3 Struktur eines Textes, Inhaltsverzeichnis

LaTeX stellt eine ganze Reihe von Strukturen bereit, um einen Text zu untergliedern. In diesem Abschnitt wollen wir uns mit ihnen vertraut machen und einen ersten Eindruck gewinnen, was LaTeX z. B. mit einem Befehl wie `\section{Überschrift}` alles anstellt.

3.1 Teil, Kapitel, Abschnitt etc.

LaTeX stellt zur Strukturierung eines Dokuments eine ganze Hierarchie unterschiedlicher Gliederungsbefehle bereit (vgl. Goossens, Mittelbach, und Samarin, 2002, Abschnitt 2.3). Dies sind im Einzelnen `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph` und `\subparagraph`. Für die Dokumentklassen `book`, `scrbook`, `report` und `scrreprt` bildet `\part` die Gliederungsebene -1, `\chapter` die Ebene 0, `\section` die Ebene 1 usw. Für die Dokumentklassen `article` und `scrartcl` gibt es keinen Befehl `\chapter`; hier ist `\part` die Ebene 0, `\section` die Ebene 1 usw.
Alle diese Befehle haben die Form `\gliederungsebene[Kurzüberschrift]{Überschrift}`. Sie erzeugen eine Überschrift, die je nach Ebene noch von einer Nummerierung und Text begleitet wird, etwa „Kapitel 1 Kapitelüberschrift“ oder „1. Abschnittsüberschrift“. Das optionale Argument dient dazu, eine Kurzform der Überschrift anzugeben, die dann gegebenenfalls im Seitenkopf (beim `\pagestyle{headings}`, sogenannte „lebende Kolumnentitel“) und im Inhaltsverzeichnis verwendet wird.
Es existiert jeweils auch ein gesternte Form des Befehls, etwa `\section*{Überschrift}`, die die Nummerierung der Überschrift unterdrückt und dafür sorgt, dass der entsprechende Gliederungspunkt weder ins Inhaltsverzeichnis noch in lebende Kolumnentitel aufgenommen wird.
Neben diesen Gliederungsbefehlen gibt es noch weitere Strukturelemente.

3.1.1 Titelseite

Mit der Umgebung `titlepage` kann eine individuell gestaltete Titelseite erstellt werden.
LaTeX kennt aber auch den Befehl `\maketitle`. Dieser Befehl verwendet die Argumente der in der Präambel einzugebenden Befehle `\author{`*Name*`}`, `\title{`*Titel*`}` und `\date{`*Datum*`}`, um je nach Dokumentklasse eine Titelseite (`book`, `scrbook` oder den Kopf des Dokuments (`article` und `scrartcl`) zu erstellen. Mehrere Autoren werden durch den Befehl `\and` getrennt. Durch einen doppleten Backslash kann eine neue Zeile erzeugt werden, z. B. für nähere Angaben zu den Autoren. Zusätzlich ermöglicht der Befehl `\thanks{`*Danksagung*`}`, Fußnoten anzubringen, in denen Autoren etwa ihren Dank an diejenigen aussprechen, die beim verfassen des Textes geholfen haben.
Für das Datum steht der Befehl `\today` zur Verfügung, der bei jedem LaTeX–Lauf das jeweils aktuelle Datum erzeugt.Dies ist sehr hilfreich für Texte, die noch im Enstehen begriffen sind.

Die Metaangaben zu diesem Skript könnten also wie folgt aussehen.

```
\title{\LaTeX{} and Friends}
\author{Jörg Naeve\thanks{Das vorliegende Skript ...}\\
          Lehrstuhl für Mikroökonomik (520c)\\
          Institut für Volkswirtschaftslehre\\
          Universität Hohenheim\\
          70593 Stuttgart
        \and
        Oliver D. Raschka\\
          Monopolkomission\\
          Adenauerallee 133\\
          53113 Bonn}
\date{\today}
```

Näheres dazu findet man beispielsweise im Kopka (2002).

3.1.2 Zusammenfassung

Die `abstract`–Umgebung erzeugt einen Abstract. Sie ist nicht vorgesehen für die Dokumentklassen `book` und `scrbook`.

