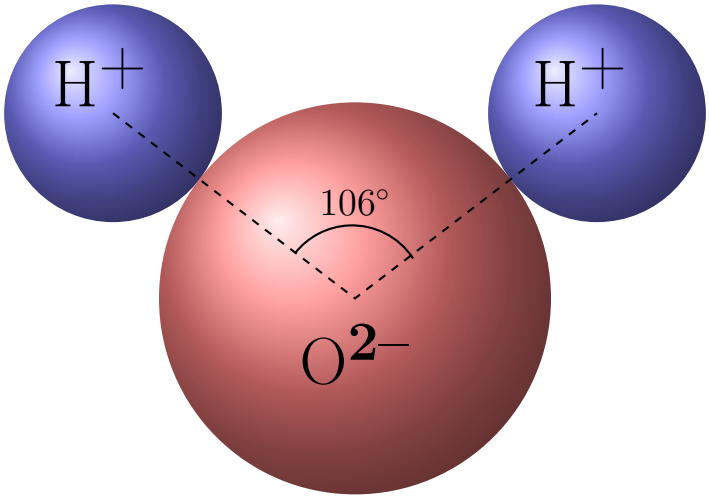
Ecce etiam ego adducam aquas super terram!



LaTeX

(opus imperfectum)

Alfred H. Gitter

Version vom 1. Januar 2021

Inhaltsverzeichnis

1 Grundlagen 3
- 1.1 Einführung . . . 3
- 1.2 Befehle . . . 8
- 1.3 Dokumentklassen und Pakete, Beispiel . . . 10
- 1.4 Gliederung (Titel, Textabschnitte und Absätze) . . . 15

2 Besondere Teilbereiche 19
- 2.1 Verzeichnisse (Inhalt, Abk., Abb., Literatur) . . . 19
- 2.2 Fuß- und Randnoten, Querverweise, Hyperlinks . . . 21
- 2.3 Bilder, Tabellen, Balkendiagramme, Kommentare . . . 22
- 2.4 Aufzählungen und Theorem-Umgebungen . . . 31
- 2.5 Programm-Code, Zeichnungen, Zähler . . . 33

3 Seite, Absatz und Wort 39
- 3.1 Seitengestaltung, mehrere Spalten . . . 39
- 3.2 Rahmen und Minipages . . . 41
- 3.3 Ausrichtung, Einrückung, Trennungen, Abstände . . . 44
- 3.4 Typographische Regeln . . . 47

4 Zeichenformatierung 50
- 4.1 Schrift (Stil, Größe, hoch/tief, Farbe) . . . 50
- 4.2 Leer- und Sonderzeichen . . . 56
- 4.3 Physikalische Einheiten . . . 62

5 Mathematik 67
- 5.1 Mathematik im Fließtext . . . 67
- 5.2 Sonderzeichen in einer *math*-Umgebung . . . 72
- 5.3 Gleichungen . . . 76

6 Grafiken mit TikZ 79
- 6.1 Strichzeichnungen und Füllungen . . . 79
- 6.2 Knoten . . . 84
- 6.3 Graphen und Funktionen . . . 86
- 6.4 Elektrische Schaltkreise und Optik . . . 91
- 6.5 Randnotizen, Kästen, Kalender und Avatare . . . 95

7 Ergänzungen 101
- 7.1 Physik, Chemie, Bioinformatik, Übungen usw. . . . 101
- 7.2 Gedichte, Spiele, Strichcode und QR-Code . . . 111
- 7.3 Präsentation mit *beamer* . . . 115
- 7.4 Briefe . . . 121
- 7.5 Literaturliste mit *biblatex* und *biber* . . . 123

1 Grundlagen

1.1 Einführung

> „Da LaTeX ein überholtes und auslaufendes System zur Formatierung von Text darstellt, entschloss ich mich vor rund dreißig Jahren, stattdessen nur das Textverarbeitungssystem eines namhaften Herstellers zu verwenden. *Vielleicht war das falsch.*“

Die vorliegende Skripte ersetzt nicht Bücher, die LaTeX fundiert erläutern. Vielmehr gibt sie Hinweise in Form von kommentierten Beispielen, die dem ungeduldigen Anwender helfen können. Die Richtigkeit wird natürlich nicht gewährleistet. Allerdings besteht die Gefahr der Ablenkung. Man sollte zunächst schreiben und die Formatierung LaTeX überlassen! Danach kann man einige der schönen Extras einfügen, welche hier dargestellt werden. Das Format der Skripte ist, dem Zweck entsprechend, nicht optimal für ein Buch, sondern stellt einen Kompromiss dar. Dieser ermöglicht ein- oder zweiseitige, Papier sparende Ausdrucke einzelner Abschnitte (Save Paper in School) und stellt (für die Freunde von Copy & Paste) beim Lesen am Bildschirm Verweise auf tex-Dateien bereit.

Kurzum, hier ist ein geschenkter Gaul, auf dem ihr reiten mögt oder nicht.

LaTeX erweitert eine alte, TeX genannte Programmiersprache zum Textsatz (von Donald E. Knuth von 1977 bis 1989 entwickelt) um benutzerfreundliche Befehle. Das La in LaTeX steht für den Entwickler Leslie Lamport.

Die seit 1994 aktuelle Version von LaTeX heißt LaTeX 2_ε; eine langfristig angelegte Weiterentwicklung läuft unter dem Namen LaTeX3. Die ältere Alternative zu LaTeX, das Eingabeformat *plain TeX* wird nur noch selten verwendet.

Kurz gesagt, LaTeX

- ist frei erhältlich für alle Betriebssysteme → verfügbar
- ergibt sehr guten Schriftsatz (Typographie) → lesbar
- eignet sich für viele Sprachen und Zeichensätze → universal
- wird in zahlreichen Anleitungen vorgestellt → dokumentiert
- unterstützt die Nutzer im Internet (CTAN, DANTE, TUG) → hilfreich
- gibt Dokument in Format Postscript oder PDF aus → druckreif
- erzeugt Formatierung durch Programmierung → anfangs schwierig
- kein Ergebnisbild (**W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et) → unübersichtlich
- erfordert jedesmal mehrere Programmdurchläufe → langsam

- bildet Dokumente aus Präambel und document-Umgebung → systematisch
- hat gewöhnungsbedürftige Fehlermeldungen → fordert Genauigkeit
- erleichtert das einheitliche Schreiben größerer Arbeiten → stabil
- erlaubt Einbindung von Tabellen, Bildern und mehr → umfangreich
- eignet sich für Mathematisches und Chemisches → wissenschaftlich
- nummeriert, formatiert und platziert Textobjekte → automatisch
- verwaltet ein bibliographisches Verzeichnis → einfach für Literaturzitate
- erstellt gute Präsentationen als PDF-Dateien → präsentabel
- speichert Text und Formatierung in Textdatei → portabel
- kann und muss durch Ergänzungen angepasst werden → erweiterbar
- bietet ansehbare Schriftarten, Symbole und Zeichnungen → schön

Bei der Textgestaltung mit LaTeX wird der Inhalt eines Dokuments mithilfe lesbarer Steuerbefehle in ein ansprechendes Druckbild übersetzt. Der Autor gibt, mit im Text eingebetteten, ausgeschriebenen Befehlen, die wesentlichen Gestaltungsmerkmale vor, überlässt Einzelheiten der Formatierung aber dem Programm. Die Befehle dienen zum Beispiel der Auswahl einer Schriftart, der Gliederung oder einer automatischen Nummerierung.

Es gibt verschiedene Ausgabeformate für die Darstellung der mit LaTeX erzeugten Dokumente. Heute wird PDF am häufigsten verwendet. Ein geeignetes Computerprogramm, welches LaTeX als Eingabeformat und PDF als Ausgabeformat hat, ist PDFLatex. Wesentliches Teil (Engine) von PDFLatex ist pdfTeX. Nachfolger von PDFLatex könnte zukünftig LuaLaTeX (Eingabeformat: LaTeX, Engine: LuaTeX, Ausgabeformat: PDF) sein. *Sub reservatione Jacobea* (so Gott will und wir leben).

Im Folgenden gehen wir davon aus, dass PDFLatex (auch pdflatex oder pdfLaTeX geschrieben) eingesetzt wird. LaTeX-Editoren haben dies meistens als Voreinstellung.

Hinweise und Hilfe erhält man unter anderem von der Deutschsprachigen Anwendervereinigung TeX e.V. (DANTE). Die Webseite ist `http://www.dante.de/`

Typographie ist die Wissenschaft oder Kunst des Schriftsatzes. Ein Text besteht aus Zeichenketten. Seine Lesbarkeit für Menschen hängt von Form und Anordnung der Zeichen, Schriftsatz genannt, ab.

Mit LaTeX wird ein guter Schriftsatz, der den Regeln der Typographie folgt, erreicht, ohne dass der Nutzer eingreifen muss. Der Nutzer kann aber, zum Beispiel durch Wahl der Schriftart, auf die automatisch erstellte Vorlage Einfluss nehmen.

Installation wird hier nicht näher beschrieben. Für Linux-Betriebssysteme kann sie ohne Besonderheiten über die Linux-Paketverwaltung erfolgen. Wenn möglich, sollten schließlich mindestens folgende Linux-Pakete vorliegen: *texlive*, *texlive-lang-german*, *texlive-generic-extra* und *texlive-latex-extra*. Wer genug Speicherplatz hat, sollte das ganze, umfangreiche LaTeX-System installieren.

Unter Linux können zusätzliche Latex-Pakete, die als Dateien mit der Endung `.sty` (oder `.cls` oder `.trsl`) vorliegen, installiert werden, indem sie in das Verzeichnis `~/texmf/tex/latex` kopiert werden (~ steht für das Benutzerverzeichnis `/home/BENUTZER`). Ebenso kopiert man alle Dateien mit der Endung `.def` (sofern vorhanden). Danach gibt man in einem Terminal) den Befehl

```
texhash ~/texmf
```

ein. Falls es das Verzeichnis noch nicht gibt, erzeugt man es mit

```
mkdir ~/texmf/tex/latex
```

Man kann statt einer Datei `PAKET.sty` auch ein Verzeichnis `PAKET`, das die Datei `PAKET.sty` enthält, in das Verzeichnis `~/texmf/tex/latex` kopieren.

Manche Latex-Pakete werden durch zwei Dateien mit den Endungen `.dtx` und `.ins` bereitgestellt. Unter Linux kopiert man beide in ein beliebiges temporäres Verzeichnis, zum Beispiel `~/Downloads`, und führt dann nacheinander die Befehle

```
cd ~/Downloads
```

und

```
latex ./PAKET.ins
```

aus. Damit erhält man eine Datei `PAKET.sty` (oder mehrere) und verfährt weiter wie oben beschrieben. Mehr zu diesem Thema findet man auf der Webseite
https://en.wikibooks.org/wiki/LaTeX/Installing_Extra_Packages

Für LaTeX unter Kleinweich-Betriebssystemen (*Microsoft Windows*) gibt es verschiedene Distributionen, unter anderem *TeXLive* und *MiKTeX*. Eine Anleitung zur Installation von TeXLive findet man auf der Webseite
https://www.tug.org/texlive/doc/texlive-de/texlive-de.pdf

Da TeXLive sehr umfangreich ist (etwa 3,5 GiB), kann eine direkte Installation über das Internet Probleme bereiten. Besser ist es, von einer DVD zu installieren. Wenn man keine hat, kann man ein ISO-Abbild herunterladen (Webseite: http://ftp.uni-erlangen.de/ctan/systems/texlive/Images/) und damit eine TeXLive-DVD erzeugen (brennen).

Auch unter Windows 10 und TeXLive können zusätzliche Latex-Pakete mithilfe der Konsole installiert werden. Wir gehen davon aus, dass die neuen Pakete (Dateien mit der Endung `.sty`, `.cls`, `.trsl`, `.dtx` oder `.ins`) nach dem Herunterladen im Unterverzeichnis *Downloads* des Benutzerverzeichnisses gespeichert sind. Durch die Tastenkombination *Windowstaste R* öffnet sich ein Fenster mit dem Titel *Ausführen*. Gibt man hier den Befehl `cmd.exe` ein, öffnet sich ein Konsolenfenster. Wichtige Befehle sind dort:

`dir` zeigt eine Liste von Verzeichnissen und Dateien im aktuellen Verzeichnis
`cd` *verz* macht das Unterverzeichnis *verz* zum aktuellen Verzeichnis
`cd ..` macht das übergeordnete Verzeichnis zum aktuellen Verzeichnis
`mkdir` *verz* erzeugt ein Unterverzeichnis *verz* im aktuellen Verzeichnis
`latex` *dat* führt das Programm pdfTeX mit der Datei *dat* aus
`texhash` *verz* führt das Programm `texhash` mit dem Verzeichnis *verz* aus

Nach dem Öffnen einer Konsole ist das Benutzerverzeichnis des aktuellen Nutzers *Nutzer* das aktuelle Verzeichnis. Mit dem Befehl

```
mkdir texmf\tex\latex
```

erzeugt man ein Unterverzeichnis, in dem neu installierte Latex-Pakete abgelegt werden können. Falls es dies Verzeichnis schon gibt, erhält man eine entsprechende Meldung vom Betriebssystem. Dann kopiert man, (nicht in der Konsole, sondern) wie üblich mit Maus und Windows-Fenstern, Dateien mit der Endung `.sty`, `.cls` oder `.trsl` vom Verzeichnis *Nutzer*`\Downloads` nach *Nutzer*`\texmf\tex\latex`. Liegt das Latex-Paket in Form zweier Dateien *dat*`.dtx` und *dat*`.ins` vor, geht man in der Konsole mit `cd Downloads` ins Unterverzeichnis `Downloads` und führt dort

```
latex dat.ins
```

aus. Danach liegen im Unterverzeichnis `Downloads` Dateien mit der Endung `.sty`, `.cls` oder `.trsl`, welche man in das Verzeichnis *Nutzer*`\texmf\tex\latex` kopiert. Nun kann man im Unterverzeichnis `Downloads` alle heruntergeladenen und erzeugten Dateien löschen. In der Konsole geht man mit `cd ..` wieder ins Benutzerverzeichnis *Nutzer* und führt den Befehl

```
texhash texmf
```

aus. Schließlich ist man fertig und kann das Konsolenfenster schließen.

Installationsprobleme Manche Pakete brauchen die folgenden Pakete (oder Teile davon, zum Beispiel `expl3`) zur Unterstützung von LaTeX 3 : *l3kernel*, *l3packages* und vielleicht auch *l3experimental*. Einige Linux-Distributionen (zum Beispiel openSUSE Leap 15.2) stellen diese Pakete in einer alten Version zur Verfügung, die oft nicht genügt (was zum Beispiel zum Fehler „support package expl3 too old“ führt).

In diesem Fall kann man die Pakete „manuell“ installieren: Zunächst lädt man die Pakete als komprimierte ZIP-Archivdateien in das Verzeichnis `~/texmf/tex/latex` (siehe oben, Seite 5) und entpackt sie dort. Nach dem Entpacken gibt es entsprechende Verzeichnisse und die ZIP-Dateien können gelöscht werden.

In jedem der neuen Verzeichnisse führt man für alle enthaltenen Dateien mit der Endung `.ins` in einem Terminal den Befehl

```
latex ./PAKET.ins
```

aus, wobei `PAKET` natürlich durch den aktuellen Dateinamen ersetzt werden muss. Anschließend gibt man den Befehl

```
texhash ~/texmf
```

und ist fertig.

Eingabe von Text und Befehlen kann mit jedem einfachen, allgemein verwendbaren Editor, unabhängig vom Betriebssystem des verwendeten Rechners, geschehen. Nachteilig ist beim Arbeiten mit einem einfachen Editor, dass das gestaltete Dokument nicht sichtbar ist, fehlerhafte Befehle nicht sofort erkannt werden und für häufig verwendete Befehle keine Abkürzung bereitsteht. Deshalb ist es bequemer, mit einem besonderen LaTeX-Editor zu arbeiten. Es gibt viele, zum Beispiel der einfache, für Anfänger geeignete *TeXworks*, und die umfangreicheren *Texmaker* und *TeXstudio*. Ein einfacher LaTeX-Editor für Linux ist auch *Gummi*.

Dateien sind Datensätze, die einen eigenen Namen haben. Es ist ratsam, Dateien so zu benennen, dass verschiedene Betriebssysteme den Dateinamen in gleicher

Weise erkennen. Jeder Dateiname ist eine Zeichenkette, die aus den Elementen einer Zeichenmenge gebildet wird. Man sollte diese Zeichenmenge (auch Alphabet genannt) auf Kleinbuchstaben, Ziffern, `_` und `.` beschränken. Insbesondere sollte man keine Leerzeichen verwenden. Das erste Zeichen des Namens sollte ein Buchstabe sein. Der Punkt sollte nur einmal enthalten sein und ihm folgen dann in der Regel drei Zeichen, welche Dateiendung heißen. (Bei der Angabe der Endung wird oft auch der vorangehende Punkt geschrieben.) Der Quellcode eines mit LaTeX an Max geschriebenen Briefs vom 13. Februar 2019 könnte zum Beispiel in einer Datei namens `brief_max_190213.tex` stehen.

Man beachte, dass manche Betriebssysteme die schlechte Angewohnheit haben, Dateinamen ohne die Endung (und ohne den vorangehenden Punkt) anzuzeigen.

Stellt man vor den Dateinamen den Namen des Verzeichnisses, in dem die Datei liegt, erhält man den Pfadnamen. Ein Datensatz wird durch seinen Pfadnamen eindeutig gekennzeichnet.

Dateien mit LaTeX-Quellcode werden gewöhnlich unter einem Namen mit der Endung `tex` gespeichert. Der Quellcode einer regelgerechten LaTeX-Datei wird als Dokument bezeichnet. Es enthält zwei Teile. Der erste heißt Präambel, der zweite *document*-Umgebung.

Druckfertiges Ergebnis ist die PDF-Datei mit der Endung `pdf`, welche vom Programm PDFLatex aus dem LaTeX-Quellcode erzeugt wurde. PDF-Dateien haben den Vorteil, auf verschiedenen Rechnern, unabhängig vom Betriebssystem, die gleiche Ausgabe auf Bildschirm und Drucker zu erzeugen. (Theoretisch jedenfalls; in der Praxis verhält sich manches PDF-Betrachterprogramm wie ein Lehmziegel im Zirkus. Um dennoch das gewünschte Druckergebnis in annehmbarer Qualität zu erhalten, kann es helfen, im erweiterten Menü das Drucken als Bild = image zu wählen. Ansonsten sollte man ein anderes PDF-Betrachterprogramm benutzen.)

Auf dem Weg vom Quelltext in Datei *x*`.tex` zum druckfertigen Ergebnis in Datei *x*`.pdf` erzeugt das Programm PDFLatex verschiedene Hilfsdateien. In *x*`.log` wird die Arbeit des Programms mitgeschrieben und Fehler und Warnungen mitgeteilt. Der Zuordnung von Querverweisen dient *x*`.aux`. Verweise auf Abschnittsüberschriften werden in *x*`.out` gespeichert. Verzeichnisse von Inhalt, Abbildungen, Tabellen und Literatur brauchen Dateien *x*`.toc`, *x*`.lof`, *x*`.lot` und *x*`.bib`. Für die Zusammenarbeit von LaTeX-Editoren mit PDF-Betrachterprogrammen kann eine Datei *x*`.synctex.gz` erzeugt werden.

Vom Quellcode zur PDF-Datei gelangt man mit LaTeX-Editoren per Knopfdruck. Bei Linux-Betriebssystemen kann man PDFLatex im „Terminal“ mit dem Befehl

```
pdflatex datei.tex
```

ausführen, wenn die Datei *datei.tex* im aktuellen Verzeichnis liegt. (Mit der Option `-shell-escape` erlaubt man die Ausführung von Shell-Befehlen. Das darf man aber nur tun, wenn man sicher ist, dass der Quellcode keine Schadsoftware enthält.)

Beim Ausdruck von PDF-Dateien auf einem Drucker, der nicht bis zum Seitenrand drucken kann, sollte man im Druck-Menü *randlosen Druck* wählen. Ansonsten werden die Druckseiten automatisch verkleinert ausgegeben. Leider unterstützen nicht alle Laserdrucker den randlosen Druck.

1.2 Befehle

LaTeX-Befehle beginnen mit einem \ (Backslash). Danach kommt der Befehlsname, der aus Buchstaben besteht. Anschließend werden oft, in geschweiften oder eckigen Klammern, Argumente übergeben. Argumente (auch Parameter genannt) beeinflussen die Befehle.

Man unterscheidet Pflichtargumente, deren Angabe notwendig ist, und optionale Argumente, mit denen der Befehl (*command*) verändert werden kann, aber nicht muss. Pflichtargumente stehen in geschweiften Klammern *{...}*, optionale in eckigen Klammern *[...]*. Mehrere Argumente in einer Klammer werden durch den Aufzählungsoperator, ein Komma, getrennt.

Das Ende eines Befehlsnamens wird durch ein folgendes Leerzeichen oder Sonderzeichen erkannt. Folgt ein oder mehrere Leerzeichen, werden sie nicht gedruckt. Will man ein Leerzeichen nach einem Befehl darstellen, kann man einen leeren Block *{}* an den Befehlsnamen anfügen. Dies wird hier anhand des Befehls *\LaTeX* gezeigt, welcher (ohne Argumente) das LaTeX-Logo ausgibt.

```
\LaTeX{} ist gut, \LaTeX nicht.
```

LaTeX ist gut, LaTeXnicht.

Eine zweiseitige, deutsche Übersicht wichtiger Befehle findet man unter der Bezeichnung *latexsheet-de.pdf*. Der Quellcode liegt in der Datei *latexsheet-de.tex* vor. Es gibt auch eine schöne Befehlsübersicht von Christian Feuersänger.

Kommentare verbessern die Verständlichkeit der verwendeten Befehle. Ein Kommentar (*comment*) wird mit % (Prozentzeichen) eingeleitet. Der anschließende Text bis zum Zeilenende wird nicht abgebildet. Mehr dazu in Abschnitt 2.3 auf Seite 30.

Die ersten drei Befehle sind

```
\documentclass[a4paper, 11pt]{scrartcl} % Präambel
\begin{document}    % Anfang der document-Umgebung
Hallo Welt\,!       % (Hier steht der Textinhalt.)
\end{document}      %   Ende der document-Umgebung
```

Dokument mit drei Befehlen, Ausgabe: Hallo Welt!

Eine detaillierte Vorgabe des Formats eines LaTeX-Dokuments geschieht mit dem ersten Befehl, welcher eine Dokumentklasse (hier: *scrartcl*) und die Schriftgröße (hier: *11 pt*) bestimmt. In der Regel stehen zumindest (ohne Laden zusätzlicher Programmpakete) die Schriftgrößen 10 pt, 11 pt und 12 pt zur Verfügung. Das Papierformat wird hier mit der Option *a4paper* (für das A4-Format) angegeben.

In der *document*-Umgebung, zwischen *\begin{document}* und *\end{document}*, steht der Textinhalt, hier: Hallo Welt!

Im Text erfolgt die Silbentrennung am Zeilenende automatisch. Die Strukturierung des Texts wird durch eingebettete Befehle vorgenommen. In obigem Beispiel fügt der Befehl \, ein kurzes Leerzeichen (Lücke) ein.

Umgebungen sind bestimmte zusammenhängende Bereiche, welche Text und Befehle enthalten. Beispiele sind eine Aufzählung, aber auch die gesamte *document*-Umgebung. Eine Umgebung (*environment*) wird mit *\begin{...}* eingeleitet und mit *\end{...}* abgeschlossen, wobei für ... der Name der Umgebung eingesetzt wird.

Gleitobjekte sind Umgebungen, die nicht unbedingt dort gedruckt werden müssen, wo sie im Quellcode beschrieben werden. Gleitobjekte sind insbesondere Abbildungen (*figure*-Umgebungen) und Tabellen (*table*-Umgebungen), die auch eine Legende (Beschriftung) haben können. Das Erscheinungsbild einer Seite ist gefälliger, wenn man LaTeX erlaubt, die Gleitobjekte automatisch zu platzieren. Die Reihenfolge der Gleitobjekte bleibt dabei natürlich unverändert.

Präambel ist der Bereich eines Dokuments, der vor der *document*-Umgebung steht. Die Präambel (*preamble*) enthält den Befehl *documentclass* und danach weitere Befehle, die zum Beispiel besondere Programmteile (Pakete) laden (siehe unten).

Schalter sind Befehle, die solange wirken können, bis sie durch einen neuen Befehl aufgehoben oder geändert werden. Es ist jedoch meistens besser, den Textbereich, auf den ein Schalter (*switch*) wirkt, von vornherein zu beschränken. Dazu dient eine Umgebung oder die Einfassung von Schalter und Text in einen (durch geschweifte Klammern begrenzten) Block.

Befehle definieren kann sinnvoll sein, um häufig gebrauchte, lange Befehlsfolgen abzukürzen. Mit dem Befehl *newcommand*∗ , gefolgt von zwei Pflichtargumenten, wird in der Präambel ein neuer Befehl definiert, vorausgesetzt, dass es einen gleichnamigen Befehl noch nicht gibt. Wenn zum Beispiel das Wort *Freiheit*, mit kleinerer Schriftgröße und kursiv geschrieben, mehrfach im Text vorkommt, kann man statt der langen Befehlsfolge *begin{small}**textit{Freiheit}**end{small}* den kurzen Befehl *F* schreiben, nachdem dieser mit

```
\newcommand*{\F}{\begin{small}\textit{Freiheit}\end{small}\xspace}%
% der Befehl ohne Argumente \F gibt das Wort Freiheit formatiert aus
```

definiert wurde. Der Name des neuen Befehls, *F* , ist das erste Pflichtargument und die neue Befehlsfolge ist das zweite Pflichtargument von *newcommand*∗.

Jeder selbst definierte Befehl sollte einen kurzen, doch verständlichen Namen bekommen. Dieser darf nur (kleine oder große) Buchstaben enthalten! Der Zweck des neuen Befehls sollte in einer Kommentarzeile erläutert werden. Alle Zeilen der Befehlsdefinition sollten mit einem % (Kommentarzeichen) abgeschlossen werden, um zu verhindern, dass unerwünschte Leerzeichen (durch Zeilenumbrüche) in der Ausgabe auftauchen.

In der Befehlsfolge des zweiten Pflichtarguments steht am Ende noch *xspace* . Dieser Befehl ist nur dann verfügbar, wenn vorher in der Präambel das Paket *xspace* mittels des Befehls *usepackage{xspace}* geladen wurde (siehe unten). Er stellt sicher, dass ein Leerzeichen, welches möglicherweise nach dem neuen Befehl *F* geschrieben wird, nicht verschwindet.

Falls man, nachdem der Text geschrieben wurde, die *Freiheit* stets groß schreiben möchte, muss man nur den Befehl *F* ändern und aus jeder *Freiheit* wird *Freiheit*.

Wenn man Zeichenketten eines bestimmten logischen Typs, zum Beispiel Tiernamen, in einem Text durch einen Schriftstil auszeichnen will, ist es sinnvoll, einen Befehl zu definieren, der eine Zeichenkette als Argument entgegennimmt und darstellt. Dafür muss die Folge der Argumente des Befehls *newcommand*∗ ergänzt werden:

```
\newcommand*{\tir}[1]{\begin{large}\textbf{#1}\end{large}\xspace}%
% der Befehl \tir{...} mit 1 Argument gibt Tiernamen formatiert aus
```

In eckigen Klammern, nach dem ersten Pflichtargument mit dem Namen des neuen Befehls, steht als optionales Argument die Anzahl der Argumente, welche der neu definierte Befehl erhalten soll, hier: 1. Im zweiten Pflichtargument, mit der neue Befehlsfolge, wird das Argument, welches der neu definierte Befehl erhalten soll, durch #1 vertreten. Gäbe es im neu definierten Befehl zwei Argumente, würde das zweite durch #2 vertreten, und so weiter. Oben definierter Befehl kann folgendermaßen angewendet werden.

```
Das \tir{Schwein} ist groß und fett.
```

Das **Schwein** ist groß und fett.

Entscheidet man sich später, alle Tiernamen lieber klein und kursiv zu schreiben, braucht man nur einmal den *tir*-Befehl zu ändern. *Schwein* gehabt!

Es gibt Befehle, die in einer sogenannten *math*-Umgebung ausgeführt werden müssen (siehe Abschnitt 5.1 auf Seite 67), zum Beispiel die Darstellung des griechischen Buchstabens Rho ρ. Wenn man einen neuen Befehl definiert, der ein ρ zeigt, kann man mithilfe des Befehls *ensuremath* erreichen, dass der neue Befehl sowohl inner- als auch außerhalb einer *math*-Umgebung ausgeführt wird.

```
\newcommand*{\R}{\ensuremath{\rho}}%
% der Befehl \R zeigt das kleine Rho
...
$\R$ mit und ohne math-Umgebung \R
```

ρ mit und ohne math-Umgebung ρ

Das Beispiel zur Darstellung eines ρ ist so einfach, dass man keinen neuen Befehl bräuchte, aber nach dem Vorbild kann man schwierigere Befehlsketten einbauen.

Mit dem Befehl *renewcommand{...}{...}* kann man einen vorhandenen Befehl neu definieren, wie die Beispiele auf den Seiten 20, 30, 50 und 109 zeigen. Wenn in der neuen Definition der alte Befehl aufgerufen wird, muss man zunächst dem Befehl, zum Beispiel *emph*, einen zweiten Namen geben, beispielsweise *tmpemph*.

1.3 Dokumentklassen und Pakete, Beispiel

Dokumentklassen erzeugen eine Grundeinstellung für das LaTeX-Dokument. Die Vorgabe eines Formats mithilfe einer Dokumentklasse (*document class*) befreit uns von der mühsamen Festlegung gestalterischer Einzelheiten und das Augenmerk kann sich aufs Wesentliche, den Inhalt, richten.

Eine Dokumentklasse wird mit dem Befehl *documentclass[...]{...}* festgelegt. Es gibt ältere Standard-Dokumentklassen, die in den USA entwickelt wurden und neuere, sogenannte KOMA-Script-Klassen (deren Name mit *scr* beginnt), welche die in Deutschland übliche Schriftgestaltung unterstützen. Häufig verwendete Dokumentklassen sind *article* beziehungsweise *scrartcl* (kurzer Bericht), *report* beziehungsweise *scrreprt* (langer Bericht) und *book* beziehungsweise *scrbook* (Buch) sowie *beamer* (Präsentation). Eine Entsprechung für die *beamer*-Klasse gibt es bei KOMA-Script nicht, aber man kann die Klasse *scrartcl* auch für Präsentationen einsetzen. Die gewünschte Dokumentklasse wird als Pflichtargument des *documentclass*-Befehls in geschweiften Klammern angegeben.

Mit optionalen Argumenten (im eckigen Klammerpaar) kann man die Gestaltung beeinflussen. Ein optionales Argument gibt das Papierformat an, in Deutschland ist

das meistens *a4paper*, ein weiteres die Schriftgröße (in der Regel 10 pt, 11 pt oder 12 pt). In den meisten Büchern beginnen Absätze mit einer Einrückung. Alternativ können Absätze durch einen vertikalen Abstand getrennt werden. Dieses Verhalten lässt sich in den KOMA-Script-Klassen mit der Option *parskip=half* (halbzeiliger Abstand) oder *parskip=full* (ganzzeiliger Abstand) einstellen (siehe Abschnitt 1.4 auf Seite 17). Das optionale Argument *twocolumn* würde bewirken, dass die Seiten zwei Spalten bekommen. Will man Vorder- und Rückseite eines Blatts nutzen (zweiseitiger Druck), kann man *twoside* wählen. Beim zweiseitigen Druck werden unter anderem die inneren und äußeren Seitenränder angepasst.

In den Dokumentklassen *article*/*scrartcl* und *report*/*scrreprt* wird (ohne das optionale Argument *twoside*) für einseitigen Druck formatiert, in *book*/*scrbook* für zweiseitigen Druck. Die Klassen *report*/*scrreprt* und *book*/*scrbook* schreiben den Titel des Werks auf eine eigene Seite, bei *article*/*scrartcl* beginnt der weitere Text nach dem Titel auf der gleichen Seite. Ohne besondere Festlegung der Schriftgröße verwenden *article*, *report* und *book* die Größe 10 pt, während *scrartcl*, *scrreprt*, *scrbook* und *beamer* 11 pt nehmen.

Mehr über die KOMA-Script-Klassen gibt es in der Anleitung von Markus Kohm. Nützlich ist auch die Kurzanleitung für Abschlussarbeiten mit Latex und KOMA-Script von Elke Schubert und Marion Lammarsch.

Pakete sind zusätzliche Teile des LaTeX-Systems, welche erst zur Verfügung stehen, wenn sie geladen wurden. Dies geschieht mit dem Befehl *\usepackage[...]{...}*. Pflichtargument ist der Name des zu ladenden Pakets. Optionale Argumente, die das Paket (*package*) beeinflussen, können in den eckigen Klammern gesetzt werden.

Ohne optionale Argumente können mehrere Paketnamen, durch Kommata getrennt, als Pflichtargumente in einem geschweiften Klammerpaar stehen.

Anpassung deutscher Texte geschieht mit folgenden Befehlszeilen. Damit werden Formatvorgaben nach deutschen Gepflogenheiten gesetzt. Die drei Befehlszeilen sollten darum für deutsche Texte immer in dieser Weise eingefügt werden.

```
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

Das optionale Argument *ngerman* veranlasst die Verwendung der neuen deutschen Rechtschreibung, zum Beispiel bei der Silbentrennung. Die Kodierung mit UTF-8 erlaubt die direkte Eingabe deutscher Umlaute und so weiter. Mit *T1* werden westeuropäische Fonts (Schriften) verwendet.

Weitere Pakete werden wie in folgenden Beispielen geladen

```
\usepackage{graphicx}      % Einbindung von Bildern aus Bilddateien
\usepackage{xspace}        % Einsetzen eines Leerzeichens bei Bedarf
\usepackage{amsmath,amsthm,amsfonts} % Erweiterungen für Mathematik
\usepackage[normalem]{ulem}  % Unterstreichen von Text, emph kursiv
\usepackage[left=3cm,right=3cm]{geometry}  % Anpassung Seitenränder
```

LaTeX-Pakete und ihre Dokumentationen werden im CTAN (Comprehensive TeX Archive Network) gesammelt. Die Webseite ist https://www.ctan.org/

Hinweise auf nützliche Pakete gibt es in dem kurzen Manuskript *A First Set of LaTeXPackages* von Jim Hefferon.

Wenn ein Paket, beispielsweise `fontenc`, einschließlich Dokumentation installiert wurde, gibt es in der Regel eine PDF-Datei dafür im Unterverzeichnis `doc` des Verzeichnisses `texmf`. Der genaue Pfadname hängt vom Betriebssystem ab; er kann in einem Linux-System beispielsweise `/usr/share/texmf/latex/base/encguide.pdf` lauten. In einem Terminal oder einer Konsole kann man dann meistens mit

```
texdoc fontenc
```

die Dokumentation zum Paket `fontenc`, oder entsprechend zu einem anderen Paket, im PDF-Anzeigeprogramm des Betriebssystems anzeigen lassen.

Beispiel für die Struktur eines LaTeX-Dokuments ist folgender Quellcode. Die Einrückungen erhöhen seine Lesbarkeit, beeinflussen aber nicht das Druckbild.

```
% Präambel
\documentclass[a4paper, 12pt, parskip=half]{scrartcl}
    \usepackage[ngerman]{babel}
    \usepackage[utf8]{inputenc}
    \usepackage[T1]{fontenc}
    \usepackage[left=2cm,right=2cm,top=1.6cm,bottom=3cm]{geometry}
    \usepackage{booktabs}
    \usepackage{marvosym}
    \usepackage{xspace}

\newcommand*{\F}{\begin{Large}\textit{Freiheit}\end{Large}\xspace}%
% der Befehl ohne Argumente \F gibt das Wort Freiheit formatiert aus

\title{So heißt mein Werk}
    \author{Karl Mustermann\thanks{E-Mail: kamu@provider.de}}
    \date{Heute ist der \today}

% document-Umgebung
\begin{document}

\maketitle
\tableofcontents

\section{Erster Abschnitt}
    \subsection{ein Unterabschnitt}
                Text kann zum Beispiel \textbf{fett}
                oder \textit{kurvig} dargestellt werden.
                \par\bigskip
                Einsteins Formel: $E = m \cdot c^2$

        \subsubsection{ein Unter-Unterabschnitt}
                \begin{tabular}{cccc}
                    \toprule
                    Cross & Frowny & Smiley & Heart \\
                    \Cross & \Frowny & \Smiley & \Heart\\
                    \bottomrule
                \end{tabular}

\section{Zweiter Abschnitt}
                Text kann auch \begin{Large}Groß\end{Large} und
                \begin{scriptsize}Klein\end{scriptsize} erscheinen.
                \par\bigskip\bigskip
                Die \F jedenfalls sollte groß geschrieben werden!

\end{document}
```

So heißt mein Werk

Karl Mustermann*

Heute ist der 22. September 2020

Inhaltsverzeichnis

1 Erster Abschnitt **1**
1.1 ein Unterabschnitt 1
1.1.1 ein Unter-Unterabschnitt 1

2 Zweiter Abschnitt **1**

1 Erster Abschnitt

1.1 ein Unterabschnitt

Text kann zum Beispiel **fett** oder *kurvig* dargestellt werden.

Einsteins Formel: $E = m \cdot c^2$

1.1.1 ein Unter-Unterabschnitt

Cross	Frowny	Smiley	Heart
✝	☹	☺	♡

2 Zweiter Abschnitt

Text kann auch Groß und Klein erscheinen.

Die *Freiheit* jedenfalls sollte groß geschrieben werden!

*E-Mail: kamu@provider.de

Abbildung 1: Druckbild des Beispiels eines einfachen LaTeX-Dokuments

1.4 Gliederung (Titel, Textabschnitte und Absätze)

Ein einfaches Dokument ist als Ganzes in drei Teile gegliedert, nämlich Titelseite, Inhaltsverzeichnis und Text. Letzterer wird in Abschnitte unterteilt. Das Inhaltsverzeichnis wird, abhängig von den Textabschnitten, durch Kompilierung automatisch erzeugt (siehe Abschnitt 2.1 auf Seite 19).

Titelseiten werden entweder mit dem Befehl *maketitle* aus vorher bereitgestellter Information erzeugt oder in einer *titlepage*-Umgebung Schritt für Schritt zusammengebaut. Wir betrachten nur die erste Methode.

In den älteren Standard-Dokumentklassen *article*, *report* und *book* können folgende Befehle in der Präambel Information für die Titelseite bereitstellen.

\title Argument: Titel, darf Formatierungsbefehle enthalten (zentriert)

\author Argument: Autor, zwei Autoren werden durch `\and` getrennt (zentriert)

\thanks Argument: Fußnote, muss im Argument von `\author` stehen

\date Argument: Datum, `\today` liefert das aktuelle Datum (zentriert)

Ohne den *date*-Befehl wird das aktuelle Datum ausgegeben. Lässt man aber das Argument weg, wird kein Datum angezeigt.

Die Titelseite wird erst in der *document*-Umgebung mit dem Befehl

```
\maketitle
```

ausgegeben. In folgendem Beispiel stehen die ersten drei Befehle in der Präambel, der letzte Befehl jedoch erst in der *document*-Umgebung:

```
\title{Die Thermodynamik der Tanzfläche}
\author{Alfred H. Gitter\thanks{E-Mail: alfred.gitter@eah-jena.de}}
\date{21. März 1999}
% ...
\maketitle
```

Im Beispiel von Abschnitt 1.3 (Seite 13) wurde der Titel in ähnlicher Weise erzeugt. Man kann die Information für die Titelseite auch erst in der *document*-Umgebung bereitstellen, jedenfalls vor dem *maketitle*-Befehl. Übersichtlicher ist es jedoch, dies (wie oben) bereits in der Präambel zu tun.

In den Dokumentklassen *report* und *book* wird eine ganze Titelseite gedruckt (ohne Ausgabe der Seitenzahl), während in *article* nach Titel, Autor und Datum der weitere Text auf der gleichen Seite anschließt.

In den KOMA-Script-Klassen *scrartcl*, *scrreprt* und *scrbook* gibt es, zusätzlich zu oben genannten, noch weitere Befehle, die Information für die Titelseite bereitstellen.

\titlehead Argument: beliebiger Text oberhalb des Titels (Blocksatz)

\subject Argument: Art oder Gegenstand des Dokuments (zentriert)

\subtitle Argument: Untertitel (zentriert)

\publishers Argument: Herausgeber oder Betreuer (zentriert)

Die Titelseite einer Bachelorarbeit könnte zum Beispiel so definiert werden:

```
\titlehead{Universität der Rolling Stones \hfill Sommersemester 1995}
\subject{Bachelorarbeit}
\title{Petrokinese}
\subtitle{Die Strafe der Rekursion}
\author{Sisy Phos\thanks{E-Mail: sissy@hades.gr}\\aus Korinth}
\date{13. Juli 1995}
\publishers{Betreuer: Prof. Dr. Albert Camus}
\dedication{Unermüdlich wie mein Eifer ist die Liebe zu Merope}
% ...
\maketitle
```

Mit dem *dedication*-Befehl wird in der Präambel der Text einer Widmung bereitgestellt, die nach der Titelseite in der Mitte einer Seite gedruckt wird.

In den Dokumentklassen *scrreprt* und *scrbook* wird eine ganze Titelseite gedruckt (ohne Ausgabe der Seitenzahl), während in *scrartcl* nach Titel, Autor und Datum der weitere Text auf der gleichen Seite anschließt.

Ein Abstract ist eine kurze Zusammenfassung, die bei Artikeln und Berichten oft vor den Haupttext gesetzt wird. In den Dokumentklassen *article*/*scrartcl* und *report*/*scrreprt*, nicht jedoch in *book*/*scrbook* gibt es dafür die Umgebung *abstract*. In den Dokumentklassen *article* und *report* erhält der Abstract eine Überschrift. Bei deutschen Texten (*babel*-Option *ngerman*) ist das standardmäßig „Zusammenfassung“. In den Dokumentklassen *scrartcl* und *scrreprt* erhält der Abstract nur dann eine Überschrift, wenn beim Dokumentklassenbefehl die Option `abstract=true` gewählt wurde. Man kann die Standard-Überschrift des Abstracts in der document-Umgebung (nach `\begindocument`) ändern, beispielsweise so:

```
\documentclass[a4paper,12pt,abstract=true]{scrartcl}
...
\begin{document}
\renewcommand\abstractname{Meine Zusammenfassungsüberschrift}
\begin{abstract}
\noindent blabla blabla
\end{abstract}
...
```

Textabschnitte sind die hierarchisch gegliederten Teile des Texts. Die Gliederung eines Textes erfolgt bei den Dokumentklassen *book*/*scrbook* und *report*/*scrreport* mit den Hierarchie-Ebenen (und entsprechenden Befehlen) *chapter* (Kapitel, Hierarchieebene 0), *section* (Abschnitt, Ebene 1), *subsection* (Unterabschnitt, Ebene 2) und *subsubsection* (Unterunterabschnitt, Ebene 3). Bei *article*/*scrartcl* fehlt die Kapitel-Ebene. Die Gliederung eines *article*/*scrartcl*-Dokuments in Abschnitte könnte (vereinfacht) ungefähr so aussehen:

```
\section{Überschrift des ersten Abschnitts}
\subsection{Überschrift des Unterabschnitts 1.1}
\subsection{Überschrift des Unterabschnitts 1.2}
\subsubsection{Überschrift des Unter-Unterabschnitts 1.2.1}
```

```
\section{Überschrift des zweiten Abschnitts}
```

Die Überschriften der Textabschnitte werden bis zu einer unteren Hierarchie-Ebene automatisch nummeriert. Welches die unterste, nummerierte Hierarchie-Ebene ist, hängt von der Dokumentenklasse ab. Man kann dies aber ändern. Der Befehl

```
\setcounter{secnumdepth}{-2}
```

in der Präambel bewirkt, dass keine Überschriften nummeriert werden. Setzt man in obigem Befehl statt -2 die Nummer einer Hierarchieebene ein, wird bis zu dieser nummeriert.

Soll ein bestimmter einzelner Abschnitt (oder Unter- oder Unterunterabschnitt) nicht nummeriert werden, fügt man einen Stern $*$ an den Gliederungsbefehl an, zum Beispiel: *\section*{Überschrift}*. Der Abschnitt mit dem Stern wird nicht im Inhaltsverzeichnis (siehe unten) aufgeführt.

Für einen Anhang, der nach dem Haupttext folgt, gibt es den Befehl *appendix*. Alle Abschnitte, die nach diesem Befehl folgen, werden als Anhangsabschnitte angesehen und erhalten automatisch eine andere Nummerierung als die Abschnitte des Haupttexts:

```
\appendix
\section{erster Anhangsabschnitt}
\section{zweiter Anhangsabschnitt}
```

Die Untergliederung eines Unter-Unterabschnitts (*subsubsection*) kann mit einem *paragraph* genannten Textbereich erfolgen, welcher nicht nummeriert und nicht ins Inhaltsverzeichnis aufgenommen wird. Eine mögliche Überschrift, die als Parameter (in geschweiften Klammern) übergeben wird, erscheint fett am Anfang der ersten Zeile. Ohne Parameter (leere geschweifte Klammern) erhält der *paragraph* keine Überschrift.

Bei Verwendung der KOMA-Script-Klassen kann die Schrift für Gliederungselemente mit dem Befehl *\setkomafont* geändert oder mit dem Befehl *\addtokomafont* erweitert werden. Der Begriff *disposition* bezieht sich auf alle Gliederungsüberschriften. Gleichbedeutend mit *disposition* ist die Bezeichnung *sectioning*.

Standardmäßig werden werden die Überschriften in serifenloser Schrift gesetzt (siehe Seite 47). Will man dies nicht, kann man in der Präambel mit

```
\setkomafont{disposition}{\normalfont\normalcolor\bfseries}
```

für alle Überschriften Serifenschrift festlegen.

Absätze können im Quellcode durch Einfügen einer Leerzeile oder mit dem Befehl *\par* erzeugt werden. Ein einfacher Zeilenumbruch genügt nicht.

Normalerweise beginnt ein Absatz mit einem (horizontalen) Einzug am Beginn der ersten Zeile, ohne vertikalen Abstand vom vorhergehenden Absatz. Ausgenommen von dieser Regel ist der erste Absatz nach einer Überschrift oder einer anderen Textunterbrechung (Leerzeile, Tabelle, Aufzählung und so weiter). Dieser beginnt ohne Einzug (stumpf). Die Länge des Einzugs der ersten Absatzzeile beträgt ein Geviert. (Geviert ist, abhängig vom Kontext, eine typographische Längen- oder Flächeneinheit.) Die letzte Zeile eines Absatzes kann sich bis zum rechten Rand erstrecken.

Sollen stattdessen Absätze durch einen vertikalen Abstand getrennt werden, kann man, bei den Dokumentklassen von KOMA-Script, nach dem Befehl *documentclass*

in eckigen Klammern die Option *parskip=half* oder *parskip=full*, für halbzeiligen beziehungsweise ganzzeiligen Abstand, angeben. Zu beachten ist bei *parskip=half* und *parskip=full*, dass die letzte Zeile eines Absatzes mindestens um ein Geviert vom rechten Rand eingerückt wird. Damit wird das Absatzende besser erkennbar. Die Klassenoption *parskip* ist eine Besonderheit von KOMA-Script. Bei den Standard-Dokumentklassen benötigt man für den Absatzabstand das Paket *parskip*.

Die Einrückung am rechten Rand der letzten Zeile eines Absatzes tritt bei den Optionen *parskip=half* und *parskip=full* auch mit einzeiligen Absätzen auf. Umgebungen, die einen oder mehrere Kästen in einem einzeiligen Absatz unterbringen sollen (zum Beispiel Minipage, siehe unten), erzeugen dann eine Warnung (*overfull \hbox*). Man kann diese bei den Dokumentklassen von KOMA-Script verhindern, indem die Option *parskip* vorübergehend ausgeschaltet wird (siehe das Beispiel zur Umgebung *tcbraster* auf Seite 42 in Abschnitt 3.2).

Die Absatzauszeichnung sollte im gesamten Dokument entweder mit Einzug oder mit Abstand erfolgen. Die gleichzeitige Verwendung der beiden wäre übertrieben. Bei kleiner Zeilenlänge (zum Beispiel in mehrspaltigen Texten) ist der Absatzabstand besser, da er eine weitere Verkürzung der ersten Zeile verhindert. Ansonsten wird meistens der Absatzeinzug bevorzugt.

2 Besondere Teilbereiche

2.1 Verzeichnisse (Inhalt, Abk., Abb., Literatur)

Das Inhaltsverzeichnis wird dort eingefügt, wo die Befehlszeile

```
\tableofcontents
```

steht. Welche Hierarchie-Ebenen des gegliederten Textes bei der Erstellung eines Inhaltsverzeichnisses einbezogen werden, hängt, sofern die diesbezügliche Voreinstellung nicht geändert wurde, von der Dokumentklasse ab. Mit *scrartcl* geht es bis zu den *subsubsections* (Unterunterabschnitten, Hierarchieebene 3) und mit *scrreport* bis zu den *subsections* (Unterabschnitten, Hierarchieebene 2). Man kann auch dies ändern. Der Befehl

```
\setcounter{tocdepth}{1}
```

in der Präambel bewirkt, dass nur die Kapitel und Abschnitte, aber keine Unterabschnitte (oder feinere Glieder) in das Inhaltsverzeichnis aufgenommen werden. Ersetzt man in obigem Beispiel die 1 durch eine andere Nummer, reicht das Inhaltsverzeichnis bis zur entsprechenden Hierarchieebene.

Die Überschrift *Inhaltsverzeichnis* kann in den KOMA-Script-Dokumentklassen zum Beispiel mit der Befehlszeile

```
\renewcaptionname{ngerman}{\contentsname}{Überschrift}
```

in der Präambel geändert werden, wenn das Paket *babel* mit der Option *ngerman* geladen wurde. Auch in anderen Dokumentklassen gelingt die Umbenennung mit

```
\addto\captionsngerman{\renewcommand{\contentsname}{Überschrift}}
```

Das Paket *tocloft* von Peter Wilson und Will Robertson ermöglicht die Anpassung der Typographie von Verzeichnissen und die Erzeugung neuer Verzeichnisse.

In den KOMA-Script-Klassen wird im Inhaltsverzeichnis die Seitenzahl von Kapiteln fett gedruckt. Will man stattdessen normale Schrift, kann man in der Präambel

```
\usepackage{tocloft}
\renewcommand\cftchappagefont{\normalfont}
```

einfügen. Damit wird auch ein Fehler vermieden, der in den KOMA-Script-Klassen beim Setzen dreistelliger Seitenzahlen in fetter Schrift (für Kapitelanfänge) im Inhaltsverzeichnis auftreten kann und eine schwer zu deutende Warnung bewirkt.

Jedes Mal, wenn man das Dokument mindestens zweimal kompiliert, wird ein aktuelles Inhaltsverzeichnis automatisch erzeugt. Aufgrund der Abhängigkeit des Inhaltsverzeichnisses von der Formatierung des gesamten Dokuments genügt die einmalige Kompilierung nicht.

Das Abkürzungsverzeichnis kann zum Beispiel mit dem Paket *acronym* von Heiko Oberdiek und Tobias Oetiker erstellt werden.

Das Abbildungsverzeichnis wird mit der Befehlszeile

```
\listoffigures
```

eingefügt. Es enthält die Legende (oder, soweit vorhanden, dessen Kurzfassung) für alle Bilder, die in einer *figure*-Umgebung mit *caption*-Befehl (siehe Seite 23) stehen.

Das Literaturverzeichnis ist ein Teil jeder wissenschaftlichen Arbeit. Für seine Erstellung gibt es drei wichtige Methoden:

1. Umgebung *thebibliography* ohne weiteres
2. Paket *natbib* und externes Programm *bibtex*
3. Paket *biblatex* und externes Programm *biber*

Statt *natbib* werden in geisteswissenschaftlichen Arbeiten andere Pakete benutzt. Experten behaupten oft, Methode 1 sei nicht effektiv. Für die große Mehrheit der Anwender ist aber die einfache Methode 1 zu empfehlen! Nur wer tatsächlich mehrere wissenschaftliche Arbeiten über das gleiche Thema schreibt, braucht Methode 2 oder 3. Wir betrachten daher in diesem Abschnitt nur Methode 1 und später (in Abschnitt 7.5 auf Seite 123) Methode 3.

Das Literaturverzeichnis kann am einfachsten mit der *thebibliography*-Umgebung erstellt werden, die in der Regel am Ende des Dokuments steht. Normalerweise lautet die Überschrift des Literaturverzeichnisses *Literatur*. Man kann dies jedoch, vor der *thebibliography*-Umgebung, mit einem Befehl ändern.

