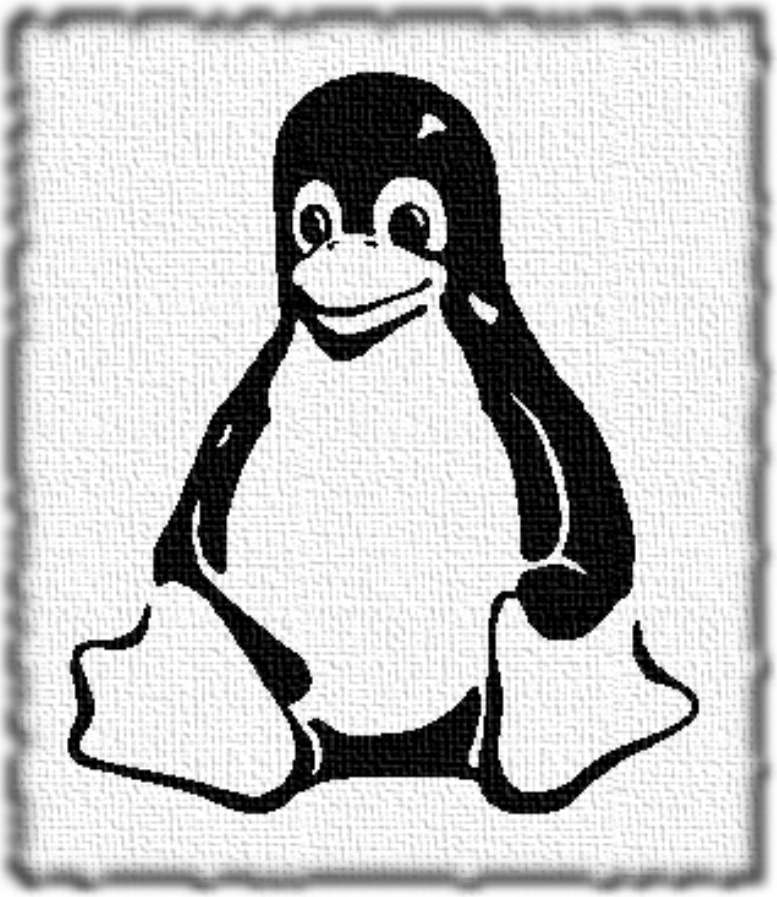
Fachschaftsrat Physik

Kurzanleitung zum Computer Crash Course



Jan N. Grieb Christian Holme Jan F. Engels
Steffen Klemer Martin Lüttich Malte Vaßholz

17. September 2012

Fachschaftsrat Physik
Friedrich-Hund-Platz 1
37077 Göttingen
fsr@fsr.physik.uni-goettingen.de
Tel. 05 51 – 39 39 38

Inhaltsverzeichnis

1 Vorwort

Dieses kleine Heftchen ist eine Gedankenstütze für die Teilnehmer des *Computer Crash Courses*. Der Kurs richtet sich vor allem an die Erstsemesterstudenten der Physik und soll einige wichtige Grundlagen in Bezug auf Linux, den Umgang mit einigen Basiswerkzeugen, das Schreiben von physikalischen Texten mit Latex sowie die Verwendung von Gnuplot vermitteln.

In dieser n-ten Überarbeitung wurde das neue Layout aus dem letzten Jahr behutsam weiterentwickelt - es sollte nun *noch* übersichtlicher sein. Auch haben wir viele Texte weiter entschlackt und natürlich wieder Fehler ausgemerzt. Hinzugekommen sind viele Hinweise auf die Verbindung der vorgestellten Grafikprogramme mit Latex und zahlreiche erklärende Worte.

Für die Freaks *g* unter Euch gibt es seit dem Jahr 2008 ein Extraheftchen über die Editoren `Emacs` und `Vim` sowie `Shell-Skripte` und `reguläre Ausdrücke`.

Solltet ihr Fehler entdecken, wartet bitte keine Sekunde und meldet sie im Wiki der Veranstaltung im Stud.IP oder via Mail an fsr@fsr.physik.uni-goettingen.de.

An dieser Stelle möchten wir allen danken, die bei der Verwirklichung dieses Skriptes geholfen haben. Vor allem geht unser Dank an Kai Bröking, Nils Kanning, Andrea Knue, Christian Hettlage, Markus Hundertmark, Kai Nörthemann, Marius Priebe, Xavier Barnabé-Thériault, Jakob Walowski und Marvin Walter die an den vorherigen Versionen des Skriptes mitgewirkt haben, auf deren Grundlage dieses Skript entstanden ist.

Außerdem danken wir Katharina Witthuhn ganz besonders für ihr alljährliches unermüdliches Korrekturlesen und die vielen hilfreichen Tipps. Nicht unbenannt dürfen auch die anderen vier fleißigen Lektoren Christoph Kolodziejski, Maren Mohler und David Swoboda bleiben – derart fleißige Fehlerfinder findet man selten.

2 Einige Worte zu Linux

Euch wird schon aufgefallen sein, dass unsere Rechner hier in der Physik etwas anders ticken als die Kisten, die ihr von zu Hause kennt oder in der Schule bedient habt. Dies liegt daran, dass wir ein anderes *Betriebssystem* – `Linux` – verwenden. Warum das so ist und welche Vor- und Nachteile dies für euch haben könnte, wollen wir in diesem Kapitel betrachten.

2.1 Was ist Linux?

In erster Linie ist Linux ein Betriebssystem[1], genau wie Windows oder MacOS es sind. Doch im Unterschied zu diesen ist es frei verfügbar und jeder kann (so er es denn *kann*) daran herumwerkeln, es anpassen und verbessern. Spezieller bezeichnet Linux aber nur die dünne Schicht, die zwischen dem Computer und der Software, die auf ihm laufen soll, vermittelt. Darauf aufbauend gibt es unendlich viele kleinere und größere Programme, die einem die Arbeit mit dem Rechner erleichtern sollen. Die meisten dieser Werkzeuge sind ebenfalls frei verfügbar, was uns gleich zu einer wichtigen Unterscheidung führt:

Software kann auf zwei Arten entwickelt werden, die öffentliche und die geschlossene. Letztere wird auch als proprietär bezeichnet und bedeutet, dass man beim Kauf eine Lizenz für die Nutzung erhält und ab da dem Wohlwollen des Herstellers ausgeliefert ist, was Fehlerbeseitigung und neue Funktionen betrifft. Typische Beispiele hierfür sind Windows, MS Office oder auch Photoshop, Acrobat Reader, Realplayer usw.

Die zweite Variante ist ein radikal anderes Modell. Es wird meist als Open-Source oder Free Software bezeichnet. Dabei bezieht sich das „Free“ auf die Freiheit, die man erhält und keineswegs auf den Preis. Entgegen weitläufiger Meinung, muss diese Software nicht kostenlos sein. In jedem Fall erhält der Nutzer aber den Quelltext zum Programm dazu. Einige offene Lizenzen, wie die älteste dieser Art, die **G**eneral **P**ublic **L**icence (GPL), *verpflichten* gar zur Weitergabe des Quelltextes. Beispiele für diese Methode sind Firefox, OpenOffice, Gimp und Teile von MacOS.

Nun kann dadurch aber jedermann die Software nach seinem Geschmack anpassen und verbessern, Fehler finden und beseitigen oder Teile des Quelltextes in eigenen Programmen einsetzen. Einziger Haken an der Sache ist, dass das eigene Programm dann ebenfalls unter der GPL stehen muss (also auch dieser Quelltext zur freien Verfügung stehen muss[2]). Die Freie-Software-Bewegung geht auf Richard Stallman zurück.

Linux wird nach diesem zweiten Modell entwickelt. 1991 von Linus Torvalds ins Leben gerufen, hat Linux seine Wurzeln im universitären Bereich, in dem damals sehr häufig Großrechner eingesetzt wurden. Auf diesen lief in den meisten Fällen ein Betriebssystem aus der *Unix*-Familie. Dies ist eine Gruppe sich stark ähnelnder Betriebssysteme. Das erste Unix wurde bereits 1969 am MIT entwickelt.

Torvalds veröffentlichte 1994 die Linux-Version 1.0. Schnell wurden die Stärken von Linux und offener Software erkannt und im Serverbereich eingesetzt. Große Serverhersteller wie IBM und Hewlett-Packard haben um 2000 angefangen, ihre Rechner ausschließlich mit Linux zu betreiben. Heutzutage ist Linux die am weitesten verbreitete Serverplattform weltweit.

[1] Ein Betriebssystem ist das grundlegende Steuerprogramm des Computers, das alles überwacht, die Benutzerschnittstelle bereitstellt usw.

[2] Es gibt auch andere Open-Source Lizenzen, die diese virulente Eigenschaft nicht haben.

Aufgrund der Offenheit gibt es nun aber leider nicht *ein* Linux, sondern verschiedene, so genannte *Distributionen*. Am bekanntesten dürften die deutsche Version von Suse (neuerdings Novell), das amerikanische RedHat, Mandriva (hier in der Physik verwendet) und das von tausenden Freiwilliger zusammengestellte Debian sein. In jüngerer Zeit stieg *Ubuntu* raketenhaft auf und überholte im privaten Bereich alle anderen.

2.2 Linux vs. Windows: Ein kleiner Vergleich

Im Gegensatz zum proprietären Betriebssystem Windows, wird Linux von verschiedenen Anbietern kostenlos zur Verfügung gestellt. Viele Distributionen wie Suse und Mandriva können aber auch gekauft werden; was man dabei bezahlt, sind jedoch die von den Firmen zusammengesuchten und in ein bequemes Installationsmenü zusammengefassten Softwarepakete und die oft umfangreiche Dokumentation. Gewöhnlich darf diese Zusammenstellung aber unentgeltlich weitergereicht (kopiert) werden.

Es ist schwierig Windows 7 abzuspecken, damit es auf betagteren Rechnern läuft. Man müsste auf eine wesentlich ältere Version von Windows zurückgreifen. Das Problem dabei ist, dass man damit die Kompatibilität zu den meisten aktuellen Programmen verliert. Ein Linuxsystem hingegen kann man so anpassen, dass es zwar nur minimale Ansprüche an die Hardware stellt, aber trotzdem aktuelle Software unterstützt. So kann man auf langsameren Computern trotzdem aktuelle Programme laufen lassen und so alten Rechnern zu neuem Leben verhelfen.