3.1.3 Anhang

Mit dem Befehl `\appendix` beginnt der Anhang. Er sorgt dafür, dass die Nummerierung der darauf folgenden Abschnitte wieder von vorne beginnt und anders aussieht als im Hauptteil.

3.1.4 Individuelle Anpassung der Gliederungsebenen

Wie man die Nummerierung der Gliederungsebenen beeinflussen kann, behandeln wir im folgenden Unterabschnitt. Um die Formatierung der Überschrift individuell anzupassen gibt es den Befehl `\@startsection`. Dieser Befehl und weitere Möglichkeiten werden ausführlich erläutert in Goossens, Mittelbach, und Samarin (2002, Abschnitt 2.3.2).
Die Namen, die LaTeX für bestimmte Gliederungsebenen automatisch erzeugt, etwa für den `abstract` lassen sich ebenfalls individuell anpassen. Dies geschieht zum Beispiel automatisch durch Sprachanpassungen, etwa durch die `german`–Option. Man kann aber auch selber eingreifen, indem man mittels des Befehls `\renewcommand` den entsprechenden Abschnittsnamen umdefiniert. Will man etwa für das Inhaltsverzeichnis (siehe Unterabschnitt 3.5) statt der Überschrift „Inhaltsverzeichnis“ lieber die Überschrift „Gliederung“ verwenden, kann man dies folgendermaßen erreichen:

```
\renewcommand{\contentsname}{Gliederung}
```

Eine Liste aller Befehle, die Abschnittsüberschriften definieren findet man in Tabelle 2.3 in Goossens, Mittelbach, und Samarin (2002).
Eine etwas andere Art der Anpassung von Gliederungsebenen möchten wir Euch nachfolgend vorstellen. Genaugenommen geht es hierbei sowohl um die Neudefinition der Struktur eines

Textes als auch um die Definition eines neuen Zählers (vgl. Abschnitt 3.2). Jeder der schon einmal in juristische Fachzeitschriften und Bücher bzw. in die Haupt- und Sondergutachten der Monopolkommission[1] geschaut hat, wird festgestellt haben, dass die Struktur des Textes durch sogenannte Randziffern/Randnummern bzw. Textziffern bestimmt wird. Zwar werden die üblichen Befehle für Gliederungsebenen wie `\section` und `\subsection` verwendet, aber zusätzlich auch eine durchgehende Nummerierung inhaltlich abgeschlossener Absätze. Diese unterscheiden sich von herkömmlichen `enumerate`–Umgebung in ihrer Satzlänge und in ihrem Abstand. Außerdem lassen sich die Aufzählungsumgebungen, wie wir sie in Abschnitt 2.3.4.2 kennengelernt haben, nicht über mehrere Gliederungsebenen automatisch durchnummerieren.[2] Eine Lösung möchten wir Euch nachfolgend vorstellen, wobei diese zugegebenermaßen noch ausbaufähig erscheint, dennoch eine einfache und „saubere“ Lösung darstellt.
Abhängig von der gewünschten Formatierung der Textziffern und der Strukturtiefe des zu erstellenden Dokuments definiert man eine bereits existierende Gliederungsebene neu. Im nachfolgenden Beispiel ist dies `\subparagraph` als niedrigst mögliche Gliederungsebene. Sie bringt im Prinzip genau die erwünschte Formatierung einer Textziffer mit sich, so dass weitere Anpassungen entfallen können.[3] Im Deklarationsteil wird der alte Befehl `\subparagraph` neu definiert:

```
\renewcommand{\thesubparagraph}{\arabic{subparagraph}.\hspace{-3.5ex}}
\newcommand{\tz}[1]{\subparagraph{}}
```

Der Befehl `\subparagraph` wird neu definiert, indem eine fortlaufende arabische Nummerierung mit einem Punkt am Ende gewünscht wird. Der anschließende Befehl bewirkt eine Abstandsverkürzung des nachfolgenden Textes zur Ziffer. Der Befehl mit der eine neue Textziffer beginnt heißt jetzt `\tz{}`. Die geschweiften Klammern bewirken, dass der Befehl abgeschlossen wird. Wichtig ist, dass die Textziffer weiterhin als Strukturelement einer `\subparagraph`–Ebene erkannt und verwendet wird. So lassen sich sowohl die gesternte Form als auch Labels weiterhin verwenden.[4]

