefüßchen” (`,,Gänsefüßchen’’`), besser aber mit der `german`–Option durch „Gänsefüßchen“ (`"‘Gänsefüßchen"’`) erreicht werden. Ähnlich erreicht man auch ‚einfache’ (`,einfache’`).

[5] Vergleiche dazu auch Trettin (2004, Abschnitt 2.2.5, S. 9 f.).

Ebenfalls verfügbar sind die Befehle `\glqq`, was „ ergibt, `\glq`, was , ergibt, `\grqq`, was “ ergibt, und `\grq`, was ‘ ergibt. Entsprechend erreicht man auch die französischen Varianten: `\flqq` ergibt «, `\flq` ergibt ‹, `\frqq` ergibt », und `\frq` ergibt ›.

2.3.2.4 Satzzeichen

Satzzeichen erhält man ganz normal, indem man sie über die Tastatur eingibt. Nach einem Punkt, einem Ausrufungszeichen, einem Fragezeichen oder einem Doppelpunkt fügt LaTeX zusätzlichen Zwischenraum ein, allerdings wird dies durch die `german`–Option ausgeschaltet. Für näheres zu diesem Thema siehe etwa `l2kurz.pdf` (Abschnitt 3.3 S. 14).

2.3.2.5 Striche und Punkte

Es gibt in LaTeX drei Strichlängen: den Trennungsstrich -, den wir durch die Eingabe `-` erzeugen, den Bindestrich – (der Duden fordert dafür allerdings auch den Trennungsstrich -), den wir durch die Eingabe `--` erzeugen und den Gedankenstrich —, den wir durch die Eingabe `---` erzeugen.
Drei Punkte sollte man nicht durch eingeben von drei einzelnen Punkten erzeugen, was die Ausgabe „...“ ergibt, sondern besser durch die Befehle `\dots` ... oder `\ldots` ... darstellen.

2.3.2.6 Hervorhebungen, Änderungen der Schriftgröße und farbiger Text

Bestimmte Textteile möchte man eventuell hervorheben, indem sie in einem anderen Font gesetzt werden. Dazu gibt es einigen Befehle.

`\textbf{Text}`	setzt den Text in **fetter Schrift**
`\textit{Text}`	setzt den Text in *kursiver Schrift*
`\textsf{Text}`	setzt den Text in serifenloser Schrift
`\textsc{Text}`	setzt den Text in Kapitälchen
`\texttt{Text}`	setzt den Text in `Schreibmaschinenschrift`
`\emph{Text}`	setzt den Text in *hervorgehobener Schrift*
`\emph{Text}`	*setzt den Text in* hervorgehobener Schrift

Wie der Text im Argument von `\emph{Text}` aussieht hängt also davon ab, wo der Befehl gebraucht wird. Dies ergibt Sinn, wenn man bedenkt, dass Hervorheben durch kursive Schrift nicht wirkt, wenn alles andere ebenfalls kursiv gesetzt ist.
Auch die Schriftgröße kann geändert werden.

`{\tiny Text}`	erzeugt
`{\scriptsize Text}`	erzeugt
`{\footnotesize Text}`	erzeugt
`{\small Text}`	erzeugt
`{\normalsize Text}`	erzeugt
`{\large Text}`	erzeugt
`{\Large Text}`	erzeugt
`{\LARGE Text}`	erzeugt
`{\huge Text}`	erzeugt
`{\Huge Text}`	erzeugt

Auch Farbe kann zur Hervorhebung bestimmter Teile eines Textes geeignet sein. Das Paket `color` von David P. Carlisle, stellt eine Unterstützung zur Verwendung von Farbe in LaTeX