Linux hat viele Gesichter. Während Windows im Wesentlichen eine einzige grafische Oberfläche zur Verfügung stellt, bietet Linux unzählige, mehr oder weniger aufwendige grafische Oberflächen, die auf dem eigentlichen Kern aufsetzen.

Ein Nachteil bei Linux ist momentan noch, dass nicht alle Hardwarehersteller Linuxtreiber für ihre Geräte anbieten. Diese müssen durch die Linuxentwickler zum größten Teil selbst geschrieben werden, und so dauert es meistens einige Zeit, bis man neue Hardware unter Linux einsetzen kann. Es zeichnet sich in letzter Zeit aber ein positiver Trend ab. So funktioniert neue Hardware von Intel, Nvidia und HP teilweise schon 3-4 Monate vor dem Verkaufsstart und die Treiber von z.B. ATI und Intel sind ihrerseits sogar Open-Source.

Der Einstieg in Linux gestaltet sich für die Meisten etwas holprig. Das mag daran liegen, dass man die meisten Programme bis ins letzte Detail konfigurieren kann, aber auch daran, dass einige Optionen *per Hand* in Textdateien vorgenommen werden müssen – es gibt (noch) nicht für alles grafische Werkzeuge. In Studien wurde aber herausgefunden, dass Personen, die noch nie mit einem Computer gearbeitet haben, mit Linux[3] schneller zurechtkommen als mit Windows. Darum dürfte die größte Hürde sein, dass es einfach *anders* ist. Wer sich einarbeitet wird mit einer sehr flotten Bedienbarkeit belohnt.

2.3 Wie und Wo bekomme ich Linux?

Da es so viele verschiedene Möglichkeiten gibt, an Linux heranzukommen, weiß man als Neuling nicht, wo man anfangen sollte. Also wollen wir an dieser Stelle einige davon vorstellen.

Relativ neu ist Ubuntu Linux, dass auf Debian (siehe unten) basiert und sich das Ziel gesetzt hat, Linux für jedermann kostenlos und im Quelltext verfügbar zu machen. Hierbei bezieht sich *jedermann* auf alle Menschen weltweit, vor allem auch in den Entwicklungsländern. Ubuntu wurde vom

[3] speziell der grafischen Oberfläche Gnome

südafrikanischen Milliardär und zweiten Weltraumtouristen Mark Shuttleworth gegründet und erfreut sich zunehmender Beliebtheit. Es ist genauso einfach zu installieren wie die großen anderen Distributionen, aber **komplett** kostenlos im Internet auf `http://www.ubuntu.com` verfügbar.

Dann sind die oben bereits erwähnten Distributionen Novell (Suse) und RedHat zu nennen. Sie sind in Deutschland am weitesten verbreitet und stellen große Pakete mit Handbüchern und Support bereit. Die Softwarepakete befinden sich auf sieben CDs oder zwei DVDs; es ist also schon alles dabei, was man braucht. Dazu gehören dann verschiedenste Programme, angefangen bei Büroaufgaben, über MP3-Player und Serversoftware bis hin zu Spielen und wissenschaftlichen Anwendungen. Alles oft in mindestens 2 Varianten. Diese Distributionen bemühen sich, stets die aktuellsten Hardwaretreiber zu beinhalten. Sie sind für Anfänger besonders empfehlenswert, da man sie in den meisten Fällen ohne Kenntnisse einfach installieren kann und sie so eingerichtet sind, dass sie automatisch neben Windows als zweites Betriebssystem laufen. Man kann diese für 30 bis 90 € im Software- oder Buchhandel beziehen, oder alternativ unter den Codenamen *Fedora* (RedHat) bzw. *OpenSuse* kostenlos runterladen.

Als letztes wollen wir noch *Debian* erwähnen. Debian ist zwar etwas für Fortgeschrittene, aber bei Administratoren aufgrund seiner sehr guten Aktualisierbarkeit recht beliebt. Debian hat 10 Jahre nach den anderen nun endlich ein aufwendig gestaltetes Drei-Klick- Installationsprogramm, das einem die ganze Einstellungsarbeit dann doch nicht abnimmt. Die Idee dahinter ist eher, das System *einmal* so zu konfigurieren, wie man es sich vorstellt und es dann für lange Zeit vergessen zu können. Dies erfordert natürlich tiefe Kenntnis der Linux-Innereien. Ferner versuchen die Anbieter von Debian nur stabil laufende Software anzubieten, was dazu führt, dass eher ältere, aber ausgereifte Versionen im Paket enthalten sind. Es ist nahezu unmöglich, ein Debian-System zum Absturz zu bringen. Die narrensichere Aktualisierbarkeit wurde ja bereits erwähnt – Es wird von Debian-Servern berichtet, die seit 1998 ohne Pause durchlaufen und trotzdem aktuelle Software einsetzen. Ubuntu hat diese Funktion übrigens geerbt.

Ein etwas anderes Konzept verfolgt Knoppix. Es ist ein sogenanntes Live-CD-Betriebssystem. Das heißt, man legt die CD in das Laufwerk, startet den Rechner neu und Knoppix läuft, ganz ohne Installation. Auf diese Weise kann man Linux einfach einmal ausprobieren, ohne Gefahr zu laufen, das bestehende System zu zerstören. Änderungen des Hintergrundbildes oder der Interneteinwahlnummer sind so natürlich nicht ohne weiteres abzuspeichern, wo sollten diese denn auch auf der nur-lesbaren CD hin? Inzwischen gibt es aber Mittel und Wege, wie beispielsweise die Auslagerung auf einen USB-Stick oder in eine kleine Datei auf der Windowsfestplatte. Die Idee hinter ersterem ist, dass man so jederzeit seinen Arbeitsplatz mit sich herumtragen kann. Die neueste Mode (2GB-Sticks sei Dank) ist, die Einstellungen *und* Knoppix auf dem Stick unterzubringen. Solche LIVE-Linuxe gibt es mittlerweile wie Sand am Meer. Auch Ubuntu, Suse und Co bieten sie an. Eine Übersicht bringt `http://www.livecdlist.com/`. Hier finden sich auch spezialisierte Varianten für die Systemrettung, Virensäuberung, für Onlinebanking, Serveraufgaben...

2.4 Wie arbeitet man mit Linux?

Unter Linux können mehrere Personen auf einmal an einem Computer arbeiten. Technisch sieht dies beispielsweise so aus, dass einer am Computer selbst sitzt, ein anderer loggt sich über das Internet von einem anderen Computer aus ein, und ein dritter, der vorher da war, lässt den Computer einfach im Hintergrund etwas ausrechnen und ruft die Ergebnisse später ab.

Klassisch mit Linux zu arbeiten, heißt mit der Kommandozeile, also textbasiert[4]. Man teilt dem Computer über Textbefehle mit, was er zu tun hat. So startet man Programme oder schaut nach, was

[4] Wer erinnert sich noch an DOS? – Das gleiche, nur komfortabel.

auf der Festplatte gespeichert ist und kann sogar im Web surfen, E-Mails lesen, Musik bearbeiten, Texte automatisch verändern... Inzwischen gibt es aber auch weit entwickelte grafische Arbeitsoberflächen, die Linux aussehen lassen wie Windows, MacOS oder auch viel intuitiver.

2.5 Arbeiten mit der Kommandozeile

Zuerst kommen wir zur „klassischen“ Methode, mit Linux zu arbeiten. Wer sich traut, drücke nun doch einmal auf Strg + Alt + F1. Es sollte die grafische Oberfläche verschwinden und ein schwarzer Bildschirm mit etwas Text erscheinen. Unter Linux stehen je nach Distribution und Einstellung um die sechs solcher Konsolen (oder „Terminals“) zur Verfügung. Es sind im Grunde sechs Arbeitsflächen, auf denen entsprechend viele verschiedene Personen arbeiten könnten. Es ist bei nur einem Bildschirm und einer Tastatur pro Computer jedoch etwas schwierig realisierbar. Praktisch ist dies eher um mehrere Dinge gleichzeitig zu machen.

Man kann z.B. mit der Tastenkombination Alt + F1 (oder wie eben mit Strg + Alt + F1 von der grafischen Oberfläche) auf die erste Konsole umschalten. Genauso ist es mit den anderen, wobei sich die Nummer der F-Taste ändert. Die Konsolen sind meistens von `tty1-tty8` durchnummeriert. Die Nummer der Konsole entspricht dann der F-Tastennummer.

Auf Konsole 7 läuft gewöhnlich der *X-Server* – die grafische Oberfläche. Linux funktioniert also komplett ohne grafisches System. Dies hat den Vorteil, dass ein Absturz des X-Servers nicht das ganze System mitreißt.

Auch unter der grafischen Oberfläche kann man textbasiert arbeiten. Dazu gibt es kleine Programme, die Namen wie X-Term oder Terminal haben[5]. Wenn es anfangs auch etwas umständlich erscheint, das Arbeiten mit der Kommandozeile lässt sich, wie wir sehen werden, sehr effizient gestalten. Dieses *Ding*, das die Befehle, die eingegeben werden, auswertet, wird meist `bash` (bourne again shell) genannt.

Grundlagen

Damit die Arbeit flott von der Hand geht, sollte man einige wichtige Tastenkombinationen beherrschen:

Mit den Tasten ↑ und ↓ kann man vorher benutzte Befehle aufrufen, ohne sie wieder eintippen zu müssen. Als ersten Test kann man nun einfach einmal `date` eintippen und betrachten, was passiert.

Die meisten Befehle stehen nicht einfach für sich selbst, sondern erwarten weitere Informationen. `mv`, der Befehl zum Verschieben von Dateien, muss natürlich wissen, was er wohin wuchten soll. Oder `gzip`, ein Packprogramm, möchte wissen, wie stark es komprimieren soll. Solche Informationen werden einfach hinter den Befehl geschrieben. So kopiert `cp datei1 datei2` die datei1 in die datei2 [6]. Oder `ls -a` zeigt alle (auch versteckte) Dateien an.

Tab-Ergänzung: ergänzt den Befehl bzw. den Dateinamen. Ist die eingegebene Buchstabenfolge eindeutig, so wird die Eingabe automatisch vervollständigt, z.B. ergibt `gnup` + TAB demnach `gnuplot`. Bei Mehrdeutigkeit werden die Möglichkeiten durch erneutes Drücken von TAB angezeigt. Das spart jede Menge Tipparbeit und beschleunigt so die Arbeit. Ausprobieren kann man dies

[5] Häufig finden sich auch die Bezeichnungen Konsole, Shell oder gar Bash, die aber eigentlich etwas spezieller sind, hier aber auf das gleiche hinauslaufen.