```
\renewcommand{\bibname}{Meine Bücherliste}
```

Man beachte, dass in einigen Dokumentklassen *refname* statt *bibname* in diesem Befehl stehen muss.

Wenn man nichts anderes wünscht, lässt man die Einträge automatisch nummerieren. Der *begin{thebibliography}*-Befehl erwartet in einem weiteren geschweiften Klammerpaar eine Zeichenkette; bei automatischer Nummerierung wird hier die maximale Zahl der Einträge eingetragen, zum Beispiel 99 oder 999.

Jeder Eintrag beginnt mit dem Befehl *bibitem*, der in geschweiften Klammern eine kurze Zeichenkette erhält. Diese ist der Schlüssel, ein eindeutiger Name, mit dessen Hilfe der Eintrag im Text zitiert wird. Der Schlüssel sollte erkennbar zum Eintrag passen. Er könnte aus dem abgekürzten Namen des Autors (drei Kleinbuchstaben), dem Publikationsjahr (zweistellig) und einem Kleinbuchstaben, der die Eindeutigkeit sicherstellt (beginnend mit a), bestehen. Damit ergibt sich folgendes Beispiel.

```
\begin{thebibliography}{99}
\bibitem{glu08a} Hans Glück, Der Elefant im Heuhaufen,
 Verlach, 6. Aufl., Buxtehude 2008.
\bibitem{glu08b} Hans Glück, Der Spatz in der Hand macht
 noch keinen Sommer, Verlach, 8. Aufl., Buxtehude 2008.
\bibitem{eol17a} Ethan Ol, Scotch your drinking problems,
 PUBlisher, Dufftown 2017.
\end{thebibliography}
```

Im Text werden Literatureinträge mit dem Befehl *cite* zitiert, welcher in geschweiften Klammern einen oder mehrere (durch Kommata getrennte) Schlüssel erhält.

```
Sprichwörter sind Glücks Sache \cite{glu08a,glu08b}.
```

Zusätzlich kann, mit einem optionalen Parameter in eckigen Klammern, eine Information zum Literatureintrag, beispielsweise eine Seitenzahl, ergänzt werden.

```
Spiritual balance is a whisky in each hand \cite[Seite 42]{eol17a}.
```

Obige Beispiele mit Zitaten erscheinen gedruckt etwa so:

> Sprichwörter sind Glücks Sache [1, 2].
>
> Spiritual balance is a whisky in each hand [3, Seite 42].

Zitate im Text können anders formatiert werden, wenn das Paket *cite* von Donald Arseneau geladen wurde. Das Paket *tocbibind* von Peter Wilson erlaubt es, das Literaturverzeichnis im Inhaltsverzeichnis aufzuführen (mit oder ohne Nummerierung).

2.2 Fuß- und Randnoten, Querverweise, Hyperlinks

Fußnoten werden mit dem Befehl *\footnote* an eine Textstelle[1] gesetzt:

```
Die Flensburger Brauerei wurde 1888, im sogenannten
Dreikaiserjahr\footnote{1888 regierten nacheinander Wilhelm I.,
Friedrich III. und Wilhelm II.}, gegründet.
```

Die Fußnote wird automatisch nummeriert, ihr Text an das Ende der Seite (oder der folgenden Seite) geschrieben. In den Dokumentklassen *article* und *scrartcl* wird das ganze Dokument durchgehend nummeriert, aber in den Dokumentklassen *report*, *scrreprt*, *book* und *scrbook* beginnt die Nummerierung in jedem Kapitel neu mit 1.

Randnoten werden in gleicher Weise (jedoch nicht nummeriert) mit dem Befehl *\marginpar* erstellt. Sie sollten aufgrund der geringen Randbreite kurz gehalten sein. Leider ist die automatische Silbentrennung beim ersten Wort des Randnotentexts ausgeschaltet. Um dennoch eine Silbentrennung zu erreichen, stellt man den Befehl *\hspace{0pt}* vor das erste Wort. Für eine linksbündige Randnote (statt Blocksatz) stellt man *\raggedright* an den Anfang. Zum Beispiel wird mit

```
\marginpar{ \raggedright\begin{large}\hspace{0pt}
Randnoten\end{large} sind keine Schand\-boten\,! }
```

die nebenstehende Randnote gebildet.

Randnoten sind keine Schandboten!

Querverweise verweisen auf eine Zeichenkette im Text oder eine Abbildung oder Gleichung. Ein Querverweis auf eine Zeichenkette wird mit einer Marke ermöglicht, indem man direkt vor der Zeichenkette den Befehl *\label* mit einem willkürlich gewählten Parameter setzt. Für Abbildungen wird eine Marke (*Label*) beim Einfügen des Bilds in eine *figure*-Umgebung erzeugt (siehe Seite 23), für Gleichungen entsprechend (siehe Seite 76). In diesen und anderen Umgebungen sollte der Befehl *\label* ans Ende gesetzt werden.

Die Befehle *\ref* beziehungsweise *\pageref* und der Parametername erlauben es nun, auf Abschnitt beziehungsweise Seitenzahl der Zeichenkette (oder des Bildes oder der Gleichung) hinzuweisen. Beispiel:

[1] 1888 regierten nacheinander Wilhelm I., Friedrich III. und Wilhelm II.

```
\label{s1} Die Sonne ist ein Stern in der Milchstraße. Blablabla ...
Über Sterne sprachen wir in Abschnitt \ref{s1}, Seite \pageref{s1}.
```

Eventuell muss man das Dokument mehrfach kompilieren (siehe Abschnitt 2.1 auf Seite 19), damit die Querverweise im Text erscheinen.

Hyperlinks sind Querverweise in einem PDF-Dokument, welches nicht als Ausdruck, sondern (mit einem PDF-Betrachterprogramm) auf dem Bildschirm gelesen wird. Nach Anklicken mit der Maus wird der Leser direkt zum Verweisziel geführt. Wurde das Paket *hyperref* von Sebastian Rahtz und Heiko Oberdiek geladen, werden die Querverweise im LaTeX-Dokument zu Hyperlinks. Da *hyperref* einige Einstellungen anderer Pakete ändert, sollte es am Ende der Präambel (nach den anderen Paketen) eingebunden werden.

So kann man zu einer beliebigen (mit \label bezeichneten) Marke springen:

```
\usepackage{hyperref}
...
Die Maus\label{maus} ist ein unwillkommenes Haustier.
\par\medskip
blablabla
\newpage
Nützlich ist die Maus (siehe Seite \pageref{maus}) am Computer.
```

Hyperlinks werden auf dem Bildschirm (nicht dagegen im Papierausdruck) standardmäßig durch eine farbige Box gekennzeichnet. Man kann aber beim Laden des Pakets durch Angabe von Optionen das Aussehen der Hyperlinks selbst bestimmen.

Außerdem kann man mit einem Hyperlink auf eine Webseite verweisen. Mit

```
The \href{https://www.nationalbeefassociation.com/}{NBA}: no sports !
```

wird der Text *NBA* in normaler Schrift gezeigt und das Verweisziel (URL) erscheint nicht. Andererseits wird mit

```
Website of the NBA: \url{https://www.nationalbeefassociation.com/}
```

das Verweisziel in nichtproportionaler Schrift gezeigt.

Mithilfe des Pakets *hyperref* kann man auch Information zu Autor und Titel des von LaTeX erzeugten Dokuments im PDF-Dokument speichern. Zum Beispiel:

```
...
\usepackage{hyperref}
\hypersetup{ pdfauthor={A. Aguecheek},pdftitle={Beef up your wit !} }
...
```

Der Dokumententitel wird in der Titelleiste des PDF-Betrachterprogramms gezeigt.

2.3 Bilder, Tabellen, Balkendiagramme, Kommentare

Bilder können in das Dolument eingebettet werden, wenn sie in einer Datei vom Typ PNG, JPG oder PDF vorliegen. Da der Bildinhalt nicht in das LaTeX-Dokument übertagen wird, muss die Bilddatei bei jeder Kompilierung (siehe Abschnitt 2.1 auf Seite 19) vorhanden sein.

Der entscheidende Befehl für das Einfügen des Bildes aus einer Datei lautet

Abbildung 2: Prostmondäne Kunst: „Handschrift“

```
\includegraphics[optionen]{datei}
```

Dabei sind *datei* der Pfad zur Bilddatei und *optionen* Angaben zur Größe des Bildes oder ähnliches. In den *optionen* kann mit *scale=x* die Bildgröße auf das x-fache vergrößert beziehungsweise verkleinert werden, wobei für x der entsprechende Zahlenwert (zum Beispiel *0.5*) eingesetzt wird. Stattdessen kann mit *width=x* oder *height=x* die gewünschte Breite oder Höhe des Bildes vorgegeben werden, wobei x eine Länge (zum Beispiel *8cm*) ist. Bei der optionalen Größenänderung wird das Seitenverhältnis des Bildes beibehalten. Eine Verzerrung des Bildes durch gleichzeitige Vorgabe von Breite und Höhe ist nicht möglich. Es gibt weitere Optionen zur Veränderung des Bildes, zum Beispiel die Möglichkeiten des Drehens und Zuschneidens, aber derartiges kann besser mit einem Bildbearbeitungsprogramm vor der Einbindung in ein LaTeX-Dokument geschehen.

Wenn die Bilddatei nicht existiert, ergibt obiger Befehl einen Fehler und der Quellcode wird nicht kompiliert. Um dies zu vermeiden, ändert man den Befehl wie folgt:

```
\IfFileExists{datei}
   { \includegraphics[optionen]{datei} }  % Bilddatei ist da
   { \textcolor{red}{Datei fehlt} }       % Bilddatei fehlt
```

So wird in jedem Fall kompiliert. Fehlt die Bilddatei, wird anstelle des Bildes der kurze Text *Datei fehlt* in Rot geschrieben.

Eine systematische Einfügung von Bildern in ein Dokument, mit Bildunterschrift, automatischer Nummerierung der Abbildungen und optimierter Platzierung auf der Druckseite erreicht man mithilfe einer *figure*-Umgebung, die so gebildet wird:

```
\begin{figure}
   \centering
   \includegraphics[width=5cm]{rlghs.png}
   \caption{Prostmondäne Kunst: \glqq Handschrift\grqq}
   \label{fig:pmk}
\end{figure}
```

Damit das Bild in der Seitenmitte erscheint (siehe Abbildung 2), wird am Anfang der *figure*-Umgebung der Befehl *\centering* gesetzt. Er ist ein Schalter (siehe Seite 9), der nur in der Umgebung wirkt, in der er aufgerufen wird. Im Gegensatz zu einer *center*-Umgebung fügt er keinen unerwünschten Leerraum ein.

Der Befehl *\caption{...}* erhält als Argument im geschweiften Klammerpaar die Bildunterschrift (Legende). Er kann als optionales Argument in eckigen Klammern (vor dem geschweiften Klammerpaar) eine Kurzfassung der Legende erhalten, welche nur im Abbildungsverzeichnis erscheint. Mit dem Befehl *\label{fig:pmk}* wird eine Marke für Querverweise (siehe oben) gesetzt.

In Dokumentklassen mit kapitelweiser Nummerierung der Abbildungen erreicht

man stattdessen eine durchgehende Nummerierung folgendermaßen in der Präambel.

```
\usepackage{chngcntr}
\counterwithout{figure}{chapter}
```

In Dokumentklassen, bei denen Abbildungen normalerweise abschnittsweise nummeriert werden, muss statt *chapter* in obigem Befehl *section* stehen.

Ein schmales Bild könnte links oder rechts von Text umflossen werden. Leider kann LaTeX Gleitobjekte, wie eine *figure*-Umgebung, nicht gut mit einem Textumlauf versehen. Das Paket *wrapfig* von Donald Arseneau stellt die *wrapfigure*-Umgebung bereit, welche das im Wesentlichen leistet, aber manuelle Nachbesserung erfordert.

Im folgenden Beispiel wird mit dem ersten Pflichtargument *r* die Lage des Bildes (l oder L: links, r oder R: rechts) bestimmt. Mit Großbuchstaben (L oder R) wird ein vertikales Gleiten erlaubt, was aber oft nicht richtig gelingt. Das zweite Pflichtargument legt die Breite fest, die dem Bild zur Verfügung steht (mindestens die Bildbreite, die im *includegraphics*-Befehl angegeben wird).

```
... Adornos, eine
reduzierte Farb- und
Formsprache.
\begin{wrapfigure}{r}
{0.5\textwidth}
\vspace{-2mm}
\centering
\includegraphics
[width=0.5\textwidth]
{rebo}
\caption{Regenbogen}
\vspace{-3mm}
\end{wrapfigure}
Die Rückbesinnung auf
wesentliche und ...
```

Das Werk zeigt, im Geiste der ästhetischen Moderne Adornos, eine reduzierte Farb- und Formsprache. Die Rückbesinnung auf wesentliche und symbolhafte Stilelemente ergibt ein spannungsreiches abstraktes Ensemble eines Weichbilds der Stadt ohne Menschen. Gelöst von perspektivischen Erwartungen und den tradierten Vorgaben ikonischer Räumlichkeit kontrastiert es unmanieriert mit einem Regenbogen surrealistischer Traumhaftigkeit. Klare Strukturierung gestaltet die ansprechende Synthese.

Abbildung 3: Regenbogen

Der Leerraum über und unter dem Bild und seiner Legende kann mit dem *vspace*-Befehl angepasst werden. Eine Alternative zur *wrapfigure*-Umgebung sind Minipages (siehe Abschnitt 3.2 auf Seite 43).

Mit dem Paket *copyrightbox* von Ives van der Flaas lässt sich ein kurzer Text direkt unter oder neben ein Bild platzieren. Das kann ein Hinweis auf den Urheber sein, wie im folgenden Bild, welches einen Langblättrigen Ehrenpreis zeigt.

Zunächst wird in der Präambel die Schriftart festgelegt. Beim Aufruf des Befehls *copyrightbox*[...]{...}{...} wird im eckigen Klammerpaar die Platzierung des Texts bestimmt (b = unten, l = links, r = rechts). Im ersten geschweiften Klammerpaar wird das Bild eingefügt, im zweiten der Text genannt.

```
\usepackage{copyrightbox}
\makeatletter\renewcommand{%
\CRB@setcopyrightfont}{%
\footnotesize}\makeatother
...
\copyrightbox[b]
{\includegraphics[scale=0.37]
{veronica.jpg} }
{Urheber: A.H.\,Gitter}
```

Urheber: A.H. Gitter

In obigem Beispiel stehen die ersten vier Zeilen in der Präambel. In der vierten Zeile kann statt *\footnotesize* auch eine andere Schriftart festgelegt werden. Die letzten vier Zeilen stehen an der Stelle, wo das Bild eingefügt werden soll.

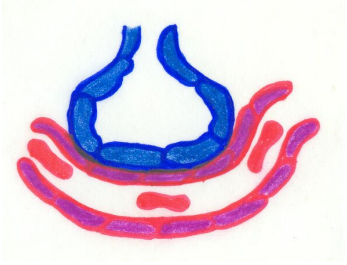
Eine Beschriftung von Bildern kann am einfachsten mit dem Paket *overpic* von Rolf Niepraschk erfolgen. Es stellt die *overpic*-Umgebung bereit. Die Anwendung wird in folgendem Beispiel anhand der Datei *alveole.jpg* gezeigt.

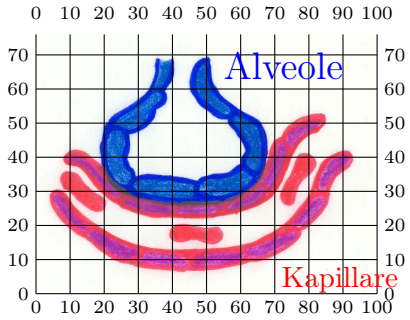
Zunächst lädt man, ähnlich wie beim *includegraphics*-Befehl, mit der *overpic*-Umgebung ein Bild aus einer Datei (hier: *alveole.jpg*) und skaliert es.

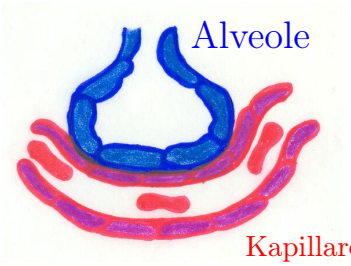
Um die Beschriftung zu positionieren legt man dann vorübergehend ein Gitternetz mit beschrifteten Achsen über das Bild. Dazu dienen die optionalen Argumente *grid* und *tics=10*. Die längere der beiden Achsen hat die Länge 100%. Der Wert, welcher *tics* zugewiesen wird (hier: 10) bestimmt den Abstand der Gitterlinien. Die Beschriftung kann mit allen graphischen Elementen erfolgen, die in einer *picture*-Umgebung möglich sind (siehe Seite 35 in Abschnitt 2.5). Hier wird Text mit dem *put*-Befehl gesetzt. In runden Klammern stehen die Koordinaten des Textanfangs und dann folgt in geschweiften Klammern ein formatierter Text.

Nach der Platzierung der Beschriftung entfernt man das Gitternetz durch Löschung der optionalen Argumente *grid* und *tics=10*. Das beschriftete Bild bleibt über.

```
\begin{overpic}[scale=0.6]
{alveole.jpg}
\end{overpic}

\par\vspace{10mm}
\begin{overpic}[scale=0.6%
,grid,tics=10]{alveole.jpg}
\put(52,63)          {
\textcolor{blue}
{\large Alveole}     }
\put(69,2)           {
\textcolor{red}
{\small Kapillare}   }
\end{overpic}

\par\vspace{10mm}
\begin{overpic}[scale=0.6]
{alveole.jpg}
\put(52,63)          {
\textcolor{blue}
{\large Alveole}     }
\put(68,2)           {
\textcolor{red}
{\small Kapillare}   }
\end{overpic}
```







Wird, wie oben, farbiger Text verwendet, muss natürlich das Paket *xcolor* (oder *color*) geladen sein. Ein waagerechter Pfeil (Richtung (1,0)) der Länge 20%, ausgehend von den Koordinaten (10,60), könnte mit

```
\thicklines \put(10,60){\vector(1,0){20}} % \thicklines: dicker Pfeil
```

eingefügt werden.

Wird der Skalierungsfaktor des Bilds nachträglich geändert, bleibt die relative Positionierung der Beschriftung bestehen, doch die Schriftgröße bleibt konstant.

Tabellen können als einfache Anordnung von Text in mehreren Zeilen gestaltet werden. Dafür kann man eine *tabbing*-Umgebung mit Tabulatoren (Sprungmarken innerhalb einer Zeile) verwenden. Zum Beispiel ergibt

```
\begin{tabbing}
XX\=XXXXXXX\=XXXXXXXXX\=\kill
\>\>Einwohner\>Bundesland\\[-1mm]
\>-----------------------------------------------------------\\
\>Köln\>1,1 Millionen\>Nordrhein-Westfalen\\
\>München\>1,5 Millionen\>Bayern
\end{tabbing}
```

	Einwohner	Bundesland
Köln	1,1 Millionen	Nordrhein-Westfalen
München	1,5 Millionen	Bayern

In obigem Beispiel werden Tabulatoren mit dem Befehl >= in einer Zeile definiert, die nicht gezeigt und gedruckt werden soll und daher mit dem Befehl *kill* abgeschlossen wird. In den folgenden Zeilen werden Tabulatoren mit dem Befehl \> aufgerufen und der Text dort fortgesetzt. Der automatische Zeilenumbruch ist in einer *tabbing*-Umgebung ausgeschaltet. Jede Zeile wird erst mit dem Befehl \\ beendet und der Text in der nächsten Zeile fortgesetzt. Einige Befehle (zum Beispiel *hline*) können in einer *tabbing*-Umgebung nicht benutzt werden. Wenn man, wie oben gezeigt, eine horizontale Linie einfügt, kann die Option eines (positiven oder negativen) vertikalen Abstands beim \\-Befehl nützlich sein.

Aufwendigere Tabellen können mit einer *tabular*-Umgebung erzeugt werden. In geschweiften Klammern werden zunächst Anzahl und Ausrichtung der Spalten festgelegt. Für jede Spalte wird mit einem Buchstaben die Ausrichtung bestimmt, wobei c für mittig, l für linksbündig und r für rechtsbündig steht. Für eine vertikale Trennlinie zwischen zwei Spalten wird ein | eingefügt. Dann werden die Zeilen und möglicherweise horizontale Trennstriche untereinander geschrieben. Die horizontalen Trennstriche werden durch den Befehl *hline* erzeugt. In jeder Zeile werden benachbarte Tabellenfelder durch & getrennt. Jede Zeile wird mit \\abgeschlossen. Mit

```
\begin{tabular}{|l|c|r|}
   \hline
   \multicolumn{2}{|c|}{Verein und Ort} & Tore\\
   \hline
   Bayern & München & 11\\
   Werder & Bremen & 0\\
   \hline
\end{tabular}
```

ergibt sich folgende Tabelle

Verein und Ort		Tore
Bayern	München	11
Werder	Bremen	0

Der Befehl *multicolumn{2}{|c|}{Verein und Ort}* verbindet zwei Zellen, richtet die neue, große Zelle mittig aus (c) und fügt links und rechts vertikale Trennstriche an (|c|). Im dritten geschweiften Klammerpaar steht der Zelleninhalt.

Tabellen sehen besser aus, wenn sie nur wenige Linien enthalten. Insbesondere auf vertikale Linien kann man oft verzichten. Das Paket *booktabs* von Simon Fear und Danie Els stellt drei horizontale Linien für Tabellen zur Verfügung. Zum Beispiel erhält man mit

```
\begin{tabular}{lcr}
   \toprule                % erzeugt die obere horizontale Linie
      \multicolumn{2}{c}{Verein und Ort} & Tore\\
   \midrule      % erzeugt eine horizontale Linie in der Tabelle
      Bayern & München & 0\\
      Werder & Bremen & 11\\
    \bottomrule            % erzeugt die untere horizontale Linie
\end{tabular}
```

eine viel schönere Tabelle:

Verein und Ort		Tore
Bayern	München	0
Werder	Bremen	11

Tabellen mit fester Spaltenbreite können mit dem Parameter p, gefolgt von der Angabe der Spaltenbreite in geschweiften Klammern, erzeugt werden.

```
\begin{tabular}{|p{3cm}|p{3cm}|}
Mozart & Verdi\\
\midrule
La Nozze di Figaro & Aida\\
\end{tabular}
```

Mozart	Verdi
La Nozze di Figaro	Aida

Nachteilig ist, dass der Zelleninhalt stets im Blocksatz ausgerichtet wird. Will man zum Beispiel eine Tabelle mit zentriertem, links- oder rechtsbündigem Zelleninhalt, benötigt man weitere Pakete und eine Definition neuer Parameter für die *tabular*-Umgebung.

Das Paket *threeparttable* von Donald Arseneau ermöglicht eine Tabelle mit Überschrift und Anmerkungen. Zum Beispiel (mit Paket *booktabs*):

```
\begin{threeparttable}[c]
   \caption{Fußball\tnote{a}}
   \begin{tabular}{lcr}
      \toprule
       \multicolumn{2}{c}{Verein
       und Ort} & Tore\\
      \midrule
       Bayern & München & 0\\
       Werder & Bremen & 11\tnote{b}\\
      \bottomrule
   \end{tabular}
   \begin{tablenotes}
      \begin{small}
       \item [a] Endspiel DFB-Pokal
       \item [b] davon ein Eigentor
      \end{small}
   \end{tablenotes}
\end{threeparttable}
```

Tabelle 2.1: Fußball[a]

Verein und Ort		Tore
Bayern	München	0
Werder	Bremen	11[b]

[a] Endspiel DFB-Pokal
[b] davon ein Eigentor

Das optionale Argument des -Befehls bestimmt die vertikale Ausrichtung (t = top, c = center, b = bottom). Markierungen für Anmerkungen, die in der Überschrift gesetzt werden, erscheinen nicht im Tabellenverzeichnis. Soll die Tabelle gleiten, muss man die *threeparttable*-Umgebung noch in eine *table*-Umgebung setzen.

Das Paket *xcolor* (siehe Seite 54) hat die Option *table*, die den Hintergrund von Tabellenzeilen alternierend einfärbt. Zum Beispiel kann jede zweite Zeile hellgrau unterlegt werden. Förderlich ist diese Orientierungshilfe aber nur, wenn wenige Zellen der Tabelle gefüllt sind.

Eine Sonderform tabellarischer Anordnungen ist die logische Gliederung durch einseitige Klammern. Das Paket *schemata* von Charles P. Schaum ermöglicht zum Beispiel folgende Darstellungen.

```
\schema
{\schemabox{a}}
{\schemabox{b\\c\\d}}
```

a { b, c, d

```
\schema[c]
{\schemabox{b\\c\\d}}
{\schemabox{a}}
```

b, c, d } a

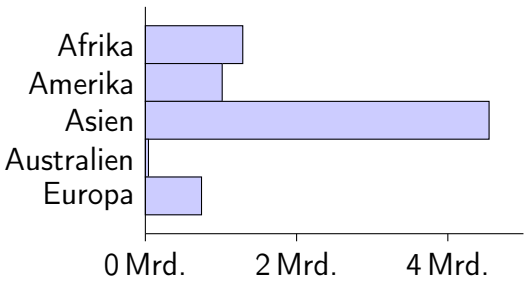
Pflichtargumente des *schema*-Befehls sind die Inhalte links und rechts der Klammer. Sie sollten in eine *schemabox* gepackt werden. Das optionale Argument [*c*] dreht die Klammer um. Innerhalb einer *schemabox* bewirkt \\ den Beginn einer neuen Zeile. Die Klammergliederungen können verschachtelt werden.

```
\begin{center} \mbox{ \schema {
\schemabox                    {
deutsche\\Millionen-\\städte} }
{
   \schema                  {
   \schemabox{im Westen:} }
                 {
   \schemabox{
   Hamburg\\München\\Köln
              }  } \bigskip
   \schema                  {
   \schemabox{im Osten:} }
   {  \schemabox{Berlin}  }
}                    } \end{center}
```

deutsche Millionen-städte { im Westen: { Hamburg, München, Köln; im Osten: { Berlin

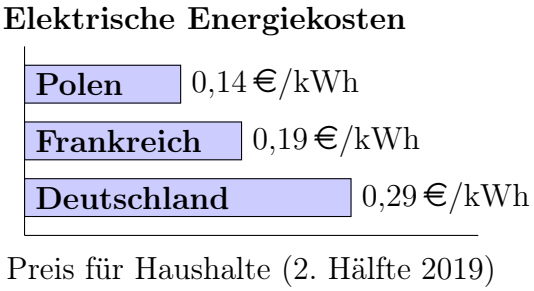
Balkendiagramme erhält man mit dem Paket *bchart* von Tobias Kuhn. Die Anpassungsmöglichkeiten sind beschränkt. Beispiele:

```
\begin{bchart}[step=2,max%
=5,unit=\,Mrd.,width=5cm]
\bcbar[plain,label=Afrika]%
{1.287920518}
\bcbar[plain,label=Amerika]%
{1.015856491}
\bcbar[plain,label=Asien]%
{4.545133094}
\bcbar[plain,label=Australien]%
{0.041261212}
\bcbar[plain,label=Europa]%
{0.742648010}
\end{bchart}
```



Das Diagramm wird in einer *bchart*-Umgebung erzeugt. Jeder Balken wird durch einen *bcbar*-Befehl gezeichnet. Die Option *plain* unterdrückt die Angabe des Zahlenwerts neben dem Balken.

```
\textbf{Elektrische Energie%
kosten}\\[1mm]
\renewcommand{\bcfontstyle}%
{\rmfamily}
\begin{bchart}[max=0.4,%
plain,width=6cm]
\bcbar[text=\textbf{Polen},%
value=\SI{0.14}{\text{\EUR}%
\per\kWh}]{0.1376}   \smallskip
\bcbar[text=\textbf{Frankreich%
},value=\SI{0.19}{\text{\EUR}%
\per\kWh}]{0.1913}   \smallskip
\bcbar[text=\textbf{Deutschlan%
d},value=\SI{0.29}%
{\text{\EUR}\per\kWh}]{0.2878}
\bcxlabel{Preis für Haushalte %
(2. Hälfte 2019)}   \end{bchart}
```



Statt der vorgegebenen serifenlosen Schrift für das Balkendiagramm wurde durch Neudefinition des *bcfontstyle*-Befehls eine Schrift mit Serifen gewählt. Die Option *plain* der *bchart*-Umgebung unterdrückt die Skalierung der Achse. Mit der *value*-Option wird ein Text festgelegt, welcher (statt des Zahlenwerts der Balkenlänge) neben dem Balken erscheint. Das Dezimaltrennzeichen ist im Paket *bchart* ein Punkt, aber in Verbindung mit dem Paket *siunitx* (siehe Abschnitt 4.3 auf Seite 62) bekommt man ein Dezimalkomma. Für den *EUR*-Befehl zur Darstellung des Euro-Symbols braucht man das Paket *marvosym*. Mit *\smallskip* wird eine vertikale Lücke zwischen die Balken gesetzt. Der *bcxlabel*-Befehl beschriftet die Achse.

Kommentare sind Bereiche des Quelltexts, die in der gedruckten Ausgabe fehlen sollen. Es können Erläuterungen des Codes sein oder Quelltext, der vielleicht später gebraucht wird (und vorerst „auskommentiert“ wird).

Kurze Kommentare werden vom Steuerzeichen % eingeleitet und reichen nur bis zum Zeilenende. Längere Kommentare kann man in einer *comment*-Umgebung unterbringen, wenn das Paket *verbatim* von Rainer Schöpf geladen wurde. Beispiel:

```
Hallo % Kommentar
Welt! % noch einer
```

```
Hello,
\begin{comment}
not fit
to print
\end{comment}
World!
```

Hallo Welt!

Hello, World!

Zu beachten ist, dass man nach *\end{comment}* erst in der folgenden Zeile weiterschreiben kann. Auch am Ende einer *verbatim*-Umgebung (siehe Abschnitt 2.5, Seite 33) kann man nach *\end{verbatim}* erst in der folgenden Zeile weiterschreiben, wenn das Paket *verbatim* geladen wurde.

Will man in der tex-Datei einen beliebigen Abschnitt (mit Befehlen und Text) speichern, der nicht von PDFLatex verarbeitet und gedruckt wird, kann man ihn einfach hinter den Befehl *\end{document}* (das heißt: außerhalb des Dokuments) setzen.

2.4 Aufzählungen und Theorem-Umgebungen

Aufzählungen listen mehrere Textteile (*items*) in einer Umgebung. Eine Aufzählungsumgebung kann beliebig viele *items* (Listenelemente) enthalten. Vor jedem *item* erscheint ein sogenanntes Label, bestehend aus einer Nummerierung (*enumerate*-Liste) oder einem Symbol (*itemize*-Liste) oder einer Zeichenkette (*description*-Liste). Aufzählungen können bis zu vier Ebenen tief geschachtelt werden. Eine geschachtelte *enumerate*-Liste entsteht zum Beispiel so:

```
\begin{enumerate}
  \item{good}
  \item{not good}
    \begin{enumerate}
      \item{bad}
      \item{ugly}
    \end{enumerate}
\end{enumerate}
```

1. good
2. not good
    a) bad
    b) ugly

Die Einrückungen vor den Befehlen sind nicht notwendig; sie dienen nur der Übersichtlichkeit und beeinflussen das Aussehen des Ausdrucks nicht. Statt `\item{Ding}` kann man auch `\item Ding` schreiben, also die geschweiften Klammern weglassen.

Soll die Zählung der Listenelemente nicht mit 1, sondern einer ganzen Zahl n beginnen, setzt man den Listenzähler *enumi* vorher auf $n-1$, wie in diesem Beispiel:

```
\begin{enumerate}
\setcounter{enumi}{6}
\item Es beginnt mit Sieben,
\item und endet mit der Acht.
\end{enumerate}
```

7. Es beginnt mit Sieben,
8. und endet mit der Acht.

Ebenso kann man eine *itemize*-Liste mit dem Befehl *itemize* erstellen. Dabei erscheint in der ersten Aufzählungsebene normalerweise das Symbol • (bullet) als Label vor jedem *item*. Möchte man ein anderes Symbol oder eine kurze Zeichenkette, gibt man diese als Option in eckigen Klammern nach dem *\item*-Befehl an. Beispiel:

```
\begin{itemize}
  \item{bullets are weaker
        than ballots}
  \item[b)]{not a), that is
            no question here}
  \item[$\star$]{stars are what
                 men are made of}
\end{itemize}
```

- bullets are weaker than ballots

b) not a), that is no question here

⋆ stars are what men are made of

Schönere Varianten des Symbols • (bullet) stehen in der Datei adfbullets.sty zur Verfügung, welche durch das Paket *adfsymbols* von Clea F. Rees bereitgestellt wird. Nach dem Laden der Datei, mit *\usepackage{adfbullets}* in der Präambel, kann man zum Beispiel eine *itemize*-Liste erzeugen, die verschiedene Symbole als Label vor jedem *item* enthält.

Eine *description*-Liste erzeugt Aufzählungen, deren Labels Zeichenketten sind. Zum Beispiel:

```
\begin{description}
  \item[Romeo] {Sohn der
       Familie Montague}
  \item[Julia] {Tochter
       der Familie Capulet}
\end{description}
```

Romeo Sohn der Familie Montague

Julia Tochter der Familie Capulet

Die bisher behandelten Aufzählungen nehmen viel Platz ein, da vor und nach den Listen und zwischen den *items* Zeilenabstände eingefügt werden. Nach dem Laden des Pakets *paralist* von Bernd Schandl stehen weitere Listenumgebungen zur Verfügung (zum Beispiel die nummerierenden Aufzählungen *inparaenum* und *compactenum*), die ohne zusätzliche Zeilenabstände arbeiten. Bei ihnen kann die Art des Labels zu Beginn der Liste als Option verändert werden, wobei einige Zeichen eine besondere Bedeutung haben: a steht für Kleinbuchstaben, A für Großbuchstaben, I für römische und 1 für arabische Zahlen.

Eine *inparaenum*-Liste bettet die *items* in den laufenden Absatz ein. Beispiel:

```
In der Revolution wurden \begin{inparaenum}[(a)]
\item{Freiheit,}   \item{Gleichheit und}   \item{Brüderlichkeit}
\end{inparaenum}von den Franzosen gefordert.
```

In der Revolution wurden (a) Freiheit, (b) Gleichheit und (c) Brüderlichkeit von den Franzosen gefordert.

Eine *compactenum*-Liste beginnt jedes *item* in einer neuen Zeile. Beispiel:

```
Heute sind alternativlos
\begin{compactenum}[1.]
  \item{Sicherheit}
  \item{Korrektheit}
  \item{Untertänigkeit}
\end{compactenum}
```

Heute sind alternativlos
1. Sicherheit
2. Korrektheit
3. Untertänigkeit

Entsprechend bilden die Listenumgebungen *compactitem* und *compactdesc* nicht nummerierende Aufzählungen (mit Symbol oder Zeichenkette als Label), welche keine Zeilenabstände einfügen.

Mehrspaltige Listen, zum Beispiel eine horizontale Auswahlliste mit Lösungsmöglichkeiten einer Übungsaufgabe, kann man mit dem Paket *tasks* von Clemens Niederberger bilden.

```
Wann fand die Seeschlacht von Lepanto statt\,?
\begin{tasks}(3)
  \task{1560} \task{1571} \task{1588}
\end{tasks}
```

ergibt

Wann fand die Seeschlacht von Lepanto statt ?

a) 1560 b) 1571 c) 1588

Die Zahl in den runden Klammern (hier: 3) gibt die Anzahl der Spalten an, welche in diesem Beispiel mit der Anzahl der Listenelemente übereinstimmt.

Das Paket *easylist* von Paul Isambert erleichtert die Erstellung von umfangreichen, automatisch nummerierten Listen, zum Beispiel einer Gliederung gemäß ISO 2145.

Soll nach einer Gleichung (siehe Abschnitt 5.3) mit, zum Beispiel physikalischen, Symbolen eine Erklärung der Symbole in Listenform folgen, kann man die Formatierung mithilfe des Pakets *eqexpl* von Konstantin Morenko vereinheitlichen.

Theorem-Umgebungen sind Gruppen von nummerierten, mit einheitlicher Überschrift versehenen Textbereichen besonderer Art, zum Beispiel mathematische Sätze oder Übungsaufgaben. Es sind im Wesentlichen ebenfalls Listen, obwohl die Listenelemente nicht zusammenhängen. Besonders wichtig sind Theorem-Umgebungen in mathematischen Texten, aber auch in anderen Anwendungen können sie nützlich sein, wie folgendes Beispiel zeigt.

Zunächst wird in der Präambel der Typ der Theorem-Umgebung definiert. Der *newtheorem*-Befehl hat zwei Pflichtargumente, nämlich den Namen des Theorem-Typs im Quellcode (zum Beispiel *beisp*) und die einheitliche Überschrift für alle Theoreme (Listenelemente) dieses Typs.

```
\newtheorem{beisp}{Beispiel}
```

Theoreme des definierten Typs werden danach mit einer Umgebung gebildet, die den Namen des Theorem-Typs hat. Die Nummerierung erfolgt später automatisch.

```
Tetrapoda sind Wirbeltiere
mit vier Extremitäten.
 \begin{beisp} Vögel sind
 Tetrapoda, weil sie zwei
 Beine und zwei Flügel haben.
 \end{beisp}
Die Extremitäten können Beine,
Arme oder Flügel sein.
 \begin{beisp}[Gegenbeispiel]
 Fische sind Wirbeltiere,
 aber keine Tetrapoda.
 \end{beisp}
```

Tetrapoda sind Wirbeltiere mit vier Extremitäten.

Beispiel 1. *Vögel sind Tetrapoda, weil sie zwei Beine und zwei Flügel haben.*

Die Extremitäten können Beine, Arme oder Flügel sein.

Beispiel 2 (Gegenbeispiel)**.** *Fische sind Wirbeltiere, aber keine Tetrapoda.*

Die Überschrift wird fett geschrieben, der Textinhalt kursiv. Als optionales Argument der Theorem-Umgebung kann (in eckigen Klammern) eine einmalige Ergänzung zur einheitlichen Überschrift folgen (hier: *Gegenbeispiel*).

2.5 Programm-Code, Zeichnungen, Zähler

Programm-Code kann in ein LaTeX-Dokument eingebettet werden, indem man ihn in eine *verbatim*-Umgebung einschließt. Gewöhnlich fügt diese aber vor- und

nachher einen großen vertikalen Abstand ein. Wenn man diese Abstände verkleinern will, kann man das Paket *etoolbox* von Philipp Lehman und Joseph Wright laden. Es stellt unter anderem den *preto*-Befehl und den *AtBeginEnvironment*-Befehl zur Verfügung. Nun kann man in der Präambel (nach Laden des *etoolbox*-Pakets) zum Beispiel die Zeilen

```
\usepackage{etoolbox}
\makeatletter
\preto{\@verbatim}{\topsep=2mm \partopsep=0mm}
\makeatother
```

einfügen. Die Abstände zu den Zeilen vor und nach jeder *verbatim*-Umgebung werden so auf 2 mm verringert. Natürlich sind auch andere Werte möglich. Die Bedeutung der Befehle *\makeatletter* und *\makeatother* wird auf Seite 53 erklärt. Programmcode wird nun mit

```
Normaler Text. Programmcode:
\begin{verbatim}
Hier steht der Programm-Code.
\end{verbatim}
Es folgt wieder normaler Text.
```

Normaler Text. Programmcode:

```
Hier steht der Programm-Code.
```

Es folgt wieder normaler Text.

in normalen Text eingebettet. Der Text erscheint in einer *\verbatim*-Umgebung ohne Einrückung mit nichtproportionaler Schriftart. Zeilenumbrüche werden dargestellt. Innerhalb des Absatzes werden LaTeX-Befehle ignoriert (außer *end{verbatim}*).

Bei Verwendung der Standardschrift *Computer Modern* (CM), auch in der europäischen Variante *European Computer Modern* (EC), wird das hochgestellte gerade Anführungszeichen in einer *verbatim*-Umgebung als hochgestelltes gebogenes (schließendes) Anführungszeichen wiedergegeben. Das kann jedoch berichtigt werden, indem man das Paket *upquote* von Michael Covington, Markus Kuhn und Frank Mittelbach lädt.

Die Befehle *\begin{verbatim}* und *\end{verbatim}* sollten jeweils allein in einer Zeile sein. Davor und danach sollte kein anderes Zeichen stehen, nicht einmal ein Leerzeichen. So vermeidet man Fehler, die unter bestimmten Bedingungen auftreten.

Hübscher kann ein Programmcode dargestellt werden, wenn das Paket *listings* von Jobst Hoffmann geladen wurde. Die Darstellungsweise wird mit dem *\lstset*-Befehl eingestellt und danach folgt Programm-Code in einer oder mehreren *lstlisting*-Umgebungen. In den geschweiften Klammern nach dem *\lstset*-Befehl wird unter anderem die verwendete Programmiersprache als Wert der Zuweisung *language=* angegeben (zum Beispiel bash, C, C++, Java oder Python).

Hier ein Beispiel für ein Python-Programm:

```
\lstset{
  language=Python, % gezeigt wird Python-Programm-Code
  tabsize=4, % Tabulator-Ersetzung durch vier Leerzeichen
  numbers=left, % Zeilennummern links
  xleftmargin=8mm, % Seitenrand links (für Zeilennummern)
  xrightmargin=8mm, % Seitenrand rechts (kleinerer Rahmen)
  numberstyle=\footnotesize\ttfamily, % Zeilennummernstil
  showstringspaces=false, % normale Leerzeichen im String
  frame=trbl,  % Rahmen an Seiten top, right, left, bottom
  frameround=tttt % Abrundung der vier Ecken des Rahmens
```

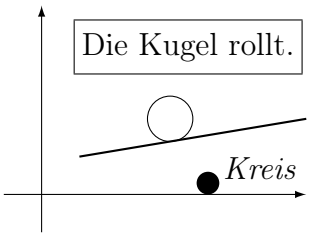
```
}
\lstset{literate=%
  {Ä}{{\"A}}1 {Ö}{{\"O}}1 {Ü}{{\"U}}1 % definiere Umlaute
  {ä}{{\"a}}1 {ö}{{\"o}}1 {ü}{{\"u}}1 % ÄÖÜäöü und das ß
  {ß}{{\ss}}1                    % für deutschen Kommentar
}
\begin{lstlisting}
#!/usr/bin/python3
# Programm zur Begrüßung
text = input("Ihr Name: ")
if text:
print("Hallo "+text)
\end{lstlisting}
```

Damit ergibt sich folgende Darstellung:

```
1 #!/usr/bin/python3
2 # Programm zur Begrüßung
3 text = input("Ihr Name: ")
4     if text:
5 print("Hallo "+text)
```

Zeichnungen können in einer *picture*-Umgebung erstellt werden. Ohne zusätzliche Pakete sind die Möglichkeiten aber begrenzt und man sollte für umfangreichere Zeichnungen das Paket *tikz* (siehe unten) verwenden. Beispiel:

```
\setlength{\unitlength}{1cm}
\begin{picture}(5,4)(-1,-0.9)
\put(-0.5,0){\vector(1,0){4}}
\put(0,-0.5){\vector(0,1){3}}
\thicklines % dickere Linien
\put(0.5,0.5){\line(6,1){3}}
\thinlines % dünnere Linien
\put(1.7,1){\circle{0.6}}
\put(2.2,0.15){\circle*{0.3}}
\put(2.4,0.2){\textit{Kreis}}
\put(0.3,1.6){ \framebox(3,0.7)
{Die Kugel rollt.} }
\end{picture}
```



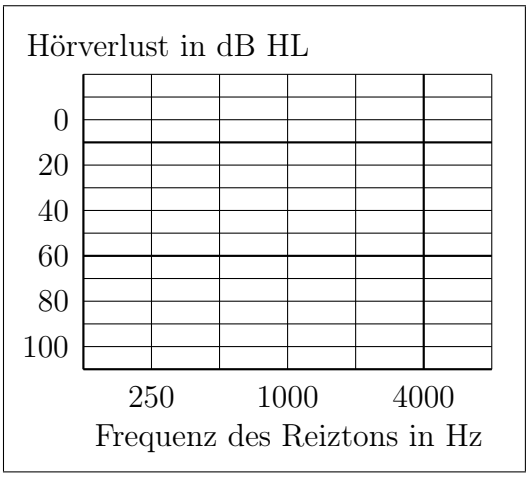
Der erste Befehl setzt die Längeneinheit fest. Danach werden die Längen ohne Einheit geschrieben. Durch Änderung der Längeneinheit lässt sich die Zeichnung vergrößern oder verkleinern. Der Befehl *\begin{picture}(5,4)(-1,-0.9)* legt im ersten runden Klammerpaar Breite und Höhe der Zeichenfläche fest und im zweiten Klammerpaar die Koordinaten der linken unteren Ecke. Dann werden mit dem *put*-Befehl verschiedene Objekte platziert. Im runden Klammerpaar stehen die Koordinaten.

Bei Linien und Vektoren (Pfeile) wird im ersten (runden) Klammerpaar die Steigung durch zwei teilerfremde Zahlen (im Bereich -6 bis 6 beziehungsweise -4 bis 4 für Vektoren) bestimmt und im geschweiften Klammerpaar die Länge. Ab dem Befehl *\thicklines* wird die Linienstärke erhöht und nach *\thinlines* wieder zurückgesetzt.

Bei Kreisen (ohne $*$) und gefüllten Kreisen (mit $*$) wird im Pflichtargument der Durchmesser angegeben. Text, der mit dem *put*-Befehl platziert wird, kann formatiert werden und auch eine *math*-Umgebung enthalten. Bei einem gerahmten Kasten mit Text, *framebox*, werden in einem runden Klammerpaar Breite und Höhe angegeben und in einem geschweiften Klammerpaar der Text.

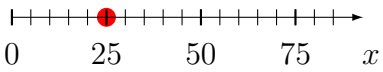
Eine *picture*-Umgebung genügt für das folgende Audiogrammformular.