3.2 Zähler

Zu jeder Gliederungsebene gehört ein entsprechender Zähler (counter), z. B. nimmt der Zähler `section` die Abschnittsnummerierung vor. Dies erfolgt automatisch. Dabei sind die Zähler niedrigerer Gliederungsebenen denen der jeweils höheren untergeordnet, d. h., dass etwa Unterabschnitte nur innerhalb eines Abschnitts nummeriert werden. Sobald ein neuer Abschnitt beginnt, beginnt die Zählung der Unterabschnitte von vorn. Einander untergeordnet sind zum Beispiel auch die Zähler ineinander verschachtelter nummerierter Aufzählungen (`enumerate`–Umgebungen), `enumi`, `enumii`, `enumiii` und `enumiv`.
Zähler können aber auch mittels einiger Befehle direkt manipuliert werden, die wir hier kurz diskutieren (vgl. Goossens, Mittelbach, und Samarin, 2002, Abschnitt A.1.3). Der Wert eines Zählers ist dabei stets eine ganze Zahl.

[1] http://www.monopolkommission.de
[2] Man kann jetzt zwar auf die Idee kommen, die Gliederungsebenen in die Aufzählungsebenen einzuschließen, dies hat aber den Nachteil, dass alle Überschriften wie die Aufzählungsebenen formatiert werden. D. h. sie erscheinen eingerückt und mit einem anderen vertikalen Abstand.
[3] Im Einzelnen sind dies der vertikale und horizontale Abstand der Ziffer zum Text und der Abschnitte zu anderen Abschnitten, sowie die Zeilenlänge, die der Textbreite entspricht.
[4] Zur Funktion von Labels siehe Abschnitt 3.6.

`\newcounter{newcounter}[oldcounter]` definiert einen neuen Zähler namens *newcounter*, der bei Angabe des optionalen Arguments dem Zähler *oldcounter* untergeordnet wird; dazu muss der Zähler *oldcounter* natürlich bereits definiert sein.
`\setcounter{counter}[value]` setzt den Zähler *counter* auf den Wert *value*.
`\addtocounter{counter}[value]` erhöht den Zähler *counter* um den Wert *value*, dabei darf dies auch eine negative Zahl sein.
`\stepcounter{counter}[value]` erhöht den Wert des Zählers *counter* um eins und setzt gleichzeitig alle untergeordneten Zähler auf Null zurück.
Zu jedem Zähler *counter* existiert ein Befehl `\the`*counter*, der die Darstellungsform ausgibt, die für den jeweiligen Zähler angegeben ist. Dabei kann ein Zählerwert auf verschiedene Art und Weise ausgegeben werden:
`\arabic{`*counter*`}` gibt den aktuellen Wert des Zählers *counter* in arabischen Ziffern aus: $1, 2, 3, 4, \ldots$
`\roman{`*counter*`}` gibt den aktuellen Wert des Zählers *counter* in kleinen römischen Ziffern aus: i, ii, iii, iv, ...
`\Roman{`*counter*`}` gibt den aktuellen Wert des Zählers *counter* in großen römischen Ziffern aus: I, II, III, IV, ...
`\alph{`*counter*`}` gibt den aktuellen Wert des Zählers *counter* als kleinen Buchstaben aus: a, b, c, d, ..., z.
`\Alph{`*counter*`}` gibt den aktuellen Wert des Zählers *counter* als Großbuchstaben aus: A, B, C, D, ..., Z.
Will man das Erscheinungsbild eines Zählers verändern, so muss dies mit dem Befehl `\` geschehen. Will man etwa Unterabschnitte mit kleinen Buchstaben nummerieren, und soll davor jeweils die Nummer des Abschnitts als Großbuchstabe gefolgt von einem Punkt stehen, so würde man definieren

```
\renewcommand{\thesubsection}{\Alph{section}.\alph{subsection}}
```

Der Zähler `secnumdepth` legt fest, bis zu welcher Gliederungsebene eine Nummerierung stattfindet. In der hier verwendeten Dokumentklasse `scrbook` ist dies standardmäßig beispielsweise die Ebene 2. Nummeriert wird also bis `\subsection`, während eine `\subsubsection` keine Nummer mehr erhält. Durch entsprechende `\setcounter` oder `\addtocounter` Befehle lässt sich dies leicht ändern (wir haben den Zähler zum Beispiel um eins erhöht).