[6] Achtung: Linux unterscheidet übrigens Groß- und Kleinschreibung!

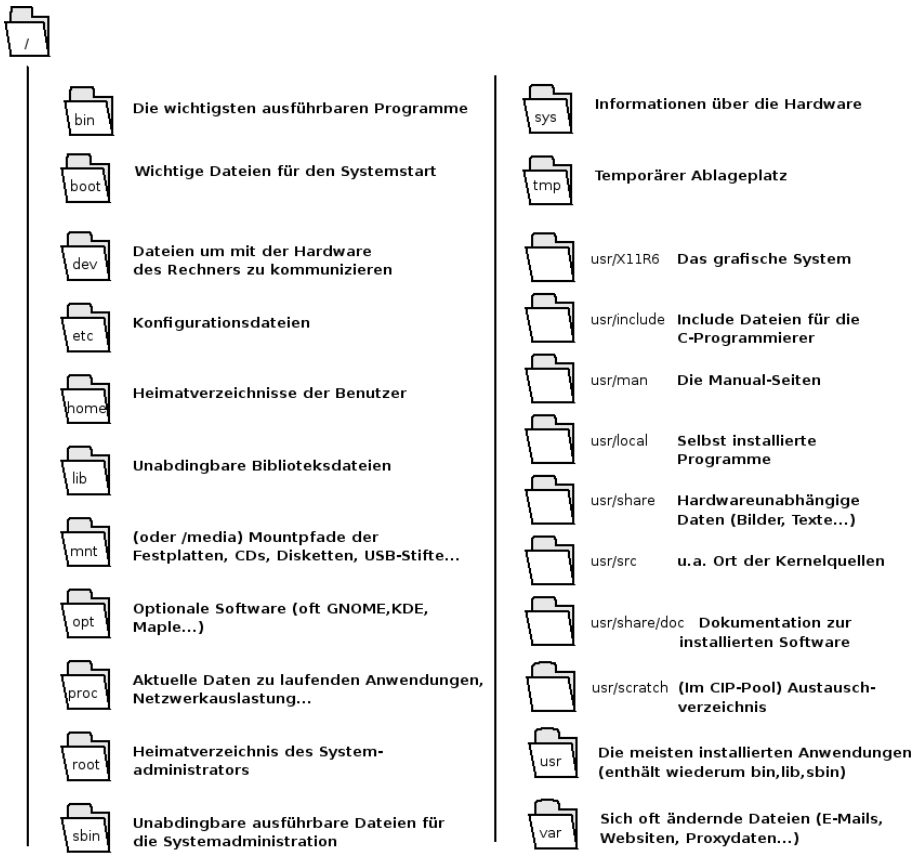


Abbildung 2.1: Struktur eines Unix Dateisystems

z.B. durch ein flottes `da`+ TAB, was nun ein `date` hervorzaubern sollte. Was dagegen passiert, wenn man nur `d`+ TAB (evt. 2x) drückt, sollte jeder selber ausprobieren...

In den allermeisten Fällen liefert ein `befehl --help` eine kurze Übersicht über seine Parameter. Zu fast jedem Programm existiert auch eine Hilfe-Datei, die man mit `man befehl` oder `info Befehl` aufrufen kann. Die Hilfe beendet man mit einem beherzten Druck auf q. Diese beiden Hilfestellen sollte man *unbedingt* im Kopf behalten. Denn im Gegensatz zu Anhängern anderer Betriebssysteme gelten Hilfe-lesende Linuxer keinesfalls als uncool oder *Schattenparker*. Anders kann man viele Dinge gar nicht herausfinden. Die Anzahl an möglichen Buchstabenkombinationen ist zwar nur abzählbar, aber immerhin doch *un*endlich...

Zur Verzeichnisstruktur von Linux

Im Gegensatz zu Windows gibt es unter Linux keine Laufwerksbuchstaben. Alle Festplatten, CD-Laufwerke, USB- Sticks usw. hängen in einem großen Verzeichnis*baum*. Das oberste Verzeichnis lautet **/**[7]. Darunter liegen dann verschiedene andere. Die genaue Struktur ist von der Distribution abhängig. Es gibt aber gewisse Konventionen, welche Abbildung 2.1 zeigt. Die für Anwender wichtigsten Verzeichnisse sind **/home**, in dem jeder Benutzer seinen eigenen Dateiablageplatz hat und **/media**, in dem für jedes CD-Laufwerk, jeden Stick und jedes Diskettenlaufwerk ein Ordner existiert.

[7] Unter Linux ist der / (*Slash*) und nicht der \ (*Backslash*) das Trennzeichen zwischen Ordnern.

Jede Datei[8] besitzt einen eindeutigen Namen, ihren Inhalt und ein paar Eigenschaften, die sogenannten *Rechte*. Jeder Datei kann dabei ein Eigentümer und eine Gruppe zugeordnet werden. Nun kann, getrennt nach *Besitzer*, *Gruppe* und *Rest der Welt* festgelegt werden, ob sie durch diesen **Lesbar**, **Beschreibbar** und **Ausführbar** ist. Damit besitzt sie also insgesamt 9 Eigenschaften.

Die Eigenschaft *versteckt*, die unter Windows recht beliebt ist, gibt es so nicht. Unter Linux gelten Dateien als versteckt, wenn ihr Name mit einem `.` beginnt. Solche Dateien werden gewöhnlich nicht angezeigt. So tummeln sich im eigenen Home-Verzeichnis hunderte Konfigurationsdateien für alle möglichen Programme, wie man leicht durch ein `ls -a` (mit a wie *alle*) sehen kann.

Soll sich ein Befehl auf mehrere Dateien beziehen, die den gleichen Anfang haben, so kann man mit der Sterntaste `*` alle Endungen zu dem gewünschten Anfang ausdrücken. `oster*` steht also für oster*hase*, oster*ei* oder auch oster*n*.

Beispiel: Wir haben ein Verzeichnis mit den Dateien

```
index.html inhalt.html was_anderes.html
```

und wollen uns direkt die ersten beiden ausgeben lassen. Dann geht dies mit

`cat index.html inhalt.html` (man bedenke die Tab-Vervollständigung, um sich Tipparbeit zu sparen...) sowie mit

```
cat in*
```

Das ist typisch für Linux - alles lässt sich auf mindestens 10 verschiedene Arten erreichen. Es geht nur darum, die für sich Effektivste zu finden. Nach dieser *kurzen* Einleitung folgen nun die wichtigsten Befehle:

Dateibefehle

> `pwd` (print working directory)
> Zeigt das Verzeichnis an, in dem man sich gerade befindet.

> `ls` (list)
> Zeigt den Inhalt des aktuellen Verzeichnisses an. Mit der Option `-a` (all) werden auch versteckte Dateien angezeigt, d.h. Dateien, deren Name mit einem Punkt anfängt, z.B. `.Dateiname`. Die Option `-l` (long) zeigt mehr Informationen zu den einzelnen Dateien an. Das Format ist das folgende:
> `Art Rechte Anz.U.v. Besitzer Gruppe Größe Mod.datum Dat.name`
> wobei bei *Art* ein `-` für eine Datei, `d` für ein Verzeichnis, `l` für einen Link steht. Die *Rechte* werden im Format `rwxrwxrwx` angegeben (**r**ead, **w**rite, e**x**ecute). Das erste Tripel gibt die Rechte des Besitzers an, gefolgt von den Rechten der Gruppenangehörigen und zum Schluss aller anderen. *Anz.U.v.* ist die Anzahl der Unterverzeichnisse. *Mod.datum* ist das Datum der letzten Änderung. Besitzer und Gruppe geben den Usernamen des Besitzers und die Gruppe an, der die Datei angehört. Die Größe wird defaultmäßig in Byte angegeben. Und *Dat.name* schließlich ist der Name der Datei bzw. des Verzeichnisses oder Links. Die Größe in sinnvollen Einheiten (Kibibyte, Mebibyte) gibt `-h` (human readable) an. Schließlich gibt es noch die Option `-t`, welche die Ausgabe nach dem Zeitpunkt der letzten Änderung sortiert.

> `cp Quelldatei Zieldatei` (copy)
> Erstellt eine Kopie der Quelldatei mit dem Namen `Zieldatei`. Handelt es sich bei

[8] Ordner sind auch nur spezielle Dateien – sie enthalten selbst wieder welche.

`Zieldatei` um ein Verzeichnis, wird die Datei mit dem alten Namen in dieses Verzeichnis kopiert.

> `mv Quelldatei Zieldatei` (move)
Verschiebt `Quelldatei` nach `Zieldatei`. Im Grunde ist es ein Umbenennen von Dateien. Dasselbe ist auch mit Verzeichnissen möglich. Sind `Quelldatei` eine Datei und `Zieldatei` ein Verzeichnis, wird `Quelldatei` in das Verzeichnis `Zieldatei` verschoben.

> `rm Datei1 Datei2 ...` (remove)
Löscht die angegebenen Dateien `Datei1`, etc. Mit der Option `-r` kann man diesen Befehl auf ein Verzeichnis anwenden. So werden alle Unterverzeichnisse und die darin enthaltenen Dateien gelöscht *(rekursiv)*. Wenn man nicht genau weiß, was man tut, sollte man zum Löschen eines Verzeichnisses den Befehl `rmdir` verwenden. Die Option `-f` (force) sorgt dafür, dass nicht nachgefragt wird, selbst wenn die zu löschende Datei schreibgeschützt ist. Soll das Löschen einer Datei bestätigt werden, verwendet man die Option `-i` (interaktiv). Vorsicht es gibt keinen Papierkorb! Was man so löscht, ist weg!

> `cd Verzeichnisname` (change directory)
Wechselt in das angegebene Unterverzeichnis. (`cd ..` wechselt in das übergeordnete Verzeichnis; `cd .` in das aktuelle[9] und `cd` ohne Argument wechselt in das eigene Home-Verzeichnis).

> `mkdir Verzeichnisname` (make directory)
Erzeugt ein neues Verzeichnis mit dem angegebenen Namen.