```
\setlength{\unitlength}{0.3mm}
\frame{   % Rahmen um die Graphik
\begin{picture}(225,205)(-35,-45)
\coordgrid*[30][10](0,0)(180,130)
\put(30,4){\sethlabel{250}}
\put(90,4){\sethlabel{1000}}
\put(150,4){\sethlabel{4000}}
\put(3,10){\setvlabel{100}}
\put(3,30){\setvlabel{80}}
\put(3,50){\setvlabel{60}}
\put(3,70){\setvlabel{40}}
\put(3,90){\setvlabel{20}}
\put(3,110){\setvlabel{0}}
\put(-25,138){Hörverlust in dB HL}
\put(5,-34){Frequenz des %
Reiztons in \si{\hertz}}
\end{picture}   } % Rahmen-Ende
```



Das Paket *coordsys* von Mogens Lemvig Hansen liefert Befehle, die unter anderem eine skalierte Achse in einer *picture*-Umgebung zeichnen.

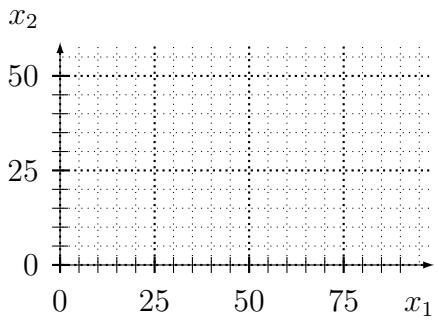
```
\setlength{\unitlength}{0.5mm}
\begin{picture}(100,20)(-5,-10)
\put(25,0){\textcolor{red}{\circle*{5}}}
\numbline[5]{0}{93}
\put(95,0){\sethlabel{x}}
\end{picture}
```



Der rote Kreis in obigem Beispiel erfordert das Paket *color* oder *xcolor*. Der Befehl *numbline* erhält im optionalen Argument den Skalenteilungswert (Abstand zweier Teilstriche). Die Pflichtargumente geben Anfang und Ende der Achse an. Mit dem *sethlabel*-Befehl wird ein Text für die x-Achsenbeschriftung geschrieben. Er wird automatisch in eine *math*-Umgebung (siehe Seite 67) gesetzt. Die entsprechenden Befehle für eine vertikale Achse sind *vnumbline* und *setvlabel*.

Auch ein ebenes kartesisches Koordinatensystem kann gezeichnet werden.

```
\setlength{\unitlength}{0.5mm}
\begin{picture}(110,88)(-10,-15)
\coordsys[5][5](0,0)(99,59)
\coordgrid[5][5](0,0)(99,59)
\put(0,0){\sethlabel{0}}
\put(0,0){\setvlabel{0}}
\put(95,0){\sethlabel{x_1}}
\put(0,65){\setvlabel{x_2}}
\end{picture}
```



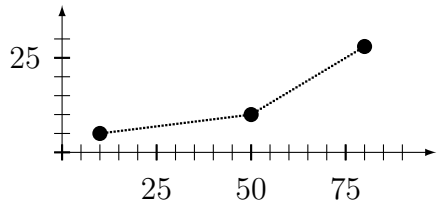
Der *coordsys*-Befehl zeichnet die die horizontale und die vertikale Achse mit Beschriftung. Er erhält in den optionalen Argumenten die Skalenteilungswerte für die beiden Achsen. Die Pflichtargumente geben Anfang und Ende der Achsen an. Der *coordgrid*-Befehl zeichnet ein Gitternetz. Er hat die gleichen Argumente wie der *coordsys*-Befehl. In obigem Beispiel wurde die Achsenbeschriftung mit der Marke 0 in horizontaler und vertikaler Achse ergänzt. Da das Argument des *sethlabel*-Befehls automatisch in eine *math*-Umgebung (siehe Seite 67) gesetzt wird, kann man Indizes mit *_1* beziehungsweise *_2* anhängen.

Das Paket *coordsys* kann nicht zusammen mit dem Paket *interval* von Lars Madsen verwendet werden, da beide einen Befehl namens *interval* definieren.

Logarithmische Achsen und Koordinatensysteme mit logarithmischen Achsen erzeugt man in ähnlicher Weise mit dem Paket *logsys* von Mogens Lemvig Hansen.

Das Paket *epic* von Sunil Podar stellt ein paar Erweiterungen für die *picture*-Umgebung bereit. Insbesondere kann man gepunktete Linien zwischen beliebigen Koordinatenpunkten ziehen. Beispiel (mit Paketen *coordsys* und *epic*):

```
\setlength{\unitlength}{0.5mm}
\begin{picture}(120,70)(-10,-20)
\coordsys[5][5](0,0)(99,39)
\put(10,5){\circle*{4}}
\put(50,10){\circle*{4}}
\put(80,28){\circle*{4}}
\linethickness{0.3mm}
\dottedline(10,5)(50,10)(80,28)
\end{picture}
```



Der *dottedline*-Befehl mit einer Folge von Koordinatenpaaren (in runden Klammern) zeichnet eine gepunktete Linie, deren Stärke vom vorhergehenden *linethickness*-Befehl gegeben wird. Das Argument des *linethickness*-Befehls ist eine Längenangabe mit Einheit (mm). Viel schönere Graphiken kann man mit den aufwändigeren Paketen *tikz* und *pgfplots* erzeugen (siehe unten).

Zähler sind Variablen in LaTeX, die eine ganze Zahl speichern. Für viele Textbereiche, beispielsweise Abschnitte oder Gleichungen (siehe unten) sind besondere Zähler bereits definiert. Außerdem kann man eigene Zähler definieren.

Die Definition eines eigenen Zählers, hier *zehler* genannt, geschieht am besten bereits in der Präambel, kann aber auch noch in der *document*-Umgebung erfolgen. Mit seiner Definition wird der Zähler auf 0 gesetzt. Durch den Befehl *setcounter* kann der Zähler auf einen beliebigen ganzzahligen Wert gesetzt werden und mit dem Befehl *stepcounter* wird der Wert des Zählers um 1 erhöht. Zum Schreiben des aktuellen Zählerwerts in den Text dient der Befehl *thezehler*, worin an *the* der Name des Zählers gehängt wurde. Will man den Wert des Zählers in einer bedingten Anweisung verwenden, ohne ihn in den Text zu schreiben, ruft man den Wert des Zählers mit *value* ab. Im folgenden Beispiel wird zweimal der Befehl *ifnum* benutzt, um Text in Abhängigkeit vom Wert des Zählers zu schreiben.

```
\newcounter{zehler}
Zählerwert: \thezehler

\setcounter{zehler}{3}
\ifnum\value{zehler}=3
Zähler ist gleich 3
\fi

\stepcounter{zehler}
Zählerwert: \thezehler

\ifnum\value{zehler}<10
Zähler kleiner als 10
\else
Zähler größergleich 10
\fi
```

Zählerwert: 0
Zähler ist gleich 3
Zählerwert: 4
Zähler kleiner als 10

Der Befehl *ifnum* erwartet einen Wert (hier: der des Zählers), ein Operatorzeichen (= oder < oder >), einen ganzzahligen Vergleichswert, dann den Text, der geschrieben wird, wenn der Vergleich wahr ist, danach möglicherweise ein *else* und ein Text, der geschrieben wird, wenn der Vergleich unwahr ist, und als Abschluss *fi*.

3 Seite, Absatz und Wort

3.1 Seitengestaltung, mehrere Spalten

Seitenränder sind manchmal zu groß (oder zu klein) voreingestellt. Mit dem Paket *geometry* von Hideo Umeki kann man die Seitenränder auf neue Werte setzen.

```
\usepackage[left=3cm,right=3cm,top=2cm,bottom=3cm]{geometry}
```

Natürlich sind auch andere Werte möglich als die im Beispiel angegebenen.

Kopf- und Fußzeile kann man mit dem Befehl *\pagestyle{...}* beeinflussen, der in der Präambel gegeben wird. Dies betrifft auch die Darstellung von Seitenzahlen. Ohne zusätzliche Pakete laden zu müssen, kann man in den geschweiften Klammern die Optionen *plain* , *empty* oder *headings* angeben. Fehlt der *pagestyle*-Befehl, entspricht dies *\pagestyle{plain}* und die Seitenzahl erscheint in der Fußleiste. Mit der Option *empty* werden keine Seitenzahlen gesetzt.

Die Wirkung von *headings* hängt von der verwendeten Dokumentklasse (und den zugehörigen Optionen) ab. Bei der Dokumentklasse *scrartcl*, ohne die Option *twoside*, erscheint der Name des Abschnitts in der Mitte der Kopfzeile und die Seitenzahl in der Mitte der Fußzeile.

Bei der Dokumentklasse *scrartcl*, mit der Option *twoside*, erscheint links in der Kopfzeile der linken (geraden) Seiten des (zweiseitig gedruckten) Dokuments der aktuelle Abschnitt (außer auf der ersten Seite eines Abschnitts) und rechts in der Kopfzeile der rechten (ungeraden) Seiten der aktuelle Unterabschnitt (falls ein solcher im Abschnitt bereits aufgemacht wurde). In den Fußzeilen steht (links auf den linken Seiten beziehungsweise rechts auf den rechten Seiten) die Seitenzahl.

Bei der Dokumentklasse *scrreprt*, ohne die Option *twoside*, erscheint der Name des Kapitels in der Mitte der Kopfzeile (außer auf der ersten Seite des Kapitels) und die Seitenzahl in der Mitte der Fußzeile.

Bei der Dokumentklasse *scrreprt*, mit der Option *twoside*, sowie bei der (zweiseitig gedruckten) Dokumentklasse *scrbook*, erscheint links in der Kopfzeile der linken (geraden) Seiten des (zweiseitig gedruckten) Dokuments das aktuelle Kapitel (außer auf der ersten Seite eines Kapitels) und rechts in der Kopfzeile der rechten (ungeraden) Seiten der aktuelle Abschnitt (falls ein solcher im Kapitel bereits aufgemacht wurde). In den Fußzeilen steht (links auf den linken Seiten beziehungsweise rechts auf den rechten Seiten) die Seitenzahl.

Bei der (zweiseitig gedruckten) Dokumentklasse *book* erscheint in der Kopfzeile der linken (geraden) Seiten links die Seitenzahl und rechts das aktuelle Kapitel. In der Kopfzeile der rechten (ungeraden) Seiten steht links der aktuelle Abschnitt (falls ein solcher im Kapitel bereits aufgemacht wurde) und rechts die Seitenzahl. Die Fußzeile bleibt hier leer. Eine Ausnahme ist die erste Seite eines Kapitels: die Kopfzeile bleibt leer und in der Fußzeile steht mittig die Seitenzahl.

Drehen von Textabschnitten ist mit dem Paket *pdflscape* von Heiko Oberdiek möglich. Was in einer *landscape*-Umgebung steht, wird um 90° gedreht. Bei den Seiten mit gedrehtem Inhalt bleiben Kopf- und Fußzeile unbeeinflusst, werden also nicht mitgedreht. Dies Verhalten ist beim Ausdruck der Seiten erwünscht. Bei der Darstellung im PDF-Betrachterprogramm werden die Seiten mit gedrehtem Inhalt im Querformat dargestellt. So kann man den Inhalt am Bildschirm besser lesen. Das Drehen um 90° kann insbesondere bei großen Tabellen hilfreich sein, die dann besser auf eine Seite passen.

scrlayer-scrpage* und *fancyhdr sind Pakete, welche vielfältige Gestaltungsmöglichkeiten für Kopf- und Fußzeilen bereitstellen.

Das Paket *scrlayer-scrpage* von Markus Kohm ist an die KOMA-Script-Klassen (*scrartcl*, *scrreprt*, *scrbook*) angepasst. Nach dem Laden desselben kann man dem *pagestyle*-Befehl die Option *scrheadings* übergeben:

```
\documentclass{ ... }
\usepackage{scrlayer-scrpage}
\pagestyle{scrheadings}
...
```

Ohne weitere Befehle des Pakets *scrlayer-scrpage* erscheinen

a) bei den Dokumentklassen *scrartcl* und *scrreprt*, jeweils ohne die Option *twoside*, die Seitenzahlen mittig in der Fußzeile und die Kopfzeile bleibt leer.

b) bei den Dokumentklassen *scrartcl* und *scrreprt*, jeweils mit der Option *twoside*, die Seitenzahlen in der Fußzeile links (auf linken Seiten) oder rechts (auf rechten Seiten) und die Kopfzeile bleibt leer.

c) bei den Dokumentklassen *scrbook* und *book* die Kopf- und Fußzeilen so wie ohne das Paket *scrlayer-scrpage* und mit *\pagestyle{headings}*, siehe oben. Ausnahme: Bei der Dokumentklasse *book* erscheint auf der ersten Seite eines Kapitel keine Seitenzahl.

Mit weiteren Befehlen des Pakets *scrlayer-scrpage* lassen sich Kopf- und Fußzeile fein und umfangreich gestalten. Näheres entnehme man der Paketbeschreibung.

Das Paket *fancyhdr* von Piet van Oostrum ist nicht an die KOMA-Script-Klassen angepasst und eignet sich für die älteren Standard-Dokumentklassen *article*, *report* und *book*.

Wasserzeichen können den Seiten hinzugefügt werden, um kenntlich zu machen, dass nur ein Entwurf (Englisch: draft) vorliegt, oder um die Urheberschaft des Texts festzuhalten. Geeignet hierfür ist das Paket *draftwatermark* von Sergio Callegari. In folgendem Beispiel wird durch Befehle des Pakets *draftwatermark* in der Präambel der Wasserzeichentext bestimmt, sowie die Helligkeit (0 = schwarz, 1 = weiß) und die Größe des Wasserzeichentexts.

```
\SetWatermarkText{Wasserzeichentext}    \SetWatermarkLightness{0.95}
\SetWatermarkFontSize{24.88pt}          \SetWatermarkScale{2.8}
```

Eine Helligkeit von 0.95 bewirkt, das der Wasserzeichentext fast unsichtbar ist.

Mehrspaltige Bereiche innerhalb eines Dokuments, welches ansonsten einspaltig gesetzt ist, kann man unter anderem mit dem Paket *multicol* von Frank Mittelbach

erzeugen. Das Paket stellt eine *multicols*-Umgebung bereit, die als Pflichtargument die Anzahl der Spalten hat. In der Präambel können, nach dem Laden des Pakets, einige Voreinstellungen geändert werden.

```
\setlength{\columnsep}{0.8cm}          % Spalten-Abstand
\setlength{\columnseprule}{0.4pt}      % Dicke der Spaltentrennlinie
\def\columnseprulecolor{\color{blue}}  % Farbe der Spaltentrennlinie
```

Ohne den zweiten Befehl gibt es keine Spaltentrennlinie. Für eine farbige Spaltentrennlinie wird das Paket *color* oder *xcolor* benötigt. Gleitobjekte, zum Beispiel eine *figure*-Umgebung, und Randnoten sind in den Spalten allerdings nicht erlaubt. Ein Beispiel zeigt die Anwendung:

```
\begin{multicols}{2}
Im Anfang war das Wort
und das Wort war bei
Gott, und das Wort war
Gott. Im Anfang war es
bei Gott. Alles ist
durch das Wort geworden
und ohne das Wort wurde
nichts, was geworden ist.
\end{multicols}
```

Im Anfang war das Wort und das Wort war bei Gott, und das Wort war Gott. Im Anfang war es bei Gott. Alles ist durch das Wort geworden und ohne das Wort wurde nichts, was geworden ist.

Breiten von Text, Zeile und Spalte werden in den Längen \textwidth, \linewidth und \columnwidth angegeben. \textwidth ist die Breite des Textbereichs, unabhängig von Einrückungen. \linewidth ist die Breite einer Zeile, die in einer Umgebung mit bestimmter Einrückung des Textes kleiner sein kann als \textwidth. \columnwidth ist die Breite einer Spalte, die in einem mehrspaltigen Textbereich kleiner als \textwidth ist.

Zum Beispiel ist in einer *itemize* Aufzählung die \linewidth kleiner als die \textwidth. In einem zweispaltigen Bereich sind bei Fließtext \linewidth = \columnwidth kleiner als \textwidth. Bei einer *itemize* Aufzählung in der Spalte eines mehrspaltigen Bereichs ist \linewidth < \columnwidth < \textwidth.

3.2 Rahmen und Minipages

Rahmen umranden einen rechteckigen Textbereich. Ein kurzer Text ohne Zeilenumbrüche kann mit dem *fbox*-Befehl gerahmt werden. Der Text kann natürlich formatiert werden. Mit dem Paket *fancybox* von Timothy Van Zandt erhält man in gleicher Weise andere Arten von Rahmen. Beispiele:

```
\fbox{\textit{abusus non tollit usum}}
```
abusus non tollit usum

```
\ovalbox{barba crescit caput nescit}
```
barba crescit caput nescit

```
\Ovalbox{\textbf{quod scripsi, scripsi}}
```
quod scripsi, scripsi

```
\doublebox{\textbf{quis leget haec?}}
```
quis leget haec?

```
\shadowbox{mors certa, hora incerta}
```

mors certa, hora incerta

Das Paket *framed* von Donald Arseneau, welches das Paket *xcolor* von Dr. Uwe Kern benötigt, enthält die *framed*-Umgebung. Damit können auch mehrzeilige Texte, mit automatischem Zeileneinbruch, gerahmt werden.

```
\begin{framed}
Silver was gone. But this was
not all. The sea-cook had not
gone empty-handed. He had cut
through a bulkhead unobserved
and had removed one of the
sacks of coin, worth perhaps
three or four hundred guineas.
\end{framed}
```

Silver was gone. But this was not all. The sea-cook had not gone empty-handed. He had cut through a bulkhead unobserved and had removed one of the sacks of coin, worth perhaps three or four hundred guineas.

Die *leftbar*-Umgebung zeichnet eine senkrechte Linie links neben den Text.

```
\begin{leftbar}
Oxen and wain-ropes would
not bring me back again to
that accursed island.
\end{leftbar}
```

Oxen and wain-ropes would not bring me back again to that accursed island.

Vielfältige Möglichkeiten zur Gestaltung von Rahmen bietet das Paket *tcolorbox* von Thomas F. Sturm. Die Umgebung *tcolorbox* zeichnet ohne weitere Befehle einen schwarzen, abgerundeten Rahmen in der aktuellen Zeilenbreite (*linewidth*).

```
\begin{tcolorbox}
‘It’s a Cheshire cat,’ said the
Duchess, ‘and that’s why. Pig!’
\end{tcolorbox}
```

‘It’s a Cheshire cat,’ said the Duchess, ‘and that’s why. Pig!’

Der *tcbox*-Befehl zeichnet einen passenden Rahmen um den kurzen Text, der als Pflichtargument übergeben wird. Die Rundung der Ecken kann vorher mit dem Befehl *tcbset{arc=...}* verändert werden, wobei ... den Innenradius der Bögen angibt (zunächst 1 mm). Der Befehl *tcbset{colframe=red}* würde einen roten Rahmen ergeben und *tcbset{colback=red!10!white}* einen rötlichen Hintergrund.

```
\tcbset{arc=3mm}
\tcbox{Adventures in Wonderland}
```

Adventures in Wonderland

Mit dem Befehl *tcbuselibrary{raster}* lädt man in der Präambel die Bibliothek *raster* nach dem Paket *tcolorbox*. Sie stellt die Umgebung *tcbraster* zur Verfügung. Damit kann man eine zusammenhängende Gruppe tabellarisch (in Zeilen und Spalten) angeordneter, automatisch nummerierter Rahmen erzeugen. Sie haben den gleichen Titel, bis auf die laufende Nummer des Rahmens. Beispiel:

```
% nur bei KOMA-Script-Klassen mit der Option parskip (half/full)
%\KOMAoption{parskip}{no}
% Rahmenraster mit 3 Spalten, alle Rahmen haben gleiche Höhe
\begin{tcbraster}[raster columns=3, raster equal height,
% Abstand der Rahmen voneinander
```

```
raster equal skip=6mm,
% Abstände zwischen Rahmen und Titel
bottomtitle=1mm, toptitle=1mm, lefttitle=1mm, righttitle=1mm,
% Abstände zwischen Rahmen und Text
bottom=1.5mm, top=1.5mm, left=1mm, right=1mm,
% Farbe des Rahmens und des Texthintergrunds
colframe=blue,colback=blue!10!white,
% Farbe des Titelhintergrunds
colbacktitle=blue!15!white,
% Farben des Titels und des Texts
coltitle=black,coltext=blue,
% Einheitlicher Titel und laufende Titelnummer (fett)
title={Platonischer Körper \textbf{\thetcbrasternum}}]
% Textinhalt aller Rahmen, der Reihe nach in einer Umgebung
\begin{tcolorbox}Tetraeder\end{tcolorbox}%   1. Rahmentext
\begin{tcolorbox}Hexaeder,\\auch %           2. Rahmentext
Würfel genannt\end{tcolorbox}%                 Fortsetzung
\begin{tcolorbox}Oktaeder\end{tcolorbox}%    3. Rahmentext
\begin{tcolorbox}Dodekaeder\end{tcolorbox}%  4. Rahmentext
\begin{tcolorbox}Ikosaeder\end{tcolorbox}%   5. Rahmentext
\end{tcbraster}
% nur bei KOMA-Script-Klassen mit der Option parskip (half/full)
%\KOMAoption{parskip}{half}   % eventuell full statt half
```

ergibt eine Tabelle der regulären, konvexen dreidimensionalen Polyeder.

Platonischer Körper 1
Tetraeder

Platonischer Körper 2
Hexaeder,
auch Würfel genannt

Platonischer Körper 3
Oktaeder

Platonischer Körper 4
Dodekaeder

Platonischer Körper 5
Ikosaeder

Minipage ist eine Umgebung, die einen Kasten bereitstellt, welcher sich fast wie eine kleine Seite verhält. Allerdings dürfen Minipages keine Gleitobjekte (*figure*, *table*, *lstlisting*) oder Randnoten (*marginpar*) enthalten.

Oft werden zwei Minipages auf der tatsächlichen Seite nebeneinander platziert. So kann man ein Bild, eine Tabelle oder eine Zeichnung neben einen Textabschnitt stellen. Pflichtargument ist die Breite, welche meistens relativ zur Textbreite angegeben wird. Die Summe der Breiten zweier nebeneinander platzierter Minipages darf natürlich die Textbreite nicht überschreiten. Beispiel:

```
Klimaforscher ist der älte%
ste Beruf der Welt.\\[5mm]
\begin{minipage}
{0.6\textwidth}
Die Rekonstruktion der Erde
vor 120000 Jahren zeigt,
wie böse Urmenschen den
Planeten erwärmten und so
die armen Eisbären in der
Eem-Warmzeit schwitzten.
\end{minipage}%
\hfill
\begin{minipage}
{0.37\textwidth}
\includegraphics[%
width=3cm]{erde.png}
\end{minipage}\\[3mm]
Aber wie haben Urmenschen
das Treibhaus gemacht?
```

Klimaforscher ist der älteste Beruf der Welt.

Die Rekonstruktion der Erde vor 120000 Jahren zeigt, wie böse Urmenschen den Planeten erwärmten und so die armen Eisbären in der Eem-Warmzeit schwitzten.

Aber wie haben Urmenschen das Treibhaus gemacht?

Beim Nebeneinandersetzen von Minipages kann es Warnmeldungen geben, wenn im Blocksatz die Zeilen zu viel oder zu wenig gefüllt werden. Derartige Schwierigkeiten wurden vor der ersten und nach der zweiten Minipage verhindert, indem Zeilenumbrüche mit \\[...] eingefügt wurden. Die mit ... bestimmten Abstände vor und nach den Minipages, hier 5 mm beziehungsweise 3 mm, sind beliebig.

Ein % nach dem ersten *\end{minipage}* ist wichtig, da der sonst folgende Zeilenumbruch zu einem Leerzeichen zwischen den Minipages führen und damit die Gesamtbreite vergrößern würde. In obigem Beispiel ist die Summe der Breiten beider Minipages 0,97 × Textbreite und der der Befehl *\hfill* sorgt dafür, dass die Restbreite von 0,03 × Textbreite zwischen den Minipages mit Leerraum aufgefüllt wird.

Während eine Minipage keine *figure*-Umgebung enthalten darf, kann umgekehrt eine *figure*-Umgebung sehr wohl Minipages enthalten. Damit kann man in einer *figure*-Umgebung Text und Bild nebeneinander setzen.

Ein vertikaler Abstand zwischen Absätzen (\parskip) wird innerhalb einer Minipage auf Null gesetzt. Man kann ihn aber manuell einsetzen. Dafür speichert man den Wert des Absatzabstands in einer Variablen mit den Befehlen

```
\newlength{\aba}
\setlength{\aba}{\parskip}
```

am Ende der Präambel und kann nun in der Minipage mit

```
\par\vspace{\aba}
```

einen neuen Absatz mit dem gewohnten Abstand beginnen.

3.3 Ausrichtung, Einrückung, Trennungen, Abstände

Ausrichtung eines Textbereichs bedeutet, dass der Text entweder linksbündig, rechtsbündig, mittig (zentriert) oder im Blocksatz dargestellt wird. Normalerweise werden Absätze im Blocksatz dargestellt. Mit einer *center*-Umgebung kann ein

Textbereich zentriert und mit einer *flushleft*-Umgebung linksbündig werden. Der Schalter *centering* ist eine Alternative zur *center*-Umgebung.

```
Nach normalem Text im Blocksatz\par
{\centering zentrierter Text\par}
und wieder ein Text im Blocksatz.
\begin{center}Hier zentrierter Text
\end{center}
und schließlich wieder Blocksatz.
```

Nach normalem Text im Blocksatz
zentrierter Text
und wieder ein Text im Blocksatz.

Hier zentrierter Text

und schließlich wieder Blocksatz.

Die *center*-Umgebung fügt vor und nach dem zentrierten Text vertikalen Abstand ein. Will man das nicht, kann man den Schalter *centering* in einem Block (in geschweiften Klammern) verwenden. Man muss dann den Text mit *par* abschließen.

Es gibt die Schalter *raggedright* für linksbündigen Text, *raggedleft* für rechtsbündigen Text, und *centering* für zentrierten Text, aber keinen Schalter, um wieder Blocksatz zu schreiben. Daher sollte man die genannten Schalter stets in einem Block { } einschließen. Nach diesem gilt dann wieder der vorgegebene Blocksatz.

Im so ausgerichteten (links- oder rechtsbündigen oder zentrierten) Text fehlt jedoch die automatische Silbentrennung (siehe unten). Abhilfe schafft das Paket *ragged2e* von Martin Schröder. Es definiert entsprechende Umgebungen und Schalter zur Textausrichtung, die die Silbentrennung unterstützen. Außerdem gibt es eine Umgebung und einen Schalter für Blocksatz. Allerdings kann die Neudefinition des Befehls \@arrayparboxrestore durch das Paket *ragged2e* in Zusammenhang mit anderen Paketen zu einer Warnung führen.

Einrückung eines Textbereichs an der linken Seite kann erreicht werden, indem der Text in eine Umgebung mit *quote* oder *quotation* gesetzt wird. Ein Beispiel:

```
Es folgt eine lateinische Redensart.
\begin{quote}
O tempora,\\o mores !
\end{quote}
\begin{quotation}
O tempora,\\o mores !
\end{quotation}
```

Es folgt eine lateinische Redensart.

O tempora,
o mores !

O tempora,
o mores !

In einer *quotation*-Umgebung wird die ersten Zeile jedes Absatzes (zusätzlich) eingerückt, bei *quote* nicht.

Um die automatische Einrückung der ersten Zeile eines Absatzes (einmalig) zu unterdrücken, kann man vor den Absatz den Befehl *noindent* setzen:

```
Dies ist ein einzeiliger Absatz, der links eingerückt wird. \par
\noindent Dies ist ein einzeiliger Absatz, der nicht eingerückt wird.
```

ergibt

Dies ist ein einzeiliger Absatz, der links eingerückt wird.
Dies ist ein einzeiliger Absatz, der nicht eingerückt wird

Trennungen und Abstände beziehen sich auf Absätze oder Textbereiche. Mit dem Befehl *newline* erzwingt man die Fortsetzung des Texts in einer neuen Zeile. Der Absatz wird dadurch jedoch nicht beendet. Der Befehl \\ hat, außerhalb bestimmter

Umgebungen, die gleiche Wirkung wie *newline*. In bestimmten Umgebungen (zum Beispiel *tabular*) hat \\ eine besondere Bedeutung, bewirkt aber meistens auch einen Zeilenumbruch.

```
Jäh \newline
kommt die Zeile\\
an ihr Ende.
```

Jäh
kommt die Zeile
an ihr Ende.

Ein vertikaler Abstand nach einem manuellen Zeilenumbruch kann durch eine Option des \\-Befehls erreicht werden, wobei der gewünschte Abstand mit Längeneinheit in eckigen Klammern angegeben wird.

```
Nach der ersten Zeile folgt \\[2mm]
mit  Abstand dann die zweite.
```

Nach der ersten Zeile folgt
mit Abstand dann die zweite.

Der angegebene Abstand darf negativ sein, wodurch die Zeilen zusammenrücken.

Mit dem Befehl *vspace* kann man einen (zusätzlichen) vertikalen Abstand zwischen zwei Absätzen einfügen, dessen Größe auch negativ sein darf. Entsprechend erzeugt *hspace* einen horizontalen Zwischenraum. Beispiel:

```
Hier gibt es eine \hspace{10 mm}Lücke.
\par
\vspace{3 mm}
Der 2. Absatz folgt mit Abstand.
```

Hier gibt es eine Lücke.

Der 2. Absatz folgt mit Abstand.

Ein horizontaler Zwischenraum,der einer bestimmten Wortbreite entspricht, kann mit dem Befehl *hphantom* eingefügt werden, dessen Argument das Wort ist.

```
Die Oper \hphantom{Aida} ist von Verdi.
```

Die Oper ist von Verdi.

Entsprechend werden mit dem Befehl *vphantom* vertikale Abstände erzeugt, so wie es das im Argument angegebene (und möglicherweise formatierte) Zeichen oder Wort tun würde (welches aber nicht gedruckt wird). Horizontale Abstände werden dadurch nicht verändert, wohl aber Zeilenabstände.

```
Mozart komponierte die Oper\\
La Nozze di \vphantom{\huge F}Figaro.
```

Mozart komponierte die Oper
La Nozze di Figaro.

Der Befehl *hfill* ermöglicht es, eine Zeichenkette ans Ende einer Zeile zu schieben:

```
Kosten: \hfill 30 ct/kWh
```

Kosten: 30 ct/kWh

Soll ein Abschnitt am Seitenanfang beginnen, kann vor dem Abschnitt mit dem Befehl *newpage* eine neue Seite angefordert werden. Beispiel:

```
\newpage
\section{Von oben herab}
Erstens kommt es anders,\newline und zweitens als man denkt.\newline
```

Silbentrennung geschieht in der Regel automatisch richtig. In Worte, die LaTeX falsch trennt (zum Beispiel *Lastregler*), kann man \- einfügen (*Last\-regler*), was bei Bedarf am Zeilenende zur Silbentrennung an der angegebenen Stelle führt. Die Trennung bestimmter Worte kann auch mit dem *hyphenation*-Befehl festgelegt werden, der am besten schon in der Präambel steht. Beispiel:

```
\hyphenation{Hüte Ti-ger-ohr}
Wie das Adlerauge ist das Tigerohr
ein sehr gutes Sinnesorgan.
```

Wie das Adlerauge ist das Tigerohr ein sehr gutes Sinnesorgan.

Die im *hyphenation*-Befehl aufgeführten Worte werden nur an den mit - angegebenen Stellen getrennt- Daher kann man die Trennung eines Wortes ganz vermeiden (wie in obigem Beispiel beim Wort *Hüte*).

Lange Zeichenketten sind zum Beispiel bioinformatische Aminosäuresequenzen im einbuchstabigen Code oder Zahlen mit sehr vielen Ziffern. Eine automatische Trennung am Zeilenende ermöglicht das Paket *seqsplit* von Boris Veytsman.

3.4 Typographische Regeln

Typographische Begriffe stammen meistens aus der Zeit, in der Texte mittels Bleisatz gedruckt wurden. Jedes gedruckte Zeichen nimmt auf dem Papier eine rechteckige Fläche ein. Deren horizontale Seitenlänge heißt *Dickte*; die vertikale Seitenlänge heißt *Kegelhöhe*. Die Gestalt des gedruckten Zeichens heißt *Glyphe*. Zwischen dem linken Rand der Rechteckfläche und der Glyphe liegt die Vorbreite, und zwischen der Glyphe und dem rechten Rand der Rechteckfläche liegt die Nachbreite. Die Dickte ist daher meistens größer als die Breite der Glyphe.

Schriftarten, deren Zeichen die gleiche Dickte haben, nennt man nichtproportionale oder *Monospace*-Schriften. In *Proportionalschriften* sind die Dickten von den Glyphen abhängig, sodass ein i eine kleinere Dickte hat als ein m.

Die Breite b, die zwei nebeneinander stehende Zeichen haben, ist mindestens die Summe ihrer Dickten $d_1 + d_2$. Wird zwischen den Zeichen ein Abstand, *Spatium* genannt, eingefügt, ist $b > d_1 + d_2$. Die Verbreiterung der normalen Schrift durch Einfügung von Spatien heißt *Sperren*.

Ein erkennbares Zeichen, insbesondere ein Buchstabe, erfordert bestimmte Striche. Die Enden dieser Striche können durch Hinzufügung feiner Linien, *Serifen* genannt, hervorgehoben werden. Man unterscheidet Schriften mit (Beispiel: H) und ohne Serifen (H). Längere gedruckte Texte sind in *Serifenschrift* besser lesbar. Für Texte, die mit geringer räumlicher Auflösung dargestellt werden, wie Kleingedrucktes oder bei der Darstellung auf Bildschirmen oder Anzeigen elektronischer Geräte, eignen sich dagegen *serifenlose Schriften*. Außerdem heben sich serifenlose Überschriften gut von einem Text in Serifenschrift ab.

Auszeichnung soll Wörter innerhalb eines Textes hervorheben, ohne den Lesefluss zu stören. In der Regel ist *kursive* Schrift am besten dafür geeignet. Unterstreichen sollte man nur ausnahmsweise, wo dies durch eine Vorgabe gefordert wird (beispielsweise für Verweise). Sperren ist verboten. GROSSBUCHSTABEN, **fette**, größere oder serifenlose Schrift gilt, außerhalb von Überschriften, als zu aufdringlich oder hässlich. KAPITÄLCHEN und Farbe sollte man, nur in besonderen Fällen,

wohlüberlegt einsetzen. `Nichtproportionale` Schrift eignet sich für den Quellcode von Computerprogrammen. Die praktische Umsetzung der Textauszeichnung wird später, in Abschnitt 4.1 (beginnend auf Seite 50) beschrieben.

Geschützte Leerzeichen verhindern, dass die Zeichen vor und nach dem Leerzeichen an einem Zeilenende getrennt werden. Es gibt nicht nur ein geschütztes Leerzeichen normaler Breite (~), sondern auch ein kleines geschütztes Leerzeichen (\,), siehe Abschnitt 4.2 auf Seite 56. Mit einem geschützten Leerzeichen bleiben Titel (oder Initiale) und Name einer Person zusammen. Ebenso ein Begriff mit zugehöriger Zahl, die vorangeht oder folgt.

```
7.~Beispiel aus Dr.~Ecks Buch, S.~9
```
7. Beispiel aus Dr. Ecks Buch, S. 9

Ein kleines geschütztes Leerzeichen trennt Maßzahl und Einheit einer Größe.

```
A.~Konrads Forelle: 98,5\,cm, 19,8\,kg
```
A. Konrads Forelle: 98,5 cm, 19,8 kg

Prozentangaben enthalten einen kleinen Zwischenraum, außer als Teil eines Wortes:

```
80\,\% Piraten trinken 40\%igen Rum
```
80 % Piraten trinken 40%igen Rum

Abkürzungen sollten sparsam verwendet werden, da sie die Lesbarkeit eines Texts meistens verschlechtern. Am Satzanfang vermeidet man sie. Wenn sie nicht allgemein üblich sind, werden sie bei der ersten Benutzung erklärt. In wissenschaftlichen Texten mit vielen Abkürzungen sollte man ein Abkürzungsverzeichnis anlegen.

```
die Süßwasser-Straßenordnung (SWO)
```
die Süßwasser-Straßenordnung (SWO)

In mehrgliedrigen Abkürzungen werden die Glieder in der Regel durch Punkt und kleines geschütztes Leerzeichen getrennt:

```
u.\,a. auch z.\,B. Freunde i.\,w.\,S.
```
u. a. auch z. B. Freunde i. w. S.

Ausnahmen sind mehrgliedrige Abkürzungen, die als Einheit betrachtet werden:

```
Globuli etc. oder Knödel usw.
```
Globuli etc. oder Knödel usw.

Steht eine Abkürzung mit Punkt am Ende eines Satzes, folgt ihr kein zweiter Punkt. Leerzeichen und Zahl nach einer Abkürzung werden nicht abgetrennt (siehe oben).

```
Siehe Bsp.~3\,ff. Diese zeigen \dots
```
Siehe Bsp. 3 ff. Diese zeigen ...

Striche können kurz oder lang, mit oder ohne Abstände davor- und dahinter, gezeichnet werden. Betrachten wir zunächst den kurzen Bindestrich (Divis):

```
rot-blau, Tee-Ei, Code 0-8-1-5
```
rot-blau, Tee-Ei, Code 0-8-1-5

Der längere Gedankenstrich dient auch für Intervalle, Gegensätze, Auslassungen:

```
Xaviere kam -- und blieb. \par
Schlossallee 6--8\par
Peine -- Meine, Eintritt 3,--\,\EUR
```
Xaviere kam – und blieb.
Schlossallee 6–8
Peine – Meine, Eintritt 3,– €

Im Englischen ist der Gedankenstrich noch länger und ohne Abstände:

```
Xaviere came---to stay.
```
Xaviere came—to stay.

Ligaturen sind Buchstabenpaare die zu einem Zeichen verbunden wurden (lateinisch *ligare* = verbinden). Eine Ligatur soll Teil einer sprachlichen Einheit sein und nicht zwei sprachliche Einheiten überbrücken. Ein Beispiel mit f und l: Ein Laden in dem gekauft wird, ist ein Kaufladen und ein Fladen, der nicht heiß genug gebacken wurde, ein zäher Kaufladen. Da LaTeX automatisch Ligaturen bildet, braucht man den Befehl `"|` zur Verhinderung einer Ligatur.

```
Kauf"|laden $\neq$ Kaufladen
```

Kaufladen ≠ Kaufladen

4 Zeichenformatierung

4.1 Schrift (Stil, Größe, hoch/tief, Farbe)

Damit die Darstellung der Zeichen eines Textes einheitlich erscheint, wird ihre Form durch eine *Schrift* gestaltet. Sofern man nichts anderes bestimmt, gibt LaTeX die Schrift *Computer Modern* (CM) vor. Da CM nicht alle westeuropäischen Zeichen (zum Beispiel deutsche Umlaute) enthält, sollte man, durch den Befehl *\usepackage[T1]{fontenc}* in der Präambel, CM durch eine europäische Variante ersetzen. Diese Schrift heißt *European Computer Modern* (EC) oder T1-codierte CM-Schrift.

In EC, und in anderen Schriften, kann man verschiedene Schriftarten unterscheiden, zum Beispiel Proportionalschrift mit Serifen (Beispiel: MMii) und ohne Serifen (Beispiel: MMii). Außerdem gibt es in EC eine nichtproportionale Schrift mit Serifen (`Beispiel: MMii`).

Voreingestellt ist Proportionalschrift mit Serifen für den Fließtext und serifenlose Schrift für Überschriften. Mit dem Befehl

```
\renewcommand{\familydefault}{\sfdefault}
```

in der Präambel erhält man durchgehend serifenlose Schrift. Wie jedoch oben (Seite 47) ausgeführt, ist für den Fließtext meistens eine Serifenschrift besser geeignet.

Bei den Schriftarten gibt es weiterhin verschiedene Stile (normal, *kursiv*, **fett** ...). Daher gibt es den Zeichensatz einer Schrift (die Menge der verfügbaren Zeichen) in mehreren Ausführungen. Außerdem müssen verschiedene Schriftgrößen bereitgestellt werden.

Da EC einige besondere Schriftarten, welche in mathematischen Ausdrücken benutzt werden (siehe unten), nicht bereitstellt, wird hierfür automatisch die Schrift *Computer Modern* (CM) herangezogen. In der Schrift EC können die verschiedenen Schriftstile nicht in allen Größen dargestellt werden oder die Darstellung ist nicht gut. Einiges kann durch das Paket *fix-cm* ergänzt werden. Will man es verwenden, muss man es mit dem Befehl *\RequirePackage{fix-cm}* laden, der als erster Befehl in der Präambel, also noch vor *\documentclass ...* stehen muss.

Die Verwendung anderer Schriften als European Computer Modern (EC) kann sinnvoll sein, wenn sich das Aussehen eines gedruckten Dokuments von anderen LaTeX-Dokumenten unterscheiden soll. Die Schrift *Times* ist eine beliebte Serifenschrift. Mit den Paketen *newtxtext* und *newtxmath* (sie ersetzen das ältere Paket *mathptmx*) und *newtxtt* von Michael Sharpe kann man sie auch im *math*-Modus und für nichtproportionale Schrift verwenden. Wenn in der nichtproportionalen Schrift die Serifen stören, kann man das Paket *inconsolata* von Michael Sharpe laden. Dies kann in der Präambel zum Beispiel mit folgenden Befehlen geschehen:

```
\usepackage{newtxtext}
\usepackage{newtxmath}
\usepackage{newtxtt}    % oder: \usepackage{inconsolata}
```

Schriftstil meint im Folgenden jede besondere Ausfertigung der Zeichen, zum Beispiel *kursiv*. Um eine kurze Zeichenkette (hier: vult) mit **fetter Schrift** (englisch: **b**old**f**ace) darzustellen, verwendet man den Befehl *textbf*, zum Beispiel:

```
Et tu, \textbf{babulus}, sinapem addis.
```
Et tu, **babulus**, sinapem addis.

und entsprechend kann man *kursive Schrift* (englisch: **it**alic) mit dem Befehl *textit* erzeugen, `nichtproportionale` (englisch **t**ele**t**ypewriter = Fernschreiber) mit *texttt* und KAPITÄLCHEN mit *textsc* (englisch: **s**mall **c**apitals). Unterstreichung ist mit dem Befehl *underline* möglich, aber es findet keine Silben- und Worttrennung in der unterstrichenen Zeichenkette statt. Die verschiedenen Befehle für Stiländerungen können kombiniert werden:

```
\textit{a team of \underline{henpecked} men}
```
a team of henpecked men

Das Paket *ulem* von Donald Arseneau ermöglicht weitere Unterstreichungen oder Durchstreichungen, zum Beispiel *\uuline{text}* für doppelt unterstrichenen Text oder *\xout{text}* für schräg durchgestrichene Buchstaben. Leider funktioniert die automatische Silbentrennung in den Befehlen dieses Pakets nicht. Aber man kann (mit \-) eine manuelle Trennung vornehmen.

Mit *\usepackage[normalem]{ulem}* verfügbare Durch- und Unterstreichungen:

~~x~~	y	xy	xy	xy
\sout{x}	\uline{y}	\uwave{xy}	\dashuline{xy}	\dotuline{xy}

Das Paket *ulem* verändert normalerweise den Befehl *\emph*, so dass er Unterstreichung statt Kursivschrift erzeugt. Die Option *normalem* bewirkt jedoch ein unverändertes Verhalten des *emph*-Befehls.

In Texten mit normaler Schrift sollte die Hervorhebung einer Zeichenkette durch kursive Schrift erfolgen. Die (empfohlene) Hervorhebung *in dieser Weise* wird durch den Befehl *emph* (englisch: emphasize = hervorheben) erreicht. Wenn der Text bereits kursiv ist, dann wird durch normale Schrift hervorgehoben.

Auch in einer *math*-Umgebung (siehe Abschnitt 5.1 auf Seite 67) stehen verschiedene Schriftstile zur Verfügung.

Das Paket *censor* von Steven B. Segletes stellt den *censor*-Befehl zur Verfügung, mit dem einzelne Worte oder kurze Abschnitte innerhalb einer Zeile durch einen gleich langen schwarzen Block ersetzt, also zensiert werden. Beispiel:

```
\censor{Jane Shore} war seine Mätresse.
```
[redacted] war seine Mätresse.

Will man die Zensur aufheben, stellt man den Befehl *\StopCensoring* an den Anfang der *document*-Umgebung. Es gibt außerdem einen *\blackout*-Befehl, um größere Textblöcke zu zensieren, aber der funktioniert nur mit Einschränkungen, darf unter anderem keine Umlaute enthalten.

Schriftgröße meint die Größe der Zeichen und kann in der Einheit *Punkt* (pt) angegeben werden. Die typographische Maßeinheit hat mit dem Satzzeichen Punkt nicht zu tun (außer der Beschreibung kleiner Objekte). Die Länge eines typographischen Punkts in LaTeX (351,459804 µm ≈ $\frac{1}{3}$ mm) unterscheidet sich von anderen typographischen Definitionen dieser Einheit (Didot-Punkt, DTP-Punkt, Pica-Punkt) geringfügig, aber das ist in der Praxis unwichtig.

Andere LaTeX-Einheiten der Schriftgröße sind *ex* und *em*. Ihre Länge in mm hängt von der Schriftart ab, denn ein ex ist die Höhe des Zeichens x und ein em die Breite eines M in der verwendeten Schrift. Außerdem können in LaTeX auch die Längeneinheiten mm, cm und in (Inch = 2,54 cm) verwendet werden. In einer *math*-Umgebung (siehe Abschnitt 5.1) wird oft eine andere Schrift verwendet als in normalem Text. Dies wird mit der Längeneinheit *mu* berücksichtigt, welche $^1/_{18}$ em in der Schrift des *math*-Modus ist.

Die Schriftgröße wird zu Beginn des Dokuments festgelegt, kann danach aber durch einen Schalter (siehe Seite 9) verändert werden. Schriftgrößenschalter sind, in aufsteigender Reihenfolge: \tiny, \scriptsize, \footnotesize, \normalsize, \large, \Large, \LARGE, \huge und \Huge.