3.3 Sätze, Definitionen etc.

In mathematischen wie auch in vielen ökonomischen Texten findet man Sätze, Definitionen, Lemmata und dergleichen. Dafür bietet LaTeX die generische `theorem`–Umgebung an.
In der Präambel wird eine neue derartige Struktur definiert durch den Befehl

`\newtheorem{`*Name*`}[`*zählen-wie*`]{`*Überschrift*`}[`*Ebene*`]`

Dabei wird eine neue Umgebung `Name` definiert. Zu dieser gehört ein Zähler `\Name`, es sei denn, mit dem optionalen Argument *zählen-wie* wird gefordert, für die neue Umgebung einen bereits bestehenden Zähler mit zu verwenden. Beispielsweise könnte man sämtliche Sätze, Lemmata, Definitionen etc. fortlaufend durchnummerieren, indem man für alle den Zähler `\satz` verwendet. Das zweite optionale Argument ist in der Regel eine Gliederungsebene. Es

legt fest, dass die entsprechende Struktur innerhalb dieser Gliederungsebene nummeriert wird und seine Nummer sich dann auch zusammensetzt aus der betreffenden Gliederungsebene und der Struktur selbst, also etwa „Axiom 4.3“ für das dritte Axiom in Kapitel 4, falls für *Ebene* `chapter` verwendet wurde. Ohne dieses Argument wird das ganze Dokument hindurch gezählt. Verwendet wird eine derartige Struktur dann durch

`\begin{`*Name*`}[`*Zusatz*`]` Text `\end{`*Name*`}`

Dies erzeugt die Nummer, gemäß der Definition des dazugehörigen Zählers und die in der Definition festgelegte *Überschrift*. Im Falle der Verwendung des optionalen Arguments erscheint der *Zusatz* dann in Klammern hinter der Überschrift.
Das Paket `theorem` bietet zusätzliche Möglichkeiten der Formatierung von Sätzen. Siehe dazu auch Goossens, Mittelbach, und Samarin (2002, Abschnitt 8.8).

3.4 Fußnoten

Fußnoten werden mit dem Befehl `\footnote{`*Text der Fußnote*`}` erzeugt. An der Stelle des Befehls wird als Markierung eine hochgestellte Zahl angebracht (der Wert des Zählers `footnote`), der Text erscheint dann in kleinerer Schrift (nämlich in `\footnotesize` vgl. Abschnitt 2.3.2.6) unten auf der Seite. Fußnoten können Text, auch mehrere Absätze, Formeln, Tabellen u. ä. enthalten.
Die Formatierung der Fußnoten kann individuell angepasst werden, was ausführlich erläutert wird in Goossens, Mittelbach, und Samarin (2002, Abschnitt 3.4.1).
Wenn das Paket `endnotes` von Uwe Lück geladen wird, können mit dem Befehl `\endnote{`*Text der Endnote*`}` auch Endnoten erzeugt werden. Sie werden an der Stelle in den Text eingefügt, an der der Befehl `\theendnotes` auftaucht. Die Überschrift der Endnotes wird durch den Befehl `\notesname` festgelegt. Eine Beschreibung dieses Pakets findet man in Lück (2006).

3.5 Inhaltsverzeichnis

Das Inhaltsverzeichnis wird durch den Befehl `\tableofcontents` automatisch erzeugt, der an beliebiger Stelle im Textteil verwendet werden darf (in der Regel am Anfang). Seine Überschrift wird durch den Befehl `\contentsname` festgelegt, den man in der Präambel nach eigenem Gusto umdefinieren kann. Der Befehl `\tableofcontents` verwendet die `.toc` Datei, in die LaTeX die Überschriften der verschiedenen Gliederungsebenen automatisch einträgt.[5] Damit das Inhaltsverzeichnis korrekt erstellt werden kann, sind also mindestens zwei LaTeX–Läufe nötig: Im ersten wird die `.toc` Datei erzeugt, und im zweiten wird sie von `\tableofcontents` eingelesen.
Der Zähler `tocdepth` legt fest, bis zu welcher Gliederungsebene das Inhaltsverzeichnis erstellt wird. `\setcounter{tocdepth}{2}` bewirkt also, dass Abschnitte (`\section`) und Unterabschnitte (`\subsection`) im Inhaltsverzeichnis erscheinen, nicht aber tiefere Untergliederungen. Ist die Überschrift eines Abschnitts zu lang für das Inhaltsverzeichnis, kann mit dem optionalen Argument des Gliederungsbefehls eine passende Kurzversion festgelegt werden.