> `rmdir Verzeichnisname` (remove directory)
Löscht ein leeres Verzeichnis. Dies funktioniert nicht, wenn sich noch Dateien in diesem Verzeichnis befinden; dies gilt insbesondere auch dann, wenn nur noch versteckte Dateien vorhanden sind. Diese fangen immer mit einem Punkt an, wie beispielsweise `.dateiname`. Sie werden mit dem Befehl `ls -a` angezeigt.

> `chmod GruppeZugriffZugriffstyp Datei` (change mode)
Ändert die Zugriffsrechte auf die Datei.
Für `Gruppe` kann man einsetzen: `u` für Benutzer, `g` für Benutzergruppe, `o` für alle anderen *(others)*. `a` steht für alle Gruppen zusammen *(all)*.
Bei `Zugriff` kann man mit `+` den Zugriff erlauben oder mit `-` sperren[10].
Für `Zugriffstyp` kann man `r` für Lesen, `w` für Schreiben und `x` für Ausführen einsetzen.
Beispielsweise kann man mittels `chmod u-w datei` die Datei `datei` schreibschützen.

Alternativ bietet sich die Eingabe mittels einem Zifferntripel an. Hierbei summiert man einfach `1` (Ausführen), `2` (Schreiben) und `4` (Lesen) auf und packt die Werte für Besitzer, Gruppe und Rest der Welt hintereinander. Dann bekommt eine Datei mit `chmod 762` also die Rechte `rwxrw-r--`.

> `du -h` (disk usage)
Zeigt die Größe des aktuellen Verzeichnisses und aller Unterverzeichnisse an. Wird die Liste zu lang, kann man mit `--max-depth=N` die Rekursionstiefe auf *N* Stufen einstellen.

> `ps` (process status)
Zeigt alle aktiven Prozesse mit Prozessnummer an. Im Wesentlichen sind dies die gerade laufenden Programme. Mit der Option `-a` werden alle eigenen Prozesse angezeigt. Ein `-x` liefert mehr Informationen und `-A` zeigt wirklich *alle* Prozesse an. Eine ähnliche Funktion hat auch

[9] Das ist etwas für die theoretischen Informatiker.
[10] Mit `=` lässt sich der Zugriff direkt setzen, statt ihn hinzuzufügen oder abzuziehen.

der Befehl `top`. Dieser zeigt laufend aktualisiert eine Tabelle aller Prozesse mit ihrer CPU-Auslastung, Speicherverbrauch u.ä. an.

> `kill Prozessnummer`
Lässt man sich mit `ps -x` oder `top` laufende Prozesse anzeigen, und merkt, dass man einige Prozesse nicht mehr braucht[11], so kann man sie mit `kill` gefolgt von den Prozessnummern beenden. Es ist besonders brauchbar, wenn Prozesse abstürzen. Wird ein Prozess durch ein einfaches `kill` nicht beendet, hilft die Option `-9` weiter. Oft einfacher anzuwenden ist `killall programmname`, das alle Prozesse mit diesem Namen beendet. Auch `killall` versteht `-9`.

> `xkill` (nur unter grafischer Oberfläche anwendbar)
Gibt man diesen Befehl in der Konsole ein, so verwandelt sich der Mauszeiger in einen Totenkopf. Klickt man nun ein Fenster an, so wird dieses und sein Ursprungsprozess geschlossen. Dies ist anwendbar, wenn Fenster abstürzen oder wenn man zu faul[12] ist mit `ps` die Prozessnummer herauszusuchen.

> `finger`
Zeigt an, wer momentan eingeloggt ist.

> `less Dateiname` (less is even more)[13]
Zeigt den Inhalt einer Datei an, ohne dass man einen Editor starten muss.

> `cat Dateiname` (concatenate)
Gibt den Inhalt einer Datei auf der Standardausgabe aus (keine „Oberfläche“ wie bei `less`). Außerdem können mit `cat` Dateien aneinandergehängt werden: `cat datei1 datei2 > result` schreibt datei1 und datei2 fein säuberlich hintereinander in `result`.

> `lpr Dateiname` (Line Printer)
Schickt die Datei `Dateiname` an den Drucker zum Ausdrucken. Von den Optionen sei nur `-P` für die Auswahl des Druckers genannt. Weitere Optionen sind im CIP-Pool auf den Druckern aufgeklebt. Da die Druckeroptionen mit Duplex u.ä. sehr umfangreich sind, bietet sich in fast allen Fällen `gtklp` als grafische Alternative mit sonst identischen Funktionen an.

> `lpq` (Line Printer Queue)
Zeigt die aktuellen Druckaufträge an. Falls man seinen doch nicht mehr gedruckt haben mag, kann man ihn mit `lprm AUFTRAGSNUMMER` löschen. Dies bietet sich auch an, wenn der Drucker mal wieder hängt und der oberste Auftrag (mit der kleinsten Nummer) der eigene ist.

> `tar czf datei.tar.gz Verzeichnisname` (Tape ARchive)
Um Speicher zu sparen oder Verzeichnisse voller Dateien einfacher verschicken zu können, sollte man diese komprimieren. Das alles erledigt dieser Befehl für uns; er fasst alle Dateien aus dem Verzeichnis `Verzeichnisname` in der Datei `datei.tar.gz` zusammen.
Mit `tar xzf datei.tar.gz` kann man die archivierten Dateien wieder entpacken.
Mit `tar tvzf datei.tar.gz` kann man einsehen, welche Dateien sich im Archiv befinden.
Die Optionen `czf, xzf` und `tvzf` bedeuten aufgeschlüsselt:

- `c` (*create*) legt ein neues Archiv an
- `x` (*extract*) entpackt ein Archiv
- `t` (*table*) zeigt den Inhalt eines Archives

[11] Oder sie tot sind
[12] Der Autor dieses Kapitels meint *beschäftigt*.
[13] Geht auf seinen Vorgänger `more` zurück.

- `f` (*file*) ermöglicht die Wahl eines Dateinamens für das Archiv
- `z` komprimiert/dekomprimiert das Archiv mit gzip (WinZip-tauglich)
- `j` komprimiert/dekomprimiert das Archiv mit bzip2 (viel langsamer, etwas kleinere Dateien, nicht mit WinZip zu öffnen)
- `v` (*verbose*) zeigt die Dateien an, die gepackt bzw. entpackt werden

> `find dir -name filename -type type-of-file`
In dieser Form kann `find` nach Dateien oder Verzeichnissen in dem Verzeichnis `dir` mit dem Namen `filename` suchen, welche vom Typ `type-of-file` sind. Beim Namen können Wildcards wie $\star$ benutzt werden (z.B. filename$\star$ liefert filename, filename1, filenameneu ...). Der Typ ist `d` für Verzeichnisse und `f` für Dateien.

> `grep Option Muster file` (GNU regular expression pattern (search))
Dieser Befehl sucht die Zeichenkette „Muster“ in einer Datei „file“. Wildcards sind nicht notwendig. Sollen auch die Dateien der Unterverzeichnisse (rekursiv) durchsucht werden, so ist die Option `-r` (für **r**ecursive) anzugeben. Mit `-n` werden zusätzlich noch die Zeilennummern hinter dem Dateinamen ausgegeben. Der Befehl ist ähnlich wie `find`, noch auf viele andere Arten nützlich, mehr dazu in der Manpage.

> `bc [-l]` (Basic Calculator)
Das Tool `bc` ist ein schlanker und praktischer Kommandozeilen-Taschenrechner. Er kennt die Grundrechenarten und mit der Option `-l` auch einige mathematische Funktionen. Im Normalfall arbeitet man im interaktiven Modus, den man durch die Eingabe von `quit` wieder verlässt. Will man dies umgehen, so lässt sich eine einzeilige Rechnung via `echo 5*7^3+1|bc` bewerkstelligen.

> `mc` (Midnight Commander)
Der mc ist ein Programm um leichter Dateien auf dem Rechner hin- und herzuschaufeln. Er ähnelt „*ein wenig*“ dem altehrwürdigen Norton Commander. Mit der Einstellung *lynx-artige Bewegung* ist er das ultimative Werkzeug für die schnelle Arbeit auf der Kommandozeile, die ihren Namen damit fast gar nicht mehr verdient.

> `mcedit`
Der passende Editor zum mc – Entweder man öffnet ihn mit F4 im Midnight Commander, oder halt mit `mcedit dateiname`. Er bedient sich, im Gegensatz zu den (wesentlich mächtigeren) Editoren `vim` und `emacs`, sehr intuitiv.

> `man Befehl` (Manpage)
Zum Schluss kommt ein Befehl, den man vielleicht als den mächtigsten und brauchbarsten bezeichnen kann. Er öffnet zu jedem `Befehl` eine Art Bedienungsanleitung, in der sich alle möglichen Optionen, Hinweise und Tipps zum `Befehl` finden lassen (z.B. `man ls`). Man kann darin mit / ein Suchwort eingeben und mit n zum nächsten Treffer manövrieren. Schließen lässt sich die Manpage mit q.

Weitere unverzichtbare Werkzeuge listet `info coreutils`. Hierunter verdienen vor allem `tr`, `sort`, `file` und `uniq` besondere Beachtung. Und auch die Tools `yes`, `nl` und `beep` sollen nicht unerwähnt bleiben.

2.5.1 Fortgeschritteneres Arbeiten mit der Kommandozeile

Pipes

Hierbei handelt es sich um die Möglichkeit, die Ausgabe des ersten Befehls als Eingabe des nächsten Befehls zu nutzen. Anzugeben sind die Befehle im Format:

> `befehl1 | befehl2`

Es wird nun das, was `befehl1` normalerweise am Bildschirm ausgibt, stattdessen als Eingabe für den zweiten Befehl benutzt. Wenn man z.B. weiß, dass die Ausgabe von `ls -alh` sehr lang ist, aber man sowieso nur an den Dateinamen `file1 ...file5` interessiert ist, würde es sich anbieten, folgendes als Befehl einzugeben:

> `ls -alh | grep file`

Umleitungen

In manchen Fällen möchte man die Ausgabe von Befehlen nicht auf dem Monitor haben, sondern lieber in einer Datei speichern. Dann greift man zur Ausgabeumleitung, die wie folgt funktioniert:

> `befehl > file`
> Die Ausgabe von `befehl` wird in die Datei `file` geschrieben. Existiert diese bereits, so wird sie überschrieben.

> `befehl >> file`
> Die Ausgabe wird auf die gleiche Weise umgeleitet, bei Existenz der Datei wird die Ausgabe allerdings an das Ende der Datei angehängt.