```
mundus{\LARGE\ vult }decipi
```

mundus vult decipi

Die Größenanpassung wird hier auf den Block beschränkt, der durch geschweifte Klammern gebildet wird. Das Leerzeichen nach dem Schalter wurde durch ein Zeichenpaar aus Backslash \und Leerzeichen erzeugt, da ein normales Leerzeichen nach dem Befehlsnamen nicht gedruckt wird (siehe Seite 8).

Zu beachten ist, dass auch der durch Leerzeichen erzeugte Abstand benachbarter Wörter von der Schriftgröße abhängt. In obigem Beispiel liegen die Leerzeichen innerhalb des Blocks, sind also groß (LARGE). Schreibt man dagegen

```
mundus {\LARGE vult} decipi
```

mundus vult decipi

liegen die Leerzeichen außerhalb des Blocks und die Wortabstände haben nur normale Größe. Meistens ist es besser, die größeren Wortabstände einzusetzen.

Ein Absatz nach dem Schriftgrößenschalter, muss (mit \par oder Leerzeile) am Ende des Blocks beendet werden.

```
{\small This script is an insult
 to a man's intelligence.\par}

{\large Even mine.\par}
```

This script is an insult to a man's intelligence.

Even mine.

Die verschiedenen Schriftgrößen entstehen nicht durch einfache Skalierung (Vergrößerung oder Verkleinerung einer Grundschrift). Vielmehr wurden die Schriften in den verschiedenen Größen jeweils eigens geformt, sind also nicht geometrisch ähnlich. So ergibt sich ein besseres Schriftbild, aber auch eine merkwürdige Abstufung der Schriftgrößen. Außerdem ist zu beachten, dass es nicht alle Schriftarten in allen Größen gibt.

In den Dokumentklassen *article*, *report*, *book*, *beamer*, *scrartcl*, *scrreprt*, *scrbook* entsprechen benannte Schriftgrößenoptionen folgenden Größen in der Einheit pt:

Option	Dokument-Schriftgröße 10 pt	11 pt	12 pt
tiny	5 pt	6 pt	6 pt
scriptsize	7 pt	8 pt	8 pt
footnotesize	8 pt	9 pt	10 pt
small	9 pt	10 pt	10,95 pt
normalsize	10 pt	10,95 pt	12 pt
large	12 pt	12 pt	14,4 pt
Large	14,4 pt	14,4 pt	17,28 pt
LARGE	17,28 pt	17,28 pt	20,74 pt
huge	20,74 pt	20,74 pt	24,88 pt
Huge	24,88 pt	24,88 pt	24,88 pt

Wie man sieht, ist beträgt die normale Schriftgröße tatsächlich 10,95 pt, wenn man im *documentclass*-Befehl die Option *11 pt* setzt.

Es ist auch möglich, eine bestimmte Schriftgröße in der Einheit pt anzugeben.

```
Das ist die normale Größe

{\fontsize{8}{18}
\selectfont
und nun das Kleingedruckte\\
mit großem Zeilenabstand.

}

Und nun ist alles wieder
ganz und gar normal.
```

Das ist die normale Größe

und nun das Kleingedruckte

mit großem Zeilenabstand.

Und nun ist alles wieder ganz und gar normal.

In obigem Beispiel wurde im Kleingedruckten die Schriftgröße auf 8 pt und der Zeilenabstand auf 18 pt gesetzt. Die Wirksamkeit dieser Änderungen beschränkt sich auf den in geschweiften Klammern gefassten Block.

Die aktuelle Schriftgröße wird mit der Befehlszeile

```
\makeatletter Schriftgröße: \f@size \,pt \makeatother
```

ausgegeben. Dabei ist *\f@size* ein Makro der zugrunde liegenden Programmiersprache TeX, welches die Schriftgröße in der Längeneinheit pt zurückgibt.

Da TeX-Makros (Befehle) nur Buchstaben enthalten dürfen und @ für TeX normalerweise kein Buchstabe ist, muss das Zeichen @ vorübergehend zum Buchstaben erklärt werden. Dazu dienen die Befehle *\makeatletter* und *\makeatother*. Dies merkwürdige Verhalten von TeX dient der Verhinderung ungewollter Veränderungen von Makros durch unbedarfte Nutzer.

Hoch- und Tiefstellung erreicht man folgendermaßen:

```
Um 10\textsuperscript{30} Uhr \\
trinke ich H\textsubscript{2}O.
```

Um 10^{30} Uhr
trinke ich H_2O.

In einer *math*-Umgebung $...$ (siehe unten) bewirkt der Befehl *\sp{...}*, oder kurz *^{...}*, eine Hochstellung und *\sb{...}*, oder kurz *_{...}*, eine Tiefstellung.

```
Um $10\sp{30}$ Uhr \\
trinke ich H$\sb{2}$O.
```

Um 10^{30} Uhr
trinke ich H_2O.

Mehr dazu findet man in Abschnitt 5.1 auf Seite 69.

Farbe kann eine Zeichenkette hervorheben. Während man in Berichten, die auf Papier gedruckt werden, bunten Text oft vermeidet, sind Farbelemente in Präsentationen die Regel. Eine stimmige Farbkombination (englisch *color scheme*) ist allerdings Geschmackssache. Man benutzt das Paket *color* von David Carlisle oder das umfangreichere Paket *xcolor* von Dr. Uwe Kern. Manche andere Pakete, die Farbunterstützung brauchen, laden *color* oder *xcolor* automatisch.

Man kann vordefinierte Farben verwenden. Mit *color* sind das: black, white, red, green, blue, cyan, yellow und magenta . Mit *xcolor* gibt es zusätzliche:

black	blue	brown	cyan	darkgray	gray	green
lightgray	lime	olive	orange	pink	purple	magenta
red	teal	violet	white	yellow		

Noch mehr vordefinierte Farben sind verfügbar, wenn das Paket *xcolor* mit der Option *[dvipsnames]* geladen wird.

Apricot	Aquamarine	Bittersweet	Black	Blue	BlueGreen	...

Die vollständige *dvipsnames*-Liste und weitere Listen benannter Farben findet man in der Beschreibung des *xcolor*-Pakets.

Beliebige Farben kann man mithilfe eines Farbmodells, zum Beispiel *rgb*, *RGB* oder *cmyk*, mischen. Für den Druck auf Papier wird das Farbmodell *cmyk* empfohlen, für den Bildschirm aber *rgb* oder *RGB*. Letztere unterscheiden sich nicht grundsätzlich, sondern verwenden nur unterschiedliche Skalen für die Grundfarben Rot, Grün und Blau. Bei *rgb* liegen die Farbanteile zwischen 0 und 1, bei *RGB* dagegen ganzzahlig zwischen 0 und 255. Beim Farbmodell *cmyk* wird die Farbzusammensetzung aus den Bestandteilen Cyan, Magenta, Gelb und Schwarz mit Werten jeweils zwischen 0 und 1 gebildet. In der Literatur werden Farben des *CMYK*-Farbmodells (CMYK mit Großbuchstaben) durch entsprechende Werte zwischen 0 und 100 angegeben. Bei der Benutzung der Pakete *color* oder *xcolor* und *cmyk* sind die Werte der *CMYK*-Farbzusammensetzung also durch 100 zu teilen.

Mit dem Befehl *textcolor* wird eine Zeichenkette farbig (zum Beispiel in der Farbe Magenta) geschrieben. Das geht auch in einer *math*-Umgebung (siehe unten). Beispiele:

```
\textcolor{magenta}{Magenta}: rotblau.
\textcolor[rgb]{0.55,0.71,0}{Apfelgrün}: lecker!
\textcolor[RGB]{220,20,60}{Purpurrot}: schön!
$3^2 + \textcolor{red}{x^2} = 5^2$
```

Magenta: rotblau.
Apfelgrün: lecker!
Purpurrot: schön!
$3^2 + x^2 = 5^2$

Bei der Mischung von Apfelgrün betragen die relativen rgb-Farbanteile (rot, grün und blau), die stets zwischen 0 und 1 liegen, r = 0.55, g = 0.71 und b = 0. Bei der

Mischung von Purpurrot betragen die relativen RGB-Farbanteile (Rot, Grün und Blau), die stets zwischen 0 und 255 liegen, r = 220, g = 20 und b = 60.

Der *definecolor*-Befehl ermöglicht es, Farben mit einem Farbmodell zu definieren. Das erste Pflichtargument gibt den Farbnamen an, das zweite das Farbmodell und das dritte die Zusammensetzung aus den Farben des Farbmodells. Farbdefinitionen stehen meistens am Ende der Präambel.

```
\definecolor{gfb}{cmyk}{0,0,0,1}          % Schwarz
\definecolor{gfr}{cmyk}{0,1,1,0}          % Rot
\definecolor{gfy}{cmyk}{0,0.12,1,0.05}    % Gold
```

Die mit dem *definecolor*-Befehl erzeugten Farben können auch von anderen Paketen, wie TikZ (siehe Abschnitt 6 auf Seite 79), verwendet werden. Damit kann man zum Beispiel die deutsche Flagge zeichnen.

```
\begin{tikzpicture}
\fill [gfb] (0mm,12mm) rectangle (30mm,18mm);
\fill [gfr] (0mm,6mm) rectangle (30mm,12mm);
\fill [gfy] (0mm,0mm) rectangle (30mm,6mm);
\end{tikzpicture}
```

Die chinesische Flagge, ebenfalls mit TikZ gezeichnet, wird auf Seite 85 gezeigt.

Während der *definecolor*-Befehl in den beiden Paketen *color* und *xcolor* zur Verfügung steht, definiert *xcolor* als Alternative den Befehl *xdefinecolor*. Dieser ermöglicht die erweiterten Farbmischungen des *xcolor*-Pakets (siehe unten), unterstützt jedoch nicht das Farbmodell *named*.

Mit dem *xcolor*-Paket kann man eine neue Farbe aus bereits definierten Farben mischen. Die Bestandteile werden mit einem Farbausdruck *farbe1*!*zahl*!*farbe2* beschrieben, wobei *zahl* den prozentualen Anteil von *farbe1* angibt. Fehlt die letzte Farbe, wird Weiß ergänzt. Der Ausdruck kann erweitert werden. Beispiele:

```
\textcolor{blue}{Blau} \newline
\textcolor{blue!45!white}{mit 55\% Weiß} \newline
\textcolor{blue!45}{ebenso} \newline
\textcolor{blue!60!red}{Rotblau} \newline
\colorlet{drb}{blue!60!red!30!black}
\textcolor{drb}{Dunkelrotblau}
```

Blau
mit 55% Weiß
ebenso
Rotblau
Dunkelrotblau

Der Farbausdruck *blue!45!white* in der zweiten Zeile mischt 45 % Blau mit Weiß, dessen Anteil also 55% beträgt. In der dritten Zeile macht *blue!45* das gleiche, da der Weißanteil automatisch ergänzt wird. Dagegen mischt *blue!60!red* in der vierten Zeile 60 % Blau mit 40 % Rot zu einem Rotblau. In der fünften Zeile definieren wir die Mischfarbe `drb`, indem zunächst wieder 60 % Blau mit 40 % Rot gemischt wird, und danach 30 % vom Rotblau mit 70 % Schwarz. In der sechsten Zeile benutzen wir die neu definierte Farbe.

Eine farbige Hinterlegung von schwarzem Text beeinträchtigt die Lesbarkeit und ist daher nur bei hellen Hintergrundfarben, zum Beispiel Gelb, empfehlenswert.

```
\colorbox{yellow}{schwarz auf gelb}
```

schwarz auf gelb

Eine Hinterlegung mit andersfarbigem Rand erhält man mit

```
\fcolorbox{red}{yellow}{gelb mit rotem Rand}
```

gelb mit rotem Rand

Mit dem Paket *xcolor* kann auch eine Spektralfarbe, deren Wellenlänge in Nanometern angegeben ist, durch eine Farbmischung angenähert wiedergegeben werden.

```
\textcolor[wave]{470}{Blau (470\,nm)}
```
Blau (470 nm)

In der Heraldik (Wappenkunde) werden aneinander grenzende Flächen mit wenigen Farben harmonisch und kontrastreich belegt. Das heraldische Farbsystem ist für deutliche Darstellungen, auch Text vor einer Hintergrundfarbe, zu empfehlen.

Unterschieden werden a) „Metalle“, das sind Gold (Gelb, RGB: 255 - 220 - 10) und Silber (Weiß, RGB: 240 - 240 - 240), und b) „Heraldische Farben“, dazu gehören Rot (RGB: 255 - 0 - 0), Blau (RGB: 0 - 0 - 255), Grün (RGB: 0 - 150 - 0) und Schwarz (RGB: 0 - 0 - 0).

Gute Farbpaare bestehen aus „Metall“ und „Heraldischer Farbe“, insbesondere: Weiß - Schwarz, Weiß - Blau, Weiß - Grün, Weiß - Rot, Gelb - Schwarz, Gelb - Rot. Schlechte Paare sind solche, bei denen „Metall“ an „Metall“ oder „Heraldische Farbe“ an „Heraldische Farbe“ grenzt, zum Beispiel Weiß - Gelb und (das leider beliebte) Rot - Blau. Weiß - Gelb wird als kontrastarm empfunden. Im Beispiel oben, mit dem *fcolorbox*-Befehl, verhindert der rote Rand eine kontrastarme Grenze zwischen Gelb und Weiß. Rotes und blaues Licht unterscheiden sich stark in der Wellenlänge und werden aufgrund der chromatischen Aberration im Auge verschieden stark gebrochen (Dispersion), so dass benachbarte Flächen dieser Farben nicht gleichzeitig scharf erscheinen. Beispiel:

```
\definecolor{gb}{cmyk}{0,0,0.15,0}
\definecolor{sw}{cmyk}{1,0.85,0.85,1}
\colorbox{blue}{\textcolor{red}
{\textbf{Rot auf Blau}}}\\[6mm]
\colorbox{sw}{\textcolor{gb}
{\textbf{Hellgelb auf Schwarz}}}
```
Rot auf Blau

Hellgelb auf Schwarz

4.2 Leer- und Sonderzeichen

Leerzeichen bewirken eine Lücke zwischen sichtbaren Zeichen. Manchmal möchte man verhindern, dass zwei durch Leerzeichen getrennte Worte (zum Beispiel *Den Haag*) am Zeilenende getrennt werden. Dann kann man zwischen sie ein geschütztes Leerzeichen ~, Tilde genannt, setzen. Das Zeichen ~ (welches im Text hochgestellt sein kann: ˜) wird als Leerzeichen normaler Breite gedruckt. Damit kann man auch einen Abstand von ein, zwei oder mehr Leerzeichen erzwingen; X~~X ergibt zum Beispiel X X. Es gibt weitere geschützte Leerzeichen anderer Breite.

Geschützte Leerzeichen verschiedener Breite:

XX XX	XX XX	XX XX	XX XX	XX XX	XX XX
XX\,XX	XX\:XX	XX~XX	XX\;XX	XX\quad XX	XX\qquad XX

In einer *math*-Umgebung (siehe unten) werden schmale geschützte Leerzeichen (nämlich \, und \:) oft eingesetzt, zum Beispiel um in einem Produkt einen Multiplikationsoperator darzustellen (Beispiel: $a\,b$ statt ab ohne Lücke).

Es gibt Leerzeichen negativer Breite, die benachbarte Zeichen zusammenrücken lassen. Folgende sind in normalem Text und in einer *math*-Umgebung verfügbar.

Leerzeichen negativer Breite:

XXXX	XXXX	XXXX
XX\negthinspace XX	XX\negmedspace XX	XX\negthickspace XX

negmedspace und *negthickspace* erfordern das Paket *amsmath*. Statt *negthinspace* kann man *!* schreiben (in normalem Text nur nach dem Laden von *amsmath*).

Sonderzeichen sind Zeichen, die nicht direkt über die Tastatur eingegeben werden können, weil es sie dort nicht gibt oder weil sie dann als Steuerzeichen gedeutet würden. Die Eingabe von Sonderzeichen erfolgt daher jeweils über eine besondere Zeichenkette (Code). $ an Anfang und Ende eines Codes kennzeichnen eine *math*-Umgebung (siehe Seite 67). Es gibt im Internet verschiedene, umfangreiche Aufzählungen der in LaTeX verfügbaren Sonderzeichen, zum Beispiel die *Liste von Scott Pakin*. Einige Sonderzeichen und Symbole werden nachfolgend vorgestellt, weitere (mathematische) werden in Abschnitt 5.2 auf Seite 72 dargestellt.

Symbole, die in LaTeX als Steuerzeichen dienen, müssen im Quelltext maskiert werden. Sie können mit folgenden Befehlen dargestellt werden:

%	$	&	#	_	{	}	¶	§	\	~
\%	\$	\&	\#	_	\{	\}	\P	\S	$\backslash$	$\sim$

\	$	^	_
\textbackslash	\textdollar	\textasciicircum	\textunderscore

Deutsche Anführungszeichen:

„deutsch“	‚halb‘
\glqq deutsch\grqq{} *oder* "‘deutsch"’	\glq halb\grq{}

Englische und französisch-schweizerische Anführungszeichen:

“double”	‘single’	«franz.-schweiz.»	‹halb›
\lq\lq englisch\rq\rq{}	\lq halb\rq{}	\flqq deutsch\frqq{}	\flq halb\frq{}

Zur Kennzeichnung der direkten Rede werden im Vereinigten Königreich meistens einfache (single quotes), in den USA dagegen in der Regel doppelte Anführungszeichen (double quotes) gesetzt. Sie können auch `so' beziehungsweise ``so'' (mit je zwei Zeichen vor und nach dem *so*) erzeugt werden.

Hochgestellte gerade Anführungszeichen werden unter anderem für die Darstellung von Programm-Code benötigt. Zur Darstellung der einfachen geraden Anführungszeichen braucht man das Paket *textcomp* von Sebastian Rahtz:

"doppelte gerade"	'einfache gerade'
\dq doppelte gerade\dq{}	\textquotesingle einfache gerade\textquotesingle{}

Ein typographisch richtiger Apostroph ’ (Auslassungszeichen) wird (für deutschen und englischen Text) durch Eingabe des Zeichens ' auf der Tastatur erzeugt.

Kursive griechische Kleinbuchstaben:

α	β	γ	δ	μ	π	ω
α	β	γ	δ	μ	π	ω

Für einige griechische Kleinbuchstaben gibt es eine alternative Schreibweise, zum Beispiel ϱ statt ρ. Man bekommt sie durch Voranstellen von *var*: *ϱ*. Dies betrifft epsilon ϵ ε, theta θ ϑ, pi π ϖ, rho ρ ϱ, sigma σ ς und phi ϕ φ.

Das Paket *upgreek* von Walter Schmidt gibt uns steile (nicht kursive) griechische Kleinbuchstaben:

$\upalpha$	$\upbeta$	$\upvartheta$	$\upmu$	$\uppi$

Man beachte: Das µ in µm kann mit \textmu geschrieben werden (siehe unten).

Steile (nicht kursive) griechische Großbuchstaben:

Δ	Λ	Π	Σ	Φ	Ψ	Ω

Kursive griechische Großbuchstaben erhält man in einer *math*-Umgebung mit dem Schriftstil *mathit* (siehe unten). So wird Σ mit *$\mathit{\Sigma}$* erzeugt.

Pfeile (einseitige Pfeile gibt es mit *left* und *right*):

$\rightarrow$	$\Rightarrow$	$\Leftrightarrow$	$\Longleftarrow$

Pfeile können mit *not* gestrichen werden, zum Beispiel $\not\Rightarrow$ mit $\not\Rightarrow$. Pfeile mit auf und unter gesetztem Text, zum Beispiel $x \xrightarrow[Addition]{+8} y$, erhält man mit

```
$x \xrightarrow[Addition]{+8} y$
```

Doppelte Reaktionspfeile mit auf und unter gesetztem Text sind mit dem Paket *mhchem* (siehe Seite 103) möglich. Zum Beispiel

```
links \ce{<-->[oben][unten]} rechts
```

links $\xrightleftharpoons[unten]{oben}$ rechts

Pfeile, die mit dem Paket *amsfonts* verfügbar sind:

\dashrightarrow	$\rightrightarrows$	$\rightleftarrows$	$\rightsquigarrow$

Punkte und Kreise:

\dots	$\cdots$	\vdots	$\bullet$	$\circ$	$\bigcirc$	\textbigcircle

Akzente:

\`{x}	\'{x}	\u{x}	\^{x}	\v{x}	\~{x}	\={x}	\r{x}	\c{c}

Der *textcircled*-Befehl zeichnet einen Kreis um oder über das im Pflichtargument genannte Zeichen: \textcircled{a} ergibt ⓐ.

Das Paket *realhats* von Matthew W. Scroggs und Adam K. Townsend malt, inner- und außerhalb einer *math*-Umgebung, einen Hut über einen einzelnen Buchstaben.

```
\begin{LARGE}$\hat[tophat]{z} = \hat%
[witch]{x} + \hat[cowboy]{y}$\end{LARGE}
```

$\hat[tophat]{z} = \hat[witch]{x} + \hat[cowboy]{y}$

Sitzen die Hüte zu tief, kann man sie, beispielsweise mit \vphantom{X}, anheben.

```
\begin{LARGE}$\hat[tophat]{\vphantom%
{X}z} = \hat[witch]{\vphantom{X}x} + %
\hat[cowboy]{\vphantom{X}y}$\end{LARGE}
```

$$\hat{z} = \hat{x} + \hat{y}$$

Weiteres Symbole sind:

…	·	£	£	†
\textellipsis	\textperiodcentered	\textsterling	\pounds	\dag
‖	␣	—	–	|
\textbardbl	\textvisiblespace	\textemdash	\textendash	\textbar
©	™	å	Å	Ø
\copyright	\texttrademark	\aa	\AA	\O
‡	*	>	<	•
\ddag	\textasteriskcentered	\textgreater	\textless	\textbullet

Will man ein Symbol, wie das für ein Registered Trade Mark ® (\textregistered), hoch setzen, kann man `\textsuperscript{\textregistered}` schreiben: ®

Mit dem Paket *textcomp* von Sebastian Rahtz im Text verfügbare Sonderzeichen:

¦	℮	№	µ	Ω
\textlbrokenbar	\textestimated	\textnumero	\textmu	\textohm
~	♪	℃	⋆	†
\texttildelow	\textmusicalnote	\textcelsius	\textborn	\textdied
°	½	×	÷	–
\textdegree	\textonehalf	\texttimes	\textdiv	\textminus
¢	$	¥	±	
\textcent	\textdollaroldstyle	\textyen	\textpm	\textleaf

Achtung: Das Paket *textcomp* definiert einige Symbole neu!

Mit *textcomp* sind im Text-Modus auch einige besondere Klammerpaare vorhanden:

⟨	⟩	⟦	⟧	{	}
\textlangle	\textrangle	\textlbrackdbl	\textrbrackdbl	\textlquill	\textrquill

Mit dem Paket *marvosym* von Martin Vogel im Text verfügbare Sonderzeichen:

€ \EUR	$ \EyesDollar	ℛ \Denarius	€ \EURcr	€ \EURhv	€ \EURtm
\Mundus	\Mobilefone	☎ \Telefon	✉ \Letter	FAX \FAX	\Email
\Stopsign	\WomanFace	\ManFace	\Lightning	☺ \Smiley	☹ \Frowny
\Bicycle	\ClockLogo	\Writinghand	\Info	♡ \Heart	\bat
♁ \Earth	♃ \Jupiter	♀ \Venus	♂ \Mars	☽ \Moon	☉ \Sun

Die Symbole € (\EURcr), € (\EURhv) und € (\EURtm) sind an die Schriftarten Courier, Helvetica und Times Roman angepasste Varianten des Euro-Symbols.

Mit dem Paket *fontawesome* im Text verfügbare Sonderzeichen:

\faMale	\faFemale	\faBicycle	\faCar	\faTruck	\faTrain	\faPlane
\faBolt	\faBomb	\faBook	\faBug	\faClockO	\faGears	\faWrench
\faDiamond	\faCube	\faFutbolO	\faBellO	\faAnchor	\faPaperclip	
\faFileO	\faImage	\faLinux	\faApple	\faWindows	\faWikipediaW	
\faFlagCheckered	\faHandScissorsO	\faHandRockO	\faHandPaperO			

Das Paket *ifsym* stellt verschiedene Symbolgruppen bereit, die jeweils als Option geladen werden. Mit *\usepackage[electronic]{ifsym}* erhält man zum Beispiel:

\PulseHigh	\ShortPulseHigh	\ShortPulseLow	\RaisingEdge	\FallingEdge

Es gibt auch Symbole, mit denen Taktdiagramme zusammengesetzt werden können. Weitere Paketoptionen sind: alpine, clock, geometry, misc, weather. Man beachte, dass einige Symbolnamen auch von anderen Paketen benutzt werden.

Mit den Paketen *epsdice* und *pifont* im Text verfügbare Sonderzeichen:

\epsdice{1}	\epsdice{2}	\epsdice{3}	\epsdice{6}	\epsdice[black]{5}

① \ding{172}	② \ding{173}	⑩ \ding{181}	✓ \ding{51}	❀ \ding{96}	☆ \ding{73}
➢ \ding{226}	♣ \ding{168}	♦ \ding{169}	♥ \ding{170}	♠ \ding{171}	✎ \ding{46}

Daniel Spittank bietet im Paket *utfsym* viele skalierbare Symbole, die als Unicode-Zeichen definiert sind, mit einer deutschen Beschreibung an und empfiehlt sie für Arbeitsblätter in der Schule. Unter anderem gibt es Symbole für Pflanzen und Tiere, Spielsteine und Spielkarten, Würfel, Emoticons und Verkehrsmittel.

Das Paket *svrsymbols* von Apostolos Syropoulos stellt verschiedene phantasievolle Symbole zur Verfügung (auch für Elementarteilchen), die man zum Beispiel für chemische Texte nutzen kann (siehe Abschnitt 7.1 auf Seite 102).

Wenn das Paket *keystroke* von Rolf Niepraschk installiert wurde, kann man kleine gerahmte Symbole für Tastatur-Tasten als Sonderzeichen wiedergeben. Beispiele für entsprechende Befehle sind: \keystroke{C} (oder statt C ein anderer Buchstabe), \Return , \Ctrl , \Shift , \Spacebar und \DArrow .

Das Paket *dingbat* von Doug Henderson und Scott Pakin stellt zum Beispiel Hände, ein Auge, eine Satellitenschüssel oder einen Bleistift dar.

Mit dem Paket *CountriesOfEurope* von Rolf Niepraschk und Herbert Voß kann man einige europäische Länder als Symbole in den Text einfügen. Es wird durch

```
\usepackage[scaled=5]{CountriesOfEurope}
```

in der Präambel geladen. Der Skalierungsfaktor, hier 5, bestimmt die Größe der Symbole. Im Dokument wird zum Beispiel mit

```
Dies
{\CountriesOfEuropeFamily\Spain}
ist Spanien.
```

Dies ist Spanien.

ein Land (hier Spain = Spanien) in den Text eingefügt. Die Zeichenbreite und -höhe ist für verschieden große Länder natürlich unterschiedlich, aber die Schriftgröße lässt sich wie bei anderen Zeichen ändern. Diese Ländersymbole gibt es:

Albania, Andorra, Austria, Belarus, Belgium, Bosnia, Bulgaria, Croatia, Czechia, Denmark, Estonia, Finland, France, Germany, GreatBritain, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Liechtenstein, Lithuania, Luxembourg, Macedonia, Malta, Moldova, Montenegro, Netherlands, Norway, Poland, Portugal, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland.

Für die Darstellung der Logos für LaTeX und ähnliches gibt es folgende Befehle. Der *hologo*-Befehl erfordert das Paket *hologo* von Heiko Oberdiek.

LaTeX	LaTeX 2ε	TeX	KOMA-Script	pdfLaTeX
\LaTeX	\LaTeXe	\TeX	\KOMAScript	\hologo{pdfLaTeX}

Folgt nach einem der ersten vier Befehle ein Leerzeichen, hängt man eine geschweifte Klammer an den Befehlsnamen, zum Beispiel: *\LaTeX{}*. Das Logo KOMA-Script ist nur in den Dokumentklassen *scrartcl*, *scrreprt* und *scrbook* verfügbar.

Römische Zahlen werden mit dem Paket *romanbar* von H.-Martin Münch und dem Befehl *Romannum{...}* geschrieben, der als Pflichtargument eine natürliche Zahl erhält. Der *romannum*-Befehl schreibt Kleinbuchstaben. Beispiel:

```
\Romannum{2018} \quad \romannum{2018}
```

MMXVIII mmxviii

Chinesische, japanische oder koreanische Schriftzeichen können, wenn sie im Format UTF-8 codiert sind, mit dem Paket *CJKutf8* von Werner Lemberg eingebettet werden. Die Zeichenkette wird in eine *CJK*-Umgebung gesetzt. Pflichtargumente

sind *UTF8* und die Bezeichnung des Zeichensatzes, zum Beispiel *gkai* oder *gbsn* für chinesische Kurzzeichen beziehungsweise *bkai* oder *bsmi* für Langzeichen.

```
Die Volksrepublik China \\(chinesisch %
\begin{CJK}{UTF8}{gkai}
中华人民共和国%
\end{CJK}%
) \\ist ein Staat im östlichen Asien.
```

Die Volksrepublik China (chinesisch 中华人民共和国) ist ein Staat im östlichen Asien.

Das Prozentzeichen % am Ende der chinesischen Zeichenkette verhindert, dass ein Leerzeichen vor der folgenden schließenden Klammer) eingefügt wird: Jedes Zeilenende wird unsichtbar durch das Zeilenendezeichen markiert, welches wie ein Leerzeichen wirkt. Das %-Zeichen bricht jedoch die Auswertung der Zeile ab, bevor das Zeilenendezeichen ausgewertet wird.

4.3 Physikalische Einheiten

Physikalische Einheiten sollten nicht kursiv, sondern steil gesetzt werden; auch in einer *math*-Umgebung (siehe unten). Zwischen Zahl und Einheit eines Wertes sollte ein kurzes Leerzeichen \, stehen. Physikalische Größen und mathematische Variable sollten dagegen kursiv sein.

In Diagrammen und Tabellen, die Werte physikalischer Größen zeigen, sollte man nicht wiederholt die gleiche Einheit wiedergeben. Besser ist, in Achsenbeschriftungen, beziehungsweise im Tabellenkopf, die Größe durch ihre Einheit zu teilen, so dass Werte ohne Einheit (Zahlen) entstehen, zum Beispiel `Länge $L$ / m`.

Wenn man nur gelegentlich Werte mit Einheiten schreibt, ist das relativ einfache Paket *units* von Axel Reichert nützlich. Es bietet, neben dem Befehl *nicefrac* (siehe Seite 70), zwei Befehle zur Formatierung von Einheiten oder Werten mit Einheiten, davon einen für die Darstellung von Einheiten als Bruch. Man kann sie in normalem Text oder in einer *math*-Umgebung $...$ (siehe unten) verwenden:

```
$v$ war \unitfrac[2{,}5]{m}{s}\,.
```

v war $2{,}5\,\mathrm{m/s}$.

```
$\unit{N} = \unit{kg}\cdot\unit{m/s^2}$
```

$\mathrm{N} = \mathrm{kg} \cdot \mathrm{m/s^2}$

Die Befehle können Einheiten, oder Zahlen mit Einheiten, wiedergeben. Ein steiles (nicht kursives) µ erhält man mit dem Befehl *\textmu*, wenn das Paket *textcomp* geladen wurde; ebenso ein Ω mit dem dem Befehl *\textohm*.

Das Paket *siunitx* von Joseph Wright bringt mehr Darstellungsmöglichkeiten für Zahlen und Einheiten. Bei der Eingabe von Dezimalzahlen können Komma oder Punkt als Trennzeichen dienen. Um a) bei der Ausgabe von Zahlen das Komma als Dezimaltrennzeichen zu bekommen (statt eines Dezimalpunkts) und b) die Einheiten bit und byte, nebst passender Vorsätze (Präfixe) zu haben, lädt man das Paket mit

```
\usepackage[output-decimal-marker={,}, binary-units]{siunitx}
```

Falls man (anders als im Folgenden) einen Punkt · statt eines Kreuzes × als Multiplikationsoperator haben möchte, fügt man in die Präambel diesen Befehl ein:

```
\sisetup{exponent-product = \cdot, output-product = \cdot}
```

Eine physikalische Einheit wird mit dem Befehl *\si{...}* gebildet. Das Pflichtargument enthält Befehle zur Gestaltung einer, möglicherweise zusammengesetzten, Einheit. Eine einfache Einheit kann mit einem dafür vorgesehenen Befehl erzeugt werden. In der Regel gibt es auch eine abgekürzte Variante.

```
die Einheit \si{\second} für Zeitangaben
```
die Einheit s für Zeitangaben

```
die Einheit \si{\s} für Zeitangaben
```
die Einheit s für Zeitangaben

Es ist möglich, im Pflichtargument eine Zeichenkette (ohne \ ) anzugeben.

```
die Einheit \si{s} für Zeitangaben
```
die Einheit s für Zeitangaben

Von dieser Möglichkeit wird jedoch abgeraten, weil damit Eingabefehler schlechter erkannt werden und einige Methoden der Formatierung fehlen. Einheiten, die aus einem Bruch bestehen, können unterschiedlich geschrieben werden. Beispiele:

```
$\si{\kilogram\meter\per\square\second}$
```
$\si{\kilogram\meter\per\square\second}$

```
$\si[per-mode=symbol]{\micro\meter\per\day}$
```
$\si[per-mode=symbol]{\micro\meter\per\day}$

```
$\si[per-mode=fraction]{\coulomb\per\mole}$
```
$\si[per-mode=fraction]{\coulomb\per\mole}$

Unter anderem sind folgende Namen von Einheiten verfügbar:

arcminute, arcsecond, ampere, angstrom, astronomicalunit, atomicmassunit, bar, barn, becquerel, bel, bit, bohr, byte, candela, clight, coulomb, dalton, day, decibel, degree, degreeCelsius, electronmass, electronvolt, elementarycharge, farad, gram, gray, hartree, hectare, henry, hertz, hour, joule, katal, kelvin, kilogram, knot, liter, litre, lumen, lux, meter, minute, mmHg, mole, nauticalmile, neper, newton, ohm, pascal, percent, planckbar, radian, second, siemens, sievert, steradian, tesla, tonne, volt, watt, weber.
litre ergibt das eigentlich richtige Symbol l, aber *liter* das besser lesbare L, das ebenfalls erlaubt ist. Statt *meter* ist auch (mit gleichem Ergebnis) die US-Schreibweise *metre* möglich. In Prozentangaben sollte, wie bei Einheiten, ein kleines Leerzeichen enthalten sein. Dies wird mit *percent* erreicht, das wie eine Einheit verwendet wird.
Einheiten des angloamerikanischen Maßsystems fehlen. Das nicht in CTAN enthaltene Paket *siunitxext* von Mitchell Paulus ergänzt einige.

Folgende Einheitenvorsätze (Präfixe) sind definiert:

yocto (10^{-24}), zepto (10^{-21}), atto (10^{-18}), femto (10^{-15}), pico (10^{-12}), nano (10^{-9}), micro (10^{-6}), milli (10^{-3}), centi (10^{-2}), deci (10^{-1}), deca (10^{1}), hecto (10^{2}), kilo (10^{3}), mega (10^{6}), giga (10^{9}), tera (10^{12}), peta (10^{15}), exa (10^{18}), zetta (10^{21}), yotta (10^{24}); alternativ zu deca ist auch die Schreibweise deka möglich.

Für viele Einheiten, auch mit Vorsätzen, gibt es Abkürzungen, zum Beispiel:

ng, ug, mg, g, kg, amu; nm, um, mm, cm, dm, m, km; ns; us, ms, s; umol, mmol, mol; pA, nA, uA, mA, A; ul, ml, l; Hz, kHz, MHz, GHz; N, kN; Pa; kohm, Mohm; uV, mV, V, kV; uW, mW, W, kW, MW; J, kJ, eV, keV, MeV, kWh; pF, F; K; dB

Potenzen (vor die Einheiten geschrieben): square, cubic, raiseto{...}, per

```
$\si{\uW\per\square\m\raiseto{-4}\K}$
```
$\si{\uW\per\square\m\raiseto{-4}\K}$

Potenzen (nach den Einheiten geschrieben): squared, cubed, tothe{...}

```
$\si{\meter\cubed\second\tothe{-4}}$
```
$\si{\meter\cubed\second\tothe{-4}}$

Für die Einheiten bit und byte sind Vorsätze üblich, die Potenzen von 2 sind: kibi (2^{10}), mebi (2^{20}), gibi (2^{30}), tebi (2^{40}), pebi (2^{50}), exbi (2^{60}), zebi (2^{70})

```
1024\,\si{\mebi\byte} = 1\,\si{\gibi\byte}
```
1024 MiB = 1 GiB

Winkel können mit dem Befehl *ang* angegeben werden.

```
Winkel $\alpha = \ang{8.5}$
$\ang{1;2;3}$ und $\ang{;10;}$
```
Winkel $\alpha = 8{,}5°$
1°2′3″ und 10′

Der Befehl *of* versieht Einheiten mit einem beschreibenden Index:

```
\si{\decibel\of{SPL}}
```
$\mathrm{dB_{SPL}}$

Wenn man das Kürzen gleicher Einheiten in Zähler und Nenner eines Bruchs zeigen möchte, sind Durchstreichungen nützlich. Man braucht dafür das Paket *cancel* von Donald Arseneau. Beispiel:

```
\si[per-mode=fraction]{\cancel\m\per\cancel\m\per\s}
```
$\frac{\cancel{\mathrm{m}}}{\cancel{\mathrm{m}}/\mathrm{s}}$

Eine Zahl ohne Einheit, aber möglicherweise mit einer Zehnerpotenz (markiert durch e, E, d oder D), kann mit dem Befehl *num* formatiert ausgegeben werden.

```
$x = \num{-5,25e-3}$
\num[group-separator = \text{~}]{1234567}
\num[negative-color = red]{-1,23E88}\\
\num[retain-explicit-plus]{+42}
```
$x = -5{,}25 \times 10^{-3}$
1 234 567
$-1{,}23 \times 10^{88}$
+42

Im zweiten Beispiel steht in eckigen Klammern eine Option, die eine Zifferngruppierung zur besseren Lesbarkeit längerer Zahlen bewirkt, hier mit einem Leerzeichen. Im dritten Beispiel bewirkt die Option, dass negative Zahlen rot gedruckt werden und in vierten wird das positive Vorzeichen ausnahmsweise ausgeschrieben.

Eine beliebige Zahl kann auch farbig gedruckt werden. Eine Liste von Zahlen (getrennt durch ;) wird mit dem Befehl *numlist* ausgegeben. Statt einer Zahl kann mit *num* oder *numlist* das Symbol π (\pi) oder das Symbol … (\dots) stehen. Ein Produkt wird mithilfe des Multiplikationszeichens (Kleinbuchstabe x) ausgegeben. Komplexe Zahlen werden mit der imaginären Einheit i oder j eingegeben und bei der Ausgabe wird, sofern nicht anders bestimmt, ein nachgestelltes i geschrieben.

```
\num[color = teal]{42}\\
\numlist{4;9;16;25}\\
\numlist{\pi;2\pi;3 \pi;\dots}\\
\num{0.1 x 2.5 x 4} = \num{1}\\
\num{3,3+i5} = \num{3,3+5i}\\
\num[output-complex-root = j]{3,3+5j}\\
\num[complex-root-position=before-number]{3,3+5j}
```
42
4, 9, 16 und 25
π, 2π, 3π und …
$0{,}1 \times 2{,}5 \times 4 = 1$
$3{,}3 + 5\mathrm{i} = 3{,}3 + 5\mathrm{i}$
$3{,}3 + 5\mathrm{j}$
$3{,}3 + \mathrm{i}5$

Der Wert einer physikalischen Größe besteht aus Zahl und Einheit. Er kann mit dem Befehl *SI* ausgegeben werden. In einer ersten geschweiften Klammer wird die Zahl angegeben (wie beim Befehl *num*) und in einer zweiten geschweiften Klammer wird die Einheit genannt (wie beim Befehl *si*). Möglich ist die Beeinflussung der Darstellung durch Optionen in einer eckigen Klammer vor den geschweiften Klammern.

```
$E = \SI{3,8E6}{\mega\volt\per\meter}$
$\SI[parse-numbers=false]{\sqrt{2}}{\m}$
```
$E = 3{,}8 \times 10^6\,\mathrm{MV\,m^{-1}}$
$\sqrt{2}\,\mathrm{m}$

Im zweiten Beispiel erlaubt die Option in eckigen Klammern die Eingabe einer formatierten Zahl, wie in einer *math*-Umgebung (siehe unten), hier eine Wurzel.

Das Ergebnis einer Messung gibt den Wert einer Größe mit einer Messungenauigkeit wieder. Diese hat die gleiche Einheit wie der Messwert. Sie kann, für die letzten Ziffern des Messwerts, in runden Klammern angegeben werden. Wenn man eine Lücke zwischen Messwert und Messungenauigkeitsklammer wünscht, gibt man die Option *uncertainty-separator={\,}* an. Alternativ kann die Messungenauigkeit durch das Zeichen ± vom Messwert getrennt werden. Dies wird mit der Option *separate-uncertainty* erreicht.

```
\SI{3,5(1)}{\volt}
\SI[uncertainty-separator={\,}]{3,5(1)}{\volt}
\SI[separate-uncertainty]{3,5(1)}{\volt}
```
$3{,}5(1)\,\mathrm{V}$
$3{,}5\,(1)\,\mathrm{V}$
$(3{,}5 \pm 0{,}1)\,\mathrm{V}$

Will man die letztgenannte Schreibweise als Standard wählen, fügt man in die Präambel (nach dem Laden des Pakets *siunitx*) den Befehl

```
\sisetup{separate-uncertainty}
```

ein. Nun kann man auf die Angabe der entsprechenden Option verzichten. Außerdem kann die Messungenauigkeit im Quelltext durch ein vorangestelltes +- angegeben werden, statt durch Klammerung. Beispiel:

```
\SI{3,5(1)}{\volt}
\SI{3,5 +- 0,1}{\volt}
```
$(3{,}5 \pm 0{,}1)\,\mathrm{V}$
$(3{,}5 \pm 0{,}1)\,\mathrm{V}$

Eine Liste von Werten mit der gleichen Einheit wird mit dm Befehl *SIlist* in normalem Text, nicht im *math*-Modus (siehe unten), ausgegeben.

```
\SIlist{1;2;3}{\meter}
```
1 m, 2 m und 3 m

Eigene oder fehlende Einheiten können definiert werden, zum Beispiel dpt als Einheit der Brechkraft eines optischen Systems oder, wenn das Paket *marvosym* geladen wurde, die Einheit € für den Anschaffungswiderstand.

```
\DeclareSIUnit{\dpt}{dpt} % muss in Präambel
\DeclareSIUnit{\euro}{\text{\EUR}} % Präambel
Brechkraft $D = \SI{5}{\dpt}$\\[3mm]
Elektrizität kostet\\in Deutschland\\
\SI{0,3048}{\euro\per\kilo\watt\per\hour}
```
Brechkraft $D = 5\,\mathrm{dpt}$

Elektrizität kostet
in Deutschland
$0{,}3048\,€\,\mathrm{kW^{-1}\,h^{-1}}$

Der Befehl *\DeclareSIUnit* muss bereits in der Präambel stehen, nach dem Laden des Pakets *siunitx*.

Für Tabellen mit Zahlenspalten stellt das Paket *siunitx* in der *tabular*-Umgebung den neuen Spaltentyp `S` bereit. Hier ein Beispiel:

```
\begin{tabular}{S}
\toprule  %   (mit Paket booktabs)
{Zahlen} \\ % keine Zahl, {}-Block
\midrule    % (mit Paket booktabs)
3.14 \\           % Gleitkommazahl
6,0224e23 \\ %    wissenschaftlich
\color{red} -42 \\     % mit Farbe
\bottomrule % (mit Paket booktabs)
\end{tabular}
```

Zahlen
3,14
$6{,}0224 \times 10^{23}$
−42

Das Dezimaltrennzeichen wird in der Mitte der Spalte platziert und alle Zahleneinträge werden entsprechend ausgerichtet. In der Zahlenspalte `S` braucht und soll kein `\num{}` verwendet werden.

5 Mathematik

5.1 Mathematik im Fließtext

Mathematische Ausdrücke, die zwischen normalem Text (Fließtext) stehen, können in einer *math*-Umgebung geschrieben werden (*inline math mode*). Es gibt drei Möglichkeiten dafür: a) \$ ... \$, b) *begin{math}* ... *end{math}* und c) \(... \), wobei der Inhalt ... im *math*-Modus ausgewertet wird, also anders als normaler Text, welcher im Textmodus ausgewertet wird. Besagte drei Möglichkeiten haben meist die gleiche Wirkung, aber in besonderen Textbereichen ist die Wirkung unterschiedlich (zum Beispiel in Befehlen mit „beweglichem Text“, wie bei Fußnoten). In der Regel ist \$ ... \$ zu empfehlen. In einer *math*-Umgebung werden im Quellcode eingefügte Leerzeichen nicht gedruckt und können zur lesbaren Gestaltung des Quellcodes eingesetzt werden.

Größere mathematische Ausdrücke werden nicht in den Fließtext eingebettet sondern abgesetzt in einer *displaymath*-Umgebung gebildet. Es gibt zwei Möglichkeiten dafür: a) *begin{displaymath}* ... *end{displaymath}* und b) \[... \] . In einer *displaymath*-Umgebung werden einige Symbole größer dargestellt als in einer *math*-Umgebung. Ansonsten verhält sich der *math*-Modus in beiden Umgebungen gleich. Beispiele:

```
\( \sum_{i=1}^{10} i = 55 \)
\quad $\lim_{n\rightarrow\infty} x_n$

\par\bigskip

Abgesetzt: \[\sum_{i=1}^{10} i = 55\]
\[ \lim_{n\rightarrow\infty} x_n \]
```

$\sum_{i=1}^{10} i = 55$ $\lim_{n\rightarrow\infty} x_n$

Abgesetzt:

$$\sum_{i=1}^{10} i = 55$$

$$\lim_{n\rightarrow\infty} x_n$$

Die *equation∗*-Umgebung entspricht im Wesentlichen der *displaymath*-Umgebung, erfordert aber das Paket *amsmath*. Für nummerierte Gleichungen gibt es die Umgebungen *equation* und *align* (siehe Abschnitt 5.3 auf Seite 76).

Bei der Klammerung größerer Ausdrücke genügt die normale Klammergröße oft nicht. Die Verwendung größenangepasster Klammern wird in Abschnitt 5.3 auf Seite 76 beschrieben.

Schriftstile in einer *math*-Umgebung stehen über besondere Befehle zur Verfügung, ohne dass Pakete geladen werden müssten.