[5] Im Notfall kann man diese Datei auch von Hand manipulieren.

Mehr zu diesem Thema, insbesondere dazu, wie das Verzeichnis formatiert werden kann und wie zusätzliche Einträge erzeugt werden, bietet Goossens, Mittelbach, und Samarin (2002, Abschnitt 2.4).

3.6 Querverweise

LaTeX bietet für Querverweise innerhalb eines Dokuments Unterstützung durch die Befehle `\label{`*key*`}`, `\ref{`*key*`}` und `\pageref{`*key*`}`.
Dabei erzeugt der Befehl `\pageref{`*key*`}` die Seitenzahl der Stelle im Text an der das Label `\label{`*key*`}` angebracht ist. Der Schlüssel *key* muss dabei innerhalb des Dokuments eindeutig sein. Dafür ist es hilfreich den *key* beispielsweise stets mit einem Kürzel für das jeweilige Strukturelement, in dem ein Label gesetzt wird, beginnen zu lassen. Also etwa `\label{sec:`*keyname*`}` für ein Label in einem Abschnitt, `\label{fig:`*keyname*`}` für ein label in einer `figure`–Umgebung (vgl. das Kapitel 7) oder `\label{fn:`*keyname*`}` für ein Label in einer Fußnote.
Der Befehl `\ref{`*key*`}` erzeugt eine Zeichenfolge, die das Element identifiziert, auf das verwiesen wird, also etwa die Nummer eines Abschnitts.
Damit alle Querverweise stimmen, sind mindestens zwei LaTeX–Läufe erforderlich. Im ersten schreibt LaTeX die nötigen Informationen in eine Hilfsdatei mit der Endung `.aux`, aus der sie im zweiten Lauf ausgelesen werden, wenn an einer Stelle ein Querverweis erfolgt. Gibt es noch Unklarheiten (Bezüge auf nicht existente Labels oder Mehrfachdefinitionen eines Labels) beschwert sich LaTeX und gibt in der `.log` Datei entsprechende Warnungen aus.

3.6.1 Zusätzliche Pakete für Querverweise

Das Paket `varioref` von Frank Mittelbach stellt einen Mechanismus bereit, der ermöglicht, Querverweise flexibel zu gestalten, so dass beispielsweise ein Verweis je nach Lage der Dinge „auf dieser Seite“, „auf der folgenden Seite“ oder „auf Seite 6“ lautet. Es ist beschrieben in Mittelbach (2004).
Mit dem Paket `xr` von David P. Carlisle kann man auf Stellen in anderen `.tex` Dokumenten verweisen, die vorher in der Präambel mit dem Befehl `\exteraldokument{`*anderesDokument*`}` angegeben werden müssen. Es ist beschrieben in Carlisle (1994).
Das Paket `hyperref` (Rahtz und Oberdiek, 2004) von Sebastian Rahtz und Heiko Oberdiek ermöglicht es unter anderem, eine PDF Datei zu erzeugen, in der Inhaltsverzeichnis und Querverweise wie HTML–Links wirken.

3.7 Gleitobjekte

Insbesondere wissenschaftliche Texte gewinnen an Verständlichkeit, wenn komplexe Inhalte vereinfacht und/oder zusammengefasst dargestellt werden. So lassen sich die relevanten Informationen mit Hilfe von Tabellen und Abbildungen besonders kompakt und übersichtlich darlegen.
Mit LaTeX sind im Prinzip alle nur erdenklichen Typen von Darstellungen möglich, so dass der Vielfalt insbesondere in diesem Bereich so gut wie keine Grenzen gesetzt sind. Allerdings sollte man vor lauter Möglichkeiten nicht vergessen, was eine gute wissenschaftliche