Auch die Eingabe für Befehle kann man aus Dateien lesen:

> `befehl < file`
> Holt die Eingabe für den Befehl, aus der Datei „file“.

Diese 4 kleinen Symbole machen die Linux-Kommandozeile zu einem mächtigen Werkzeug. Um beispielsweise mal eben die Rechte einer Datei in einem vollen Verzeichnis zu erfahren reicht ein `ls -la|grep datei`. Oder um 2 Dateien zusammenzufügen benutzt man `cat datei1 datei2 > zusammen`[14]. Auch können mit `grep` große (Messwert-)Dateien nach wichtigen Punkten durchsucht, mit `sort` sortiert oder nur die 2. und 7. Spalte extrahiert und in eine neue geschrieben werden. Der Kreativität sind keine Grenzen gesetzt. Eine Lektüre über Reguläre Ausdrücke (z.B.: in der Manpage von grep) lohnt sich hierbei sehr[15].

Kommandosubstitution

Es kann vorkommen, dass man die Ausgabe eines Befehls als Argument in einem weiteren Befehl benötigt. In diesem Fall lässt sich eine Kommandosubstitution anwenden:

> `befehl2 ` `befehl1` `` [16] Die Ausgabe von befehl1 ersetzt die Zeichenkette ‘befehl1‘ und dient somit als Argument von `befehl2`.

Anwendung findet dies häufig bei der Arbeit mit Dateien. `rm ` `find ./ -name *zack*` `` löscht z.B. alle Dateien mit *zack* im Namen.

[14] Das funktioniert sogar mit avi-Filmen.
[15] Eine kleine Einführung findet ihr außerdem im Zusatzheft.
[16] Das ` (Backtick) bekommt man mittels Shift+´ (neben dem ß). Es ist verschieden vom Hochkomma und vom Apostroph.

2.6 Arbeiten mit grafischer Oberfläche

Unter Linux gibt es, wie bereits erwähnt, eine Vielzahl von grafischen Oberflächen, wobei die grafische Oberfläche als ganzes meist X11 genannt wird. Man findet spartanische Oberflächen wie fvwm, welche eigentlich nur ein Menü besitzen und das Handhaben von Fenstern ermöglichen. Die persönlichen Vorlieben konfiguriert man hier meist noch *per Hand* in einer Textdatei. Zu den kompakten, aber schon einfacher zu benutzenden Oberflächen, gehört unter anderem WindowMaker. Dieser bringt eigene Programme und grafische Konfigurationseditoren mit. Zu guter Letzt gibt es noch Gnome, XFCE und KDE. Hierbei handelt es sich um Oberflächen, welche der normalen Windows-Oberfläche in nichts nachstehen. Sie lassen sich bequem bedienen und bringen sogar noch ein auf die Oberfläche zugeschnittenes Programmpaket mit. Für Anfänger eignen sich besonders die aufwendigen Oberflächen, da diese sehr einsteigerfreundlich sind. Ein großer Unterschied besteht darin, dass man mit KDE und Gnome viele administrative Aufgaben grafisch durch Anklicken erledigen kann.

Letztendlich ist die Wahl der Oberfläche komplette Geschmackssache und nur im Fall älterer Rechner[17] ist von KDE oder Gnome aufgrund der geringen Rechenleistung abzuraten. Es gibt durchaus hartgesottene Figuren – Linus Torvalds ist so eine –, die meinen, die grafische Oberfläche diene nur dem gleichzeitigen Öffnen unzähliger Terminals.

Ausgewählt werden kann die Oberfläche beim Einloggen im Menü *Sitzung* oder *Session* während man seinen Benutzernamen eingibt.

Dabei ist die Entscheidung für eine Oberfläche keineswegs endgültig. Programme der Einen laufen jeweils auch auf der Anderen und die Bedienung ist häufig ebenfalls recht ähnlich. Gemein ist allen, dass man (außer mit den Menüpunkten und den bekannten Tastenkürzeln) markierten Text mit der mittleren Maustaste einfügen kann.

Auch bieten fast alle Oberflächen die Möglichkeit, die Taste Alt zu betätigen und dann bei gedrückter Maustaste Fenster zu verschieben (bzw. sie bei mittlerer/ rechter zu verkleinern oder zu vergößern).

Außerdem gibt es immer die Möglichkeit mehrere *virtuelle* Desktops zu benutzen, zwischen denen man meist mit Strg + Alt + ← (oder →) und einem kleinen Umschalter in der Menüleiste wechseln kann. Dadurch bietet sich die Möglichkeit, einen Desktop zum surfen, einen zum Mailen, einen zum Arbeiten und einen zum Spielen zu benutzen. Vor allem arbeitswütigen Menschen gefällt dies oft auf Anhieb. Es gibt noch tausende weiterer nützlicher Tastenkürzel, die aber alle in den Hilfesystemen dokumentiert sind. Erwähnt seien noch

- Alt + Tab Umschalten zwischen Fenstern
- Alt + F2 Ausführen-Dialog
- Alt + F1 Das Programme-Menü (In Windowskreisen: *Startmenü*)
- Strg + Alt + ← Backspace Schießt die grafische Oberfläche ab (z.B. wenn sie sich doch mal aufhängt)

2.7 Arbeiten aus der Ferne

2.7.1 Linux – Linux

Für Linux gibt es (im Gegensatz zu Windows) eine einfache Möglichkeit Anwendungen auf einem anderen Rechner aus der Ferne zu benutzen. Das dazu nötige Zauberwort heißt `ssh` und steht für

[17] Da *alt* relativ ist: Systeme unter 256MB RAM

Secure SHell. Mit den richtigen Argumenten (`ssh username@hostname`) aufgerufen baut `ssh` eine verschlüsselte Verbindung zu einem Rechner mit dem Host-Namen `hostname` auf und versucht dich dort unter dem Benutzernamen `username` anzumelden. Wenn das klappt fragt `ssh` nach dem zu `username` gehörigen Passwort. Anschließend steht dir eine ganz normale Kommandozeile auf dem entfernten Rechner zur Verfügung.
Startet man `ssh` mit der Option `-X`[18], so wird die grafische Ausgabe eines entfernt gestarteten Programms umgeleitet.[19] Da dabei recht große Datenmenge übertragen werden, sollte man eine recht schnelle (Internet-)Verbindung haben, komfortabel wirds ab $\gtrsim$ 10 Mbit/s.[20]

Kochrezept für Fernzugriff auf den CIP-Pool

Als erstes sollte man wissen, dass der einzige aus dem Internet erreichbare Host des CIP-Pools auf den wunderbaren (und leicht zu merkenden) Namen `login.cip.physik.uni-goettingen.de` hört.
Möchte beispielsweise die Physikstudentin `hilde.gard` eine `ssh`-Verbindung inkl. Umleitung der grafischen Ausgabe aufbauen, so gibt Hildegard ein: `ssh -X hilde.gard@login.cip.physik.uni-goettingen.de`. Nach korrekter Passwordeingabe befindet sie sich auf dem Login-Server, auf dem aber nur wenige Programme installiert sind. Um nun auf einen der CIP-Rechner zu gelangen, gibt sie `ssh -X cNNN` ein, mit $13 \leq$`NNN`≤ 140; also beispielsweise `ssh -X c023`.

Dateiübertragung

Um Daten zwischen Rechnern hin und her zu kopieren, verwendet man am besten `scp dateiname username@login.cip.physik.uni-goettingen.de:.` (Punkt).
Hier bedeutet das `scp` secure copy. Mit diesem Beispiel würden wir die Datei `dateiname` von unserem Rechner in unser Homeverzeichnis im CIP-Pool kopieren. Das Zeichen `:` bedeutet, dass an dieser Stelle die Adresse endet. Der Punkt `.` deutet das Homeverzeichnis an. Alternativ kann man auch den ganzen Verzeichnisnamen ausschreiben. Ein Beispiel wäre also:
`scp dateiname username@...:verzeichnis/neuer_dateiname`
Man kann auch mehrere Dateien auf einmal kopieren, dazu muss man einfach nur die einzelnen Dateinamen hintereinander vor die Remote-Adresse schreiben. Vom CIP-Rechner nach Hause gelangen die Daten, wenn man einfach beide Optionen tauscht.

Falls das `scp` mal nicht funktioniert, könnt ihr auch `sftp` benutzen. Funktioniert ähnlich wie ssh, nur die `-X` Option ist nicht vorhanden. Habt ihr euch einmal auf dem Rechner in der Ferne angemeldet, holt ihr eure Daten von dort mit `get Datei` auf euren Rechner. Falls ihr Daten auf den entfernten Rechner kopieren wollt, müsst ihr diese mit `put Datei` dorthin „legen“.

Als einfachere Alternative findet man auch hier wieder den Midnight Commander, der nach einem Druck auf F9 das Menü hervorholt. Nun drücke man ↓ und aktiviere **Shell link...**. Im sich nun öffnenden Dialog tippt man einfach
`username@login.cip.physik.uni-goettingen.de` und schon ist man nach einem Enter bei der Passwortabfrage. Links ist nun der Rechner in der Physik, rechts der, an dem man gerade sitzt.

[18] Wirklich groß X, klein x bewirkt das Gegenteil.
[19] `ssh -X` wird gern genutzt um ein bekanntes Computer-Algebra-System von zu Hause zu nutzen.
[20] Wobei Steffen es auch über Modem geschafft haben soll....

Wem das Kopieren auf der Kommandozeile nicht so zusagt, der kann mit dem Programm `gftp` etwas bequemer Dateien über ssh kopieren. Dazu wählt man oben rechts `ssh2`, als Server das bekannte `login.cip.physik.uni-goettingen.de` und Benutzer/ Passwort sollten bekannt sein. Das Feld **Port** lässt man einfach frei.

2.7.2 Windows – Linux

Natürlich sollte man auch in der Lage sein, „remote“ zu arbeiten, wenn man keinen Zugang zu einem Linuxrechner hat. Für Windows gibt es ebenfalls einige Programme, mit denen man sich auf Linuxrechnern einloggen kann. Diese wären Putty, (entspricht `ssh`), Pscp (entspricht `scp`) und WinSCP, das beides in einem Programm verbindet. WinSCP kann von `http://winscp.net` heruntergeladen werden. Man erhält es dort ganz herkömmlich mit einem Installer, aber auch sehr praktisch als *Standalone Application*, die man einfach auf einem USB-Stick mit sich herumtragen kann, um jederzeit von jedem Windowsrechner an seine Daten zu gelangen.