```
ohne Befehl: $ABCabcd\beta\gamma\Phi\Psi 2389$
$\mathnormal{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathrm{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathbf{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathsf{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathtt{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathit{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathcal{ABCXYZ}$
```

ohne Befehl: $ABCabcd\beta\gamma\Phi\Psi 2389$
$\mathnormal{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathrm{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathbf{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathsf{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathtt{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathit{ABCabcd\beta\gamma\Phi\Psi 2389}$
$\mathcal{ABCXYZ}$

Der kalligraphische Stil *mathcal* kann nur lateinische Großbuchstaben darstellen. Beim *mathit*-Stil gibt es bei ff eine Ligatur (Buchstabenverbindung), zum Beispiel im Wort *Suffix* statt $Suffix$, was bei Funktionsnamen wichtig sein kann. Mit dem Paket *amsfonts* der American Mathematical Society (AMS) gibt es außerdem den Stil *mathfrak*, der aber keine griechischen Großbuchstaben erzeugt: $\mathfrak{ABCabcd\beta\gamma 2389}$. Um in *math*-Umgebungen kursive Zeichen fett zu drucken, lädt man das Paket *bm*.

```
$\mathrm{v},\bm{\mathrm{v}}$ und $\bm{\varphi{}}$
```

$\mathrm{v},\bm{\mathrm{v}}$ und $\bm{\varphi{}}$

Unterschiedliche Schriftstile für mathematische Symbole sollen die mathematischen Objekte leichter erkennbar machen. Es gibt verschiedene Empfehlungen dazu, zum Beispiel diese: Variablen im weiteren Sinne (einschließlich Vektoren, Tensoren und Matrizen) werden kursiv gesetzt, alles andere (einschließlich Zahlen, Einheiten, Operatoren, Konstanten und Namen unveränderlicher Objekte) wird gemäß ISO 80000-2:2009 geradestehend (steil) gesetzt. Dies gilt sowohl für lateinische als auch für griechische Buchstaben. Demnach ist $u = 2 \times \pi \times r$ richtig. Es gilt auch für Indizes, also h_i (i geradestehend) für eine Anfangshöhe und h_i (i kursiv) für das i-te Element einer Folge. Auch Vektoren, Tensoren und Matrizen sind Variable und werden kursiv geschrieben. Im Unterschied zu Skalaren werden sie fett gedruckt, nicht aber ihre Indizes. Vektor-Operatoren, zum Beispiel $\boldsymbol{\nabla}$, sind geradestehend und fett. Vektoren werden mit Kleinbuchstaben, Tensoren und Matrizen mit Großbuchstaben geschrieben.

Farbe erhält man mit dem *textcolor*-Befehl des Pakets *xcolor* (siehe Seite 54).

```
$a^2 + \textcolor{red}{x} + b^2 = c^2$
```

$a^2 + \textcolor{red}{x} + b^2 = c^2$

Argumente des *textcolor*-Befehl werden aber zu ordinären Zeichen gemacht, das heißt: ohne Abstände vorn und hinten.

```
$a \textcolor{red}{-} b \textcolor{red}{=} c$
```

$a \textcolor{red}{-} b \textcolor{red}{=} c$

In obigem Beispiel sollte das Minuszeichen die für binäre Operatoren üblichen Abstände bekommen und das Gleichheitszeichen die für Relationen üblichen Abstände. Dies kann man erreichen, indem man den *textcolor*-Befehl als Argument des Befehls `\mathbin{}` beziehungsweise `\mathrel{}` setzt.

```
$a \mathbin{\textcolor{red}{-}} b \mathrel{\textcolor{red}{=}} c$
```

Nun erhält man die Gleichung mit richtigen Abständen: $a \mathbin{\textcolor{red}{-}} b \mathrel{\textcolor{red}{=}} c$

Es ist zu erwarten, dass die Programmierung des *textcolor*-Befehls zukünftig geändert wird, um die Abstandskorrektur überflüssig zu machen.

Rahmen können auch in in einer *math*-Umgebung mit dem *fbox*-Befehl (siehe Abschnitt 3.2 auf Seite 41) erzeugt werden. Der Inhalt der *fbox* ist zwar zunächst im Textmodus, aber sie kann auch eine *math*-Umgebung enthalten.

```
$a^2 + \fbox{$x^2$} = c^2$
```
$a^2 + \fbox{$x^2$} = c^2$

Der Abstand des Rahmens vom umgebenden Text kann mit dem *hspace*-Befehl vergrößert werden, dessen Pflichtargument den Zusatzabstand angibt.

```
wenn \hspace{2mm}\fbox{$x=y^2$}\hspace{2mm} gilt
```
wenn \hspace{2mm}\fbox{$x=y^2$}\hspace{2mm} gilt

Entsprechend erhält man eine farbige Hinterlegung mit dem *colorbox*-Befehl:

```
$a^2 + \colorbox{yellow}{$x^2$} = c^2$
```
$a^2 + \colorbox{yellow}{$x^2$} = c^2$

Der *fcolorbox*-Befehl fügt noch einen andersfarbigen Rahmen hinzu:

```
$a^2 + \fcolorbox{red}{yellow}{$x^2$} = c^2$
```
$a^2 + \fcolorbox{red}{yellow}{x^2} = c^2$

Unterstreichungen sind auch im *math*-Modus mit dem *underline*-Befehl möglich:

```
$a^2 + \underline{10 x^2} = c^2$
```
$a^2 + \underline{10 x^2} = c^2$

Hoch- und Tiefstellung eines Zeichens wird durch ^ beziehungsweise _ eingeleitet. Zeichenketten müssen als Block in geschweiften Klammern stehen

```
$x_1          +x_{99}= x_{50}{}^2$
```
$x_1 + x_{99} = x_{50}{}^2$

Auf der rechten Seite der Gleichung wurde vor der Hochstellung ein leerer Block {} eingefügt, damit tief- und hochgestellte Zeichen nicht in einer Spalte (x_{50}^2), sondern horizontal versetzt (${x_{50}}^2$) stehen. Hoch- und Tiefstellungen können auch vor einem Zeichen stehen, indem man einen leeren Block voranstellt. Mehrfache Hoch- oder Tiefstellungen sind möglich.

```
${}^{13}_{~6}$C~~~$f'' = a^{-x^2}$
```
${}^{13}_{~6}$C~~~$f'' = a^{-x^2}$

Die Ableitungsstriche bestehen in einer *math*-Umgebung aus Apostrophzeichen. Man erhält sie genauso f'' mit $f^{\prime\prime}$. Eine höher gestellte, aber nicht empfohlene Alternative $f^{''}$ gibt es mit $f^{''}$.

Der Apostroph ist in einer *math*-Umgebung ein sogenanntes aktives Zeichen. Dies bedeutet, dass eine Folge von Apostrophen zu einem Symbol für eine mehrfache Ableitung wird. Das Symbol einer zweifachen Ableitung ist also nicht zweimal das Symbol einer einfachen Ableitung.

Bei der Tiefstellung im *math*-Modus mittels _{} wird in geschweiften Klammern eingeschlossener Text (nicht jedoch Zahlen) kursiv und mit vergrößerten Zeichenabständen ausgegeben, wenn er nicht als Argument des Befehls *mathrm*{} oder (nach Laden des Pakets *amsmath*) als Argument des Befehls *text*{} gesetzt wurde.

Das Paket *subtext* von Palle Jørgensen vereinfacht die Tiefstellung im *math*-Modus durch den Befehl _[], dessen Argument, das in eckigen Klammern steht, automatisch mit *text*{} formatiert wird.

Brüche und Dezimalzahlen. Der Befehl *\frac{...}{...}* erzeugt einen Bruch. Mit dem Paket *units* (oder seiner Teilmenge *nicefrac*) kann man Brüche in einer normalen Textzeile oder in einer *math*-Umgebung mithilfe des Befehls *\nicefrac{...}* kompakter schreiben. Es folgen einige Beispiele.

```
Hässlich ist $18/37$ und
der Bruch $\frac{18}{37}$ ist zu hoch,
aber $\nicefrac{18}{37}$ passt besser,
im Textmodus (\nicefrac{18}{37}) auch.
```

Hässlich ist 18/37 und der Bruch $\frac{18}{37}$ ist zu hoch, aber $^{18}/_{37}$ passt besser, im Textmodus ($^{18}/_{37}$) auch.

Soll ein Bruch in einer *math*-Umgebung so groß geschrieben werden wie in einer *displaymath*-Umgebung, braucht man den *displaystyle*-Befehl.

```
${\displaystyle\frac{18}{37}}$ statt $\frac{18}{37}$
```

$\frac{18}{37}$ statt $\frac{18}{37}$

Die Zeilenabstände vergrößern sich dadurch natürlich. Das Kürzen von Brüchen lässt sich mit Durchstreichungen veranschaulichen, wenn das Paket *cancel* von Donald Arseneau geladen wurde.

```
$\frac{70}{130} = \frac{\cancel{2} \cdot \cancel{5}
 \cdot 7}{\cancel{2} \cdot \cancel{5} \cdot 13}$
```

$\frac{70}{130} = \frac{\cancel{2}\cdot\cancel{5}\cdot 7}{\cancel{2}\cdot\cancel{5}\cdot 13}$

Wenn man Kettenbrüche mit einem *frac*-Befehl schreibt, werden die Teilzähler und Teilnenner höherer Glieder immer kleiner. Eine gleichbleibende Größe erhält man dagegen, wenn das Paket *amsmath* geladen ist, mit dem *cfrac*-Befehl, wie folgendes Beispiel zeigt:

```
\[ \sqrt{2} =
1 + \frac{1}{
2 + \frac{1}{
2 + \frac{1}{
2 + \frac{1}{\dots}}}} \]
```

$$\sqrt{2} = 1 + \frac{1}{2 + \frac{1}{2 + \frac{1}{2 + \frac{1}{\dots}}}}$$

```
\[ \sqrt{2} =
1 + \cfrac{1}{
2 + \cfrac{1}{
2 + \cfrac{1}{
2 + \cfrac{1}{\ddots}}}} \]
```

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{\ddots}}}}$$

Das Sonderzeichen $\ddots$ wird mit dem Befehl *$\ddots$* geschrieben.

In der *math*-Umgebung wird ein Komma normalerweise als Aufzählungsoperator mit nachfolgendem kleinem Abstand dargestellt. Das gilt auch für Dezimalzahlen, die, wie im Deutschen üblich, mit Dezimalkomma geschrieben werden sollen. In deutschen Dezimalzahlen muss man daher *{,}* als Dezimalkommazeichen schreiben. Obiges Beispiel verwendet das schmale Leerzeichen *\,* als Multiplikationsoperator.

Intervalle, Matrizen und geschweifte Klammern können in der *math*-Umgebung leicht gestaltet werden, wie folgende Beispiele zeigen. Mit

```
$x \in [0,1], y \in (0,1)$
```

$x \in [0,1], y \in (0,1)$

stellt man ein geschlossenes und ein offenes Intervall dar. Man kann offene Intervalle auch durch umgekehrte eckige Klammern einfassen, statt durch runde Klammern. Das Paket *interval* von Lars Madsen sorgt dafür, dass bei dieser Schreibweise die Abstände stimmen.

Mit dem Quellcode

```
$\begin{smallmatrix} a_1&a_2 \\ a_3&a_4 \end{smallmatrix} \text{ und}
\left( \begin{smallmatrix} x \\ y \\ z \end{smallmatrix} \right)$
```

werden kleine Matrizen ohne und mit Klammern, zum Beispiel $\begin{smallmatrix} a_1 & a_2 \\ a_3 & a_4 \end{smallmatrix}$ und $\left(\begin{smallmatrix} x \\ y \\ z \end{smallmatrix} \right)$, in der Textzeile erzeugt. Dagegen werden geklammerte große Matrizen mit

```
$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$
```

erzeugt, was $\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$ geschrieben wird. Mit *matrix* statt *pmatrix* entstehen entsprechend ungeklammerte Matrizen und mit *vmatrix* Determinanten, $\begin{vmatrix} a_1 & a_2 \\ a_3 & a_4 \end{vmatrix}$.

Die Umgebung *cases* ermöglicht eine Fallunterscheidung. Mit

```
$x = \begin{cases}a^{-1}&\text{für }a\ne0\\0&\text{sonst}\end{cases}$
```

erhält man $x = \begin{cases} a^{-1} & \text{für } a \neq 0 \\ 0 & \text{sonst} \end{cases}$

Die Befehle *\overbrace* und *\underbrace* erzeugen waagerechte geschweifte Klammern. Zum Beispiel ergibt

```
$\underbrace{1\cdot2\cdot3}_{6} = \overbrace{1+2+3}^{6}$
```

$$\underbrace{1 \cdot 2 \cdot 3}_{6} = \overbrace{1 + 2 + 3}^{6}$$

Einfache Rechenoperationen kann man im LaTeX-Dokument unter anderem mit dem Paket *fltpoint* von Eckhart Guthohrlein durchführen. Der Befehl *\fpDecimalSign{,}* beziehungsweise *\fpDecimalSign{.}* in der Präambel legt Komma beziehungsweise Punkt als Dezimaltrennzeichen fest. Ein Befehlsname kann als Variable dienen, der in der Präambel ein Wert zugewiesen wird.

```
\fpDecimalSign{,}  % Dezimaltrennzeichen: ,
\newcommand*{\wertX}{8,8}% Variable: \wertX
```

In der *document*-Umgebung stehen dann verschiedene arithmetische Operatoren zur Verfügung: Addition geht mit dem Befehl *\fpAdd{...}{...}{...}*. Weitere binäre arithmetische Operatoren sind entsprechend *fpSub*, *fpMul* und *fpDiv*. Unäre Operatoren sind *\fpNeg{...}{...}* und entsprechend *fpAbs*. In der ersten geschweiften Klammer steht jeweils ein Befehlsname, welcher als Variable das Ergebnis aufnimmt (im nächsten Beispiel: *\wertA*). In den folgenden ein oder zwei geschweiften Klammern stehen die Operanden, entweder als Zahl oder als Variable (Befehlsname), der ein Wert zugewiesen wurde.

```
\fpDiv{\wertA}{108}{4} Ergebnis: \wertA \par
\fpAdd{\wertB}{\wertX}{2,5} Ergebnis: \wertB
```

Ergebnis: 27
Ergebnis: 11,3

Außerdem gibt es einen unären Operator zur Rundung einer Zahl, bei dem als weiteres Argument die Zehnerpotenz angegeben wird, zu der gerundet werden soll.

```
\fpRound{\wertC}{3,141}{-1} Ergebnis: \wertC
```
Ergebnis: 3,1

Hier noch ein Beispiel:

```
Variable x hat den Wert \wertX.\\
\fpSub{\wertY}{\wertX}{6,66}
Ungerundet: x - 6,666 = \wertY\\
\fpRound{\wertZ}{\wertY}{0}
und gerundet ergibt das \wertZ.
```
Variable x hat den Wert 8,8.
Ungerundet: x - 6,666 = 2,14
und gerundet ergibt das 2.

Literatur zu Mathematik mit Latex gibt es auf der Webseite der American Mathematical Society, http://www.ams.org/publications/authors/tex/amslatex. Die Seite
http://ftp.fau.de/ctan/info/short-math-guide/short-math-guide.pdf
ist ebenfalls hilfreich mit einem *Short Math Guide* for LaTeX.

5.2 Sonderzeichen in einer *math*-Umgebung

wurden teilweise bereits in Abschnitt 4.2 (ab Seite 57) aufgelistet, zum Beispiel Pfeile. Man kann zahlreiche Sonderzeichen schreiben (zum Beispiel griechische Buchstaben, siehe oben), die bei der normalen Texteingabe fehlen. Der Quellcode

```
es gelten $c^2=a^2+b^2-2\,a\,b \cos\gamma$ und $\uppi \approx 3{,}14$
```

ergibt als gedruckte Zeile (der Befehl *\uppi* benötigt das Paket *upgreek*)

es gelten $c^2 = a^2 + b^2 - 2\,a\,b\cos\gamma$ und $\pi \approx 3{,}14$

Es folgen weitere Beispiele. Vorausgesetzt wird das Paket *amsmath*.

Mathematische Operatoren:

–	±	·	∗	×	÷	⊗	⊕
--	$\pm$	$\cdot$	$\ast$	$\times$	$\div$	$\otimes$	$\oplus$

Brüche, Summen und Produkte:

$\frac{a}{b}$	$^a/_b$	$\sum_{i=1}^{3} x_i$	$\prod_{1}^{3} i$
$\frac{a}{b}$	$\^a/_b$	$\sum_{i=1}^{3} x_i$	$\prod_{1}^{3} i$

Wurzel, Exponentialfunktion, Binomialkoeffizient und Grenzwert:

$\sqrt[2]{x}$	x^{88}	$\binom{n}{k}$	$\lim_{x\to\infty}$
$\sqrt[2]{x}$	$\x^{88}$	$\binom{n}{k}$	$\lim_{x \to \infty}$

Index, Exponentialfunktion, Logarithmen, Minimum, Maximum, modulo:

x_1	exp	log	ln	min	max	mod
x_{1}	$\exp$	$\log$	$\ln$	$\min$	$\max$	$\mod$

Differentiale und Differentialquotienten:

$\mathrm{d}x$	$\frac{\mathrm{d}y}{\mathrm{d}x}$	∂
`$\mathrm{d}x$`	`$\frac{\mathrm{d}y}{\mathrm{d}x}$`	`$\partial$`

Integrale (*iint* und *iiint* sind nur mit *\usepackage{amsfonts}* verfügbar):

$\int_{t_0}^{t_{max}} t^2\,\mathrm{d}t$	$\oint$	$\iint$	$\iiint$	
`$\int_{t_0}^{t_max} t^2\,\mathrm{d}t$`	`$\oint$`	`$\iint$`	`$\iiint$`	

Mit dem *limits*-Befehl können die oberen und unteren Grenzen eines Integrals über und unter, statt rechts vom Integralzeichen gesetzt werden. Beispiel:

```
$\int\limits_{a}^{b} f(x) \,\mathrm{d}x$
```

$$\int\limits_{a}^{b} f(x)\,\mathrm{d}x$$

Entsprechend ist dies auch mit Summen und Produkten möglich.

Integralzeichen werden standardmäßig nach rechts lehnend gezeichnet, wie im Englischen üblich. Will man aufrechte Integralzeichen entsprechend der deutschen Tradition, kann das Paket *cmupint* von Uroš Stefanović helfen.

Relationen:

$\neq$	$\leq$	$\geq$	$\ll$	$\gg$	$\approx$	$\equiv$
`$\neq$`	`$\le$`	`$\ge$`	`$\ll$`	`$\gg$`	`$\approx$`	`$\equiv$`
$\sim$	$\nsim$	$\propto$	$\in$	$\notin$	$\ni$	$\ngtr$
`$\sim$`	`$\nsim$`	`$\propto$`	`$\in$`	`$\notin$`	`$\ni$`	`$\ngtr$`

Mit `$x\stackrel{def}{=}y$` setzt man eine Zeichenkette auf eine Relation: $x \stackrel{def}{=} y$. Die Definition eines entsprechenden Befehls kann in der Präambel erfolgen.

Logik:

$\land$	$\lor$	$\forall$	$\exists$	$\exists!$	$\nexists$
`$\land$`	`$\lor$`	`$\forall$`	`$\exists$`	`$\exists!$`	`$\nexists$`
$\Leftrightarrow$	$\iff$	$\implies$	$\to$	$\gets$	$\lnot$
`$\Leftrightarrow$`	`$\iff$`	`$\implies$`	`$\to$`	`$\gets$`	`$\lnot$`

Eine negierte Form des Symbols $\implies$ (\implies) erhält man mithilfe des Pakets *centernot* von Heiko Oberdiek. `$\centernot\implies$` ergibt $\centernot\implies$.

Geometrie:

$\triangle$	$\square$	$\parallel$	$\perp$	$\angle$	$\measuredangle$
`$\triangle$`	`$\square$`	`$\parallel$`	`$\perp$`	`$\angle$`	`$\measuredangle$`

Mengenlehre:

$\cup$	$\cap$	$\setminus$	$\subseteq$	$\not\supset$	$\emptyset$
`$\cup$`	`$\cap$`	`$\setminus$`	`$\subseteq$`	`$\not\supset$`	`$\emptyset$`

Trigonometrische Funktionen:

sin	arccos	cot	arctan	sinh	sec	csc
`$\sin$`	`$\arccos$`	`$\cot$`	`$\arctan$`	`$\sinh$`	`$\sec$`	`$\csc$`

Die Funktion arccot fehlt in der *math*-Umgebung.

Komplexe Zahlen:

$r\angle\varphi$	$\Re$	$\Im$	$\lvert z\rvert = 1$	$\overline{a+b}$
$r \angle \varphi$	$\Re$	$\Im$	$\left\lvert z \right\rvert=1$	$\overline{a+b}$

Die imaginäre Einheit ist i mit i oder j mit j.

Weitere mathematische Symbole:

∇	Δ	∞	$\vert$	$\Vert$	$\mid$	$\hbar$
∇	Δ	∞	$\vert$	$\Vert$	$\mid$	$\hbar$

Betrag und Norm:

$\lvert +1 \rvert = \lvert -1 \rvert$	$\lVert -x \rVert = \lVert x \rVert$
$\lvert +1 \rvert = \lvert -1 \rvert$	$\lVert -x \rVert = \lVert x \rVert$

Diakritische Zeichen:

$\acute{x}$	$\grave{x}$	$\bar{x}$	$\check{x}$
$\acute{x}$	$\grave{x}$	$\bar{x}$	$\check{x}$
$\tilde{x}$	$\widetilde{abcde}$	$\hat{x}$	$\widehat{abcde}$
$\tilde{x}$	$\widetilde{abcde}$	$\hat{x}$	$\widehat{abcde}$
$\dot{x}$	$\ddot{x}$	$\vec{x}$	$\overrightarrow{AB}$
$\dot{x}$	$\ddot{x}$	$\vec{x}$	$\overrightarrow{AB}$

Weitere „Bedeckungen“ sind $\dddot{x}$, $\dot{\vec{x}}$ mit $\dot{\vec{x}}$ und $\ddot{\vec{x}}$ mit $\ddot{\vec{x}}$, $\breve{x}$ mit $\breve{x}$, sowie $\mathring{x}$ mit $\mathring{x}$. Mithilfe des *ddot*-Befehls kann man auch deutsche Umlaute in einer *math*-Umgebung bilden (obwohl das nicht empfehlenswert ist):

```
$\ddot{A}\ddot{O}\ddot{U}\ddot{a}\ddot{o}\ddot{u}$
```

$\ddot{A}\ddot{O}\ddot{U}\ddot{a}\ddot{o}\ddot{u}$

Für diakritische Zeichen auf i und j gibt es diese Buchstaben auch ohne Punkt: $\imath$ mit $\imath$ und $\jmath$ mit $\jmath$.

Weitere diakritische Zeichen sind zum Beispiel mit dem Paket *wsuipa* (ursprünglich von Janene Winter und betreut von Dean Guenther) möglich. Schönere Vektorpfeile stellt das Paket *esvect* von Eddie Saudrais bereit.

Überstreichungen können mit dem *overline*-Befehl oder dem *bar*-Befehl erzeugt werden. Während *overline* die als Argument übergebene Zeichenkette auf voller Länge überstreicht, ergibt *bar* einen Balken fester Länge. Kleinliche Menschen wollen vielleicht den Überstrich von *overline* etwas kürzen. Das geht zum Beispiel so:

```
\newcommand{\kurzoverline}[1]{\mkern %
2mu\overline{\mkern-2mu#1\mkern-2mu}\mkern 2mu}
$\bar{M}$ ~ $\overline{M}$ ~ $\kurzoverline{M}$
```

$\bar{M}$ ~ $\overline{M}$ ~ $\kurzoverline{M}$

Mit *\usepackage{amsfonts}* verfügbare Symbole für Zahlenmengen:

$\mathbb{N}$	$\mathbb{Z}$	$\mathbb{R}$	$\mathbb{C}$
$\mathbb{N}$	$\mathbb{Z}$	$\mathbb{R}$	$\mathbb{C}$

Mit dem Paket *amsfonts* verfügbare hebräische Buchstaben und andere Symbole:

ℵ	ℶ	ℷ	ℸ	⊡	⊟
$\aleph$	$\beth$	$\gimel$	$\daleth$	$\boxdot$	$\boxminus$
⊞	⊠	⋐	⋑	⋒	⋓
$\boxplus$	$\boxtimes$	$\Subset$	$\Supset	$\Cap	$\Cup$
⊚	≦	≧	≨	∄	∦
$\circledcirc$	$\leqq$	$\geqq$	$\lneqq	$\nexists	$\nparallel$

Ein Doppelpunkt kann als Relation $a : b$ mit $a:b$ geschrieben werden oder mit anderem Zeichenabstand als Interpunktion $f\colon x \to y$ mit $f\colon x\to y$. Mathematische Symbole, die einen Doppelpunkt enthalten, werden schön mit dem Paket *colonequals* von Heiko Oberdiek dargestellt, sowohl in normalem Text als in einer *math*-Umgebung (als Relation). Wer den Unterschied zwischen := und := nicht sieht, braucht das Paket natürlich nicht. Beispiele:

::	:=	=:	:−	:≈
\coloncolon	\colonequals	\equalscolon	\colonminus	\colonapprox

Das Paket *shuffle* von Antoine Lejay und Julian Gilbey definiert zwei Symbole für das Shuffle-Produkt, welches bei algebraischen Rechnungen verwendet wird.

```
Shuffle-Produkt \quad $\shuffle$
Vollst. Shuffle-Produkt \quad $\cshuffle$
```

Shuffle-Produkt ⧢
Vollst. Shuffle-Produkt ⧣

Halloween hat eine beliebte Symbolik, mit der sich auch fürchterliche Mathematik betreiben lässt. Dazu dient das Paket *halloweenmath* von G. Mezzetti. Vorausgesetzt, die nötigen Pakete können mit

```
\usepackage[pdftex]{pict2e}
\usepackage{halloweenmath}
```

in der Präambel (nach dem Paket *amsmath*) geladen werden, stehen die im folgenden Beispiel gezeigten gruseligen Symbole in einer *math*-Umgebung zur Verfügung.

```
Kürbisse: $\pumpkin+\pumpkin=\bigpumpkin$
\par\smallskip $\mathwitch f(x^2)=y \quad
a+b\rightbroom c$ \par\medskip$\bigskull$
\quad $\xrightswishingghost{\text{Geist}}
\quad\mathghost$\par\smallskip$\mathcloud
\quad\mathbat\quad \mathwitch*$ mit Katze
```

Kürbisse: 🎃 + 🎃 = 🎃
$f(x^2) = y \quad a + b \rightbroom c$
💀 Geist 👻
☁ 🦇 mit Katze

Das Paket enthält noch weitere Varianten der soeben vorgestellten Symbole.

Weiterhin gibt es das Paket *skull* von Henrik Christian Grove, welches in einer *math*-Umgebung mit dem Befehl `\skull` ein Totenkopfsymbol zeichnet.

5.3 Gleichungen

erlauben die Einbettung komplexer mathematischer Aussagen in das Dokument. Dafür wird mit *\begin{equation}* und *\end{equation}* eine *equation*-Umgebung gebildet, innerhalb der die Schreibweise der *math*-Umgebung entspricht, die in Abschnitt 5.1 (Seite 67) beschrieben wird. Da Gleichungen in einer eigenen Zeile stehen, können sie umfangreichere mathematische Ausdrücke wiedergeben. Einige Befehle, zum Beispiel $\sum_{u}^{o}, werden in einer Gleichung ausladender geschrieben als in einer *math*-Umgebung.

Gleichungen werden automatisch nummeriert. Wurde das Paket *amsmath* mit *\usepackage{amsmath}* geladen, hat man zusätzliche mathematische Schreibweisen zur Verfügung und kann auch nicht nummerierte Gleichungen bilden, indem man ein $*$ an *equation* hängt. Folgende Beispiele verwenden einige der oben aufgeführten mathematischen Sonderzeichen.

```
\begin{equation*}   % nicht nummerierte Gleichung
\text{Es sei } \Delta x=\left(\frac{b-a}{n}\right) \end{equation*}
```

ergibt als Druckbild:

$$\text{Es sei } \Delta x = \left(\frac{b-a}{n}\right)$$

Durch den Befehl *\text* kann normaler Text eingebettet werden. Mit *\left(* und *\right)* werden runde Klammern gesetzt, deren Größe automatisch an den Inhalt angepasst wird. Entsprechendes gilt für eckige [ ] und geschweifte Klammern { }.

```
\begin{equation}\label{riemann}   % nummerierte Gleichung mit Marke
\int_{a}^{b} f(x)\,\mathrm{d}x=
\lim_{n\to\infty} \sum_{k=1}^{n} f(x_k)\,\Delta x
\end{equation}
```

ergibt als Druckbild:

$$\int_a^b f(x)\,\mathrm{d}x = \lim_{n\to\infty} \sum_{k=1}^{n} f(x_k)\,\Delta x \tag{5.1}$$

Hier wurde zum Trennen von Faktoren mit \, ein schmales Leerzeichen gesetzt. Mithilfe der Marke, die mit dem Befehl \label{riemann} gebildet wurde, kann später auf die Gleichung verwiesen werden. Der Quellcode

```
siehe Gleichung \eqref{riemann} zur Flächenberechnung
```

ergibt

> siehe Gleichung (5.1) zur Flächenberechnung

Es gibt verschiedene Möglichkeiten mehrzeilige Gleichungen zu schreiben. Man benötigt dafür das Paket *amsmath* und in allen Fällen gilt: Mit \\ wird eine Zeile beendet und in der nächsten Zeile eine weitere Gleichung begonnen. In der letzten Zeile wird \\ weggelassen.

Ist eine Gleichung länger als eine Zeile, kann man sie (statt in eine *equation*- oder *equation*$*$-Umgebung) in eine *multline*- (für eine nummerierte Gleichung) beziehungsweise *multline*$*$-Umgebung (ohne Nummerierung) stellen. Die erste Zeile wird linksbündig, die folgenden zentriert, die letzte rechtsbündig gesetzt. Beispiel:

```
\begin{multline}
y(x)=88x^8+7x^7+666x^6+5x^5+
42x^4\\+33x3+22x^2+111x-123
\end{multline}
```

$$y(x) = 88x^8 + 7x^7 + 666x^6 + 5x^5 + 42x^4 + 33x3 + 22x^2 + 111x - 123 \quad (5.2)$$

Eine Alternative ist die *split*-Umgebung. Sie wird in eine *equation*- oder *equation*∗-Umgebung gesetzt. Hier kann man die Ausrichtung steuern, indem man die Stellen, welche direkt untereinander stehen sollen, mit dem Steuerzeichen & markiert. In folgendem Beispiel werden die Zeilen am Zeichen = ausgerichtet.

```
\begin{equation} \begin{split}
y&=x^3\,(23x^8+13x^4+18x^2+21)\\
 &=21x^3+18x^5+13x^7+23x^{11}
\end{split} \end{equation}
```

$$\begin{aligned} y &= x^3\,(23x^8 + 13x^4 + 18x^2 + 21) \\ &= 21x^3 + 18x^5 + 13x^7 + 23x^{11} \end{aligned} \quad (5.3)$$

Um mehrere Gleichungen zentriert untereinander zu schreiben, kann man sie in eine *gather*-Umgebung (mit Nummerierung) oder eine *gather*∗-Umgebung (ohne Nummerierung) setzen. Beispiel:

```
\begin{gather}
5 x_2 + 7 x_3 = 9 k \\
3 x_1 + 5 x_2 = x_3 + 10
\end{gather}
```

$$5x_2 + 7x_3 = 9k \quad (5.4)$$

$$3x_1 + 5x_2 = x_3 + 10 \quad (5.5)$$

Sollen mehrere Gleichungen untereinander geschrieben und so ausgerichtet werden, dass beispielsweise die Gleichheitszeichen untereinander stehen, verwendet man eine *align*-Umgebung (mit Nummerierung) oder eine *align*∗-Umgebung (ohne). Beispiel:

```
\begin{align}
            k & = 8 \nonumber \\
2 x_1 + 8 x_3 & = 3 k \\
5 x_2 + 7 x_3 & = 9 k \\
3 x_1 + 5 x_2 & = x_3 + 10
\end{align}
```

$$\begin{aligned} k &= 8 \\ 2x_1 + 8x_3 &= 3k \quad (5.6) \\ 5x_2 + 7x_3 &= 9k \quad (5.7) \\ 3x_1 + 5x_2 &= x_3 + 10 \quad (5.8) \end{aligned}$$

Die Ausrichtung erfolgt an den Zeichen, welche dem Steuerzeichen & folgen. Der Befehl \nonumber am Ende einer Zeile verhindert deren Nummerierung.

Das Paket *witharrows* von François Pantigny erlaubt es, untereinander geschriebene Gleichungen ohne Nummerierung mit einem Pfeil am rechten Rand und einer Erklärung zu versehen, wie folgendes Beispiel zeigt.

```
$\begin{WithArrows}
x & = (a + b)^2
\Arrow{ausmultiplizieren} \\
  & =  a^2 + 2ab + b^2
\end{WithArrows}$
```

$$\begin{aligned} x &= (a+b)^2 \\ &= a^2 + 2ab + b^2 \end{aligned} \quad \Big) \textit{ausmultiplizieren}$$

Da die Pfeile mit TikZ (siehe unten) gezeichnet werden, können sie (ebenso wie der erklärende Text) anders (beispielsweise farbig) gestaltet werden.

Mit dem Paket *framed* von Donald Arseneau (siehe Abschnitt 3.2 auf Seite 41) können auch Gleichungen eingerahmt (*framed*-Befehl) oder farbig hinterlegt werden (*snugshade*-Befehl oder *shaded*-Befehl). Für letzteres muss man mit dem Befehl

```
\definecolor{shadecolor}{named}{yellow} % Farbe für shaded-Umgebungen
```

in der Präambel, nach Laden des ebenfalls benötigten Pakets *xcolor*, die Hintergrundfarbe (hier: gelb) festlegen. Beispiele:

```
\begin{framed} \begin{equation}
m = \frac{m_0}{\sqrt{1 - (v/c)^2}}
\end{equation} \end{framed}
```

$$m = \frac{m_0}{\sqrt{1 - (v/c)^2}} \tag{5.9}$$

```
\begin{snugshade} \begin{equation}
m = \frac{m_0}{\sqrt{1 - (v/c)^2}}
\end{equation} \end{snugshade}
```

$$m = \frac{m_0}{\sqrt{1 - (v/c)^2}} \tag{5.10}$$

6 Grafiken mit TikZ

Innerhalb eines LaTeX-Doluments kann man mit entsprechenden Befehlen Vektorgrafiken erzeugen, wenn das Paket *tikz* von Till Tantau und Christian Feuersänger mit *\usepackage{tikz}* geladen wurde. TikZ stützt sich auf das Graphikpaket PGF und ist sehr umfangreich. Hier folgt nur eine Einführung anhand von Beispielen, aber es gibt zum Glück eine hervorragende Anleitung.

TikZ stellt eine Zeichenfläche bereit, die durch ein zweidimensionales kartesisches Koordinatensystem beschrieben wird, dessen Ursprung (0,0) in der linken unteren Ecke der Fläche liegt. Wenn nichts anderes angegeben wird, ist 1 cm die Längeneinheit. Man sollte bei der Platzierung grafischer Elemente andere Längeneinheiten möglichst vermeiden und die Gesamtgröße der Grafik bei Bedarf durch Skalierung mit der Option *scale* anpassen. Sehr kleine Grafiken, zum Beispiel ●, können mit \tikz{ ... } in eine Textzeile eingebaut werden:

```
... Beispiel \tikz{\fill[red] (0,0) circle[radius=1mm];}, können ...
```

Das Paket *tikz* stellt die Farben des Pakets *xcolor* zur Verfügung (siehe Abschnitt 4.1 auf Seite 54). Aufwendigere Grafiken werden übersichtlicher in einer Umgebung beschrieben, welche mit *\begin{tikzpicture}* beginnt und mit *\end{tikzpicture}* endet.

6.1 Strichzeichnungen und Füllungen

In folgendem Beispiel wird mit drei *draw*-Befehlen eine Grafik gezeichnet. Jeder *draw*-Befehl wird mit einem Semikolon abgeschlossen. Der erste *draw*-Befehl zeichnet eine einzelne Strecke, der zweite einen Pfad, der aus zwei Strecken zusammengesetzt ist. Die Zahlenpaare in runden Klammern geben kartesische Koordinaten an. Ohne Option zur Linienart erzeugt -- eine Strecke mit durchgezogener Linie. Die Option |-> in eckigen Klammern nach *draw* erzeugt einen Querstrich am Anfang und einen Rechtspfeil am Ende des Pfades. Der dritte *draw*-Befehl zeichnet einen Kreis, dessen Mittelpunkt bei (5,2) liegt und dessen Radius 18 mm beträgt.

```
\begin{tikzpicture} [thick, scale=0.3]
\draw (0,2) -- (2,2) ;
\draw[|->] (0,0)--(1,4)--(2,0) ;
\draw (5,2) circle (18mm);
\end{tikzpicture}
```

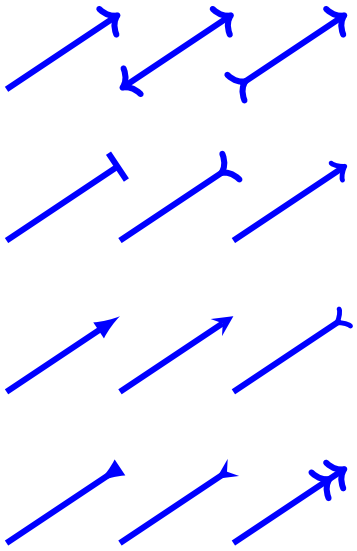
In den eckigen Klammern nach *\begin{tikzpicture}* stehen Optionen, mit denen das Erscheinungsbild verändert wird: *thick* ergibt dickere Striche und mit *scale* wird ein Maßstabsfaktor angegeben, um die Gesamtgröße der Grafik anzupassen.

Die Pfeilspitzen sind in der Standardeinstellung von TikZ ziemlich klein. Ihre Größe wird zwar an die Linienbreite angepasst, aber man sollte doch in der Präambel, nach *\usepackage{tikz}*, die Zeile

```
\usetikzlibrary{arrows.meta}
```

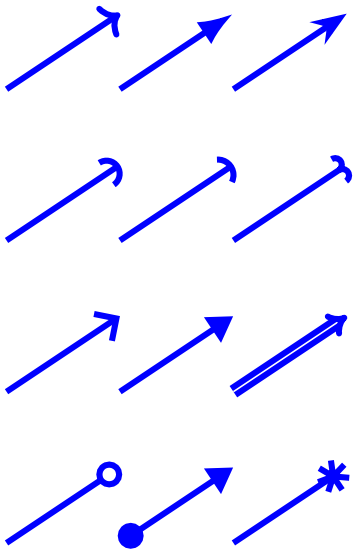
einfügen (dies wurde im vorliegenden Manuskript getan) und erhält dann einige etwas größere Pfeilspitzen. Folgendes Beispiel zeigt diejenigen Pfeile, welche TikZ bereits ohne *arrows.meta* bereitstellt, deren Spitzen aber mit *arrows.meta* teilweise größer gezeichnet werden.

```
\begin{tikzpicture} [line width=0.8mm,blue]
\draw[->] (0,0) -- (1.5,1) ;
\draw[<->] (1.5,0) -- (3,1) ;
\draw[>->] (3,0) -- (4.5,1) ;
\draw[-|] (0,-2) -- (1.5,-1) ;
\draw[-<] (1.5,-2) -- (3,-1) ;
\draw[-to] (3,-2) -- (4.5,-1) ;
\draw[-latex] (0,-4) -- (1.5,-3) ;
\draw[-stealth] (1.5,-4) -- (3,-3) ;
\draw[-to reversed] (3,-4) -- (4.5,-3);
\draw[-latex reversed] (0,-6)--(1.5,-5);
\draw[-stealth reversed] (1.5,-6)--(3,-5);
\draw[->>] (3,-6) -- (4.5,-5) ;
\end{tikzpicture}
```



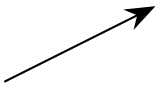
Außerdem hat man mit *arrows.meta* mehr Pfeilarten, zum Beispiel:

```
\begin{tikzpicture} [line width=0.8mm,blue]
\draw[-To] (0,0) -- (1.5,1) ;
\draw[-Latex] (1.5,0) -- (3,1) ;
\draw[-Stealth] (3,0) -- (4.5,1) ;
\draw[-Arc Barb] (0,-2) -- (1.5,-1) ;
\draw[-Parenthesis] (1.5,-2) -- (3,-1) ;
\draw[-Hooks] (3,-2) -- (4.5,-1) ;
\draw[-Straight Barb] (0,-4) -- (1.5,-3) ;
\draw[-Triangle] (1.5,-4) -- (3,-3) ;
\draw[-Implies,double] (3,-4) -- (4.5,-3);
\draw[-{Circle[fill=none]}] (0,-6)--(1.5,-5);
\draw[Circle-Triangle] (1.5,-6)--(3,-5);
\draw[-{Rays[n=7]}] (3,-6) -- (4.5,-5) ;
\end{tikzpicture}
```



Überdies kann man die Pfeilspitzen noch selbst verändern, etwa so:

```
\begin{tikzpicture}[thick]
\draw[ ->,
>={ Stealth[width=3mm,length=4mm] } ]
(0,0) -- (2,1) ;
\end{tikzpicture}
```



Die Option *Stealth* gibt die Form des Pfeils an; eine Alternative ist *Latex*. Die Parameter *width* und *length* kann man natürlich nach Belieben ändern.

Für die Linienstärke sind folgende Namen möglich: ultra thin, very thin, thin, semithick, thick, very thick und ultra thick. Alternativ kann die Linienstärke auch durch einen Wert festgelegt werden, zum Beispiel mit *line width=2mm*.

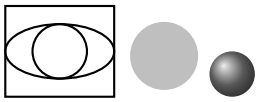
Ein Pfad oder *path* ist eine Aneinanderreihung von Strecken oder (mit aufwändigerer Formatierung) kurvenförmigen Linien. Mit *cycle* kann er zu seinem Anfang zurückgeführt werden und bildet dann einen Zyklus. Die von einem Zyklus eingeschlossene Fläche kann gefärbt werden. Beispiel:

```
\begin{tikzpicture} [thick, scale=0.8]
\path[fill=orange]
 (0,0)--(0,2)--(2,1)--(2,0)--cycle;
\path[draw, line width=3pt, shade]
 (3,0)--(5,2)--(6,0)--cycle;
\end{tikzpicture}
```

Der links gezeigte trapezförmige Zyklus hat keine Umrandung, da die Option *draw* fehlt. Die Option *fill* färbt das Trapez einheitlich in der angegebenen Farbe (orange), während *shade* den grauen Farbverlauf in der rechts gezeigten Fläche erzeugt. Mit der Option *draw* werden die Grenzlinien der Dreiecksfläche gezeichnet. Die Liniendicke kann mit der Option *line width* bestimmt werden, hier in der Längeneinheit pt. Mehr zu Längeneinheiten findet man in Abschnitt 4.1 auf Seite 51.

Nun zeichnen wir ein Rechteck, dessen untere linke Ecke bei (0,0) ist, und dessen obere rechte Ecke bei (48 mm, 40 mm). Darin liegt eine Ellipse mit Mittelpunkt (2.4,2), horizontalem Radius 2.4 und vertikalem Radius 1.2. Ganz innen befindet sich ein Kreis mit Mittelpunkt bei (2.4,2) und einem Radius von 1.2 . Weiterhin wird mit dem *fill*-Befehl ein farbig ausgefülltes Objekt, hier ein Kreis, dargestellt. Schließlich erzeugen wir mit dem *shade*-Befehl einen Farbverlauf, der hier einen grauen Ball erscheinen lässt.

```
\begin{tikzpicture} [thick, scale=0.3]
\draw (0,0) rectangle (48mm,40mm) ;
\draw (2.4,2) ellipse (24mm and 12mm) ;
\draw (2.4,2) circle (12mm) ;
\fill [lightgray] (7,1.8) circle (15mm) ;
\shade[ball color=gray] (10,1) circle (1);
\end{tikzpicture}
```



Ein Kreisbogen (englisch *arc*) wird zum Beispiel so (hier als Pfeil) gezeichnet:

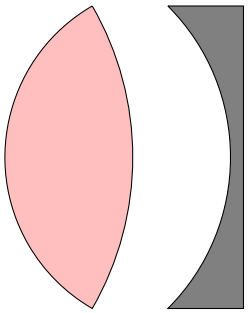
```
\begin{tikzpicture} \draw[->] (0,0) arc [
start angle=10, end angle=330,
radius=5mm ] ; \end{tikzpicture}
```

In der runden Klammer stehen die x- und y-Koordinate des Anfangspunkts des Bogens. Der Anfangswinkel *start angle* gibt den Winkel der Strecke an, die Kreismittelpunkt und Anfangspunkt des Bogens verbindet (auch wenn die Strecke nicht gezeichnet wird). Dabei wird der Winkel mathematisch positiv (linksdrehend) von der x-Achse aus gemessen. Entsprechendes gilt für den Endwinkel *end angle*. Der Radius *radius* wird, wie üblich, als Länge gegeben. Statt des Endwinkels kann auch die Differenz von End- und Anfangswinkel *delta angle* angegeben werden. Eine Kurzschreibweise für obigen Zeichenbefehl ist: `\draw[->] (0,0) arc (10:330:5mm);`

Im folgenden Beispiel werden zwei optische Linsen gezeichnet. Beide haben einen vertikalen Durchmesser von 4 cm. Links ist eine sphärische bikonvexe Sammellinse mit $R_1 = {}^4\!/_3\sqrt{3}\,\mathrm{cm}$ und $R_2 = -4\,\mathrm{cm}$, rechts eine sphärische konkavplane Zerstreuungslinse mit $R_1 = -2\sqrt{2}\,\mathrm{cm}$ (gemäß Linsenmachergleichung, Vorzeichen nach

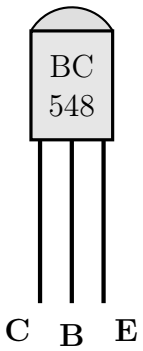
Abbe). Beide Linsen werden als geschlossener Linienzug gezeichnet, der rosa beziehungsweise grau gefüllt wird.