bereit. Es gehört zum `graphics`–Bündel. daher findet sich eine Anleitung zu seiner Benutzung in der entsprechenden Dokumentation Carlisle (1999). Uwe Kern hat die Funktionalitäten mit seinem `xcolor`–Paket nochmals erweitert (Kern, 2005).
Beide Pakete enthalten unter anderem den Befehl `\textcolor{`*Farbe*`}{`*Text*`}`, der den Text *Text* in der Farbe *Farbe* ausgibt. So kann Text auch farbig dargestellt werden, etwa gelb (yellow), rot (red), blau (blue) oder grün (green).
Generell sollte man von derlei Möglichkeiten eher sparsam Gebrauch machen (eine genauere Betrachtung dieser Seite mag Zweifel daran aufkommen lassen, ob wir uns an unsere eigenen Empfehlungen halten; vielleicht kann sie aber auch einen Eindruck vermitteln, was ein übertriebener Einsatz der vorhandenen Gestaltungsmöglichkeiten anrichtet). Zudem handelt es sich hier um eine klare Verletzung der TEX–Philosophie des „generischen Markups“.
Der folgende Absatz ist ein Beispiel für schlechten LATEX–Code:

```
\textsc{John Maynard Keynes} ist ein wichtiger Autor der
\textbf{Makroökonomik}. Sein Hauptwerk
"‘\textit{The General Theory of Employment, Interest, and Money}"’
ist 1936 erschienen.
```

Der selbe Absatz sollte eher so aussehen:

```
\eigenname{John Maynard Keynes} ist ein wichtiger Autor der
\begriff{Makroökonomik}. Sein Hauptwerk
\buchtitel{The General Theory of Employment, Interest, and
Money} ist 1936 erschienen.
```

Wie Eigennamen, Begriffe und Buchtitel dargestellt werden sollen, wird dann in der Präambel festgelegt, in der die drei entsprechenden Befehle definiert werden. In unserem Fall haben wir sie so definiert, dass beide Eingaben das selbe Ergebnis liefern, nämlich:
John Maynard Keynes ist ein wichtiger Autor der **Makroökonomik**. Sein Hauptwerk "‘*The General Theory of Employment, Interest, and Money*"’ ist 1936 erschienen.
Das zweite Vorgehen hat zwei Vorteile:

1. Es hilft dabei (und zwingt dazu), sich auf die logische Struktur des Textes zu konzentrieren.

2. Es macht Änderungen leicht und hilft dabei, später beispielsweise ein Namensregister zu erstellen.

2.3.3 Definition eigener Befehle

Wie definiert man nun einen eigenen Befehl. Dies geschieht mittels des Befehls `\newcommand`. Wenn wir etwa den oben benutzen Befehl `\eigenname` definieren wollen, der dafür sorgt, dass der Name im Argument in Kapitälchen erscheint, so definieren wir
`\newcommand{\eigenname}[1]{\textsc{#1}}`.
Das erste Argument von `\newcommand` gibt den Namen des zu definierenden Befehls an. Es folgt ein optionales Argument, das festlegt, wie viele Argumente der Befehl bekommt, den wir

definieren wollen. Schließlich folgt die Angabe dessen, was der Befehl tun soll, dabei wird das n-te Argument durch `#n` angesprochen.
Falls wir versuchen, einen Namen für unseren Befehl zu vergeben, der dem eines bereits definierten Befehls entspricht, gibt LaTeX eine entsprechende Fehlermeldung aus. In diesem Falle können wir mit dem Befehl `\renewcommand` arbeiten, der es erlaubt, vorhandene Befehle neu zu definieren; davon ist allerdings generell dringend abzuraten.
Siehe zu detaillierten Ausführungen zum Thema Befehlsdefinitionen auch den Anhang A.1.1 in Goossens, Mittelbach, und Samarin (2002).
Will man sicherstellen, dass ein Befehl nicht undefiniert bleibt, aber gleichzeitig verhindern, eine bereits vorhandene Definition zu überschreiben — z. B. in Dateien, die in verschiedenen Dokumenten eingesetzt werden sollen — existiert dafür der Befehl `\providecommand`. Seine Funktion entspricht der von `\newcommand`, allerdings wird er ignoriert, falls der Name des Befehls, der definiert werden soll, bereits vergeben ist.
Praktisch für die Definition von Befehlen ohne Argument ist das Paket `xspace` von David P. Carlisle, das einen Befehl `\xspace` bereitstellt. Was dieser Befehl tut, kann man an einem einfachen Beispiel illustrieren. Angenommen, wir wollen den Befehl `\UHOH` definieren, der „Universität Hohenheim“ (ohne die Anführungszeichen) erzeugen soll. Wenn wir definieren

```
\newcommand{\UHOH}{Universität Hohenheim}
```

besteht das selbe Problem, das wir schon beim `\LaTeX`–Befehl gesehen hatten: Die Eingabe

```
die \UHOH ist eine tolle Uni.
```

würde als Ausgabe folgendes erzeugen:

> die Universität Hohenheimist eine tolle Uni.