Darstellung auszeichnet. Wichtige Kriterien für eine sachgerechte Aufbereitung von Text sind Einfachheit, Eindeutigkeit und Verständlichkeit. Bieten Darstellungen keinen Mehrwert im Sinne dieser Kriterien, so kann auch auf sie getrost verzichtet werden. Für einen Leitfaden zur Erstellung sachgerechter Tabellen und Abbildungen im wissenschaftlichen Kontext siehe die Ausführungen von Theisen (2000, S. 162–169).
Einige der Pakete, die in jeder Basis–Version von LaTeX enthalten sind und von denen wir nachfolgend einige vorstellen, ermöglichen schon mit ihren Grundoptionen komplexe Tabellen oder Abbildungen. Da derartige Darstellungen oft eine gewisse Ausdehnung aufweisen, die es erschweren, sie in den normalen Fließtext einzufügen (insbesondere Seitenumbrüche können Probleme bereiten), werden sie in der Regel als Gleitobjekte eingefügt. Tabellen oder Abbildungen werden dabei an der Stelle in den Text eingebunden, an die sie inhaltlich gehören. Mit Hilfe von Optionen berechnet LaTeX dann an welcher Stelle die Darstellung tatsächlich erscheint. Dies kann entsprechend der Vorgaben und dem zur Verfügung stehende Platz direkt im Anschluss an den Text oder erst auf der folgenden Seite sein.
Aus didaktischen Gründen werden wir eine detaillierte Diskussion von Gleitobjekten auf das Kapitel 7 vertagen, in dem außerdem die Erstellung von Tabellen behandelt wird. Die Einbindung externer Grafiken und die Möglichkeiten, die LaTeX bzw. zusätzliche Pakete bieten, Grafiken selbst zu erzeugen, besprechen wir in Kapitel 8.

3.8 Unterteilung von Dokumenten

Wenn man ein längeres Dokument in LaTeX verfasst, bietet es sich möglicherweise an, nicht den gesamten Text in einer einzigen Datei abzuspeichern. Dies ist besonders hilfreich, wenn etwa mehrere Autoren gemeinsam an einem Text arbeiten; man denke etwa an ein Buch, zu dem mehrere Autoren Kapitel liefern.
LaTeX bietet zwei verschiedene Befehle, die dies unterstützen, nämlich `\input{`*Datei*`}` und `\include{`*Datei*`}`. Dabei sucht LaTeX nach einer Datei namens *Datei*`.tex`, die es an der Stelle einfügt, an der der Befehl steht.
Eine Möglichkeit wäre etwa, eine Datei `meinkopf.tex` zu erstellen, die alle Eingaben enthält, die man regelmäßig in der Präambel verwendet. Dann kann eine konkrete `.tex` Datei, die man bearbeitet so aussehen

```
\input{meinkopf.de}
\begin{document}
   Text
\end{document}
```

Der Befehl `\input{`*Datei*`}` fügt einfach den Inhalt der Datei *Datei*`.tex` ein; er kann überall verwendet werden. Es ist auch möglich, ihn geschachtelt zu verwenden, d. h. ein Dokument kann per `\input` eine Datei laden, die ihrerseits wieder `\input` Befehle enthält.
Der Befehl `\include{`*Datei*`}` hingegen ist nur im Textteil eines LaTeX–Dokuments erlaubt und darf nicht geschachtelt werden. Erlaubt sind hingegen `\input` Befehle in einer durch `\include` eingelesenen Datei. Dieser Befehl fügt ebenfalls den Inhalt der Datei *Datei*`.tex` ein, beginnt dabei aber gleichzeitig eine neue Seite.
Bei beiden Befehlen bewirkt das Fehlen der Datei, die eingelesen werden soll einen Stopp der Verarbeitung durch TeX mit einer Fehlermeldung.

Werden mehrere Dateien mittels `\include` eingefügt, sagen wir beispielsweise dies seien die Dateien `Datei1.tex`, `Datei2.tex`, ...`Datei9.tex`, kann in der Präambel mit dem Befehl `\includeonly{Datei1,Datei2,Datei7}` dafür gesorgt werden, dass lediglich die entsprechenden Teile, hier also die in den Dateien `Datei1.tex`, `Datei2.tex` und `Datei7.tex`, des gesamten Dokuments bearbeitet werden. Dabei werden aber sämtliche Informationen aus den `.aux` Dateien zu allen Teilen aus früheren LaTeX–Läufen verwendet, d. h. beispielsweise funktionieren sämtliche Querverweise weiterhin. Zum Schluss sollte man allerdings stets das gesamte Dokument übersetzen und zwar mehrfach, damit sämtliche Querverweise und ähnliches korrekt erstellt werden.