```
\begin{tikzpicture}
\draw [fill=pink] (0,2) arc[start angle=120,%
   end angle=240,radius=(4/3)*sqrt(3)]
arc[start angle=-30,%
   end angle=30,radius=4];
\draw [fill=gray] (1,2)--(2,2)--(2,-2)--(1,-2)
arc[start angle=-45,%
   end angle=45,radius=2*sqrt(2)];
\end{tikzpicture}
```



Das folgende Beispiel zeigt die drei Anschlüsse eines Bipolartransistors BCE 548: Basis (B), Collektor (C) und Emitter (E). Teil des Bilds ist ein Kreissegment mit einem Mittelpunktswinkel von 120°. Der Kreisbogen beginnt rechts mit einem Winkel von 30° und endet links mit einem Winkel von 150°. Der Wert 10,392 305 mm für die Länge des Radius ergibt sich aus der der halben Länge der Kreissehne, 9 mm, geteilt durch sin(60°).

```
\begin{tikzpicture}[scale=0.6]
\fill[fill=gray!25!white] (9mm,12mm) arc [start angle=30,
   end angle=150, radius=10.392305mm] -- cycle;
\draw[thick] (9mm,12mm) arc [start angle=30,
   end angle=150, radius=10.392305mm] ;
\draw[very thick,fill=gray!20!white]
   (-9mm,-12mm) rectangle (9mm,12mm);
\draw[ultra thick] (-7mm,-12mm) -- (-7mm,-48mm) ;
\draw[ultra thick] (0,-12mm) -- (0,-48mm) ;
\draw[ultra thick] (7mm,-12mm) -- (7mm,-48mm) ;
\node[align=center] at (0,0) {BC\\548};
\node at (-12mm,-54mm) {\textbf{C}};
\node at (0,-55mm) {\textbf{B}};
\node at (12mm,-54mm) {\textbf{E}};
\end{tikzpicture}
```



Die Verbindung zwischen zwei Punkten muss keine gerade Strecke, sondern kann eine gekrümmte Linie sein. In folgendem Beispiel werden die Winkel, unter denen eine Linie ihren Anfangspunkt verlässt beziehungsweise ihren Endpunkt erreicht, mit einer Winkelangabe in Grad vorgegeben. Da -- stets eine gerade Strecke ergibt, schreibt man stattdessen *to* zwischen Anfangs- und Endpunkt.

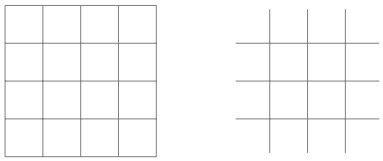
```
\begin{tikzpicture} [thick]
\draw[out=90, in=90, dashed] (0,0) to (2,1) ;
\draw[out=30, in=-90, dotted] (1,0) to (3,1) ;
\end{tikzpicture}
```

Die Optionen *dashed* und *dotted* erzeugen eine gestrichelte beziehungsweise gepunktete Linie. Weitere Linienarten sind *loosely dashed*, *densely dashed*, *loosely dotted* und *densely dotted*.

Bei der Entwicklung zusammengesetzter Grafiken kann es hilfreich sein, ein Raster mit definierten Abständen zu zeigen. Vor und nach *grid* werden die Koordinaten der

linken unteren beziehungsweise der rechten oberen Ecke festgelegt. Die Schrittweite der Rasterlinien wird mit der Option *step* angegeben.

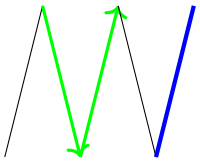
```
\begin{tikzpicture}[help lines]
\draw [ step=0.5]
(-1,-1) grid (1,1) ;
\draw [ step=0.5]
(2.05,-0.95) grid (3.95,0.95) ;
\end{tikzpicture}
```



Im Beispiel hat das linke Raster einen Außenrand, das rechte jedoch nicht, weil deren Begrenzungen keine Vielfachen der Schrittweite sind. Die Option *help lines* bewirkt, dass Linien dünn und grau gezeichnet werden.

In folgendem Beispiel wird die Formatierung der Grafik in einem Teilbereich geändert. Dieser beginnt mit *\begin{scope}* und endet mit *\end{scope}*.

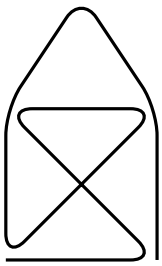
```
\begin{tikzpicture}[thin, scale=0.5]
\draw (0,0) -> (1,4) ;
\begin{scope}[very thick, green, ->]
\draw (1,4) -- (2,0) ;
\draw (2,0) -- (3,4) ;
\end{scope}
\draw (3,4) -- (4,0) ;
\draw[ultra thick, blue] (4,0) -- (5,4) ;
\end{tikzpicture}
```



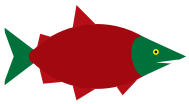
In eckigen Klammern stehen nach *\begin{scope}* Optionen, die nur im Teilbereich gelten sollen (hier *very thick*, *green* und *->* für eine dicke grüne Linie mit Pfeilspitze rechts). Die Optionen *ultra thick* und *blue* (die, in eckigen Klammern, direkt hinter einem *draw*-Befehl stehen) wirken nur auf die Strecke, die durch diesen *draw*-Befehl beschrieben wird.

Relative Koordinaten, bezogen auf die letzte Position, können angegeben werden, indem zwei Pluszeichen vor das Koordinatenpaar gesetzt werden. Mit *(1,1) -- ++(1,1)* wird eine Linie von (1,1) zum Punkt (2,2) gezogen. Folgendes Beispiel zeigt eine Strichzeichnung mit gerundeten Ecken.

```
\begin{tikzpicture}[
very thick,
rounded corners=10pt,
scale=0.5 ]
\draw (0,0)  --  ++(4,0) -- ++(-4,4)
-- ++(4,0) -- ++(-4,-4) -- ++(0,4)
-- ++(2,3) -- ++(2,-3) -- ++(0,-4) ;
\end{tikzpicture}
```



Bei manchen Zeichnungen, wie zum Beispiel beim nebenstehenden Rotlachs (Nerka), überlappen sich verschiedene Farbflächen. Ein Punkt, der in mehreren Flächen liegt, erhält nicht eine Mischfarbe, sondern die Farbe der zuletzt erzeugten Fläche. Das Auge des Fisches wurde als gelbe Kreisfläche auf einen grünen Hintergrund gesetzt und die schwarze Pupille wiederum auf den gelben Kreis.



6.2 Knoten

Jedem Ort im Koordinatensystem kann ein Objekt, Knoten (englisch: node) genannt, zugeordnet werden. Jeder Knoten hat Koordinaten und eine geometrische Form (englisch: shape). Verfügbar sind immer Rechteck (rectangle), Kreis (circle) und Punkt (coordinate). Ohne Abgabe der Form wird ein Rechteck verwendet. Weitere Formen sind mit der Bibliothek *shapes* verfügbar. Die Form eines Knotens wird nur gezeichnet, wenn dies ausdrücklich befohlen wird.

Ein Knoten kann benannt werden, um ihn später (ein weiteres Mal) zu benutzen. Allen Knoten muss eine Zeichenkette (Text) zugeordnet werden, die, außer beim Punkt, in der Mitte innerhalb des Knotens gezeigt wird. Sie kann aber leer sein und dann wird nichts gezeigt (das ist die Logik der Mathematiker). Es gibt noch viele weitere Eigenschaften von Knoten, die wir aber erst später betrachten.

Ein Punkt (coordinate) namens *rudi* wird durch

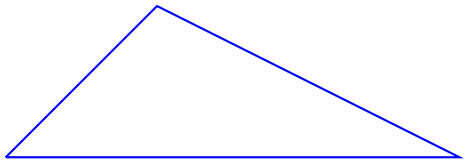
```
\node[shape=coordinate] (rudi) at (-1,2) {} ;
```

mit den Koordinaten (-1,2) definiert. In den geschweiften Klammern steht eine leere Zeichenkette. Bei anderen Knotenformen könnte hier ein Text stehen. Eine abgekürzte Schreibweise für obigen Befehl ist

```
\coordinate (rudi) at (-1,2) ;
```

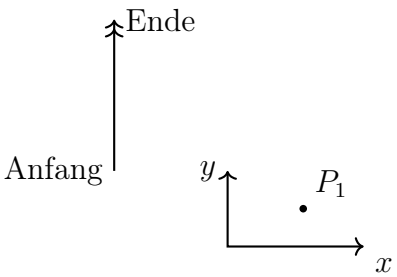
Man kann zum Beispiel die Eckpunkte eines Dreiecks erst definieren und danach das Dreieck zeichnen:

```
\begin{tikzpicture} [blue,thick]
\coordinate (A) at (-3,0) ;
\coordinate (B) at (3,0) ;
\coordinate (rudi) at (-1,2) ;
\draw (A)--(B)--(rudi)--(A) ;
\end{tikzpicture}
```



Mit jedem Knoten, außer einem Punkt (coordinate), wird eine Zeichenkette geschrieben. Die Position der Zeichenkette kann aus der Mitte des Knotens (center) verschoben werden. Dies soll anhand einer (mit Doppelpfeil versehenen) Strecke gezeigt werden, deren Anfang und Ende wir beschriften, sowie anhand eines Textknotens in einem kleinen Achsenkreuz.

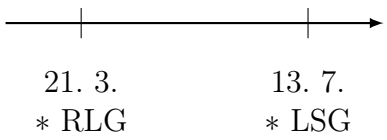
```
\begin{tikzpicture} [thick]
\draw [->>] (0,1) -- (0,3) ;
\node [left] at (0,1) {Anfang} ;
\node [right] at (0,3) {Ende} ;
\draw [<->] (1.5,1)--(1.5,0)--(3.3,0);
\node [below right] at (3.3,0) {$x$};
\node [left] at (1.5,1) {$y$};
\fill (2.5,0.5) circle (0.5mm) ;
\node[above right] at (2.5,0.5) {$P_1$};
\end{tikzpicture}
```



Die Beschriftung eines Knotens erfolgt oft nach oben (*above*), unten (*below*), links (*left*) oder rechts (*right*) versetzt, um die Linie oder das Objekt am Ort des Knotens nicht zu überschreiben.

Mehrzeilige Beschriftungen, wie in folgendem Beispiel, sind mithilfe von \\ möglich, wenn die Ausrichtung bestimmt wird (*align=center* oder *align=left*).

```
\begin{tikzpicture}
\draw[-latex,thick] (0,0)--(5,0);
\draw (1,-.2)--(1,.2) (4,-.2)--(4,.2);
\node[align=center,below] at (1,-0.5)
{21. 3.\\ \textasteriskcentered~RLG};
\node[align=center,below] at (4,-0.5)
{13. 7.\\ \textasteriskcentered~LSG};
\end{tikzpicture}
```



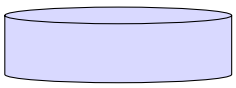
Verschiedene geometrische Formen, zum Beispiel Sterne und Zylinder, sind verfügbar wenn man in der Präambel, nach *usepackage{tikz}*, die Zeile

```
\usetikzlibrary{shapes.geometric}
```

einfügt. Damit kann man zum Beispiel die chinesische Flagge oder eine zylindrische Scheibe zeichnen. (Die deutsche Flagge wurde bereits auf Seite 55 gezeigt.)

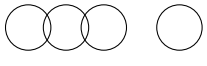
```
\begin{tikzpicture}
[inner sep=0pt,star point ratio=2.617]
    \definecolor{cfr}{HTML}{FF0000}
    \definecolor{cfy}{HTML}{FFFF00}
\fill[cfr] rectangle (30mm, 20mm);
\node[star,fill=cfy, minimum size=6mm,
 rotate=0] at (5mm,15mm) {};
\node[star, fill=cfy, minimum size=2mm,
 rotate=50] at (10mm,18mm) {};
\node[star, fill=cfy, minimum size=2mm,
 rotate=25] at (12mm,16mm) {};
\node[star, fill=cfy, minimum size=2mm,
 rotate=0] at (12mm,13mm) {};
\node[star, fill=cfy, minimum size=2mm,
 rotate=50] at (10mm,11mm) {};
\end{tikzpicture}
```

```
\begin{tikzpicture}
\node[cylinder,draw=black,fill=blue!15,
minimum width=3cm, minimum height=1cm,
aspect=0.8, shape border rotate=90] {};
\end{tikzpicture}
```



Sollen Objekte mehrfach gezeichnet werden, kann es sinnvoll sein, eine Schleife zu programmieren. Die Schleife beginnt mit dem Befehl *foreach* und einer Variablen, zum Beispiel *x*, gefolgt von *in* und einer Liste in geschweiften Klammern. Danach folgt ein Befehl, der mit einem Semikolon abgeschlossen wird, oder ein Befehlsblock in geschweiften Klammern.

```
\begin{tikzpicture}
\foreach \x in {1,1.5,2,3} \draw (\x,0) circle (0.3);
\end{tikzpicture}
```



Endliche Zahlenfolgen, deren Glieder mit konstantem Abstand angeordnet sind, können mit ... beschrieben werden.

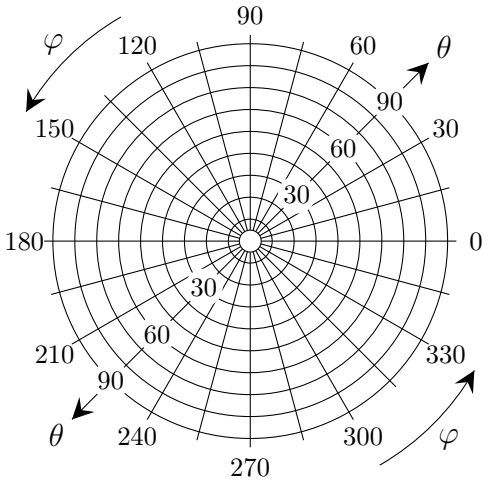
```
\begin{tikzpicture}
\foreach \x in {1,3,...,9}
{
\draw (\x/2,0) circle (0.3);
\node at (\x/2,0) {\x};
}
\end{tikzpicture}
```

Das Beispiel rechts zeigt ein Gesichtsfelddiagramm. Mehrere, ähnliche Zeichnungselemente, wie die konzentrischen Kreise, wurden mit einer Schleife programmiert.

```
\def\wis{0.29mm} % Längeneinheit
\foreach \x in {10,20,...,90}
\draw (0,0) circle (\x*\wis) ;
```

Die Kreisbögen wurden mit *arc* erzeugt. Schöne Pfeilspitzen erforderten eine kompliziert zusammengesetzte Option *decoration*.

```
\draw[decoration={markings,%
mark=at position 1 with
{\arrow[scale=2.5,>=stealth]{>}}},%
postaction={decorate}]
(120:\wis*118) arc [start angle=120%
,end angle=150,radius=\wis*118] ;
\node at (135:\wis*127) {$\varphi$};
```

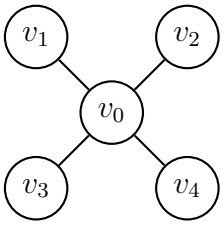


Linien zur Bemaßung in technischen Zeichnungen stellt das Paket *dimline* von Sébastien Gross als Ergänzung von *tikz* bereit.

6.3 Graphen und Funktionen

Graphen im Sinne der Graphentheorie sind mathematische Objekte, die Knoten und Verbindungen enthalten. Ein Graph wird gezeichnet, indem zunächst alle Knoten benannt und dargestellt, und dann die Verbindungen gesetzt werden.

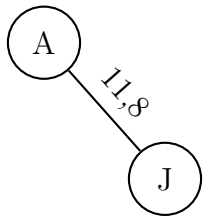
```
\begin{tikzpicture} [thick,
every node/.style={circle,draw}]
\node (v0) at (0,0) {$v_0$} ;
\node (v1) at (-1,1) {$v_1$} ;
\node (v2) at (1,1) {$v_2$} ;
\node (v3) at (-1,-1) {$v_3$} ;
\node (v4) at (1,-1) {$v_4$} ;
\draw (v0)--(v1) (v0)--(v2)
      (v0)--(v3) (v0)--(v4) ;
\end{tikzpicture}
```



In obigem Beispiel steht nach *node* in Klammern ein Knoten-Name, den man beim Setzen der Verbindungen benutzt.

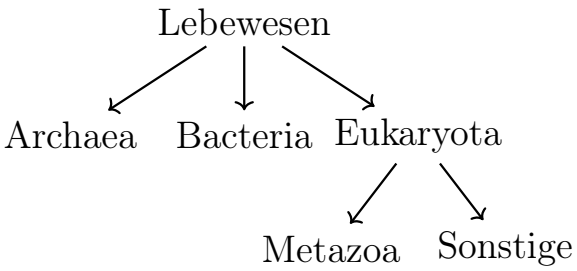
In einem kantengewichteten Graphen werden die Kanten mit Zahlenwerten beschriftet. Folgendes Beispiel zeigt die Entfernung von Apolda nach Jena in km.

```
\begin{tikzpicture} [thick,scale=2]
\node[circle,draw,text width=0.5%
cm,align=center] (A) at (0,0.9) {A} ;
\node[circle,draw,text width=0.5%
cm,align=center] (J) at (0.8,0) {J} ;
\draw (A)--(J) node [pos=0.5,%
above, sloped] {11{,}8} ;
\end{tikzpicture}
```



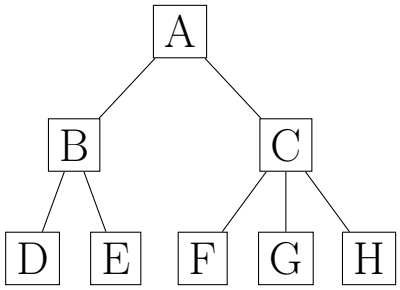
Folgendes Beispiel zeigt ein Baumdiagramm mit gerichteten Verbindungen.

```
\begin{tikzpicture}
 [ sibling distance=23mm,
 font=\large,
 line width=0.3mm,
 style={->} ]
\node {Lebewesen}
 child { node {Archaea}  }
 child { node {Bacteria} }
 child {
  node {Eukaryota}
   child {node {Metazoa}}
   child {node {Sonstige}}
       };
\end{tikzpicture}
```



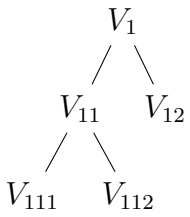
Ein weiteres Beispiel eines Baumdiagramms, hier mit rechteckigen Knoten.

```
\begin{tikzpicture} [ font=\Large,
every node/.style={rectangle,draw},
level 1/.style={sibling distance=28mm},
level 2/.style={sibling distance=11mm}  ]
\node {A}
 child { node {B}
  child { node {D} } child { node {E} } }
 child { node {C}
  child { node {F} } child { node {G} }
  child { node {H} } } ;
\end{tikzpicture}
```



Baumdiagramme mit Wurzel (ein ausgezeichneter Knoten an der Spitze) lassen sich gut mit dem Paket *forest* von Sašo Živanović zeichnen, das auf *tikz* aufbaut. Ein Baum wird in einer *forest*-Umgebung definiert und automatisch möglichst kompakt, das heißt: nicht unnötig breit, dargestellt. Die Baumstruktur wird durch verschachtelte eckige Klammern wiedergegeben, wie folgendes Beispiel zeigt.

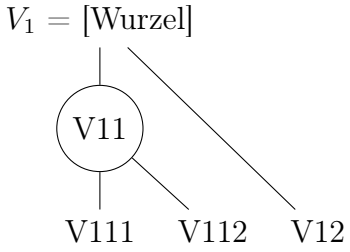
```
\begin{forest}
   [ $V_1$
      [ $V_{11}$
         [ $V_{111}$ ]
         [ $V_{112}$ ]
      ]
      [ $V_{12}$ ]      ]
\end{forest}
```



Zur richtigen Darstellung des Baumdiagramms muss die LaTeX-Datei zweimal übersetzt werden. Die Beschriftung der Knoten kann durch Befehle verändert werden. Soll sie Zeichen mit besonderer Bedeutung (zum Beispiel eckige Klammern, Gleichheitszeichen oder Leerzeichen) enthalten, schließt man sie in geschweifte Klammern ein.

Die Knoten können mit den Optionen gestaltet werden, die *tikz* bereitstellt. Das Erscheinungsbild lässt sich mit Optionen verändern, die der Knotenbeschriftung, und einem Komma, folgen. Im Beispiel

```
\begin{forest}
   [ {$V_1$ = [Wurzel]},
     for tree={calign=first}
      [ V11, circle, draw
           [ V111, tier=t1 ]
           [ V112, tier=t1 ]   ]
      [ V12, tier=t1           ]
    ]
   \node at (current bounding box.south)
    [below=3mm]
    {Ein Beispiel für \emph{forest}.};
\end{forest}
```



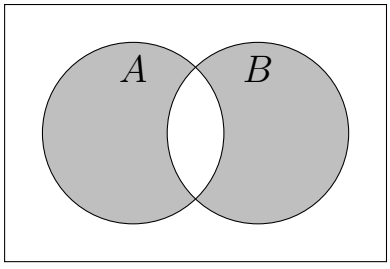
Ein Beispiel für *forest*.

bewirkt die Option *for tree={calign=first}*, dass im ganzen Baum die ersten Kindknoten senkrecht unter ihrem Vorfahren platziert werden. Die mit der Option *tier=t1* versehenen Knoten werden in einer Zeile angeordnet. Dabei ist *t1* ein willkürlich gewählter Name, der für die Knoten der Zeile gleich sein muss. Das Baumdiagramm kann mit einer Unterschrift ergänzt werden, wie oben gezeigt.

Venn-Diagramme stellen Relationen von Mengen bildlich dar. Das Paket *venndiagram* von Nicola Talbot zeichnet mit den Umgebungen *venndiagram2sets* und *venndiagram3sets* Venndiagramme mit zwei beziehungsweise drei Mengen. Beispiel:

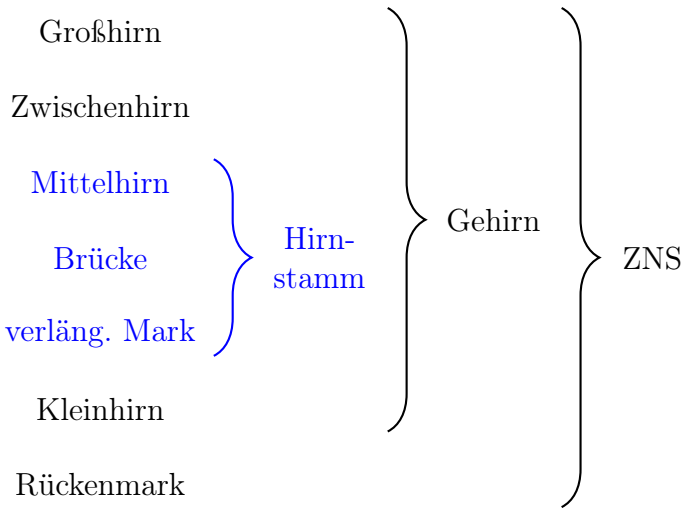
```
\begin{large}
$(A \setminus B) \cup
(B \setminus A)$ \quad
\begin{venndiagram2sets}
\fillOnlyA \fillOnlyB
\end{venndiagram2sets}
\end{large}
```

$(A \setminus B) \cup (B \setminus A)$



Gliederung mit geschweiften Klammern kann mit dem Paket *schemata* erreicht werden, siehe Seite 28. Alternativ kann man Textbausteine und geschweifte Klammern selbst platzieren, wie folgendes Beispiel zeigt.

```
\begin{tikzpicture}[decoration={brace,amplitude=5mm}]
\node at (0,3) {Großhirn}; \node at (0,2) {Zwischenhirn};
\node[blue] at (0,1) {Mittelhirn}; \node[blue] at (0,0) {Brücke};
\node[blue] at (0,-1) {verläng. Mark}; \node at (0,-2) {Kleinhirn};
\node at (0,-3) {Rückenmark};
\draw[decorate,thick,blue]
(1.5,1.3) -- (1.5,-1.3);
\node[blue,align=center]%
at (2.9,0) {Hirn-\\stamm};
\draw[decorate,thick]%
(3.8,3.3) -- (3.8,-2.3);
\node[align=center]%
at(5.2,0.5) {Gehirn};
\draw[decorate,thick]%
(6.1,3.3) -- (6.1,-3.3);
\node[align=center] at%
(7,0) {ZNS\hspace{-6mm}};
\end{tikzpicture}
```



Für die geschweiften Klammern wird in der Präambel mit

```
\usetikzlibrary{decorations.pathreplacing}
```

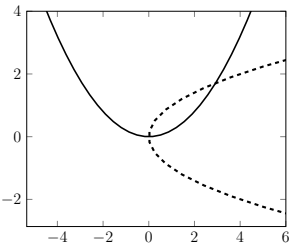
eine besondere Bibliothek für TikZ geladen.

Mathematische Funktionen können berechnet und dargestellt werden, wenn man zusätzlich das Paket *pgfplots* von Christian Feuersänger lädt. Das folgende Beispiel zeigt mit wenigen Befehlen eine ansprechende Darstellung zweier Kurven. In der Präambel steht

```
\usepackage{tikz,pgfplots} \pgfplotsset{compat=newest}
```

Und in der *document*-Umgebung folgt

```
\begin{tikzpicture} [thick, scale=0.5]
\begin{axis}[xmax=6,ymax=4, samples=30]
   \addplot[very thick] (x,0.2*x*x);
   \addplot[dashed, ultra thick] (x*x,x);
\end{axis}
\end{tikzpicture}
```

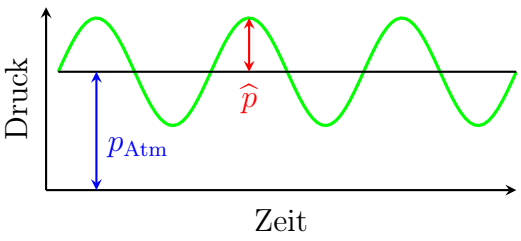


Mit einer *axis*-Umgebung wird sehr einfach ein Koordinatenkreuz erzeugt. Es genügt die Angabe der maximalen Achsenwerte und der Anzahl von Funktionswerten, die (pro Funktion) dargestellt werden. Innerhalb der *axis*-Umgebung wird in nur einer Zeile sowohl die Gleichung einer Funktion als auch die zugehörige Formatierung der Kurve festgelegt. Die in eckigen Klammern angegebene Formatierung bestimmt eine Farbe (ohne Angabe bedeutet: *black*) und Linienstärke und -art (ohne Angabe bedeutet: *solid*).

In folgendem Beispiel wird der Schalldruckverlauf eines Sinustons dargestellt. Die Zeichnung ist nicht maßstabsgetreu, da p_{Atm} sehr viel größer ist als $\widehat{p}$.

```
\begin{tikzpicture}\begin{axis}[width=7.8cm,height=4cm,ticks=none,
```

```
axis x line=bottom,thick,xmin=-0.5,xlabel={Zeit},x label style=
{yshift=-1mm},axis y line=left,ymin=-1.2, ymax=2.2,ylabel={Druck},
y label style={yshift=+1mm}]
\addplot[very thick,green,domain=0:6*pi,samples=100]{1+sin(deg(x))};
\draw (0,1)--(6*pi,1);\draw[thick,
red,stealth-stealth] (2.5*pi,1)--
(2.5*pi,2);\node[red] at (2.5*pi,
0.5) {$\widehat{\hspace{0.02mm}p}$};
\draw[thick,blue,stealth-stealth]
(0.5*pi,1)--(0.5*pi,-1.2);
\node[right,blue] at (0.5*pi,-0.4)
{$p_{\mathrm{Atm}}$};
\end{axis} \end{tikzpicture}
```

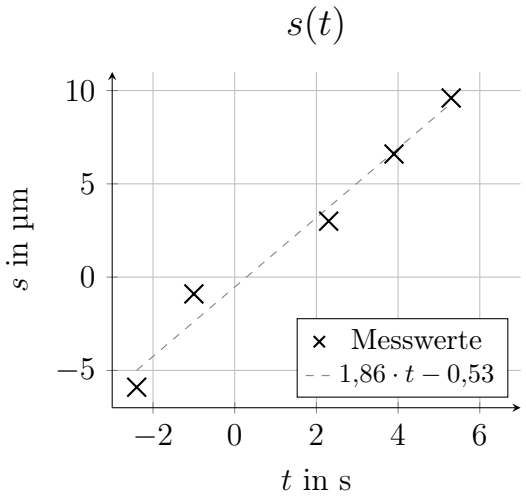


Man kann auch Messwerte in einem Diagramm darstellen und zum Beispiel eine Ausgleichsgerade berechnen lassen. In der Präambel steht

```
\usepackage{pgfplots} \pgfplotsset{compat=newest}
\usepackage{pgfplotstable}
\usepackage[output-decimal-marker={,}]{siunitx} % Dezimaltrennung ,
\SendSettingsToPgf % Übernahme obiger siunitx-Einstellungen in PGF
```

Und in der *document*-Umgebung folgt

```
\begin{tikzpicture} \begin{axis}[width=0.95\linewidth, % width=10cm,
title={\large $s(t)$}, axis x line=bottom, xmin=-3, xmax=7, xlabel=%
{$t$ in \si{\second}}, axis y line=left, ymin=-7, ymax=11, ylabel=%
{$s$ in \si{\micro\meter}},grid, legend pos=south east, % north west
% kleinere Symbole in der Legende, Kurvenformgenauigkeit:
legend image post style={scale=0.6},samples=50,] % Komma stört nicht
% Messwerte plotten:
\addplot[only marks,mark={x},thick,
mark size=5pt]    % mark=* oder o
table[row sep=\\] % Datentabelle
{  -2.4   -5.9  \\
   -1.0   -0.9  \\
    2.3    3.0  \\
    3.9    6.6  \\
    5.3    9.6  \\
};
\addlegendentry{\small Messwerte}
% Ausgleichsgerade plotten:
\addplot[color=gray,style=dashed]
table[row sep=\\, % gleiche Daten
y={create col/linear regression}]
{  -2.4   -5.9  \\
   -1.0   -0.9  \\
    2.3    3.0  \\
    3.9    6.6  \\
    5.3    9.6  \\
};
```



```
\addlegendentry{\footnotesize $\pgfmathprintnumber{%
\pgfplotstableregressiona} \cdot t \pgfmathprintnumber%
[print sign]{\pgfplotstableregressionb}$}
\end{axis} \end{tikzpicture}
```

Verschiedene (auch farbige) Symbole für Datenpunkte in Liniendiagrammen stellt das Paket *oplotsymbl* von B. Michel Döhring bereit. Es beruht auf TikZ. Die Symbole können als Sonderzeichen in normalem Text oder in einer *math*-Umgebung benutzt werden. Neben den Grundformen sind zahlreiche Varianten verfügbar.

Symbole für wissenschaftliche Diagramme, die mit *oplotsymbl* verfügbar sind:

△	○	⬠	☆	◇	⬡	□
\triangleplepa	\circlet	\pentago	\starlet	\rhombus	\hexago	\squad

Einige Varianten von *trianglepa*:

▽	▷	◁	◬	▲
\trianglepb	\trianglepr	\trianglepl	\trianglepadot	\trianglepafill[blue]

Als optionales Argument kann man die Farben des Pakets *xcolor* verwenden.

6.4 Elektrische Schaltkreise und Optik

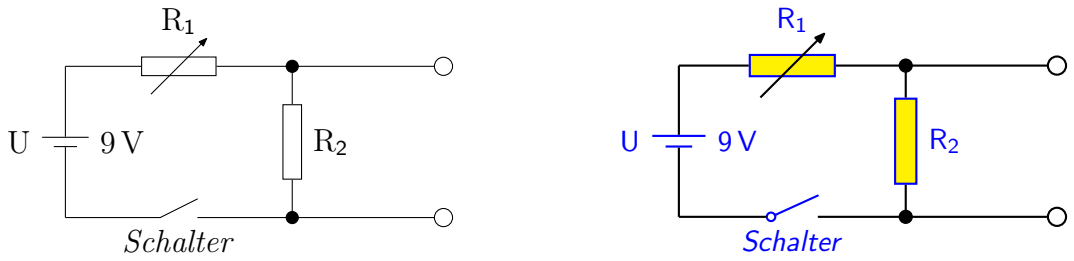
Um Schaltkreise mit TikZ zu zeichnen, kann man die Programmbibliothek *circuits.ee.IEC* oder das Paket *circuitikz* hinzuladen. Erstere ist vielleicht einfacher anzuwenden, letzteres bietet mehr Schaltsymbole, zum Beispiel auch Transistoren.

circuits.ee.IEC ist eine Programmbibliothek für in Deutschland übliche elektrische Symbole, welche man mit dem Befehl *usetikzlibrary{circuits.ee.IEC}* lädt. Zu Beginn der *tikzpicture*-Umgebung wird die Option *circuit ee IEC* , sowie weitere Optionen für den Stil der Zeichnung angegeben. In der *tikzpicture*-Umgebung stehen die Schaltkreissymbole als *tikz*-Knoten (nodes) zur Verfügung. Zum Beispiel ergibt

```
\begin{tikzpicture} [                    % Anfang des Optionsblocks
circuit ee IEC,          % elektrische Schaltungssymbole benutzen
font=\sffamily\small,        % Schrift ohne Serifen, Größe: small
thick,                                    % Linienbreite: thick
every node/.style=blue,  % Stil für nodes, auch Schaltungssymbole
large circuit symbols,       % Größe der Schaltungssymbole: large
every info/.style=blue,   % Stil der Beschriftungen (info, info')
every resistor/.style={fill=yellow},       % Stil der Widerstände
set make contact graphic= var make contact IEC graphic % Schalter
]                                          % Ende des Optionsblocks
\draw (0,2) to [battery={info={9\,V},info'={U}}] (0,0) ;
\draw (0,2) to [resistor={adjustable}] (3,2) ;
\node at (1.5,2.6) {R$\mathsf{_{1}}$} ;
\draw (3,0) to [resistor={info'={R$\mathsf{_{2}}$}}] (3,2) ;
\draw (0,0) to [make contact={info'={\textit{Schalter}}}] (3,0) ;
\draw[fill=black] (3,0) circle (0.8mm)
                  (3,2) circle (0.8mm) ;
```

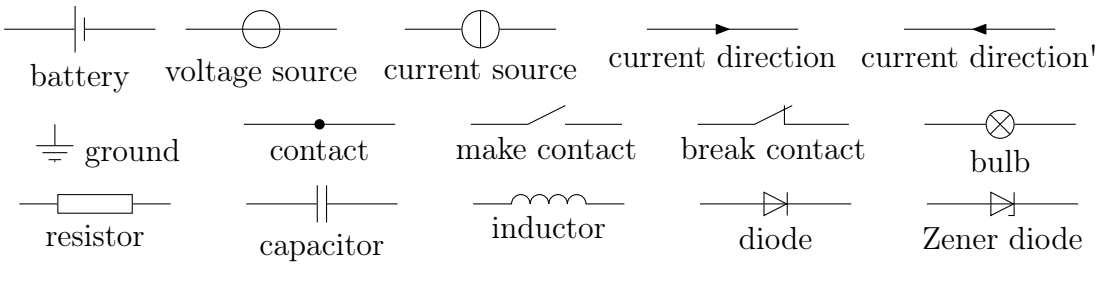
```
\draw (3,0)--(5,0) (3,2)--(5,2) ;
\draw[fill=white] (5,0) circle (1.2mm)
                  (5,2) circle (1.2mm) ;
\end{tikzpicture}
```

den unten rechts gezeigten Schaltkreis. Der linke Schaltkreis entsteht, wenn der Befehl *\begin{tikzpicture}* nur mit der Option *[circuit ee IEC]* gegeben wird.



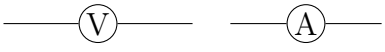
Der Befehl zur Erzeugung eines Schaltkreissymbols kann den Schlüssel *info* oder *info'* enthalten, dessen Wert eine Zeichenkette ist, welche als Text über, unter, links oder rechts vom Symbol gezeigt wird (abhängig von der Ausrichtung des Symbols und von *info* beziehungsweise *info'*). Alternativ kann die Beschriftung eines Symbols auch durch einen *node* erfolgen, wie allgemein in *tikz*.

Unter anderem sind folgende Schaltkreissymbole bereits definiert:



Eine Leuchtdiode erhält man mit folgendem Befehl:

```
\draw (0,0) to [diode={light emitting, info'={LED}}] (2,0) ;
```

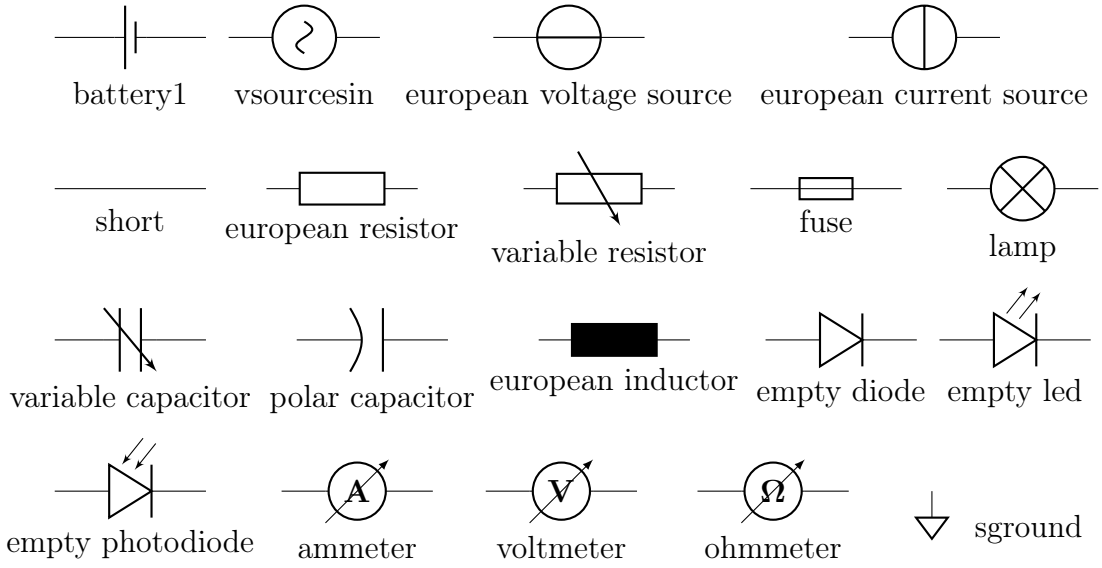


Die Schaltkreissymbole vmeter ameter kann man selbst definieren:

```
\tikzset{circuit declare symbol = ameter} % Amperemeter definieren
\tikzset{set ameter graphic=
{draw,generic circle IEC,minimum size=5mm,info=center:A}}
\tikzset{circuit declare symbol = vmeter}  % Voltmeter  definieren
\tikzset{set vmeter graphic=
{draw,generic circle IEC,minimum size=5mm,info=center:V}}
```

Eine übersichtliche Beschreibung von *circuits.ee.IEC* findet sich auf der Webseite https://matheplanet.com/matheplanet/nuke/html/article.php?sid=1609.

CircuiTikZ ist ein Paket von M. A. Redaelli, S. Lindner und S. Erhardt, das (einschließlich der Option für die in Deutschland üblichen elektrischen Symbole) mit *\usepackage[european]{circuitikz}* geladen wird. Damit werden Schaltkreise in einer *circuitikz*-Umgebung gezeichnet. Da das Paket auf TikZ aufbaut, werden die gleichen Befehle benutzt, aber es gibt besondere *nodes* für Schaltsymbole. Zu beachten ist, dass die Symbol-Kurzbezeichnungen des Pakets *circuitikz* nicht verfügbar sind, wenn die Programmbibliothek *circuits.ee.IEC* geladen wurde. Da man *circuits.ee.IEC* mit *circuitikz* nicht braucht, sollte man darauf verzichten. Bei den folgenden Beispielen werden besagte Kurzbezeichnungen jedoch nicht verwendet. Unter anderem sind folgende Schaltkreissymbole bereits definiert:



Weitere Informationen zu CircuiTikZ gibt es in der Beschreibung des Pakets.

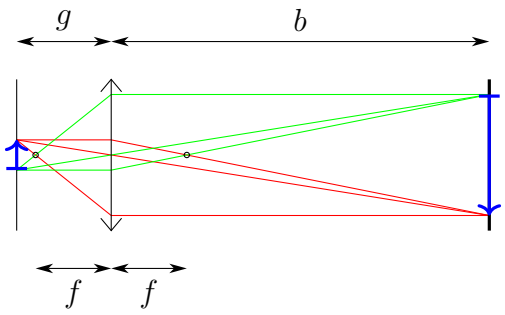
Optik unterstützt das Paket *optics* von Michel Fruchart. Verschiedene, einfach gezeichnete graphische Objekte dienen als Symbole für optische Bauelemente, die als besondere *tikz*-Knoten (nodes) in einer *tikzpicture*-Umgebung, zusammen mit anderen Teilen von TikZ, eingesetzt werden. In der Präambel lädt man TikZ und die notwendigen Programmbibliotheken mit

```
\usepackage{tikz}
\usetikzlibrary{shapes,shapes.misc,shapes.geometric}
\usetikzlibrary{optics}
```

Im folgenden Beispiel wird die Abbildung eines Gegenstands auf ein vergrößertes Bild durch eine Sammellinse dargestellt. Ist $f = \ell$ die Brennweite, wobei ℓ eine beliebige Länge ist, beträgt die Gegenstandsweite $g = 1{,}25\,\ell$ und die Bildweite $b = 5\,\ell$. Rot beziehungsweise grün sind die jeweils drei Hauptstrahlen des Lichts gezeichnet, das vom oberen beziehungsweise unteren Ende des Gegenstands ausgeht.

```
\begin{tikzpicture}[use optics]
\draw[red] (-1.25,0.2)--(5,-0.8) (-1.25,0.2)--(0,-0.8) -- (5,-0.8);
\draw[red] (-1.25,0.2)--(0,0.2)--(5,-0.8);
\draw[green](-1.25,-0.2)--(5,0.8) (-1.25,-0.2)--(0,0.8)--(5,0.8);
\draw[green] (-1.25,-0.2)--(0,-0.2)--(5,0.8);
\node[thin optics element] (G) at (-1.25,0) {};
```

```
\node[lens,focal length=1cm,draw focal points={circle}]
   (L1) at (0,0) {};
\node[screen] (S) at (5,0) {};
\draw[|->,blue,very thick] (-1.25,-0.2) -- (-1.25,0.2);
\draw[|->,blue,very thick] (5,0.8) -- (5,-0.8);
\coordinate (ao1) at (0,1.5cm);
\draw[>=technical,<->] (ao1 -| G) -- (ao1 -| L1)
   node[midway,above] {$g$};
\draw[>=technical,<->] (ao1 -| L1) -- (ao1 -| S)
   node[midway,above] {$b$};
\coordinate (ao2) at (0,-1.5cm); \coordinate (fl) at (-1cm,0);
\coordinate (fr) at (1cm,0);
\draw[>=technical,<->] (ao2 -| fl) -- (ao2 -| L1)
   node[midway,below] {$f$};
\draw[>=technical,<->] (ao2 -| L1) -- (ao2 -| fr)
   node[midway,below] {$f$};
\end{tikzpicture}
```

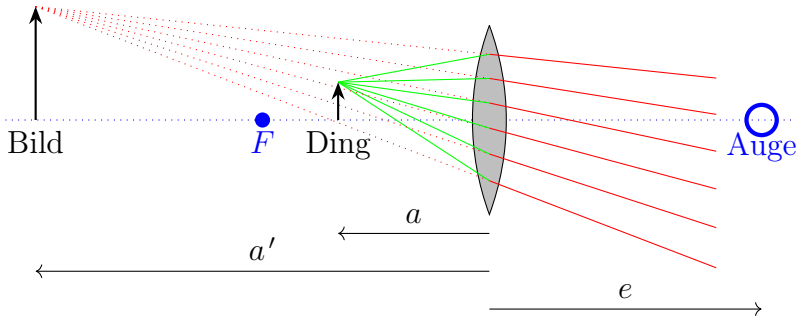


Optische Bauelemente des Pakets *optics* sind Sammel- und Zerstreuungslinse, Spalt, Doppelspalt, Spiegel (plan, konvex, konkav), Polarisator, Strahlteiler, Geradsichtprisma, allgemeines dünnes oder dickes optisches Element, Wärmeschutzfilter, Bildschirm, optisches Gitter, Raster, halbdurchlässiger Spiegel, Blende, Lampen und Sensoren, Laser.

Optische Zeichnungen können mit Tikz auch ohne das Paket *optics* erstellt werden. Oben (Seite 82) wurden optische Linsen abgebildet. Im folgenden Beispiel wird der Verlauf von Lichtstrahlen bei Verwendung einer sphärischen Sammellinse als Lupe dargestellt (maßstabsgetreu für $n_{\text{Glas}} = 1{,}6$, nicht das Auge). Die zum virtuellen Bild fortgesetzten Strahlen sind gepunktet.

```
\begin{tikzpicture}
% opt. Achse, Dingbrennpunkt (x-Koordinate = -3)
\draw[blue,dotted] (-6.4,0)--(4,0); % optische Achse
\fill[blue] (-3,0) circle (0.1) node[below] {$F$}; % F
% symmetrische (L=R) bikonvexe Linse, Brennweite 3
\pgfmathsetmacro{\WL}{asin(2.5/(2*3.6))} % d=2.5, r=3.6
\draw [fill=lightgray] (0,2.5/2) % oberster Linsenpunkt
arc[start angle=180-\WL,delta angle=2*\WL,radius=3.6]
arc[start angle=-\WL,delta angle=2*\WL,radius=3.6];
% Lichtstrahlen
\foreach \WI in {-7,...,-2} % 6 Strahlen von -7*3 bis -2*3 Grad
```

```
{ \pgfmathsetmacro{\DY}{tan(\WI*3)} % Bild: x = -6, y = 1.5
\draw[red,dotted](-6,1.5)--(0,1.5+6*\DY); % vom Bild zur Linse
\draw[red](0,1.5+6*\DY)--(3,1.5+9*\DY); % Linse zum Auge
\draw[green](-2,0.5)--(0,1.5+6*\DY); } % vom Ding zur Linse
% Ding (x-Koordinate = -2, y = 0.5) und Bild (x = -6, y = 1.5)
\draw[Stealth-,black,thick] (-2,0.5)--(-2,0) node[below] {Ding};
\draw[Stealth-,black,thick] (-6,1.5)--(-6,0) node[below] {Bild};
% Auge (x-Koordinate = 3.6, Radius = 0.2)
\draw[blue,ultra thick] (3.6,0) circle (0.2) node[below] {Auge};
% Streckenpfeile (Dingweite a, Bildweite a', Augenabstand e)
\draw[<-] (-2,-1.5)--(0,-1.5) node[midway,above] {$a$};
\draw[<-] (-6,-2)--(0,-2) node[midway,above] {$a^{\,\prime}$};
\draw[->] (0,-2.5)--(3.6,-2.5) node[midway,above] {$e$};
\end{tikzpicture}
```



Die Stealth-Pfeile erfordern hier den Befehl \usetikzlibrary{arrows.meta} in der Präambel nach \usepackage{tikz} (siehe Abschnitt 6.1, Seite 79).

Zum Zeichnen einer einfachen Linse (ohne Füllung), zum Beispiel einer bikonvexen Linse bestimmter Dicke, genügt vielleicht das Paket *simpleoptics* von Justin Cawood.

6.5 Randnotizen, Kästen, Kalender und Avatare

Randnotizen können in einem unfertigen Manuskript Stellen markieren, die noch bearbeitet werden müssen. Mit dem, auf Tikz aufbauendem Paket *todonotes* von Henrik Skov Midtiby kann man, in einem Kasten mit farbigem Hintergrund, Randnotizen setzen und man kann auch auf fehlende Bilder hinweisen.