Grund: Das Leerzeichen nach dem Befehl wird als Ende des Befehls interpretiert und kann nicht gleichzeitig auch noch ein Leerzeichen zwischen „Hohenheim“ und „ist“ erzeugen. Kämen wir auf die Idee, unsere Befehlsdefinition zu ändern, indem wir nach „Hohenheim“ noch ein Leerzeichen einfügen, also

```
\newcommand{\UHOH}{Universität Hohenheim },
```

so würde dies zwar in obigem Beispielsatz gut funktionieren, aber zu unerwünschten Effekten führen, wenn auf den Befehl ein Satzzeichen folgt. Zum Beispiel würde die Eingabe

```
Ich studiere an der \UHOH.
```

folgende Ausgabe erzeugen

> Ich studiere an der Universität Hohenheim .

Das Problem ist das Leerzeichen vor dem Punkt. Laden wir stattdessen das Paket `xspace` und definieren

```
\newcommand{\UHOH}{Universität Hohenheim\xspace},
```

so wird ein Leerzeichen eingefügt, wenn auf den Befehl ein Leerzeichen und dann ein anderes Wort folgt, nicht aber, wenn das folgende Zeichen ein Satzzeichen ist.[6] Falls es zu sehr nervt, immer `\LaTeX{}` eingeben zu müssen, kann man natürlich auch einen Befehl `\LTeX` folgendermaßen definieren `\newcommand{\LTeX}{\LaTeX\xspace}`. Für eine Beschreibung des Pakets siehe Carlisle (1997).

2.3.4 Einige Umgebungen

2.3.4.1 Zitate

Für längere Zitate gibt es die Umgebung `\begin{quote}`, `\end{quote}`, die den Inhalt eingerückt in der jeweils aktiven Schrift darstellt.
Zum Beispiel wird jedem, der in den Genuss gekommen ist, Lateinunterricht zu erhalten, der Beginn von Caesars Gallischem Krieg noch im Ohr sein.

> Gallia est omnis divisa in partes tres, quarum unam incolunt Belgae, aliam Aquitani, tertiam, qui ipsorum lingua Celtae, nostra Galli appellantur.

2.3.4.2 Aufzählungen

Die `itemize`–Umgebung bietet die Möglichkeit verschiedene Punkte mit Spiegelstrichen aufzulisten.
Etwa die entscheidenden Vorzüge von LaTeX als

```
\begin{itemize}
  \item stabil und fast beliebig anpassbar
  \item schöner Output
  \item Betonung der logischen Struktur statt WYSIWIG
  \item unabhängig vom Betriebssystem
  \item kostenlos
\end{itemize}
```

Was folgenden Output liefert.

- stabil und fast beliebig anpassbar
- schöner Output
- Betonung der logischen Struktur statt WYSIWIG
- unabhängig vom Betriebssystem
- kostenlos

[6] Nicht ganz den erwünschten Effekt hat `\xspace` wenn man direkt an den Befehl, in dessen Definition `\xspace` verwendet wird, eine Fußnote anbringen möchte. Gibt man z. B. `\UHOH\footnote{`*Fußnotentext*`}` ein, erscheint das Fußnotenzeichen im Text mit einem Abstand von UHOH getrennt. Um dies zu verhindern müssen wir in diesem Fall `\UHOH{}\footnote{`*Fußnotentext*`}` eingeben.

Ähnlich ist die `enumerate`–Umgebung, nur dass sie die einzelnen Punkte als nummerierte Liste ausgibt.