3.8.1 Projekte im WinEdt

Hat man den Quelltext eines Dokuments in mehrere Teile unterteilt, steht man häufig vor dem Problem, dass man gerade einen dieser Teile bearbeitet, dann aber gern das Dokument übersetzen würde (den einzelnen Teilen fehlt der Kopf, so dass sie einzeln nicht übersetzbar sind). Oder es fällt plötzlich auf, dass man einen anderen Teil benötigt, zum Beispiel um einen Querverweis anzubringen.
Der Editor WinEdt macht das Arbeiten mit einem Dokument, dessen Quelltext über mehrere Dateien verteilt ist sehr komfortabel, wenn man die **Projekt** Funktion nutzt.
Nehmen wir an, wir schreiben ein Buch. Die entsprechende Hauptdatei `meinbuch.de.tex` sehe aus wie folgt.

```
\input{meinbuchkopf.de.tex}

\begin{document}
  \include{Kapitel1.de.tex}
  \include{Kapitel2.de.tex}
  \include{Kapitel3.de.tex}
  \include{Kapitel4.de.tex}
  \include{Kapitel5.de.tex}
\end{document}
```

Um daraus ein Projekt zu machen öffnet man die Datei `meinbuch.de.tex`, geht ins WinEdt–Menue `Project` und wählt dort den Punkt `Set Main File`. Nun kann man mittels der Funktion `Tree` einen Baum sämtlicher Dateien erstellen, die zu dem Main File gehören und bei Bedarf diese auch gleich öffnen.
Bearbeitet man nun eine dieser untergeordneten Dateien und benutzt einen der Icons zum Übersetzen, wird automatisch der Main File übersetzt, also genau das getan, das man will.
Die Funktion `Gather` sammelt alle wesentlichen Informationen zum vorliegenden Projekt. Zum Beispiel findet man im Reiter `TOC` ein Inhaltsverzeichnis, aus dem man direkt an die entsprechende Stelle der passenden Quelldatei springen kann.
Im Main File kann man zusätzliche Dateien einbinden, indem man Zeilen der folgenden Art einfügt.

```
%GATHER{meinbuch.bib}
%GATHER{meinbuch.de.bbl}
```

Wegen des Kommentarzeichens am Anfang werden sie von LaTeX ignoriert, bringen WinEdt aber dazu, die entsprechenden Dateien ebenfalls dem Projekt zuzuordnen und die entsprechenden Informationen mit bereit zu stellen. Hier wären die beiden Dateien eine BibTeX–Quelldatei, die eine Literaturdatenbank ist, sowie die daraus erzeugte Datei `meinbuch.de.bbl`, die die grundlegenden Informationen zum Literaturverzeichnis enthält. Näheres zu diesem Thema folgt später in Kapitel 5.
Um ein Projekt zu speichern wählt man `Save Project` aus dem Menue `Project` oder dort `Close Project` und dann „ja“ bei der Frage, ob man speichern will.

Literaturverzeichnis

Carlisle, D. P. (1994): *The* ***xr*** *package* `../texmf/doc/latex/tools/xr.dvi`.

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

Kopka, H. (2002): *LaTeX I: Einführung*, Bd. 1. Pearson Studium, München, ISBN: 3-8273-7038-8, 544 S. + 1 CD, 39,95 €.

Lück, U. (2006): *for critical editions with LaTeX* `../texmf/doc/latex/endnotes/endnotes.dvi`.

Mittelbach, F. (2004): *The* ***varioref*** *package* `../texmf/doc/latex/tools/varioref.dvi`.

Rahtz, S. P. Q., und H. Oberdiek (2004): *Hypertext marks in LaTeX: a manual for* ***hyperref*** `../texmf/doc/latex/hyperrf/manual.pdf`.

Theisen, M. R. (2000): *Wissenschaftliches Arbeiten*. Verlag Vahlen, München, 10 Aufl., ISBN: 3-8006-2628-4, 292 S., 13,00 €.