```
Gallia est omnis divisa in partes tres,
\todo[color=green!30]{Belgae?}\\aliam Aquitani,
tertiam qui ipsorum lingua Celtae, nostra Galli appellantur.
```

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

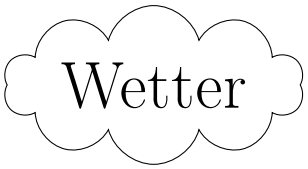
Die im optionalen Argument festgelegte Hintergrundfarbe sollte, wie im Beispiel, transparent sein, damit der Vordergrundtext gut lesbar ist. In Fußnoten und einigen Umgebungen (zum Beispiel minipage) funktionieren die *todonotes* leider nicht.

Gerahmte Kästen in verschiedenen Formen und Farben kann man mit TikZ um Text oder mathematischen Formeln setzen. Mit der Zeile

```
\usetikzlibrary{shapes.symbols}
```

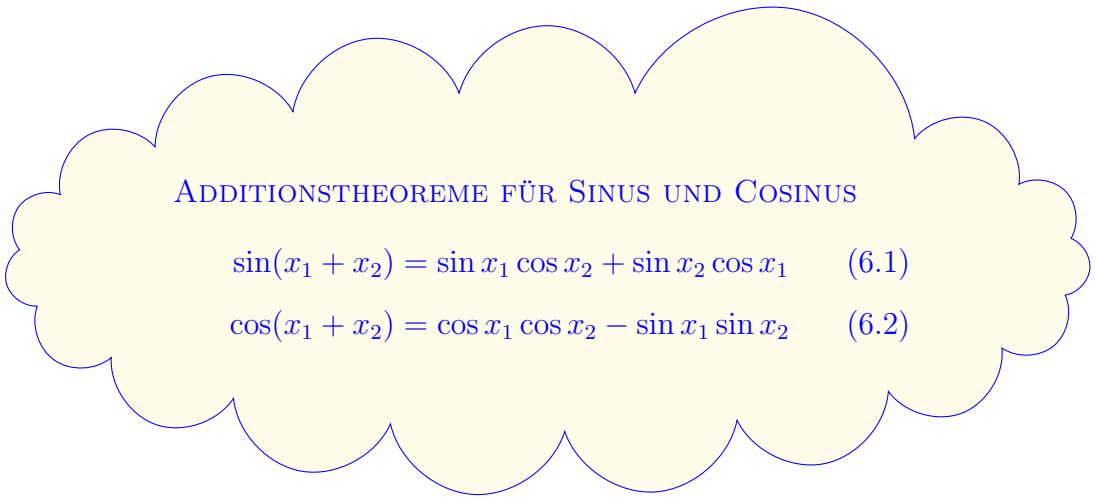
lädt man zunächst eine geeignete Programmbibliothek in der Präambel (nach dem Laden von *tikz*) und kann danach die bereitgestellten Formen nutzen. Es folgen einige Beispiele.

```
\begin{tikzpicture}
\node [cloud,draw,cloud %
ignores aspect]{\Huge Wetter};
\end{tikzpicture}
```




```
\begin{tikzpicture}
\node [starburst,starburst %
points=28, starburst point %
height=5mm,draw,thick,red,%
fill=red!10,inner sep=3mm] {%
\Large \textbf{Revolution}};
\end{tikzpicture}
```

```
\begin{tikzpicture}
\node [cloud,cloud puffs=18.8,cloud ignores aspect,minimum width%
=8cm,minimum height=3cm,align=center,draw,blue,fill=yellow!10]{%
  \begin{minipage}{0.65\textwidth}
    \begin{large}
      \textsc{Additionstheoreme für Sinus und Cosinus} \vspace{-2mm}
      \begin{align}
        \sin(x_1+x_2) &= \sin x_1 \cos x_2 + \sin x_2 \cos x_1 \\
        \cos(x_1+x_2) &= \cos x_1 \cos x_2 - \sin x_1 \sin x_2
      \end{align}\vspace{-6mm}
    \end{large}
  \end{minipage}
};
\end{tikzpicture}
```



Gibt man für die Anzahl der Puffen (Auswölbungen der Wolke) einen Wert an, der nicht ganzzahlig ist (hier: *cloud puffs=18.8*), wird die Wolke unsymmetrisch. Ohne die Option *cloud ignores aspect* ergäbe sich eine rundliche Wolke (Höhe = Breite).

```
\begin{tikzpicture}
\node [rectangle, rounded corners,draw=blue,very thick,%
inner sep=3mm] (kasten){%
\begin{minipage}{0.925\textwidth} \rule{0pt}{7mm}%
He had had a nice, good, idle time all the while---plenty of
company---and the fence had three coats of whitewash on it!
If he hadn't run out of whitewash he would have bankrupted
every boy in the village.
\end{minipage}
};
\node[right=8mm, rectangle,rounded corners,fill=yellow!20,%
draw=blue,very thick,text=blue,inner xsep=3mm,%
inner ysep=1.8mm] at (kasten.north west)%
 {\vphantom{\Large W}\textbf{Efficient Delegation}};
\end{tikzpicture}
```

Efficient Delegation

He had had a nice, good, idle time all the while—plenty of company—and the fence had three coats of whitewash on it! If he hadn't run out of whitewash he would have bankrupted every boy in the village.

Die Einfügung von *\vphantom{\Large W}* in die Zeichenkette der Überschrift bewirkt hier einen größeren Abstand von Überschrift und darüber liegender Linie, sodass der Überschriftenkasten ausgewogener erscheint.

Das leider fehlerhafte Paket *pgfornament* von Alain Matthes enthält allerlei Verzierungen, mit denen man zum Beispiel schöne Trennlinien bekommt. Gewarnt sei vor merkwürdigen Fehlern, die bei manchen PDF-Betrachterprogrammen auftreten können, wenn *pgfornament* benutzt wird und bestimmte andere Pakete ebenfalls geladen wurden.

Kästen mit Logo können mit dem Paket *bclogo* von Patrick Fradin und Maxime Chupin erzeugt werden, das mit

```
\usepackage[tikz]{bclogo}
```

geladen wird und die *bclogo*-Umgebung für einen schönen Textkasten bereitstellt.

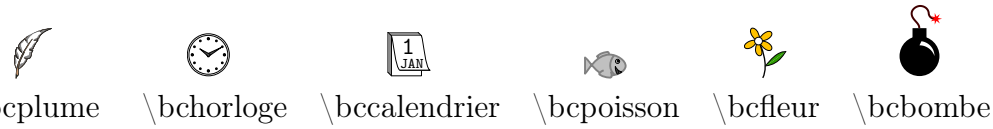
```
\begin{bclogo}[logo=\bcplume,
noborder=true, couleurBarre=red]
{~Cicero: In Catilinam}
O tempora, o mores! Senatus haec
intellegit, consul videt; hic tamen
vivit. Vivit? Immo vero etiam in
senatum venit! \end{bclogo}
```

Cicero: In Catilinam

O tempora, o mores! Senatus haec intellegit, consul videt; hic tamen vivit. Vivit? Immo vero etiam in senatum venit!

Optionale Argumente der *bclogo*-Umgebung sind unter anderem *logo=...* mit einem Logo (siehe unten) als Wert, *noborder=...* mit *true* (ohne Rahmen) oder *false* (mit Rahmen) als Wert, und *couleurBarre=...* mit einem Farbwert für den linken Balken. Mit *barre=none* statt *couleurBarre=...* wird der Balken weggelassen.

Unter anderem sind folgende Logos verfügbar:



Die Logos können auch ohne *bclogo*-Umgebung in normalen Text eingefügt werden, sind dort aber nicht wie Zeichen skalierbar. Zum Beispiel gibt \bclampe .

Wenn man über die Platonischen Körper schreibt, sind folgende Logos interessant:

Kalender können mit der TikZ-Programmbibliothek *calendar* gezeichnet werden. Um deutsche Namen für Monate und Wochentage zu erhalten, lädt man in der Präambel vor *\usetikzlibrary{calendar}* das Paket *translator* mit der Sprachoption, die auch als Dokumentklassenoption angegeben wurde (in der Regel *ngerman*):

```
\usepackage[ngerman]{translator}
\usepackage{tikz}
\usetikzlibrary{calendar}
```

Dann lässt sich im Dokument ein Kalender wie in folgendem Beispiel erstellen.

```
\begin{tikzpicture}
[every day/.style={anchor=center}]
\calendar[
name=ahgcal, week list, day text={\%d=},
dates=\year-\month-01 to \year-\month-last,
month label above centered,
month text={\textit{\%mt \%y0}}   ]
if (Sunday) [red!70!black]
if (equals=\year-\month-\day)
{\draw (0,0) circle (8pt);}   ;
\end{tikzpicture}
```

January 2021

			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Avatare sind graphische Figuren, die Menschen, Tiere oder Fabelwesen darstellen und Menschen im Internet symbolhaft vertreten. Derartige Graphiken sind vielleicht nicht notwendig, aber lustig. Hierfür gibt es unter anderem das Paket *tikzpeople* von Nils Fleischhacker. Die Figuren werden als Knoten in einer *tikzpicture*-Umgebung gezeichnet. Der Typ wird als optionales Argument angegeben. Definiert sind:

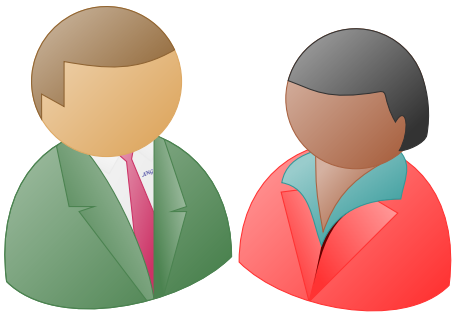
alice, bob, bride, builder, businessman, charlie, chef, conductor, cowboy, criminal,

dave, devil, duck, graduate, groom, guard, jester, judge, maninblack, mexican, nun, nurse, person, physician, pilot, police, priest, sailor, santa, surgeon.

Die Figuren können durch Optionen ergänzt oder verändert werden, nämlich: evil (böse, zum Beispiel mit Kinnbart und Hörnern), female (längere Haare), good (mit Heiligenschein), mirrored (guckt von rechts nach links, statt von links nach rechts), monitor (mit Computerbildschirm), saturated (mit gesättigten Farben), shield (mit Schild), sword (mit Schwert). Bestimmten Figuren kann jeweils nur eine Teilmenge aller Optionen zugeordnet werden. Gesichtszüge (Mund, Nase Augen) fehlen den Figuren. Verschiedene Teile können anders gefärbt werden (Beispiel: hair=brown). Die Größe der Figuren kann mit der Option *minimum size=...* verändert werden. Als Pflichtargument muss für die Figur eine Beschriftung angegeben werden, zum Beispiel: {Romeo}, die aber auch leer sein kann: {}.

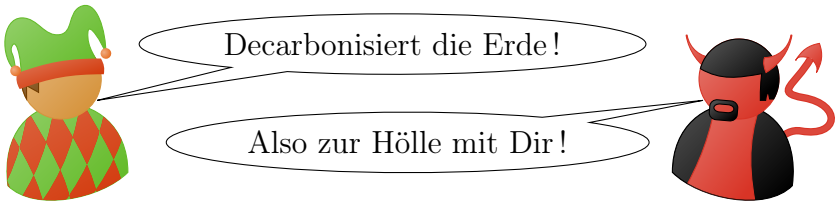
Nun endlich zwei Beispiele:

```
\begin{tikzpicture}
\node[businessman, tie=purple,
 monogramtext=AHG, % auf dem Hemd
 minimum size=3cm]at(0,0.16){Romeo};
\node[alice,mirrored, shirt=red,
 undershirt=teal, minimum %
 size=2.8cm] at (3,0) {Julia};
\end{tikzpicture}
```



Romeo Julia

```
\begin{center} \begin{tikzpicture}
\node[jester, saturated, minimum size=1.6cm] (a) at (0,0) {};
\node[ellipse callout, draw,yshift=1cm, xshift=4.3cm,
  callout absolute pointer={(a.mouth)}] {Decarbonisiert die Erde\,!};
\node[devil,mirrored,saturated,minimum size=1.6cm] (b) at (8.8,0) {};
\node[ellipse callout, draw,yshift=-0.3cm, xshift=4.5cm,
  callout absolute pointer={(b.mouth)}] {Also zur Hölle mit Dir\,!};
\end{tikzpicture} \end{center} % in Präambel: \usetikzlibrary{shapes}
```



Schneemänner können ganzjährig mit dem Paket *scsnowman* von Hironobu Yamashita in mehreren Varianten gezeichnet werden. Zum Beispiel:

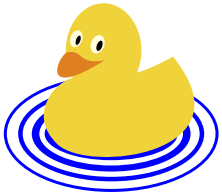
```
Schneemann
\scsnowman
[scale=6, adjustbaseline,
mouthshape=frown,sweat=blue]
```

Schneemann

Optionale Argumente sind *body*, *sweat*, *hat*, *arms*, *muffler*, *buttons* und *snow*. Man kann ihnen eine Farbe zuweisen. Die Option *mouthshape* hat die möglichen Werte *smile*, *tight* und *frown*. Die Option *scale* mit einem Zahlenwert bestimmt die Größe und *adjustbaseline* setzt den Schneemann auf die Grundlinie der Textzeile.

Gummienten sind ja wohl das Letzte, was hier erwähnt werden soll. Für alle, die nicht an Anatidaephobie leiden, malt das Paket *tikzducks* von samcarter bunte Entchen. Hier ein einfaches Beispiel:

```
\begin{tikzpicture}
\duck[water=blue]
\end{tikzpicture}
```



Wir fragen unsere chinesischen Freunde: Wann gibt es ein Paket für Pandabären?

7 Ergänzungen

7.1 Physik, Chemie, Bioinformatik, Übungen usw.

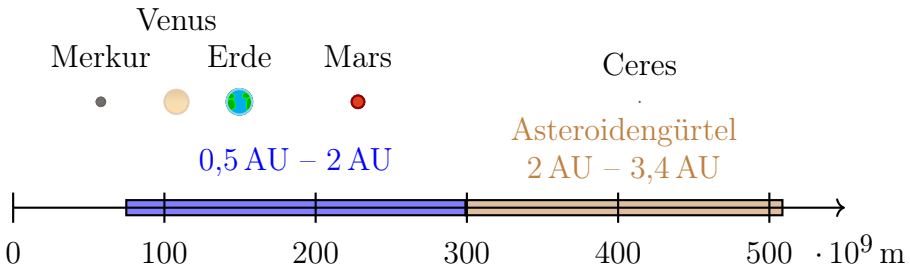
Physikalische Grundlagen können mit dem Paket *mandi* von Paul J. Heafner beschrieben werden. Es enthält besondere Befehle für physikalische und astronomische Gleichungen. Leider fehlt eine Anpassung an deutsche Schreibweisen und die Kompatibilität mit anderen Paketen ist nicht gut. Nützlich sind zum Beispiel die (mit Symbol und Wert) definierten physikalischen und astronomischen Konstanten.

Physikalische Konstanten werden im Paket *physconst* von Brian W. Mulligan mit aktuellen Werten aufgelistet.

Pakete für physikalische Einheiten (Seite 62), Optik (Seite 93) und elektrische Schaltkreise (Seite 91) wurden bereits vorgestellt (siehe oben).

Kleine Zeichnungen von Sonne, Mond und den Planeten des Sonnensystems, und auch Mondphasen, stellt das Paket *planets* von Isabelle M. Santos für TikZ bereit. Damit kann man zum Beispiel die habitable Zone um die Sonne darstellen.

```
\begin{tikzpicture}[thick]
%   Zuerst werden die Achsen und die Beschriftung gezeichnet:
\node[blue] at (1.25*149.6/50,-0.8) {0{,}5\,AU -- 2\,AU};
\draw[fill=blue!50] (0.5*149.6/50,-1.5) rectangle (299.2/50,-1.3);
\node[brown] at (2.7*149.6/50,-0.4) {Asteroidengürtel};
\node[brown] at (2.7*149.6/50,-0.9) {2\,AU -- 3{,}4\,AU};
\draw[fill=brown!50] (299.2/50,-1.5) rectangle (508.64/50,-1.3);
\draw[->] (0,-1.4)--(550/50,-1.4);
\foreach \d in {0,100,...,500}
{ \draw(\d/50,-1.6)--(\d/50,-1.2); \node at (\d/50,-2.0) {\d}; }
\node at (11.2,-1.947)
{$\cdot$\,10\textsuperscript{\footnotesize 9}\,m};
\node[right,align=left] at (-0.3,-3) {\textcolor{blue}{Habitable %
Zone} nach Petigura et al., PNAS 110, 19273-8 (2013),\\Planeten %
im gleichen Maßstab (\,$\neq$ Entfernungsmaßstab) gezeichnet.};
%   Dann werden die Planeten und der Zwergplanet gezeichnet:
\pgfmathsetmacro{\x}{58/50}   \node at (\x,0.6) {Merkur};
\planet[surface=mercury, scale=4.8794/70, centerx=\x]
\pgfmathsetmacro{\x}{108/50}   \node at (\x,1.1) {Venus};
\planet[surface=venus, scale=12.1036/70, centerx=\x]
\pgfmathsetmacro{\x}{149.6/50}   \node at (\x,0.6) {Erde};
\planet[surface=earth, scale=12.7349/70, centerx=\x]
\pgfmathsetmacro{\x}{228/50}   \node at (\x,0.6) {Mars};
\planet[surface=mars, scale=6.7724/70, centerx=\x]
\pgfmathsetmacro{\x}{414.26/50}   \node at (\x,0.5) {Ceres};
\planet[surface=moon, scale=0.9394/75, centerx=\x]
\end{tikzpicture}
```



Habitable Zone nach Petigura et al., PNAS 110, 19273–8 (2013), Planeten im gleichen Maßstab (≠ Entfernungsmaßstab) gezeichnet.

Das Paket *tikz-truchet* von Matthew Scroggs zeichnet Truchet-Fliesen und auch zum Beispiel einfache schwarz-weiße geometrische Würfel, bei denen man eine Seitenfläche hervorheben kann. Leider verändert das Paket einige Einstellungen, so dass es zu unerwünschten Verschiebungen kommen kann.

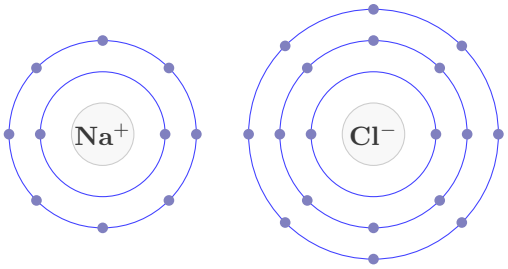
Phantasievolle chemische Symbole werden vom Paket *svrsymbols* von Apostolos Syropoulos als Sonderzeichen in einer *math*-Umgebung zur Verfügung gestellt. Hier einige Beispiele:

```
Atom ~~ $\atom$
Wassermolekül ~~ $\water$
ionische Bindung ~~ $\ionicbond$
metallische Bindung ~~ $\metalbond$
kovalente Einfachbindung ~~ $\covbond$
Wasserstoffbrückenbindung ~~ $\hbond$
```

Atom
Wassermolekül
ionische Bindung
metallische Bindung
kovalente Einfachbindung
Wasserstoffbrückenbindung

Atommodelle mit elektronenbesetzten Schalen stellt das Paket *bohr* von Clemens Niederberger dar. Beispiel:

```
\setbohr{ name-options-set
={font=\footnotesize},
electron-radius = {2pt} }
\bohr{10}{$\mathbf{Na^+}$}
\quad % Bohr-Schalenmodell
\bohr{18}{$\mathbf{Cl^-}$}
```



Der erste Befehl in obigem Beispiel setzt die Schriftgröße für die Kernbezeichnung und den Radius der Elektronen fest. Der Befehl *\bohr{...}{...}* erhält im ersten Pflichtargument die Zahl der Elektronen, im zweiten den Text, welcher den Kern bezeichnet.

Das Paket *elements*, welches vom Paket *bohr* automatisch geladen wird, stellt die Elektronenkonfiguration der Atome bereit und erlaubt auch, andere Elektronenkonfigurationen im gleichen Format zu schreiben. Beispiel:

```
\elconf{Cl} \par % e-Konfig. Chlor
\writeelconf{2,2+6,2+6}  % Chlorid
```

$1s^2 2s^2 2p^6 3s^2 3p^5$
$1s^2 2s^2 2p^6 3s^2 3p^6$

In obigem Beispiel werden die Elektronenkonfigurationen von Chlor (dem Programm bekannt) und Chlorid (als Argument des Befehls *\writeelconf{...}*) ausgegeben.

Summenformeln und chemische Gleichungen können mit dem Paket *mhchem* von Martin Hensel dargestellt werden. Es benötigt einige andere Pakete, die automatisch geladen werden. Umgekehrt wird *mhchem* automatisch vom Paket *chemexec* geladen. Mit *chemexec* sollte man daher auf das Laden von *mhchem* verzichten. Da die neuen Versionen von *mhchem* nicht kompatibel zu älteren sind, sollte man beim Laden eine Versionsnummer als Option angeben:

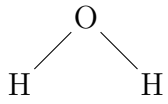
```
\usepackage[version=4]{mhchem}
```

Der Befehl *\ce{...}* des Pakets *mhchem* erlaubt es, einzelne Summenformeln oder ganze Gleichungen zu schreiben, wie folgende Beispiele zeigen.

`\ce{H3O+} \ce{NaCl(aq)} \ce{^{13}_{6}C}`	$\mathrm{H_3O^+}\ \mathrm{NaCl(aq)}\ \mathrm{^{13}_{6}C}$
`\ce{Ba^2+ + SO4^2- -> BaSO4 v }`	$\mathrm{Ba^{2+} + SO_4^{2-} \longrightarrow BaSO_4\downarrow}$
`\ce{H2O -> H2 ^ + 1/2 O2 ^}`	$\mathrm{H_2O \longrightarrow H_2\uparrow + \tfrac{1}{2}O_2\uparrow}$
`\ce{O=O} \quad \ce{H-C#N}`	$\mathrm{O{=}O} \quad \mathrm{H{-}C{\equiv}N}$
`\ce{H2C=CH2 ->[$\mathrm{H_2}$] H3C-CH3}`	$\mathrm{H_2C{=}CH_2 \xrightarrow{H_2} H_3C{-}CH_3}$
`\ce{A <--> B} \quad \ce{A <=> B}`	$\mathrm{A \rightleftarrows B} \quad \mathrm{A \rightleftharpoons B}$
`\ce{A <=>> B} \quad \ce{A <<=> B}`	$\mathrm{A \rightleftharpoons B} \quad \mathrm{A \rightleftharpoons B}$

Chemische Strukturformeln können mit dem Paket *chemfig* von Christian Tellechea gezeichnet werden. Beispiele:

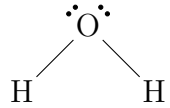
```
\chemfig{H-[1]O-[7]H}
```



Die Molekülbeschreibung besteht aus den Buchstaben für die Atome (oder Atomgruppen) und den Bindungen (- Einfach-, = Doppel-, ∼ Dreifachbindung). Für Bindungen kann in eckigen Klammern ein Winkel angegeben werden. Am einfachsten ist die Verwendung vordefinierter Winkel (0: 0 °, 1: 45 °, , 2: 90 °, . . . , 7: 315 °).

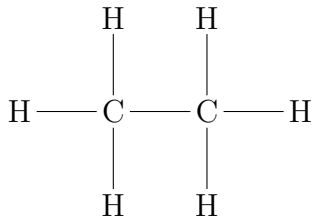
Die Elektronenformel (Lewis-Struktur) erhält man, wenn statt des Atomsymbols der Befehl *\lewis{Winkel Elektronen Winkel Elektronen,...,Atomsymbol}* steht.

```
\chemfig{H-[1]\lewis{1:3:,O}-[7]H}
```



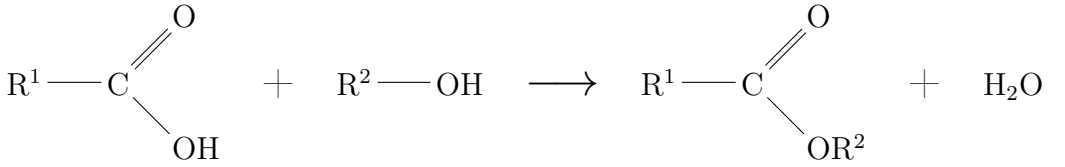
Elektronen können durch einen Punkt (.) oder Doppelpunkt (:) bezeichnet werden. Wird keine Angabe zu den Elektronen (weder . noch :) gemacht, erscheint ein Strich (für ein Elektronenpaar). Es werden vordefinierte Winkel wie für Bindungen angegeben (siehe oben).

Verzweigungen werden durch Ausdrücke in runden Klammern angegeben.

Ethan  wird mit dem Befehl

```
\chemfig{H-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-H}
```

gezeichnet. Die Reaktionsgleichung



Carbonsäure Alkohol Carbonsäureester Wasser

erhält man mit den Befehlen

```
\chemname{\chemfig{R^{1}-C(-[7]OH)=[1]O}}{Carbonsäure}
\quad \begin{Large}+\end{Large} \quad
\chemname{\chemfig{R^{2}-OH}}{Alkohol}
\quad \begin{Large}$\longrightarrow$\end{Large} \quad
\chemname{\chemfig{R^{1}-C(-[7]OR^{2})=[1]O}}{Carbonsäureester}
\quad \begin{Large}+\end{Large} \quad
\chemname{\chemfig{H_2O}}{Wasser}
```

Der Befehl *\chemname{Molekül}{Name}* setzt den gewählten Namen unter das Molekül. Hoch- oder Tiefstellung einer Zeichenkette *zk* nach einem Atomsymbol erreicht man, wie in einer *math*-Umgebung, mit `^{zk}` beziehungsweise `_{zk}` . In obigem Beispiel wurde die Nummerierung der Organylgruppen, R^1 und R^2, hoch gesetzt. So kann man auch Ladungssymbole (+ und -) hochsetzen.

Chemische Übungsblätter können leichter mit dem Paket *chemexec* von Clemens Niederberger geschrieben werden. Wenn der Befehl *\aufgabe{...}* (siehe unten) zur Verfügung stehen soll, muss das Paket mit

```
\usepackage[exercise]{chemexec}
```

geladen werden. Sollen englische statt deutsche Bezeichnungen geschrieben werden, ist beim Laden zusätzlich die Option *english* anzugeben.

```
\usepackage[exercise,english]{chemexec}
```

Die im Paket definierten Umgebungen und einige Befehle sind auch gut für Übungen in anderen Fachgebieten geeignet. Das Paket kann jedoch nicht nicht in der Dokumentklasse *beamer* verwendet werden. Zunächst einige nützliche Befehle, die chemische Sonderzeichen ergeben:

`Ladungen \om{} und \op, Ca\op[2]`	Ladungen $^{\ominus}$ und $^{\oplus}$, $Ca^{2\oplus}$
`\el{} \Hpl{} und Hydroxid \Hyd{}`	$e^{\ominus}$ $H^{\oplus}$ und Hydroxid $OH^{\ominus}$
`Oxidationszahlen \ox{+1}{K}\ox{-1}{Cl}`	Oxidationszahlen $\overset{+1}{K}\overset{-1}{Cl}$
`Säurereste \carbonat{} \nitrat{} \nitrit{}`	Säurereste CO_3 NO_3 NO_2
`\phosphat{} \phosphit{} \sulfat{} \sulfit{}`	PO_4 PO_3 SO_4 SO_3

Wenn die Befehle *\om*, *\op* oder *\ox* in einer Umgebung oder einem Befehl des Pakets *mhchem* benutzt werden, muss vor ihnen ein Leerzeichen sein.

Die *beispiel*-Umgebung erzeugt einen Rahmen, in dem ein nummeriertes Beispiel gesetzt wird.

```
\begin{beispiel}
Wenn dir die Hose reißt und eine Dame
sagt, sie wisse deine Offenheit zu
schätzen, dann ist das Ironie.
\end{beispiel}
```

Beispiel 1:
Wenn dir die Hose reißt und eine Dame sagt, sie wisse deine Offenheit zu schätzen, dann ist das Ironie.

Sollen zwei Beispiele im gleichen Rahmen gezeigt werden, trennt man sie mit dem Befehl *\bsp*. Die *definition*-Umgebung erzeugt einen Kasten, in dem nach der Überschrift *Definition* eine Definition steht.

```
\begin{definition}
Ironie ist ein sprachlicher
Ausdruck, aus dem  eine andere
Aussage abgeleitet werden kann
als das wörtlich Gesagte.
\end{definition}
```

DEFINITION Ironie ist ein sprachlicher Ausdruck, aus dem eine andere Aussage abgeleitet werden kann als das wörtlich Gesagte.

Die *exkurs*-Umgebung mit einem Pflichtargument *{...}* erzeugt einen Rahmen aus zwei farbigen Balken (oben und unten), in dem die Überschrift *EXKURS: ...* (wobei *...* der im Pflichtargument angegebene Titel ist) und der Exkurstext stehen. Die Exkurse werden in das Inhaltsverzeichnis aufgenommen.

```
\begin{exkurs}{Wetter}   % Exkurs übers Wetter (Zitat von Mark Twain)
Alle schimpfen aufs Wetter, aber keiner tut was dagegen. \end{exkurs}
```

Der Befehl *\aufgabe{}* mit leerem Pflichtargument gibt eine nummerierte Überschrift *Aufgabe ...* aus. Wird im Pflichtargument eine Zeichenkette gegeben, wird diese statt des Wortes *Aufgabe* in der Überschrift wiedergegeben.

```
\aufgabe{}
Wie lange dauerte der
Siebenjährige Krieg?%
\aufgabe{Siebengescheites}
Was ergibt sieben
mal sieben?
```

1. Aufgabe

Wie lange dauerte der Siebenjährige Krieg?

2. Siebengescheites

Was ergibt sieben mal sieben?

Der Befehl *\loesung{...}* definiert im Pflichtargument die Lösung zur vorhergehenden Aufgabe. Man kann auch ein optionales Argument mit dem Aufgabentitel (oder einer anderen Zeichenkette) angeben. Der Befehl *\makeloesung*, der nur einmal aufgerufen werden kann, gibt alle Lösungen aus.

```
\aufgabe{Erster Kreuzzug}
Wann lebte
Gottfried von Bouillon?
\loesung[Erster Kreuzzug]
{Als er nichts anderes
zu essen hatte.}
\makeloesung
```

3. Erster Kreuzzug

Wann lebte Gottfried von Bouillon?

3. Erster Kreuzzug

Als er nichts anderes zu essen hatte.

Wird beim *loesung*-Befehl ein optionales Argument angegeben, erhält die Lösung dieses als Überschrift, ansonsten ist das Wort *Lösung* die Überschrift.

Ein Alternative zu *chemexec* ist das Paket *exercise*, siehe unten.

DNA- und Proteinsequenzen werden für menschliche Leser üblicherweise als Folge von Zeichenketten aus jeweils 10 Zeichen und mit einer Nummerierung am Zeilenanfang dargestellt. Dafür gibt es das Paket *dnaseq* von Bjørn Pedersen. Man kann Teilsequenzen mit einer Hintergrundfarbe hervorheben.

```
\DNA! CTCGA'{green}GGGG'{white}CCTAGAC%
ATTGCCCTCCAGAGAGAGCACCCACACCCTCCAGGCTT!
```

```
  1 CTCGAGGGGC CTAGACATTG
 21 CCCTCCAGAG AGAGCACCCA
 41 CACCCTCCAG GCTT
```

Die DNA-Zeichenkette wird hier mit dem Zeichen `!` abgeschlossen. Ebenso kann man Proteinsequenzen darstellen, deren Aminosäuren im einbuchstabigen Code wiedergegeben werden.

Lange Zeichenketten wie DNA- oder Proteinsequenzen, die ohne Unterbrechung durch Leerzeichen in mehreren Zeilen dargestellt werden, können meistens an beliebiger Stelle umgebrochen werden. Dafür sorgt der Befehl *seqsplit* aus dem gleichnamigen Paket (siehe Seite 47). Farbige Zeichen sind (mit dem Paket *xcolor*) möglich, wenn diese in geschweiften Klammern stehen.

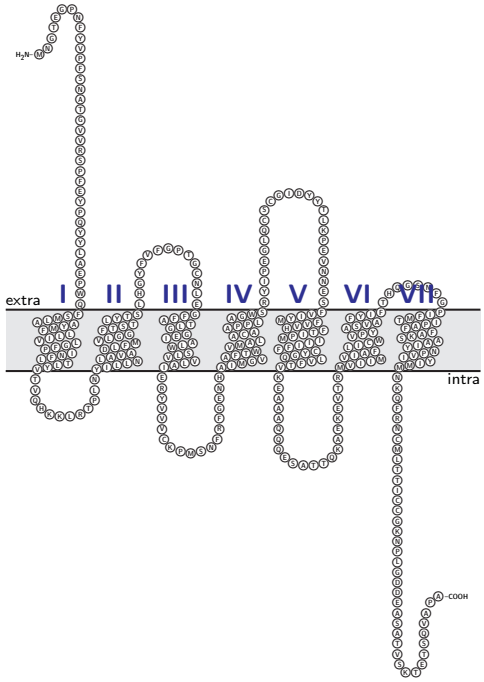
```
\noindent\texttt{\seqsplit{MALWMRLLPLLA
LLALWGPDPAAAFVNQHL{\textcolor{red}{C}}%
GSHLVEALYLV{\textcolor{red}{C}}GERGFFY%
TPKTRREAEDLQVGQVELGGGPGAGSLQPLALEGSLQK%
RGIVEQ{\textcolor{red}{C}}{\textcolor{%
red}{C}}TSI{\textcolor{red}{C}}SLYQLEN%
Y{\textcolor{red}{C}}N}}
```

```
MALWMRLLPLLALLALWGPD
PAAAFVNQHLCGSHLVEALY
LVCGERGFFYTPKTRREAED
LQVGQVELGGGPGAGSLQPL
ALEGSLQKRGIVEQCCTSIC
SLYQLENYCN
```

Obiges Beispiel zeigt menschliches Präproinsulin mit rot gedruckten Cysteinen.

Membranproteine können mit dem Paket *textopo* von Eric Beitz dargestellt werden. Im folgenden Beispiel werden die sieben Transmembrandomänen des menschlichen Proteins *Rhodopsin* abgebildet.

```
\begin{textopo}
\membranecolors{Black}{Gray10} % Farben
\scaletopo{1} % Verkleinerung des Plots
\sequence{ MNGTEGPNFYVPFSNATGVVRSPFEYPQYYLAEPWQ
   [ FSMLAAYMFLLIVLGFPINFLTLYV ]    % 1. TMD
TVQHKKLRTPLN
   [ YILLNLAVADLFMVLGGFTSTLYTS ]    % 2. TMD
LHGYFVFGPTGCNLE
   [ GFFATLGGEIALWSLVVLAI ]         % 3. TMD
ERYVVVCKPMSNFRFGENH
   [ AIMGVAFTWVMALACAAPPLAGWS ]     % 4. TMD
RYIPEGLQCSCGIDYYTLKPEVNNES
   [ FVIYMFVVHFTIPMIIIFFCYGQLVFTV ] % 5. TMD
KEAAAQQQESATTQKAEKEVTR
   [ MVIIMVIAFLICWVPYASVAFYIF ]     % 6. TMD
THQGSNFG
   [ PIFMTIPAFFAKSAAIYNPVIYIMM ]    % 7. TMD
NKQFRNCMLTTICCGKNPLGDDEASATVSKTETSQVAPA }
\Nterm{extra} % N-Terminus extrazellulär
\end{textopo}
```



Mit *membranecolors{...}{...}* werden die Farben der Membran (Rand und Inneres) gewählt und mit dem Befehl *scaletopo{1}* wird die Graphik verkleinert. Der Befehl *sequence{...}* enthält die Aminosäuresequenz im einbuchstabigen Code, wobei Transmembrandomänen in eckige Klammern gefasst sind. Der Befehl *Nterm{extra}* bestimmt, dass das Amino-Ende extrazellulär liegt.

Bioinformatische Sequenzalignments sind Vergleiche von Zeichenketten (informatische Sequenzen), bei denen ähnliche Teilsequenzen sichtbar werden. In der Bioinformatik werden biologische Sequenzen durch Zeichenketten dargestellt, $A = a_1a_2a_3 \ldots a_m$, $B = b_1b_2b_3 \ldots b_n$, wobei die Zeichen Elemente eines Alphabets Σ sind. Beim Alignment werden die Zeichenketten zeilenweise, unter Hinzufügung von Leerzeichen, mit konstanter Zeichenbreite geschrieben, so dass die Bereiche größter Ähnlichkeit direkt untereinander stehen. Das Paket *gotoh* von Takuto Asakura führt ein paarweises Alignment mit dem Gotoh-Algorithmus durch, welcher die Methode der dynamischen Programmierung anwendet.

In folgendem Beispiel sind zwei Nukleotidsequenzen $A =$ `ATCGGCGCACGGGGGA` und $B =$ `TTCCGCCCACA` über dem Alphabet $\Sigma = \{A, C, G, T\}$ gegeben. Das Alignment erhält man nun einfach durch

```
\Gotoh{CCGTATTCACTAC}
{AGTATCAATACCGGGGCGAC}
\texttt{\GotohResultA}\\
\texttt{\GotohResultB}%
\\[3mm] Score \GotohScore
```

```
CCGTATTCA........CTAC
.AGTATCAATACCGGGGCGAC

Score -17
```

Der *Gotoh*-Befehl hat als Pflichtargumente die beiden Zeichenketten. Er speichert das Ergebnis des Alignments in den Befehlen *GotohResultA* und *GotohResultB*. Diese können dann mit normalen LaTeX-Befehlen untereinander geschrieben werden.

Es ist sinnvoll, dafür eine nichtproportionale (dicktengleiche) Schrift zu benutzen, hier mit dem *texttt*-Befehl. Die Kosten des Alignments (Score) finden sich im Befehl *GotohScore*. Den Parametern des Gotoh-Algorithmus (*match*, *mismatch*, *d* und *e*) können vor dem *Gotoh*-Befehl mit dem *GotohConfig*-Befehl Werte zugewiesen werden. Darauf wurde in obigem Beispiel verzichtet, so dass die Standardwerte eingesetzt sind.

Eine graphische Darstellung bioinformatischer Sequenzalignments ist mit dem Paket *texshade* von Eric Beitz möglich. Werden die beiden Pakete *texshade* und *textopo* verwendet, lädt man zuerst *texshade*, danach *textopo*. Ein multiples Sequenzalignment (Alignment mehrerer Sequenzen) wird mit Programmen wie CLUSTALW erzeugt. Das Ergebnis kann in einer Datei gespeichert sein.

Das Paket *texshade* kann Alignmentdateien lesen, die im MSF-format oder im ALN-format geschrieben wurden. In folgenden Beispielen gehen wir von einer Datei *ins_A.aln* aus, welche sich im gleichen Verzeichnis befindet wie die tex-Datei. Diese Alignmentdatei enthält die Aminosäuresequenz der A-Kette des Insulins von fünf verschiedenen Wirbeltierarten.

```
\begin{texshade}
{ins_A.aln}
\seqtype{P}
\shadingmode
{identical}
\threshold
[100]{50}
\showlegend
\hidenumbering
%\hideconsensus
\end{texshade}
```

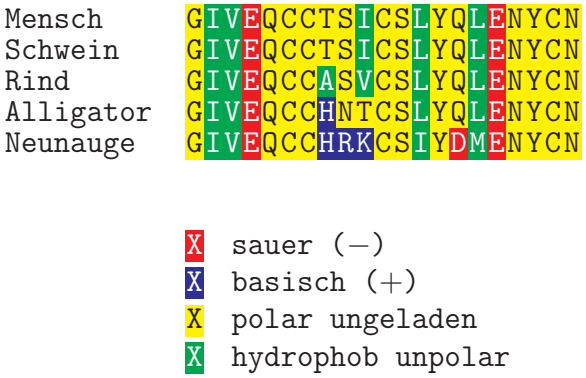
```
Mensch      GIVEQCCTSICSLYQLENYCN
Schwein     GIVEQCCTSICSLYQLENYCN
Rind        GIVEQCCASVCSLYQLENYCN
Alligator   GIVEQCCHNTCSLYQLENYCN
Neunauge    GIVEQCCHRKCSIYDMENYCN
Konsensus   !!!!!!! **!!*!**!!!!!

X  nicht konserviert
X  ≥ 50% konserviert
X  alle identisch
```

Die vom Paket bereit gestellte *texshade*-Umgebung erhält als Pflichtargument den Namen oder Pfad zur Alignmentdatei. Der Befehl *seqtype{P}* gibt an, dass es sich um Aminosäuresequenzen handelt (Alternative: *N* für Nukleotidsequenzen). Der Befehl *shadingmode{identical}* bestimmt, dass spaltenweise die Anzahl der Übereinstimmungen von Zeichen gekennzeichnet wird. Die Hintergrundfarben der Zeichen hängen von der prozentualen Übereinstimmung der Zeichen ab und werden mit dem Befehl *threshold[100]{50}* festgelegt. Die Legende dieser Festlegung wird mit *showlegend* gezeigt und *hidenumbering* unterdrückt die Anzeige der Zeichenzahl pro Zeile. Mit dem Befehl *hideconsensus* (welcher hier durch % auskommentiert wurde) könnte man die Anzeige der Konsensuszeile abschalten.

```
\begin{texshade}
{ins_A.aln}
\seqtype{P}
\shadingmode
[1]{diverse}
\hidenumbering
\end{texshade}
```

```
Mensch      GIVEQCCTSICSLYQLENYCN
Schwein     ....................
Rind        .......a.v..........
Alligator   .......hnt..........
Neunauge    .......hrk..i.dm.....
```

```
\begin{texshade}
{ins_A.aln}
\seqtype{P}
\shadingmode
[hydropathy]
{functional}
\shadeallresidues
\showlegend
\hidenumbering
\end{texshade}
```



Übungsblättter können mit verschiedenen Paketen erstellt werden, zum Beispiel mit *chemexec*, siehe oben. Einfach zu handhaben ist auch das Paket *exercise* von Paul Pichaureau. Es passt die Namen an die verwendete Sprache an und stellt je eine Umgebung für Übungen und Lösungen zu Verfügung. Wenn man es mit

```
\usepackage[lastexercise]{exercise}
```

lädt, schreibt man die Lösung direkt nach der Übung auf. Jede Übung wird mit einem *label* gekennzeichnet. Am Anfang der *document*-Umgebung, vor den Übungen, wird, anhand der *label*, ausgewählt, welche Übungen ausgedruckt werden sollen. Damit kann man aus einer Aufgabensammlung leicht ein besonderes Übungsblatt zusammenstellen. Die Auswahl erfolgt beispielsweise mit

```
\ExerciseSelect[label={4mal5,25qua}]    % Übungen 4mal5 und 25qua
```

Wenn man den standardmäßigen Namen *Übung* durch *Aufgabe* ersetzen möchte, fügt man anschließend den Befehl

```
\renewcommand{\ExerciseName}{Aufgabe}
```

hinzu. Mit dem *ExerciseSelect*-Befehl werden nur die ausgewählten Übungen, nicht aber die Lösungen gedruckt. Will man dagegen alle Aufgaben und Lösungen ausgeben, lässt man den *ExerciseSelect*-Befehl weg (oder kommentiert ihn aus).

Danach folgen die Übungen und Lösungen. Als optionales Argument sollte man in jeder *Exercise*-Umgebung ein *label* haben. Außerdem kann man einen Titel angeben und die Schwierigkeit der Übung mit einer Anzahl Sternchen bezeichnen. Hier ein kurzes Beispiel ohne *ExerciseSelect*-Befehl:

```
\begin{Exercise}
[ label=4mal5,
difficulty=1 ]
Was ist $4 \cdot 5$ ?
\end{Exercise}
\begin{Answer}
20
\end{Answer}

\begin{Exercise}[
label=25qua,
%name=Rechenaufgabe,
title={Quadrat},
difficulty=2
]
Was ergibt $25^2$ ?
\end{Exercise}
\begin{Answer}
625
\end{Answer}
```

* Übung 1

Was ist $4 \cdot 5$?

Lösung zu Übung 1

20

** Übung 2 Quadrat

Was ergibt 25^2 ?

Lösung zu Übung 2

625

Hyperlinks können benutzt werden. Man muss jedoch das Paket *hyperref* vor dem Paket *exercise* laden.

Schriftliche Prüfungen kann man mit der von Philip Hirschhorn geschaffenen Dokumemtklasse *exam* erstellen. Das folgende Beispiel erscheint hier ziemlich gedrängt. Die Anwendung ist jedoch einfach.

```
\documentclass[addpoints,a4paper,ngerman,12pt]{exam}
 \usepackage[left=2cm,right=1.5cm]{geometry}      % Seitenränder
 \usepackage{babel}                \usepackage[utf8]{inputenc}
 \usepackage[T1]{fontenc}          \usepackage{sectsty}
% Deutsche Worte statt der vorgegebenen englischen
   \pointpoints{Punkt}{Punkte}     \hpgword{Seite:}
   \hpword{Punkte:}                \hsword{Ergebnis:}
   \hqword{Aufgabe:}               \htword{Summe}
% Schriftgröße und Zeilenabstand für Abschnittsüberschriften
   \sectionfont{\fontsize{12}{15}\selectfont} % benötigt sectsty
 \newcommand{\vd}{\vspace{10mm}}    % Abstand nach einer Aufgabe
% Festlegung des allgemeinen Formats von Kopf- und Fußzeilen
   \header{Klausur Umbratikulogie}{1. April 2017}{Prof. Barcala}
   \headrule   % horizontale Linie unter der Kopfzeile
   \footer{}{}{\thepage\,/\,\numpages} % Fußzeile mit Seitenzahl
\begin{document}
\noprintanswers % ohne die Lösungen, mit Lösungen: \printanswers
\parbox{16cm}{ \makebox[7.8cm]{Name:\enspace\hrulefill}   % Name
\makebox[8.1cm]{Vorname:\enspace\hrulefill} } % Vorname ins Feld
   \begin{questions}
      \section*{Künstliche Intelligenz}
\question[2] Was ist Intelligenz?   % 2 Punkte für die Aufgabe
```

```
\begin{solution} Das, was der Test misst. \end{solution}   \vd
      \section*{Natürliche Dummheit}
\question[3] Wer erfand das Rad?    % 3 Punkte für die Aufgabe
\begin{checkboxes} % Mehrfachauswahl-Aufgabe (Multiple Choice)
\choice Moses \choice Newton \choice Einstein \end{checkboxes}
   \end{questions}   \vspace{3mm}
\gradetable[h][questions]    % Tabelle mit Punkten pro Aufgabe
\end{document}       % pages statt questions: Punkte pro Seite
```

Vorteilhaft sind: a) die automatische Nummerierung der Aufgaben, b) eine automatisch aktualisierte Tabelle mit den Punktzahlen pro Aufgabe (oder pro Seite), und c) die Möglichkeit, Lösungen im Quellcode einzufügen und mit einem besonderen Befehl sichtbar zu machen. Um die Punktetabelle automatisch zu aktualisieren, muss drei Mal kompiliert werden. Man kann auch Aufgaben in Teilaufgaben zerlegen (hier nicht gezeigt).

7.2 Gedichte, Spiele, Strichcode und QR-Code

Gedichte können mit der Umgebung *verse* dargestellt werden. Beispiel:

```
Es folgt ein Gedicht:
\begin{verse}
Good Guys aimed to please,\\
Bad Guys used the club.

Life is live and cruel,\\
Man, ' trusts man, is fool.