Beim ersten Öffnen sollte man bei **Host name** `login.cip.physik.uni-goettingen.de`, bei **User name** entsprechend den eigenen Benutzernamen und bei **Password** selbiges eintragen und dies dann ein für alle mal mit `Save...` abspeichern. Unter **SSH** kann und sollte man die Datenkomprimierung mit `Enable compression` einschalten. Außerdem ist es möglich, je nach Geschmack bei **Preferences** das Aussehen anzupassen. Hier sollte jeder selbst mal probieren, was ihm / ihr am liebsten ist. Ein beherzter Klick auf `Login` in den **Stored sessions** loggt uns ein. Nun muss beim ersten Mal die Frage beantwortet werden, ob man dem Server der Physik traut, was man tun sollte, da sonst an dieser Stelle Schluss ist.

Schließlich sind links die Dateien des eigenen Rechners und rechts die in der Physik zu sehen. Außerdem kann man über `Commands→Open Terminal` ein Terminal öffnen und wie vor Ort arbeiten. WinSCP ist ein tolles Programm, es gibt aber noch eine Alternative, welche Open-Source (GPL) und für Windows, Mac und Linux zu haben ist. Da wir ab und zu mit beiden Programmen Transferabbrüche hatten, scheint hier ein Problem seitens des CIP-Login-Rechners vorzuliegen.

FileZilla arbeitet wie WinSCP mit SFTP und zudem mit FTP, wofür es ursprünglich geschrieben wurde, und anderen FTP-Erweiterungen. Das übersichtliche und einfach zu konfigurierende Programm lässt sich hier[21] herunterladen. Wie bei WinSCP lassen sich verschiedene Server vorkonfigurieren. Der Datei- und Ordnerbetrachter kann zwischen dem Explorer- und dem MC-Modus je nach eigenen Vorlieben gewechselt werden. Sehr hilfreich ist die Transfer-Warteschlange mit sehr ausgetüftelten Was-tun-wenn-Datei-bereits-vorhanden-Auswahlmöglichkeiten.

Zur Konfiguration des CIP-Servers im Servermanager: „Neuer Server“ anwählen und unter Server `login.cip.physik.uni-goettingen.de` eintragen (Port kann leer gelassen werden). Als Servertyp ist SFTP - SSH File Transfer Protocol zu wählen. Bei Verbindungsart ist entweder „Normal“ zu wählen, dann ist Benutzername und Passwort einzugeben und dieses wird intern gespeichert, oder man wählt – wenn man dem Programm nicht vertraut – „Nach Passwort fragen“ und muss das Passwort bei jeder neuen Verbindung erneut eingeben. Nun lässt sich die Verbindung per „Verbinden“ öffnen (Einstellungen werden automatisch gespeichert). Die nachfolgende Abfrage nach dem Vertrauen in das Serverzertifikat ist mit „ja“ zu beantworten.

Cygwin

Alternativ kann man unter anderem in Form von Cygwin eine Linux-Oberfläche mitsamt bash, cp, grep, ssh, und allem was sonst noch dazugehört, installieren. In diesem Fall kann man in der von

[21] `http://filezilla-project.org`

Linux bekannten Weise `ssh` und `scp` verwenden und, was vielleicht vielen ans Herz wachsen wird, `kile` auch zu Hause laufen lassen.

Cygwin macht sich zu nutze, dass diese und alle anderen Grundfunktionen im Quellcode zugänglich sind. Mittels einer speziellen Laufzeitbibliothek, welche die Linux-Funktionalität bereitstellt, ist es möglich die meisten Linux-Anwendungen für Windows zu kompilieren.

KDE on Windows Initiative

Ähnlich veranlagt ist die KDE on Windows Initiative[22], welche KDE-Programme für Windows anbietet. Dies ist wahrscheinlich die einfachere Variante, ein Windows-`kile` zu starten[23].

Hier wird ausgenutzt, dass die Grundlage von KDE, die Qt-Bibliothek auch unter Windows unter einer Open-Source-Lizenz veröffentlicht wurde und damit auch alle darauf aufbauende Anwendungen leicht auf Windows zu portieren sind.

[22] `http://windows.kde.org/`
[23] `http://sourceforge.net/apps/mediawiki/kile/index.php?title=KileOnWindows`

3 Editoren

3.1 Was ist ein Editor?

Ein (Text-)Editor ist ein Programm zum Bearbeiten von Text, genauer gesagt von rohem Text, der ohne Formatierungen wie Fett und Kursiv vorliegt. Er sollte nicht mit einem Textverarbeitungsprogramm wie Word verwechselt werden und ist unverzichtbar für alle, die programmieren, ihre Texte in Latex schreiben oder Konfigurationsdateien ändern wollen. Ferner kann ein Editor für kleinere Notizen eingesetzt werden und stellt noch immer die Standardumgebung zum E-Mail schreiben bereit.[1]

3.1.1 Die verschiedenen Editoren

Da es so viele verschiedene Aufgaben gibt, die man mit einem Editor bewältigen muss, gibt es die verschiedensten Varianten und es scheint manches Mal eine Glaubensfrage zu sein, welchen man benutzt. Hier werden wir nur auf eine kleine Auswahl hinweisen.

Die Palette reicht unter Linux von sehr einfachen Textanzeige- und Editierprogrammen (wie `mousepad`), über semi-professionelle Editoren (wie `gedit` oder `kate`) bis hin zu eierlegenden Wollmilchsäuen wie es `emacs` oder `vim` sind. Die Wahl *seines* Editors ist reine Geschmackssache und man sollte durchaus einige durchprobieren.

Einige erweiterte Funktionen helfen im Alltag aber schon. Praktisch ist beispielsweise, dass er zumindest Syntaxhighlighting, also die farbliche Hervorhebungen wichtiger Wörter und Konstrukte, und Bracketmatching, das Anzeigen und Auffinden einer zugehörigen, schließenden Klammer, beherrscht.

Es gibt einige spezialisierte Editoren wie `bluefish`, `screem` für Websiten oder `texmaker`, `kile` und `winefish` für Latex. Diese bieten viele vorgefertigte Textschnipsel, die leicht mit der Maus angeklickt oder mit einer Tastenkombination aufgerufen werden können, was den unschätzbaren Vorteil hat, dass man sich nicht alles merken muss. Das können sie dann aber auch nur für ihre jeweilige Sprache. Auch Gliederungsansichten, Codefolding, das *Einklappen* ganzer Absätze, oder integrierte Ausführung von Filterprogrammen und Compilern sind wertvolle Ergänzungen. Viele Editoren bieten darüber hinaus erweiterte Textersetzung, automatische Vervollständigung von Schlüsselwörtern, automatische Klammerung uvm. Hier alles aufzuzählen würde den Rahmen bei Weitem sprengen; die Websiten und Dokumentationen der angesprochenen Programme helfen weiter.

Da die meisten Linuxneulinge zuerst mit KDE Kontakt aufnehmen, weisen wir auf Kate, den „**K**DE **a**dvanced **t**ext **e**ditor“ hin, den Standardeditor unter KDE. Er hat eine Menüleiste, eine Knopfleiste und auf der linken Seite kann man einsehen, welche Dateien sich noch im aktuellen Verzeichnis befinden. Praktisch ist an ihm auch das Syntaxhighlighting. In Abbildung 3.1 ist der Kate mit einem Latex-Dokument, das in Kapitel 4 näher erläutert wird, zu sehen.

Ein weiterer Editor, Kile (**K**DE **I**ntegrated **L**aTeX **E**nvironment), baut auf Kate auf. Er ist nicht allzu vielseitig, aber bestimmt eine Erleichterung für alle, die Texte in Latex verfassen wollen. Viele Latex-Befehle sind bereits als Knöpfe vorhanden, so dass man nicht so viel auswendig lernen und tippen muss. Die von Kate bekannte Verzeichnisanzeige wurde hier auf die Bedürfnisse von Latex

[1] Auch kann er für künstlerische Gestaltungen genutzt werden: `http://www.chris.com/ASCII/`

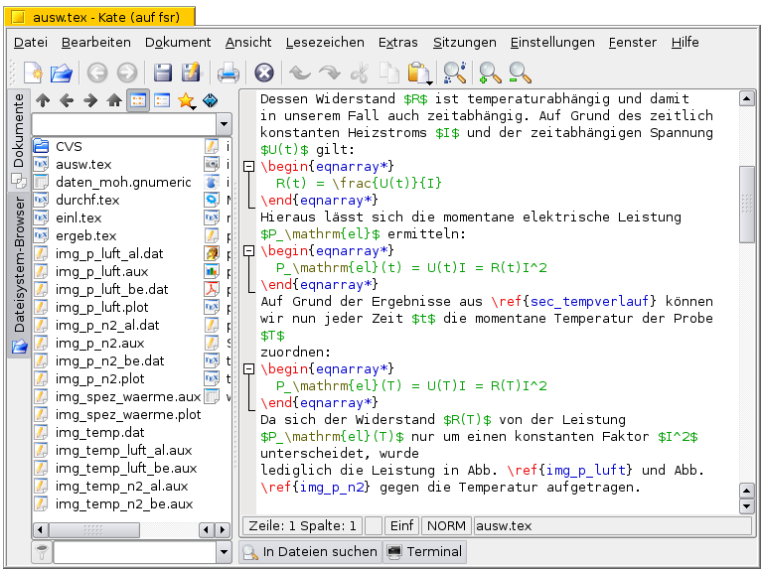


Abbildung 3.1: Der Kate mit einem Latex-Dokument.