1. stabil und fast beliebig anpassbar
2. schöner Output
3. Betonung der logischen Struktur statt WYSIWIG
4. unabhängig vom Betriebssystem
5. kostenlos

Dritte im Bunde ist die `description`–Umgebung, die sich zum Beispiel für eine Liste wichtiger Begriffe eignet.

```
\begin{description}
  \item[Begriff] Erklärung des Begriffs
\end{description}
```

ergibt

Begriff Erklärung des Begriffs

Auch in den anderen beiden Umgebungen kann man `\item` mit einem optionalen Argument in eckigen Klammern verwenden, das dann anstelle des Spiegelstrichs bzw. der laufenden Nummer verwendet wird (in diesem Fall wird der entsprechende Punkt nicht mitgezählt, d.h., der folgende Punkt erhält die Nummer, die auf die des letzten Items ohne optionales Argument folgt.
Alle drei Arten der Aufzählung können bis zu einer Tiefe von vier Ebenen ineinander verschachtelt werden. Sie sollten allerdings in normalen Texten eher sparsam verwendet werden.

2.3.4.3 verbatim

Gelegentlich möchte man exakt das als Ausgabe sehen, was man eingibt. Dazu gibt es zwei Möglichkeiten. Für längere derartige Textstücke gibt es die `verbatim`–Umgebung, die den entsprechenden Abschnitt abgesetzt und in einer „Schreibmaschinenschrift“ ausgibt. Kürzere Textpassagen lassen sich mit Hilfe des Befehls `\verb` darstellen. Dieser Befehl interpretiert das unmittelbar auf den Befehlsnamen folgende Zeichen als öffnende Klammer, das nächste Auftauchen dieses Zeichens als schließende Klammer und gibt Alles dazwischen wörtlich und in „Schreibmaschinenschrift“ wieder. Dabei werden keine Zeilenumbrüche gemacht. Die Klammerfunktion für den `\verb`–Befehl können alle Sonderzeichen außer `*` übernehmen (um Konfusion für den menschlichen Leser des LaTeX–Quellcodes zu vermeiden sollte man aber zum Beispiel davon absehen, an dieser Stelle die geschweifte Klammern `{` oder `}` oder mit der `german`–Option die Anführungszeichen `"` zu verwenden).
Eine gesternte Version gibt Leerzeichen zur Verdeutlichung als ␣ aus, d. h. z. B. die Eingabe `\verb*"text text"` erzeugt `text␣text`.

2.3.4.4 Definition eigener Umgebungen

Wie bei befehlen gibt es auch für Umgebungen die Möglichkeit, selbst eigene Umgebungen zu definieren. Siehe dazu Anhang A.1.2 in Goossens, Mittelbach, und Samarin (2002).
Häufig wird man dabei auf vordefinierte Umgebungen zurückgreifen, die auf die eigenen Bedürfnisse angepasst werden. Ein Beispiel dafür ist die Definition theoremartiger Umgebungen mit Hilfe des Befehls `\newtheorem`, auf die wir im Abschnitt 3.3 eingehen werden.

2.4 Zeilenumbruch, Silbentrennung

LaTeX bemüht sich, den Text eines Absatzes optimal zu formatieren. Dabei gilt es beim voreingestellten Blocksatz abzuwägen zwischen einer gleichmäßigen Verteilung der Wörter, so dass die Zwischenräume weder zu stark gestaucht noch zu sehr gedehnt werden, und dem Bestreben, Trennungen von Wörtern möglichst zu vermeiden.
Das Problem entfällt, wenn wir mittels des Befehls `\raggedright` auf Flattersatz umstellen.
Indem wir den Wert von `\hyphenpenalty` verändern, können wir Silbentrennungen generell verbieten (Wert 10000) oder ohne weiteres zulassen (Wert 0). Werte dazwischen verlagern die Gewichtung zwischen Trennen und ungleichmäßiger Wortverteilung.
LaTeX lädt mit der `german`–Option Trenntabellen für die deutsche Sprache. Wir können in der Präambel für Wörter, die das Programm nicht kennt, selbst angeben, wie sie getrennt werden sollen. Das geht so:

```
\hyphenation{Do-nau-dampf-schiff-fahrts-ka-pi-täns-mütze}
```