The Good Guys rest in peace,\\
And Bad Guys rest in pub.
\end{verse}
```

Es folgt ein Gedicht:

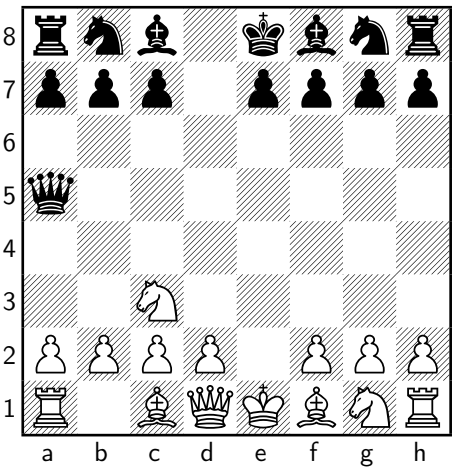
> Good Guys aimed to please,
> Bad Guys used the club.
>
> Life is live and cruel,
> Man, ' trusts man, is fool.
>
> The Good Guys rest in peace,
> And Bad Guys rest in pub.

Die Zeilen des Gedichts, außer der letzten Zeile jeder Strophe, werden mit \\ abgeschlossen. Die Strophen werden durch Leerzeilen getrennt. Das Gedicht erscheint eingerückt und zwischen den Strophen ist ein kleiner Abstand. Die ohne weiteres verfügbare *verse*-Umgebung kann, durch Laden des Pakets *verse* (von Peter R. Wilson), ersetzt werden. Damit erhält man mehr Darstellungsmöglichkeiten.

Schach wird mit dem Paket *skak* von Torben Hoffmann dargestellt. Es kann entweder den Verlauf einer Partie wiedergeben oder eine vorgegebene Stellung. Die in einer Partie oder Schachaufgabe erreichte Stellung kann als Schachbrettzeichnung abgebildet werden. Bei der Beschreibung einer Partie werden die eingegebenen Züge gespeichert, so dass der Verlauf einer Partie wiedergegeben und die erreichte Stellung jederzeit gezeigt werden kann. Der Befehl *newgame* beginnt ein Spiel und mit *mainline* können die Züge aufgeschrieben werden. Die Offiziere werden bei der Angabe der Züge für Weiß und Schwarz mit englischen Abkürzungen gekennzeichnet: König K, Dame Q, Turm R, Läufer B und Springer N. Eine Variante wird mit *variation* notiert und *lastmove* gibt den letzten Zug aus. Die erreichte Stellung wird mit *showboard* abgebildet. Folgendes Beispiel demonstriert die Anwendung der Befehle.

```
\newgame % neues Spiel
\mainline{1.e2e4 d5}
Skandinavisch \par
Der Zug \lastmove{}
ist aggressiv. \par
  % der Zug d7d5 wird
  % kurz d5 geschrieben
\mainline{2.e4xd5}
nicht \variation{2.e5}
\par \mainline{2...Qd5
3.Nc3 Qa5}
  % Offiziere mit Groß-
  % buchstaben, Q=Queen
  % K=King, R=Rook, B=
  % Bishop, N=Knight
\par \bigskip
\showboard % Stellung
```

1 e4 d5 Skandinavisch
Der Zug 1. . . d5 ist aggressiv.
2 e×d5 nicht 2 e5
2. . . ♕d5 3 ♘c3 ♕a5



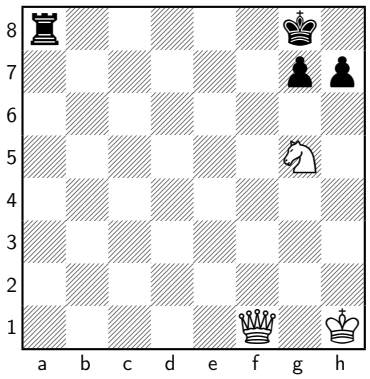
Eine Schachaufgabe mit vorgegebener Stellung wird mit

```
\newgame
\fenboard{r5k1/6pp/8/6N1/8/8/8/5Q1K w - - 0 1}
\scalebox{0.8}{\showboard}
```

dargestellt (Abbildung, siehe unten). Der Befehl *\newgame* steht am Anfang einer neuen Zugfolge oder Stellung und der Befehl *\fenboard{. . . }* beschreibt eine Stellung in der Forsyth-Edwards-Notation (FEN), welche hier nicht erläutert werden soll. Zu beachten ist, dass bei der FEN (abweichend von obiger Zug-Notation) die Figuren von Weiß mit K, Q, R, B, N und P (Bauer) und die von Schwarz mit k, q, r, b, n und p bezeichnet werden. Die Abbildung des Schachbretts erfolgt wieder mit dem Befehl *showboard*. Da die Abbildungsgröße meistens angepasst werden soll, skaliert man sie (im Beispiel auf 0,8 mal die Originalgröße) mit dem Befehl *scalebox{. . . }{. . . }*, der verfügbar ist, wenn das Paket *graphicx* (oder das ältere *graphics*-Paket) geladen wurde.

Schachaufgabe von Siegbert Tarrasch.

Weiß am Zug setzt in 5 Zügen matt.

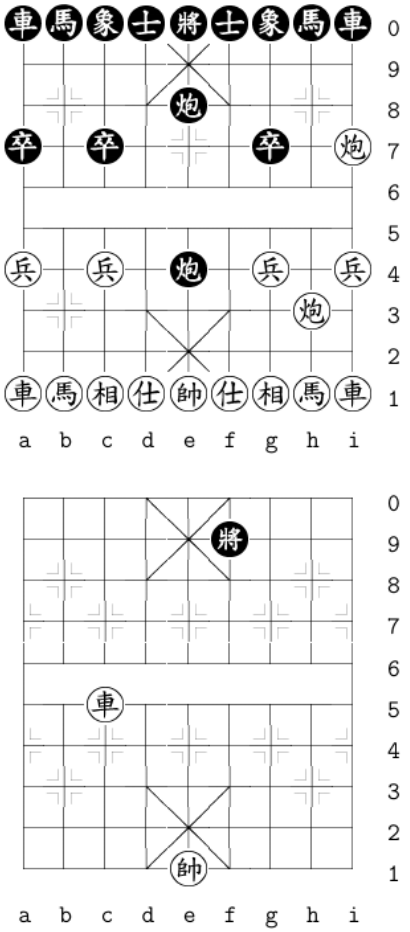


Xiangqi oder chinesisches Schach kann mit dem Paket *xq* von Stephan Weinhold und Sebastian Pipping aufgezeichnet werden. Leider hat das Paket einige Fehler, so dass es beispielsweise mit der Dokumentklasse *article*, nicht aber mit *scrartcl* läuft. Das unten gegebene Beispiel schließt daher den *documentclass*-Befehl ein. Zunächst wählt man eine Sprache für die Zeichen, welche den Spielfiguren zugeordnet werden.

Möglich sind unter anderem Deutsch (*german*) und Englisch (*english*). Die roten Spielfiguren und ihre Zeichen (in Deutsch beziehungsweise Englisch) sind: Feldherrr (F, K), Mandarin (M, A), Elefant (E, E), Wagen (W, R), Kanone (K, C), Pferd (P, H) und Soldat (S, P). Die entsprechenden schwarzen Spielfiguren werden mit Kleinbuchstaben bezeichnet. Der Befehl *newgame* beginnt ein neues Spiel. Züge werden mit dem Befehl *move* eingegeben, wobei roter und schwarzer Zug nacheinander mit jeweils vier Zeichen angegeben werden. Jeweils zwei Zeichen stehen für Start- und Zielkoordinaten. Deren Bedeutung ergibt sich aus der Abbildung des Spielfelds. Soll der rote oder schwarze Zug fehlen, wird stattdessen *xxxx* geschrieben. Der Befehl *cr* beziehungsweise *cb* fügt dem folgenden Zug von Rot beziehungsweise Schwarz einen Kommentar hinzu. Das Spielfeld und die erreichte Stellung werden mit *showboard* ausgegeben.

Der Befehl *textpiece* bindet Spielsteine in den Text ein. Mit *resetboard* erzeugt man ein leeres Spielbrett, auf dem man mit *piece* Spielsteine setzen kann. Danach wird das Brett mit *showboard* gezeigt. Folgendes Beispiel verdeutlicht die Anwendung der Befehle.

```
\documentclass{article}
\usepackage{xq}
\begin{document}
\mylanguage{german}
\newgame   % Spiel beginnen
\move b3e3 b8e8   % 1. Zug
\move e3e7 e8e4   % 2. Zug
\cr{ ?}  % Kommentar, 3. rot
\move e7i7 xxxx   % 3. rot
\cb{ !}  % Komm., 3. schwarz
\move xxxx h8e8 % 3. schwarz
\showboard \textbf{Matt\,!}
%
\par Der rote Feldherr
\textpiece{F} sieht anders
aus als der schwarze
\textpiece{f} . \par
%
\resetboard % leeres Brett
\piece ff9 % schwarzer und
\piece Fe1 % roter Feldherr
\piece Wc5 % und ein Wagen
\showboard % zeige Stellung
\end{document}
```



Sudoku ist ein mathematisches Rätselspiel auf Grundlage einer gezeichneten quadratischen Tabelle mit 9 Teilquadraten. Jedes Teilquadrat besteht aus 3×3 Feldern, die mit den Ziffern 1 bis 9 gefüllt werden sollen. Einige Ziffern sind bereits vorgegeben. Die fehlenden Ziffern sollen durch logische Überlegungen gefunden werden. Mit dem Paket *sudoku* von Paul Abraham kann man solch eine Rätseltabelle abbilden, zum Beispiel:

```
\scalebox{0.65}{
\begin{sudoku-block}
| | | |8|6| | | | |.
| | | | | | | |5|2|.
|3| | |7| | | |4| |.
|6|9| | | |2| | |7|.
| |1| |3| | | | | |.
| |2|4| | |1| |8|9|.
| | |7| |4| |6| | |.
| | |8| |7|9|4| | |.
|2| |1| | | |7| | |.
\end{sudoku-block}
}
```

			8	6				
							5	2
3			7				4	
6	9				2			7
	1		3					
	2	4			1		8	9
		7		4		6		
		8		7	9	4		
2		1				7		

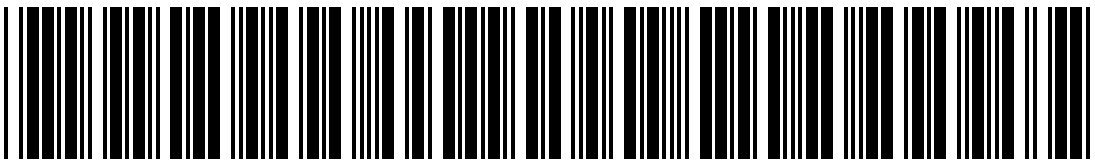
Strichode ist eine maschinenlesbare Schrift aus unterschiedlich breiten, parallelen Strichen. Das Paket *makebarcode* von Zdeněk Wagner ermöglicht es, verschiedene eindimensionale Strichcodes zu erzeugen. Beim Laden des Pakets kann man als optionale Argumente den Typ des Strichcodes und dessen Größe angeben. Mit

```
\usepackage[code=Code39,X=0.5mm,ratio=3,H=2cm]{makebarcode}
```

lädt man *makebarcode* mit Standard Code 39, Modulbreite 0,5 mm, Breitenverhältnis (zwischen schmalen und breiten Balken) 3, Höhe 2 cm. Der Zeichensatz (Alphabet) enthält die englischen Großbuchstaben, die Ziffern 0 bis 9, das Leerzeichen und die sechs Zeichen `+-.$/%`. Die Länge ist grundsätzlich nicht beschränkt. Bei der Wiedergabe des Codes im Klartext werden manchmal Start und Stopp als $*$ geschrieben. Vorteile von Standard Code 39 sind sind die weite Verbreitung (bei den meisten Strichcodelesegeräten) und gute Maschinenlesbarkeit. Nachteile sind der kleine Zeichensatz und die geringe Informationsdichte (großer Platzbedarf). Zum Beispiel wird mit

```
\barcode{ALFRED H. GITTER}
```

der Strichcode

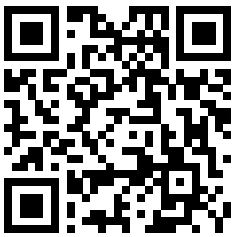


ausgegeben, welcher die Zeichenkette ALFRED H. GITTER darstellt.

QR-Code ist eine quadratische Zeichnung, die aus kleinen schwarzen Quadraten besteht und eine Zeichenkette codiert. Mobiltelefone und Computer mit Kamera verfügen oft über ein Programm, welches QR-Code lesen und decodieren kann. Beispielsweise können so Internetadressen bequem von einem Papierausdruck abgelesen und in einem Browser geöffnet werden. Maximal kann ein QR-Code knapp 3 kB an Information beinhalten. QR-Code kann mit dem Paket *qrcode* von Anders O. F. Hendrickson erzeugt werden. Beispiel:

QR-Code mit 3 cm Seitenlänge
für eine Wikipedia-Webseite

```
\qrcode[height=3cm]{
https://de.wikipedia.org/wiki/QR-Code
}
```



7.3 Präsentation mit *beamer*

Für die Bild-Projektion während eines Vortrags empfiehlt sich die Dokumentklasse *beamer*. Der folgende Quellcode zeigt ein kurzes Beispiel. Die Abfolge der Befehle wird durch die zugehörigen Erläuterungen unterbrochen.

Die Präambel enthält einen einleitenden *documentclass*-Befehl, in dem die Dokumentklasse *beamer* gewählt wird. Im Gegensatz zu anderen Dokumentklassen wird bei *beamer* der Text linksbündig (nicht im Blocksatz) und ohne automatische Silbentrennung dargestellt. Aufgrund der geringen Textmenge pro Seite kann man bei Bedarf Trennungen manuell einsetzen.

```
\documentclass[14pt]{beamer}                    % Dokumentklasse
```

Die Schriftgröße wird hier mit dem relativ großen Wert 14 pt festgelegt, da die Textmenge bei der Präsentation gering, die Entfernung des Lesers jedoch sehr groß sein kann. Zur Auswahl stehen bei *beamer* mehr Schriftgrößen als bei anderen Standard-Dokumentklassen, nämlich 8 pt, 9 pt, 10 pt, 11 pt, 12 pt, 14 pt, 17 pt und 20 pt. Ohne besondere Festlegung der Schriftgröße verwendet *beamer* die Größe 11 pt. Das Format einer Seite, bei einer Projektion (aus historischen Gründen) auch Folie genannt, braucht nicht angegeben zu werden. Es beträgt 128 mm × 96 mm, aus dem sich ein Seitenverhältnis von 4:3 ergibt.

Wir fügen nun eine Zeile ein, die den Befehl *Tiny* anweist, nicht Schriftgröße 4, sondern (wie der Befehl *tiny*) Schriftgröße 5 zu verwenden.

```
\let\Tiny=\tiny       % Warnung (size <4> not available) verhindern
```

Damit verhindern wir die Warnung, dass Schriftgröße 4 nicht zur Verfügung steht. Da diese Schriftgröße ohnehin nicht gebraucht wird, ist die Sache eigentlich belanglos, aber wir möchten keine überflüssige Warnung (*Font shape 'OT1/cmss/m/n' in size <4> not available*) bekommen.

Dann werden, wie in anderen Dokumentklassen, Pakete geladen.

```
\usepackage[ngerman]{babel}                     % Pakete laden
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{graphicx, multimedia}
\usepackage{tikz, amsfonts, amsmath}
```

Mit den ersten drei Befehlen wird die deutsche Sprachanpassung vorgenommen. Dann werden Pakete für die Einbindung von Bildern, Filmen, Grafiken, mathematischen Symbolen und Gleichungen geladen. Die Pakete *xcolor* für farbigen Text und *hyperref* für Hyperlinks werden von der Dokumentklasse *beamer* automatisch geladen. Damit stehen die Farben des Pakets *xcolor* zur Verfügung, die in Abschnitt 4.1 (Seite 54) genannt wurden.

Das Layout einer Präsentation wird im Wesentlichen durch Wahl eines Themas vorgegeben, wozu der Befehl *\usetheme* dient.

```
\usetheme{default}                                  % Layout-Vorgabe
```

Hier wird das einfache Standard-Thema *default* gewählt, welches eine Seite ohne Rand-Spalten darstellt und schwarz auf weißem Hintergrund schreibt.

Will man für die Seitenüberschrift einen farbigen, statt eines weißen Hintergrunds (Standardeinstellung, *default*), kann man den Befehl *usecolortheme{*option*}* anfügen, wobei *option* zum Beispiel *seahorse* oder *wolverine* sein kann. Derartige Malkunst lenkt jedoch vom Inhalt ab.

Einige Standardvorgaben sollte man ändern. Mit

```
\setbeamertemplate{navigation symbols}{}          % Navi-Zeile löschen
```

wird die Einblendung der *navigation symbols* ausgeschaltet, die ansonsten am unteren Rand jeder Seite erscheinen würden. Dann wird mit

```
\setbeamerfont{page number in head/foot}{size=\small}   % S.zahlgröße
```

die Schriftgröße für die Seitenzahl auf *small* gesetzt und mit

```
\setbeamertemplate{footline}[frame number]        % Seitenzahl am Fuß
```

in der Fußzeile (*footline*) die Angabe der aktuellen Seite und der Seiten-Anzahl mit der Option *frame number* angefordert. Nun wird das Format der später automatisch eingefügten Abschnittsseiten (*sectionpage*) festgelegt.

```
\setbeamertemplate{section page}                    % Format der
{  \begin{centering}                                % Abschnitts-Seite
   \usebeamerfont{section title}\insertsection\par
   \end{centering}  }
```

In einer *centering*-Umgebung mittig ausgerichtet, wird hier nur der Abschnittstitel eingesetzt (in der dafür standardmäßig vorgesehenen Schriftart). Ähnlich wie in anderen Dokumentklassen wird die Titelseite vorbereitet.

```
\title{Dieser Tag ein Leben}                        % Format der
   \author{Melcher Melcherson}                      % Titelseite
%   \institute{Familie Melcherson aus Stockholm}
   \date{}
```

Bei wissenschaftlichen Vorträgen wird unter dem Redner (*author*) noch das Institut genannt, aus dem dieser kommt. Hier entfällt das (die betreffende Zeile ist auskommentiert). Zu Beginn jedes Abschnitts soll automatisch eine *sectionpage* eingefügt werden, die den Titel des Abschnitts mitteilt. Dies wird mit

```
\AtBeginSection{\frame{\sectionpage}}               % Abschnitts-Seite
```

veranlasst. Nun haben wir (bis auf später erläuterte, mögliche Ergänzungen) das Ende der Präambel erreicht.

Der Textinhalt beginnt mit der *document*-Umgebung.

```
\begin{document}                        % Anfang der document-Umgebung
```

Nach dem Anfang der *document*-Umgebung folgt die Titelseite.

```
\begin{frame} \titlepage \end{frame}                    % Titelseite
```

Eine Präsentationsseite heißt *frame* und wird daher in einer *frame*-Umgebung erzeugt. Hier wird die Titelseite aus den in der Präambel gemachten Angaben automatisch gestaltet. Die Präsentation kann mit dem *section*-Befehl in verschiedene Abschnitte unterteilt werden. Der Titel eines Abschnitts wird in geschweiften Klammern angegeben.

```
\section{der Ferien erster Teil}                        % Abschnitt 1
```

Eine Liste aller Abschnittstitel kann, bei geeignetem Layout (nicht mit der hier empfohlenen *default*-Einstellung), am Rand jeder Seite in einer Navigationsspalte gezeigt werden.

Es ist in Vorträgen üblich, einzelne Teile einer Seite nacheinander sichtbar zu machen. Dies lässt sich einfach erreichen, indem die Anzeige mit dem Befehl *pause* unterbrochen wird, bis der Vortragende durch Drücken der Eingabetaste oder der linken Maustaste die Fortsetzung bewirkt.

```
\begin{frame}{Die Familie}
   Meine Kinder:
   \begin{itemize}
      \pause \item{Malin}
      \pause \item{Johann}
      \pause \item{Niklas}
      \pause \item{Pelle}
         \begin{itemize}\item{Jocke}\end{itemize}
   \end{itemize}
\end{frame}
```

In obigem Beispiel werden die Namen der Kinder der Familie Melcherson nacheinander sichtbar gemacht. Im *handout*-Modus (siehe unten) werden die *pause*-Befehle nicht beachtet und alles ohne Unterbrechung auf eine Seite geschrieben. Die Einblendung einzelner Teile der Seite nennt man Overlay. Die schrittweise Darstellung des Seiteninhalts soll die Aufmerksamkeit des Lesers auf den gerade besprochenen Teil lenken. Der Leser wird dadurch jedoch bevormundet und die Einblendung lenkt *per se* den Leser vom Inhaltlichen ab. Daher sollte man Overlays, jedenfalls in wissenschaftlichen Vorträgen, sparsam verwenden.

Oft braucht man zwei Spalten auf einer Seite, zum Beispiel um Text neben ein Bild zu platzieren. Die Dokumentklasse *beamer* stellt hierfür (im Gegensatz zu anderen Dokumentklassen) die *columns*-Umgebung bereit, innerhalb derer man Spalten mit jeweils einer *column*-Umgebung erzeugen kann.

```
\begin{frame}{Die Vegetation}
   \begin{columns}
      \begin{column}{0.5\textwidth}
      \IfFileExists{bild1.jpg}    % prüfe, ob die Bilddatei existiert
        { \includegraphics[scale=0.5]{bild1.jpg} } % Bilddatei ist da
        { \textcolor{red}{Bild-Datei fehlt} }       % Bilddatei fehlt
      \end{column}
      \begin{column}{0.3\textwidth}
         \centering % centering-Befehl zentriert (in dieser Umgebung)
            blau blüht der Ehrenpreis\\
```

```
            neben dem Schreinerhaus
      \end{column}
   \end{columns}
\end{frame}
```

In obigem Beispiel werden zwei Spalten gebildet. In der linken Spalte wird mit dem *includegraphics*-Befehl ein Bild aus der Datei `bild1.jpg` geladen. In der rechten steht ein erklärender Text, der mit *\centering* mittig gesetzt wird.

Nach dem Beginn des zweiten Abschnitts soll ein Bild gezeigt werden, das aus urheberrechtlichen Gründen nicht im Handout (siehe unten) erscheinen darf. Um die Befehle zur Darstellung des Bildes nur bei der Präsentation wirken zu lassen, schließt man sie in die geschweiften Klammen nach einem *only<beamer>*-Befehl ein.

```
\section{der Ferien zweiter Teil}                    % Abschnitt 2
\begin{frame}{Blick über die große Ostsee}
   \centering       % centering-Befehl zentriert (in dieser Umgebung)
   \only<beamer>{
%   \hspace{-0.9cm}  % der hspace-Befehl verrückt das Bild horizontal
      \IfFileExists{bild2.jpg}    % prüfe, ob die Bilddatei existiert
        { \includegraphics[scale=0.7]{bild2.jpg} } % Bilddatei ist da
        { \textcolor{red}{Bild-Datei fehlt} }       % Bilddatei fehlt
              }
\end{frame}
```

In obigem Beispiel wird ein Bild aus der Datei `bild2.jpg` eingefügt. Wenn das Bild sehr groß dargestellt werden soll, kann man es nach links verschieben und den leeren Textrand nutzen. Dies würde durch den *hspace*-Befehl mit einer negativen Längenangabe erreicht werden. Manchmal möchte man ein Bild seitenfüllend, also mit maximaler Höhe und Breite, darstellen. Dabei nimmt man in Kauf, dass das Bild verzerrt wird, wenn das Seitenverhältnis Höhe / Breite der Bildvorlage nicht mit dem der Seite übereinstimmt.

```
\IfFileExists{bild3.jpg}          % prüfe, ob die Bilddatei existiert
 { { \setbeamertemplate{background canvas}{\includegraphics[%
  height = \paperheight,width = \paperwidth]{bild3.jpg}}
\begin{frame}[plain] \end{frame} } }               % Bilddatei ist da
 { \textcolor{red}{Bild-Datei fehlt} }              % Bilddatei fehlt
```

Obige Befehle dehnen das Bild automatisch. Da das Bild die ganze Seite ausfüllt, ist kein Platz für eine Überschrift oder die Anzeige der Seitenzahl. Die Seitenzahl wird aber um 1 erhöht.

Filme kann man mit dem *movie*-Befehl in die Seite einbetten, wenn der PDF-Viewer diese selbst abspielt oder auf einen vom Betriebssystem des Rechners bereitgestellten Mediaplayer zugreifen kann, welcher das gewünschte Film-Format unterstützt. Ein Film sollte natürlich nur im *beamer*-Modus, nicht im Handout erscheinen.

Besonderheiten bestehen in der Dokumentklasse *beamer* bei der Einbindung bestimmter Umgebungen in eine *frame*-Umgebung. Betroffen sind unter anderem die *verbatim*-Umgebung, die *lstlisting*-Umgebung des Pakets *listings*, und die *comment*-Umgebung des Pakets *comment*. In diesen Fällen muss der *frame*-Umgebung die Option *fragile* hinzugefügt werden:

```
\begin{frame}[fragile]{Test}
\begin{verbatim}
\LaTeX
\end{verbatim}
\end{frame}
```

Inhaltsverzeichnisse werden mit dem Befehl *tableofcontents* gebildet.

```
\begin{frame}{Inhalt meines Vortrags}
   \tableofcontents
\end{frame}
```

Ein zweispaltiges Inhaltsverzeichnis erlaubt es, mehr Gliederungspunkte darzustellen. Da die Aufteilung in Spalten automatisch erfolgen soll, benutzt man nicht die *column*-Umgebung, sondern lädt (in der Präambel) das Paket *multicol* mit der Umgebung *multicols* (mit s am Ende) und schreibt

```
\begin{frame}{Inhalt meines Vortrags}
\begin{multicols}{2}
   \tableofcontents
\end{multicols}
\end{frame}
```

Backup-Seiten werden bereit gehalten, um bei Bedarf zusätzlich gezeigt zu werden. In einer Präsentation soll auf jeden Fall eine bestimmte Mindestanzahl von Seiten gezeigt werden. Die möglicherweise darüber hinaus gezeigten Seiten (*Backup*) sollen bei der Berechnung der Seiten-Anzahl nicht berücksichtigt werden. Um einen Bereich mit solchen Backup-Seiten bilden zu können, werden in der Präambel (vor *begin{document}*) zwei neue Befehle definiert. Zunächst der Befehl *beginbackup*

```
\newcommand{\beginbackup}                              % Beginn der
{  \newcounter{framenumbervorappendix}                 % Backup-Seiten
   \setcounter{framenumbervorappendix}{ \value{framenumber} }  }
```

und dann der Befehl *backupend*

```
\newcommand{\backupend}                        % Ende der Backup-Seiten
{  \addtocounter{framenumbervorappendix}{ -\value{framenumber} }
   \addtocounter{framenumber}{ \value{framenumbervorappendix} }  }
```

Am Ende des regulären Vortragsteils beginnt mit unserem neu definierten Befehl *beginbackup* der, Backup genannte, Zugabe-Teil der Präsentation.

```
\beginbackup                                                  % Backup
\begin{frame}<handout:0>{Bootsfahrt im Schärengarten}
   \only<beamer>{
   \movie[autoplay, width=200pt, height=150pt]{}
   {film.mp4}   }
\end{frame}
\backupend
```

Mit dem ebenfalls oben definierten Befehl *backupend* endet das Backup. Die Seiten des Backup, welche nicht im Handout erscheinen sollen, werden mit der Option

<*handout:0*> begonnen (siehe unten).

```
\end{document}                          % Ende der document-Umgebung
```

Nach dem Backup endet die *document*-Umgebung.

Wasserzeichen können den Präsentationsseiten mit dem Paket *background* von Gonzalo Medina hinzugefügt werden. Folgendes Beispiel zeigt die Anwendung.

```
\documentclass[14pt]{beamer} \let\Tiny=\tiny % Warnung verhindern

\usepackage{background}
\backgroundsetup{angle=30,scale=6,contents={Beispiel},opacity=0.1}
\setbeamertemplate{background}{\BgMaterial}
\setbeamertemplate{navigation symbols}{} % Navi-Zeile löschen

\begin{document}
\begin{frame}{Ich bin nur ein Beispiel.}Hallo Welt\,!\end{frame}
\end{document}
```

Handout ist eine gedruckte Kurzfassung der Präsentation. Dafür wird der *documentclass*-Befehl um die Option *handout* erweitert. Will man mehrere, zum Beispiel 8 (möglich sind auch 2, 4, oder 16), Präsentationsseiten auf einer Druckseite unterbringen, kann man das Paket *pgfpages* verwenden (eine Alternative ist das Paket *pdfpages* von Andreas Matthias). Folgende Befehle erzeugen ein Handout auf DIN-A4-Seiten im Hochformat.

```
\documentclass[14pt,handout]{beamer}
\usepackage{pgfpages}
\pgfpagesuselayout{8 on 1}[a4paper,border shrink=5mm]
```

Will man 4 oder 16 Präsentationsseiten auf eine DIN-A4-Seite drucken, sollte die Option *landscape* im *pgfpagesuselayout*-Befehl ergänzt werden.

Wenn, wie oben (Seite 118), ein Teil der Präsentationsseite nur im *beamer*-Modus gezeigt wird, dann wird im *handout*-Modus immernoch die Seite mit Überschrift (und eventuell weiterem Text) dargestellt. Soll dagegen eine ganze Seite im Handout fehlen, fügt man an den Seitenbeginn die Option <*handout:0*> an, zum Beispiel (für obige Seite) *begin{frame}*<*handout:0*>*{der Ferien zweiter Teil}* . Umgekehrt würde mit der Option <*beamer:0*> eine Seite nur im Handout erscheinen und im *beamer*-Modus fehlen. Damit kann man zwei Ausführungen einer Seite erstellen, eine für die Präsentation und die Alternative für das Handout.

Mehr Information zur Dokumentklasse *beamer* gibt es im ausführlichen Benutzerhandbuch von Till Tantau, Joseph Wright und Vedran Miletić, in einer Präsentation von Ki-Joo Kim und in Beispielen von Sascha Frank.

Eine Alternative zu *beamer* ist zum Beispiel die Dokumentklasse *elpres*, welche im Paket *elpres* von Volker Kiefel definiert wird. Damit kann man Präsentationen einfacher erstellen. Es fehlen die umfangreichen Gestaltungsmöglichkeiten des Layouts und andere Extras von *beamer*. Aber vielleicht braucht man die ja nicht.

7.4 Briefe

Mehrere Pakete stellen besondere *Dokumentklassen für Briefe* bereit, zum Beispiel *letter* von Leslie Lamport, Frank Mittelbach und Rainer Schöpf. Im folgenden wird die Dokumentklasse *scrlttr2* verwendet, welche Teil des Pakets *KOMA-Script* von Markus Kohm ist. Sie bietet zahlreiche Anpassungsmöglichkeiten, aber wir beschränken uns auf eine einfache, weitgehend automatische Formatierung. Der folgende Quellcode zeigt ein Beispiel mit Erläuterungen.

Da ein Brief oft als Vorlage für den nächsten Brief dient, ist es sinnvoll, alle Textbausteine als Variablen zu definieren, die wir möglicherweise in einem Brief verwenden könnten. Bei sehr umfangreicher Korrespondenz mag es sinnvoll sein, derartige Definitionen in einer besonderen Datei auszulagern. Wir bescheiden uns hier aber mit der einfachen Variante *Alles in einer Datei*.

In den Optionen des *documentclass*-Befehls wird nur entschieden, welche Textbausteine im aktuellen Brief gezeigt werden. Die Wertzuweisung erfolgt später. Man beginnt die Präambel mit

```
\documentclass[paper=a4,fontsize=12pt%
,fromalign=center   % andere Werte: left, right
,fromphone          % Telephonnummer zeigen
,frommobilephone    % Handynummer zeigen
,fromemail          % E-Mail-Adresse zeigen
%,fromurl          % URL der Homepage zeigen
%,fromlogo         % Logo (Bild) zeigen
]{scrlttr2}
```

wobei die Schriftgröße hier auf 12 pt festgelegt wurde, um eine gute Lesbarkeit auch für Menschen mit schlechterem Sehvermögen zu erreichen. DIN 5008 empfiehlt eine Schriftgröße von mindestens 10 pt. Die Platzierung der Absenderdaten im Oberteil der ersten Seite wird mit der Variablen *fromalign* bestimmt. Die Nennung weiterer Variablen bewirkt, dass die zugehörigen Textbausteine gezeigt werden. Durch Voranstellung des %-Zeichens (Auskommentieren) wird eine Zeile unwirksam und der zugehörige Textbaustein nicht gezeigt. Entsprechend wird mit folgender Zeile (ohne %) das Bankkonto am Ende der ersten Seite angegeben.

```
%\setkomavar{firstfoot}{\usekomavar{frombank}}
```

Anschließend werden die Pakete für deutsche Sprachunterstützung geladen.

```
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
```

Will man eine andere Schrift, lädt man das entsprechende Paket.

```
\usepackage{libertine} % benutze freie Schriftart Linux Libertine
```

Will man Bilder, zum Beispiel ein Logo, einfügen, braucht man das *graphicx*-Paket.

```
\usepackage{graphicx}
```

Nun erfolgt die Wertzuweisung an die Variablen für Textbausteine, welche Name, Adresse und Kommunikationsdaten des Absenders beschreiben.

```
\setkomavar{fromname}{Jim Knopf}
```

```
\setkomavar{fromaddress}{Bahnhofstraße 1\\12345 Lummerland}
\setkomavar{fromphone}{(\,01\,23\,) 20\,76\,33}
\setkomavar{frommobilephone}{01\,51\,25\,76\,24\,57}
\setkomavar{fromfax}{(0123) 45679}
\setkomavar{fromemail}{jimknopf@mollymail.lul}
\setkomavar{fromurl}[Homepage: ]{http://www.eah-jena.de/$\sim$gitter}
```

In der Absenderadresse erzeugt \\ einen Zeilenumbruch. Mit *$\sim$* wird in der URL-Zeichenkette das Zeichen ~ (Tilde) erzeugt. Um ein Logo aus einer Bilddatei *logo* einfügen zu können, wird der Dateiname bekannt gemacht:

```
\setkomavar{fromlogo}{\includegraphics{logo}}   % Bilddatei für Logo
```

Die Bankverbindung des Absenders wird in der Variablen *frombank* gespeichert.

```
\setkomavar{frombank}{IBAN: DE49 3002 0900 1805 6122 25\\
bei der TARGOBANK (Düsseldorf), BIC: CMCIDEDD}
```

In die *document*-Umgebung wird eine *letter*-Umgebung gesetzt, welche den Brieftext enthält. Pflichtargument ist die Adresse des Empfängers.

```
\begin{document}
\begin{letter}{Wilde 13\\c/o Mahlzahn\\Alte Straße 133\\Kummerland}
```

In der Empfängeradresse erzeugt \\ einen Zeilenumbruch.

Nach der Empfängeradresse wird rechts ein Datum ausgegeben, wobei das aktuelle Datum voreingestellt ist. Will man ein anderes Datum, kann man es selbst, zum Beispiel mit der Befehlszeile

```
\date{21. März 1999}
```

beliebig setzen. Die alphanumerische Schreibung ist gefällig, aber häufige Datumsangaben im Brieftext können numerisch kürzer dargestellt werden. Für Datumsangaben in numerischer Form empfiehlt DIN 5008 die Schreibweise 1999-03-21, aber auch 21.03.1999 ist noch erlaubt. Dabei werden Monat und Tag zweistellig geschrieben, das Jahr vierstellig, und die Zeichenkette enthält keine Leerzeichen.

Der Brieftext muss mit dem *opening*-Befehl beginnen, welcher die Anrede setzt. Briefe an Ämter, Firmen und Autoritäten sollte man formal höflich beginnen:

```
\opening{Sehr geehrte Damen und Herren,}
```

Wird ein Mensch mit akademischen Graden oder anderen Ehrentiteln im Brief angesprochen, stellt man den höchstrangigen Titel und Grad dem Namen voran. Im Zweifelsfall wird aufgerundet! Private Briefe kann man meistens mit *Lieber ...* oder *Liebe ...* beginnen. Altmodische Anrede- und Grußformeln wirken im Schriftverkehr nicht nachteilig und sind allemal besser als ein *Hallo!* wie in der Telefonwarteschleife.

Dann folgt der eigentliche Brieftext.

```
Ich wäre Ihnen sehr verbunden, wenn Sie mir den gegenwärtigen
Aufenthaltsort der Prinzessin Li Si aus Mandala mitteilen würden,
sofern er Ihnen bekannt sein sollte.
```

In einem Brief sollte man Bildelemente grundsätzlich nur verwenden, wenn sie a) einen wohldefinierten Zweck erfüllen, b) persönlich und originell erscheinen oder c) die übergeordnete Leitung es so will. Emoticons gehören in der Regel nicht dazu.

Die abschließende Grußformel wird mit dem *closing*-Befehl gesetzt. Meistens ist *Mit freundlichen Grüßen,* gut. Oft ist ein vorangestellter Dank noch besser.

```
\closing{Ich danke für Ihre Bemühungen und verbleibe\\
mit freundlichen Grüßen,}
```

Die Grußformel kann also auch aus mehreren (mit \\ getrennten) Zeilen bestehen. Anlagen können mit dem *encl*-Befehl aufgeführt werden.

```
\encl{Bild der Prinzessin, von Herrn Pi Plu}   % Anlagen
```

Geht der Brief an mehrere Empfänger, kann man sie im Verteiler nennen.

```
\cc{Die Wilde 13\\Kopie an Kaiser Pung Ging}   % Verteiler
```

Am Ende werden *letter*- und *document*-Umgebung geschlossen.

```
\end{letter}
\end{document}
```

Um eine einfache (deutsche oder englische) Rechnung zu schreiben, kann man zum Beispiel die Umgebung *invoice* des Pakets *invoice2* von Simon Dierl einfügen. Man sollte eine neue Version des Pakets (1.2 vom 15. 1. 2018) verwenden.

7.5 Literaturliste mit *biblatex* und *biber*

Die Erstellung einer Literaturliste (Literaturverzeichnis) kann mit dem Paket *biblatex* und dem externen Programm *biber* geschehen. Im Cheat Sheet von Clea F. Rees gibt es eine Übersicht. Zunächst braucht man eine Bibliographiedatei.

Bibliographiedatei

Eine Bibliographiedatei ist eine Textdatei, die als Literaturdatenbank dient. Ihr Name sollte die Dateiendung `bib` haben. Da es üblich ist, die Bibliographiedatei mit der Zeit zu erweitern, sollte ihr Name eindeutig die Version benennen. Ein möglicher Name ist `lit_`*JJMMTT*`.bib`, wobei *JJMMTT* das sechsstellige Datum (Jahr, Monat, Tag) der letzten Änderung darstellt.

Die Bibliographiedatei kann mit einem LaTeX-Editor oder einem einfachen Texteditor bearbeitet werden, bei Kleinweich-Betriebssystemen zum Beispiel mit *Notepad* oder *Edify*. Später wird mit dem LaTeX-Paket *biblatex* und dem externen Programm *biber* eine Literaturliste aus der Bibliographiedatei entnommen.

In der Bibliographiedatei gibt es für jede Literaturquelle einen *Eintrag*, der einen eindeutigen Namen hat (BibTeX-Key). Dieser Name kann aus dem Nachnamen des Erstautors, dem Veröffentlichungsjahr und einem Buchstaben bestehen, wobei Kleinbuchstaben verwendet werden. Der Buchstabe am Ende dient zur Unterscheidung mehrerer Arbeiten mit gleichem Erstautor und Jahr.

Jeder Eintrag beginnt mit @ und dem Dokumenttyp (Referenzart) der Literaturquelle. Es folgen `{`, der BibTeX-Key und Felder. Jedes Feld enthält ein Paar bestehend aus Schlüssel = Wert. Dokumenttyp und Schlüssel können mit Groß- oder Kleinbuchstaben geschrieben werden. Leerzeichen außerhalb der Felder sind unbedeutend und dürfen beliebig gesetzt werden. Der Eintrag wird mit `}` beendet. Das Format einer Bibliographiedatei ersieht man aus folgendem Beispiel.

```
@online{gitter2019a,
author = {Gitter, Alfred H.},
title = {Latex (opus imperfectum)},
howpublished = {Website},
url = {http://www.mt.eah-jena.de/doc/AGitt/ahg_latex.pdf},
date = {2019-01-06},
urldate = {2019-01-31},
}
@book{hook2013a,
author = {Hook, James and Crook, A.},
title = {Extinction of the crocodiles},
publisher = {Broken Jaw Press},
address = {Fredericton},
date = {1908},
}
@article{pan1906a,
author = {Pan, Peter and Bell, Tinker and Darling, Wendy},
title = {Faith, Trust and Pixie dust},
journaltitle = {Archives of Pure and Applied Magic},
volume = {7},
pages = {39--97},
date = {1906},
}
@thesis{verne2015a,
author = {Verne, Jules-Gabriel},
title = {Dunkle Energie aus Lunarzellen},
institution = {Utopia Universität},
location = {Berlin},
type = {Bachelorarbeit},
date = {2015-03-15},
}
```

Die Reihenfolge der Einträge ist beliebig, aber der Übersichtlichkeit halber ist es gut, sie nach dem BibTeX-Key alphabetisch zu ordnen. In obigem Beispiel wurden die Dokumenttypen `online` (Internetquelle), `book` (Buch), `article` (Zeitschriftenartikel) und `thesis` (akademische Abschlussarbeit) vorgestellt.

Für akademische Abschlussarbeiten gibt es die Dokumenttypen `mastersthesis` (Magister- oder Masterarbeit) und `phdthesis` (Doktorarbeit). Es ist aber meistens besser, den allgemeineren Dokumenttyp `thesis` zu verwenden und im Feld `type` den genauen Typ mit einer beliebigen Zeichenkette (Bachelorarbeit, Masterarbeit oder Dissertation) anzugeben.

Das Veröffentlichungsdatum steht im Feld `date` entweder als Jahreszahl oder als Datum im ISO-Format (Jahr-Monat-Tag). `date` löst die früher gebrauchte Feldbezeichnung `year` ab. Quellen im Internet haben den Dokumenttyp `online`. Das Feld `urldate` gibt das Abrufdatum an.

LaTeX Quellcode mit *biblatex*

In der Präambel des LaTeX Quellcodes werden mindestens die Pakete *biblatex* (Philipp Lehman, Philip Kime) und *csquotes* (Philipp Lehman, Joseph Wright) geladen. Das Paket *csquotes* wird (mit *babel* oder *polyglossia*) benötigt, um die Anführungszeichen in Literaturzitaten der Sprache (bei uns Deutsch) anzupassen. Es wird empfohlen auch das Paket *xpatch* zu laden. Mit dem Paket *hyperref*, das nach *biblatex* geladen werden soll, werden Zitate zu Verweisen (Links), was beim Lesen am Bildschirm praktisch ist.

Außerdem wird mit dem Befehl *addbibresource* die Bibliographiedatei angegeben (hier: `lit_190213.bib`). Sie sollte im gleichen Verzeichnis stehen wie die Datei mit dem LaTeX Quellcode. Andernfalls muss man den Pfad zur Bibliographiedatei angeben. Somit werden die Zeilen

```
\usepackage[sortlocale=auto,sorting=nyt,style=authoryear]{biblatex}
\addbibresource{lit_190213.bib}
\usepackage{csquotes,xpatch}
\usepackage{hyperref}
```

in der Präambel gebraucht, wobei als Stil *authoryear* festgelegt wurde. Damit werden in den Literaturverweisen Autoren und Publikationsjahr angegeben und die Literaturliste erhält keine Nummerierung. Ähnlich ist der Stil *authoryear-comp*. Hier werden gleichzeitig zitierte Literaturverweise desselben Autors zusammengefasst (kürzere Darstellung). Beim Stil *numeric-comp* enthält jeder Literaturverweis eine Zahl, welche sich aus der Nummerierung der Literaturliste ergibt.

Die Optionen `sortlocale=auto,sorting=nyt` bestimmen die Sortierung der Literaturliste. `sortlocale=auto` bewirkt eine Sortierung gemäß der Spracheinstellung des Pakets *babel* und `sorting=nyt` legt fest, dass zuerst nach dem Namen des Autors (`n`), dann nach Publikationsjahr (`y`) und schließlich nach dem Titel (`t`) sortiert wird. Entsprechend ist auch `nty` statt `nyt` möglich. Die Option `sorting=none` würde die Literaturliste dagegen in der Reihenfolge der Zitate ausgeben.

Die mögliche Option `backend=biber` wurde nicht verwendet, da `biber` der Standardwert von `backend` ist. Es gibt Pakete, die nicht zusammen mit *biblatex* geladen werden dürfen (zum Beispiel *cite* und *titlesec*).

In der *document*-Umgebung können Literaturverweise mit dem Befehl *cite* eingefügt werden, der als Pflichtargument einen BibTeX-Key oder eine Liste von BibTeX-Keys (mit Komma als Trennzeichen) hat.

```
\cite{hook2013a,pan1906a}
```

Damit wird beim Stil *numeric-comp* jedem BibTeX-Key die zugehörige Nummer der Literaturliste zugeordnet und in eckige Klammern gesetzt. Beim Stil *authoryear* erscheinen Autor und Publikationsjahr im Literaturverweis ohne Klammern.

Beim Stil *authoryear* möchte man den Literaturverweis jedoch oft in runden Klammern ausdrucken. Dies ist mit dem Befehl *parencite* möglich.

```
\parencite{verne2015a}
```

Beim Stil *numeric-comp*, aber nicht bei *authoryear* gibt es den Befehl *supercite*, der die Nummer der Literaturliste hochgestellt ohne Klammern schreibt.

```
\supercite{gitter2019a}
```

Aus den im Quelltext für Literaturverweise verwendeten Einträgen der Bibliographiedatei (siehe oben) wird in der *document*-Umgebung mit dem Befehl

```
\printbibliography
```

die Literaturliste bei der Kompilerung erzeugt (siehe unten). Sie wird an der Stelle des Befehls *\printbibliography* eingefügt.

Hier ein kurzes Beispiel für ein LaTeX-Dokument mit Literaturliste:

```
% Präambel
\documentclass[a4paper,12pt,parskip]{scrartcl}
\usepackage[ngerman]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[sortlocale=auto,sorting=nyt,style=numeric-comp]{biblatex}
    \addbibresource{lit_190213.bib}
\usepackage{csquotes,xpatch}
\usepackage{hyperref}
\pagestyle{empty}
% document-Umgebung
\begin{document}
Piraten haben es nicht leicht \cite{hook2013a}.
\printbibliography
\end{document}
```

Kompilierung mit *biber*

Die Kompilierung des LaTeX-Quellcodes erfordert das externe Programm *biber*. Sie muss in mehreren Schritten erfolgen und dabei werden verschiedene Hilfsdateien erzeugt (unter anderem ein *biber control file* mit der Endung `bcf`). Ohne LaTeX-Editor, also bei Kleinweich-Betriebssystemen in der „Eingabeaufforderung“ oder der „Konsole“, beziehungsweise bei Linux im „Terminal“, kann man die Befehlsfolge

```
pdflatex datei.tex
biber datei.bcf
pdflatex datei.tex
pdflatex datei.tex
```

eingeben, wobei `datei.tex` die Datei mit dem LaTeX Quellcode ist oder, wenn die Datei nicht im aktuellen Arbeitsverzeichnis steht, der Pfad zur Datei.

Die Kompilierung im LaTeX-Editor erfolgt ebenfalls in mehreren Schritten. Voraussetzung ist eine Konfiguration des LaTeX-Editors. Nach einer Kompilierung mit PDFLatex ruft man *biber* auf. Dann kompiliert man noch zwei Mal mit PDFLatex.

Der Aufruf von *biber* erfolgt bei *TeXworks*, indem man im Auswahlfeld des Hauptmenüs „Biber“ (statt „pdfLaTeX“) wählt und mit `Strg-T` startet. Für „pdfLaTeX“ muss das Auswahlfeld entsprechend zurückgesetzt werden. Bei *Texmaker* und *TeXstudio* wird *biber* mit der Taste `F11` aufgerufen.