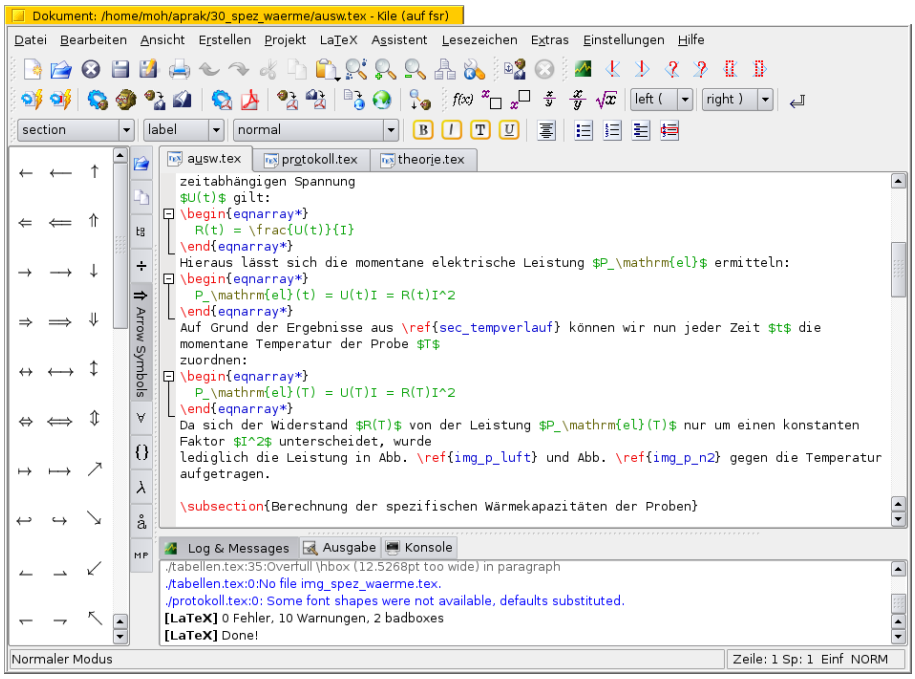


Abbildung 3.2: Der Kile mit demselben Latex-Dokument wie beim Kate.

spezialisiert. Man kann alle selbsterstellten Referenzen einsehen oder, wenn man sein Dokument auf mehrere Dateien verteilt hat, den Zusammenhang übersichtlich halten. Einen kleinen Vorgeschmack gibt Abbildung 3.2.

Man kann sowohl Kate, als auch Kile nur unter X11 benutzen (siehe Kapitel 2.6). Sie verfügen beide nur über ein sogenanntes **g**raphical **u**ser **I**nterface (GUI)[2].

Nicht unerwähnt bleiben sollen auch `mcedit` und `scribes`[3] für Puristen; `ed` und `sed` für Sadis-

[2] ...also bunte Fenster zum drinne rumklicken ;)

[3] scribes.sf.net ist als Homepage von Steffens Lieblingseditor auf jeden Fall einen Blick wert, vor allem das Beispielvideo.

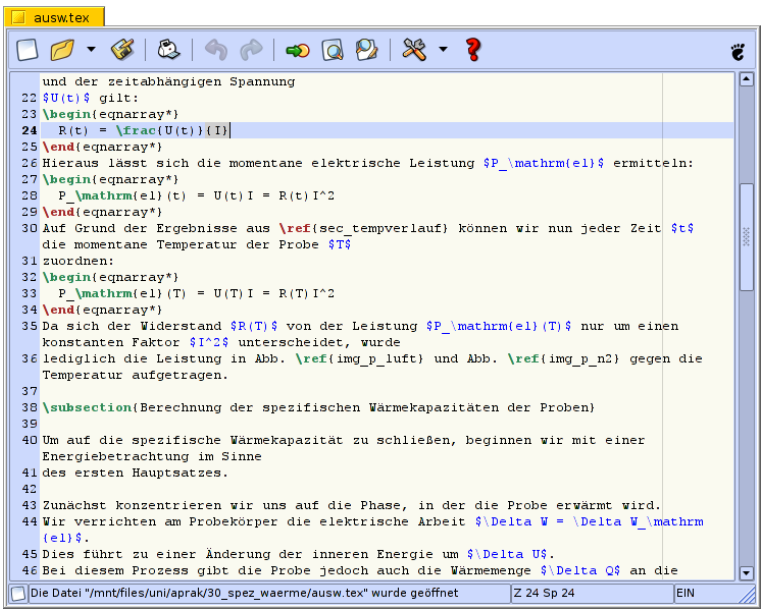


Abbildung 3.3: Und Scribes mit noch einmal dem gleichen Dokument.

ten sowie als funktionelle (und ressourcenverbratende) Riesen `eclipse` und `jedit`. Alle Editoren sind oft vorinstalliert oder schnell mit Google oder auf `Freshmeat.net`[4] zu finden. Letzteres bietet mit seiner Suchfunktion auch noch wesentlich mehr Stoff für eigene Versuche.

Die beiden Standardeditoren unter Linux sind aber `vi` (oder besser sein großer Bruder `vim`) und `emacs`, deren Verfechter bereits einige Glaubenskriege ausgeführt haben. Wir stellen hier in diesem Dokument beide Editoren kurz vor. Falls ihr euch dann entschieden habt, doch mal einen oder gar beide auszuprobieren, steht für euch ein Extraheftchen bereit. Dieses soll als eine kleine Einführung euren Weg in die "profi"-Editierung mit `vim` oder `emacs` einleiten.

Da der kleine Bruder vom vim, der vi auf *jedem* Linuxsystem zu finden ist, ob alt, ob neu, ob im Handy, auf der Playstation oder der DBOX, kann es nicht schaden eine Einleitung zu ihm zu lesen und sich ein bisschen einzuarbeiten. Für die Ungeduldigen, die ihn jetzt schon geöffnet haben: ESC gefolgt von :q! und einem Enter beendet vi ohne zu speichern.

Der Emacs ist in der Unix-/Linuxwelt ebenfalls recht verbreitet, in Ansätzen grafisch und nach einiger Gewöhnung sehr flott zu bedienen. Man kann ihn sowohl in der Konsole, als auch unter X11 (siehe Kapitel 2.6) benutzen. Er ist sicher von den erwähnten Editoren der Mächtigste, wobei man sich fragen kann, ob man wirklich Tetris in einem Editor spielen können muss.

Ein kurzer Rundblick auf andere Betriebssysteme soll aber nicht fehlen. So können wir unter Mac-OS `textwrangler` und `Smultron` empfehlen. Unter Windows helfen `texmaker`, `notepad++`, `TeXnicCenter` und auch `Eclipse` (mit Latex-Plugin) über erste Hürden hinweg. Und um der Frage vorzubeugen, ja es gibt Möglichkeiten `kile` unter Windows laufen zu lassen, diese sind aber steinig und schwer. Als Tipp seien die Worte `cygwin`/ `mingw` sowie `ssh -X` erwähnt. Emacs und vi laufen selbstverständlich auch unter den beiden anderen Betriebssystemen.

[4] Freshmeat ist ein Verzeichnis (*fast*) aller Programme für Linux/ Unix.

3.1.2 Warum der Emacs?

Emacs ist im Gegensatz zu den meisten der bis hierhin erwähnten Editoren richtig alt. Er wurde schon 1976 von Richard Stallman entwickelt und zählte zu den ersten Editoren mit denen man, wie heute üblich, mit dem Cursor navigieren und so den Text bearbeiten konnte. Dies war in den 1970ern keineswegs die Regel, sondern Komfort pur!

Heutzutage stehen natürlich andere Features im Vordergrund. Neben gängigen Hilfsmitteln wie Syntax- und Klammer-Highlighting oder Rechtschreibprüfung gibt es zum Beispiel die Möglichkeit, lästige Standardaufgaben über Makros zu automatisieren. Es gibt eigentlich für fast alles den passenden Emacsbefehl ... so auch `M-x doctor`.[5]

Außerdem stehen für die meisten Dokumenttypen spezielle Modi bereit. So existiert ein Latex-Modus der unter anderem beim Referenzieren von Tabellen und Bildern hilft und vor allem beim Eingeben von Formeln sehr viel Tipparbeit spart.

Das Ganze hat natürlich auch einen Haken: Um richtig schnell mit Emacs arbeiten zu können, braucht man etwas Einarbeitung und man sollte sich auch innerlich von der Maus verabschieden. Da in den kommenden Semestern aber unzählige Praktikumsprotokolle mit Latex verfasst werden wollen, lohnt es sich jedoch alle Mal, sich mit einem der "komplizierteren" Editoren anzufreunden – zum Beispiel mit Emacs!

3.1.3 Warum Vim?

Das Kürzel vi steht für *visual*, da es den Zeileneditor (ja, so wurde früher mal editiert) namens `ex`[6] visualisiert. Dieser wurde von Bill Roy im Jahre 1976 aufgewertet und hat sich daraufhin als der Standardeditor in der Unix-Welt bewährt.

In den folgenden Jahren wurde der vi von vielen verbessert, und die Version, welche sich durchsetzen konnte, ist "VI Improved" oder kurz Vim, welches in den 90er Jahren von Bram Moolenaar herausgebracht wurde.

Öffnet ihr vi, habt ihr einen sehr einfachen, aber effektiven und ressourcenfreundlichen[7] Editor vor euch. Startet ihr dagegen Vim, öffnet sich ein Editor, welcher von seiner Mächtigkeit nahezu identisch zum vorhin angesprochenen Emacs ist. So kann Vim ebenfalls Syntaxhighlighting, unterstützt die Maus, hat eine Wortvervollständigung, eine Rechtschreibprüfung und noch viel, viel mehr. Vorhanden sind auch tausende von Plugins, wie z.B. Latex-suite, welche euch das Leben mit Latex-Dokumenten vereinfachen wird.

Der wesentliche Unterschied zu Emacs sind die verschiedenen Modi. Während man bei Emacs alle Befehle durch das Drücken von Modifikationstasten, wie z.B. der `Strg` oder `Alt` Taste, beginnt, besitzt vi (und dann natürlich auch Vim) einen eigenen Befehlsmodus. Dadurch können sehr einfach mehrere Befehle hintereinander abgesetzt werden[8]. Die verschiedenen Modi an sich sind jedoch für Neueinsteiger gewöhnungsbedürftig und abschreckend. Denn:

"Sure vi is user-friendly; it's just peculiar about who it makes friends with."[9] - Unbekannter Autor

Viel Erfolg beim Flirten mit Vim!

[5] Man drückt `Alt` und `x` gleichzeitig, tippt dann `doctor` ein und schließt mit `Enter` ab.
[6] Diesen Editor gibt es jetzt immernoch, also einfach mal ausprobieren und sich dann glücklichschätzen, dass es jetzt Vim oder Emacs gibt.
[7] Da vi schneller als Emacs startet, wird dieser auch von Emacs-Benutzern für kleine Arbeiten verwendet.
[8] z.B. löscht `3dd` drei Zeilen auf einmal.
[9] „Sicher ist vi benutzerfreundlich, es ist nur eigen in der Wahl seiner Freunde".