Dabei können wir eine ganze Liste von Worten eingeben, die jeweils durch ein Leerzeichen getrennt werden; die möglichen Stellen zum Trennen werden durch einen Ternnstrich `-` (Minuszeichen) gekennzeichnet.
Falls es Probleme gibt, kann man auch im Text für ein bestimmtes Wort angeben, wo es getrennt werden soll oder kann. Der Befehl `\-` gibt explizit an, dass an dieser Stelle und nur dort getrennt werden kann. Geben wir `ab\-rollen` ein, ist damit die Trennung zwischen den beiden „l“ ausgeschlossen. Hingegen bewirkt der Befehl `"-`, den die `german`–Option bereitstellt, dass an dieser Stelle getrennt werden kann, erlaubt aber auch weiterhin die in der Trenntabelle vorgesehenen Trennungen. Für besondere Fälle wie Worte mit Bindestrich kann man auch dafür sorgen, dass beim Trennen kein Trennstrich erzeugt wird; dies bewirkt der Befehl `""`, den die `german`–Option bereitstellt.
Ein besonderer Befehl sorgt dafür, dass Trennungen so vorgenommen werden, wie im deutschen üblich. `Dru"cker` erzeugt im Falle der Trennung korrekt „Druk-ker“. Ähnlich erzeugt `Ro"lladen` die Trennung „Roll-laden“, dies ist aber nur noch für Anhänger der alten Rechtschreibung relevant (und konsequenterweise in der Option `ngerman` für das `babel`–Paket nicht mehr enthalten).
Will man Trennungen an bestimmten Stellen vermeiden, kann man das entsprechende Wort in eine Box packen. Schreiben wir `\mbox{Müller}`, wird stets „Müller“ ohne Silbentrennung erscheinen. Mit Hilfe der Tilde kann man auch Zeilenumbrüche an einer Leerstelle zwischen zwei Worten verhindern, so verhindert etwa die Eingabe `Dr.~Who`, dass Herr Who von seinem Doktortitel getrennt wird.
Auf Silbentrennung sollte man prinzipiell von vornherein achten, z. B., um grundsätzlich auszuschließen, dass an Stellen ein Zeilenumbruch vorkommt, wo er nichts zu suchen hat, wie etwa bei Dr. Who. Andererseits verschiebt man eine explizite Angabe von Stellen zur Trennung

sinnvollerweise bis zur Endredaktion, wenn Inhalt und Formatierung des Textes ansonsten feststehen.

Literaturverzeichnis

Braams, J. (2005a): *Babel, a multilingual package for use with LaTeX's standard document classes (package description)* `../texmf/doc/generic/babel/babel.pdf`.

——— (2005b): *Babel, a multilingual package for use with LaTeX's standard document classes (userguide)* `../texmf/doc/generic/babel/user.pdf`.

Carlisle, D. P. (1997): *The* `xspace` *package* `../texmf/doc/latex/tools/xspace.dvi`.

——— (1999): *Packages in the* `graphics` *bundle* `../texmf/doc/latex/graphics/grfguide.pdf`.

Goossens, M., F. Mittelbach, und A. Samarin (2002): *Der LaTeX Begleiter*. Pearson Studium, München, ISBN: 3-8273-7044-2, 600 S., 39,95 €.

Jeffrey, A., und F. Mittelbach (2004): *inputenc.sty* version 1.0d, `../texmf/doc/latex/base/inputenc.dvi`.

Kern, U. (2005): *Extending LaTeX's color facilities: the* `xcolor` *package* v2.09, `../texmf/doc/latex/xcolor/xcolor.pdf`.

Kohm, M., und J.-U. Morawski (2004): *Die Anleitung: KOMA–Skript* `../texmf/doc/latex/koma-script/scrguide.pdf` oder `ftp://ftp.dante.de/tex-archive/macros/latex/contrib/koma-script/scrguide.pdf`.

Raichle, B. (1998): *Kurzbeschreibung* `german.sty` *und* `ngerman.sty` *(Version 2.5)* `../texmf/doc/latex/german/gerdoc.dvi`.

Trettin, M. (2004): *Das LaTeX 2ε–Sündenregister oder veraltete Befehle, Pakete und andere Fehler* version 1.7, `../texmf/doc/guides/l2tabu/german/l2tabu.pdf`.