4 Der Textsatz mit Latex

4.1 Einleitung: Latex

Auf den ersten Blick wirkt Latex (eigentlich LaTeX) auf Neulinge wie wild gehackter Code. Am Anfang denkt man sich: „Wieso einfach, wenn es auch kompliziert geht?“ Bei dem, was heutige sogenannte WYSIWYG[1] Textverarbeitungsprogramme optisch leisten, fragt man sich, warum man überhaupt noch „primitiv“in einen Editor Befehle eingeben sollte, um dann einen formatierten Text zu erhalten.
Doch bietet LaTeX auch überzeugende Vorteile, diese sind vor allem ein sauberes Layout, welches LaTeX im Gegensatz zu den meisten WYSIWYG-Programmen dem Nutzer fast vollständig abnimmt. Außerdem besticht es durch einen sehr guten Formelsatz und kaum Kompatibilitätsprobleme, wie man sie u.a. von den ständig wechselnden Word-Versionen gewohnt ist. Auch die Möglichkeit professioneller Präsentationen und Briefe sollten nicht unterschätzt werden.

Weiterhin fertigt LaTeX vollautomatisch Referenzen, Fußnoten, Bibliographien (also Literaturangaben) und Inhaltsverzeichnisse sehr einfach an.

4.1.1 Womit bearbeite ich Latex?

Zum einen gibt es WYSIWYG-Interfaces für LaTeX, wie z.B. LyX unter Linux, die gleich den formatierten Text anzeigen und es für Anfänger erleichtern, einen Text zu verfassen.

Andererseits gibt es Editoren (nicht WYSIWYG), welche die Arbeit mit LaTeX auf verschiedene Art und Weise unterstützen. Hierbei ist vor allem das Syntaxhighlighting zu nennen, welches von allen gängigen Editoren unterstützt wird. Dabei werden Kommandos, Klammern oder Bereiche logisch markiert. Z.B. würde von `\textbf{Fettertext}` das „`\textbf`“ in pink, und das „`{Fettertext}`“ in dunkelgrün (oder hellblau, oder purpurrot, oder ...) dargestellt. So erkennt man besser, was zusammengehört und was nicht. Emacs stellt z.B. fette oder kursive Schrift bereits beim Eingeben entsprechend dar. Andere Editoren bieten darüber hinaus umfangreiche Schaltflächen und Menüs, in welchen Umgebungen und Kommandos per Knopfdruck abgerufen werden können, sodass man nicht alle auswendig parat haben muss, hier wäre unter Linux vor allem Kile zu nennen. Unter Windows ist das TeXnicCenter relativ beliebt.

4.1.2 Literatur

Die richtige Literatur kann beim Umgang mit LaTeX helfen, es ist jedoch Geschmackssache, ob man lieber ein Buch in der Hand hat oder die Internetrecherche vorzieht. Die folgende Auswahl ist daher als Anregung zu verstehen...

- **The Not So Short Introduction to LaTeX2e**
  T. Oetiker, H. Partl, I. Hyna, E. Schlegel
  `http://tobi.oetiker.ch/lshort/lshort.pdf`

[1] **W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et

- **Das LaTeX Handbuch**
  L. Lamport
  Addison-Wesley
- **Der LaTeX Begleiter**
  M. Goossens, F. Mittelbach, A. Samarin
  Addison-Wesley
- **LaTeX Band 1 - Einführung**
  H. Kopka
  Addison-Wesley
- **LaTeX–Eine Einführung und ein bißchen mehr...**
  M. Jürgens, T. Feuerstack
  `http://www.fernuni-hagen.de/imperia/md/content/zmi_2010/a026_latex_einf.pdf`
- **LaTeX–Fortgeschrittene Anwendungen. Oder: Neues von den Hobbits...**
  M. Jürgens
  `http://www.fernuni-hagen.de/imperia/md/content/zmi_2010/a027_latex_fort.pdf`
- **The Comprehensive LaTeX Symbol List**
  S. Pakin
  `http://www.ctan.org/tex-archive/info/symbols/comprehensive`
- **LaTeX GE-PACKT**
  Karsten Günther
  mitp-Verlag

Die NASA stellt unter der Internetadresse `www.giss.nasa.gov/latex/` eine Sammlung von nahezu allen gebräuchlichen LaTeX-Anweisungen und -Umgebungen bereit.

4.2 Die Grundstruktur eines Latex-Dokumentes

4.2.1 Der Dokumentenheader

Alle die Dinge, die man beispielsweise Word oder dem LibreOffice-Writer normalerweise per Klick auf Schaltflächen mitgibt (Schriftgröße, Schriftart, ...), müssen in LaTeX durch einen geeigneten Befehl ausgedrückt werden. Bei Word fängt man standardisiert auf einer A4 Seite im Hochkantformat zu schreiben an, Sonderwünsche muss man extra anmelden, für Präsentationen bedarf es eines neuen Programmes. Da LaTeX viele Dokumententypen zulässt, müssen hier grundsätzlich einige *Deklarationen/Festlegungen* getroffen werden, bevor man mit dem Schreiben loslegen kann. Für all dies ist der Dokumentenheader gut. Beispielhaft könnte der Beginn eines Dokumentes wie folgt aussehen:

```
1 \documentclass[12pt,a4paper]{article}
2 \usepackage{a4}
3 \usepackage[utf8]{inputenc}
4 \usepackage[ngerman]{babel}
5 \usepackage{graphicx}
6 \usepackage{amsfonts,amstext,amsmath}
```

Der Befehl `\documentclass{dokumentenklassenname}` legt fest, welcher Dokumenttyp erstellt werden soll (*dokumentenklassenname* wird dann durch den Namen der jeweilige Klasse ersetzt). Im Praktikum ist vor allem die Dokumentenklasse *article* wichtig. Weitere sind z.B. *book*, *beamer* und *dinbrief*. Diese Klassen legen bereits einige Parameter für das Dokument fest, etwa die Seitenränder, die Schriftgröße oder die Schrift selbst. Durch Befehle lassen diese sich jedoch auch individuell variieren.
Im oben dargestellten Fall soll ein Dokument der Dokumentenklasse *article* erstellt werden. In den eckigen Klammern stehen noch weitere Deklarationen, die für das gesamte Dokument gelten sollen. Durch `a4paper` wird das Papierformat DIN-A4 festgelegt (andernfalls wird US-Letter benutzt) und durch `12pt` die Schriftgröße auf 12pt gesetzt (Standard ist 10pt).

Zur Erstellung von zweiseitig bedruckten Dokumenten gibt man als Option *twoside* an, dies führt dazu, dass die Seitenränder nach grader und ungrader Seite angepasst werden. *twocolumn* gibt das Dokument zweispaltig aus.

Wie man den Zeilenabstand verändert, steht in Tabelle 4.1.

Möchte man etwas mehr als nur die allernötigsten Befehle benutzen, so benötigt man **Pakete**. Diese Pakete stellen die Definition von weiteren Befehlen u.ä. bereit. Werden die Befehle einfach ohne weiteres benutzt und die Pakete nicht eingebunden, so können diese beim Kompilieren auch nicht umgesetzt werden. Diese Pakete werden mit dem Befehl `\usepackage[optionen]{paketname}` eingebunden. `paketname` steht dabei für den Namen des Paketes. In der eckigen Klammer können weitere Optionen für das jeweilige Paket angegeben werden. Die eckigen Klammern können allerdings auch weggelassen werden, wenn keine Optionen notwendig sind.

Bemerkung: Benötigt ein LaTeX-Befehl ein Argument, d.h. verlangt LaTeX dafür eine Angabe, dann kommt diese in {geschweifte Klammern}. Optionale Argumente, die also nicht zwingend erforderlich sind, stehen in `[eckigen Klammern]`. Dies macht man sich leicht am Beispiel von `\usepackage[`*option*`]{paketname}` klar: wird kein Paketname angegeben, dann bleibt die geschweifte Klammer leer oder sie fehlt ganz. Es fehlen daher benötigte Angaben (der Befehl macht keinen Sinn mehr) und es wird beim Kompilieren eine Fehlermeldung herausgegeben.

Die Optionen werden vor der geschweiften Klammer in der eckigen angegeben. Beispielsweise benötigt das Paket *ngerman* keine weitere Angaben: `\usepackage{ngerman}`. Zur Vereinfachung können Pakete, die dieselben Optionen haben (insbesondere gar keine Optionen) mit Kommata (aber ohne Leerzeichen!) getrennt in die geschweifte Klammer eines `\usepackage{...}` Befehls geschrieben werden, z.B. `\usepackage{ngerman,float}`.

4.2.2 Der Text für das Dokument

... kommt in die sog. *document*-**Umgebung**!
Umgebungen sind Bereiche innerhalb des LaTeX-Quellcodes, in die bestimmte Objekte geschrieben werden. Beispielsweise wird es sich als nützlich erweisen, für Grafiken oder Tabellen einen gesonderten Bereich zu deklarieren, welchen LaTeX innerhalb des Dokumentes an eine geeignete Stelle verschieben darf, damit die Seiten so besser ausgestaltet werden.

Eine Umgebung fängt innerhalb des Dokumentes mit dem Befehl

```
\begin{Umgebungsname}
```

an und endet mit dem Befehl

```
\end{Umgebungsname}.
```

`inputenc`	Hierbei handelt es sich um eine Deklaration der Eingabekodierung: Dieses Paket ermöglicht die direkte Eingabe von nicht ASCII-Zeichen (wie ä,ö,ü) und somit auch deren Umsetzung im Dokument. Andernfalls müssten diese Zeichen durch Befehle umschrieben werden. Unter *Linux* ist die Option *utf8* gut geeignet.
`babel`	Ist das Paket für die Unterstützung verschiedener Sprachen; u.a. werden mit diesem Paket die richtigen Trennregeln eingebunden. Durch die Option `ngerman` wird die neue deutsche Rechtschreibung verwendet.
`a4`	Passt auch die Seitenränder auf DIN-A4 an. Die Option `4apaper` der Dokumentenklasse legt nur das Format fest.
`graphicx`	Wird benötigt um später Grafiken einbinden zu können.
`amsfonts`, `amstext`, `amsmath`	Diese drei Pakete der Amerikanischen Mathematischen Gesellschaft erweitern den LaTeX-Mathemodus um nützliche Funktionen. Im Folgenden wird angenommen, dass sie geladen sind.
`setspace`	Enthält unter anderem das Kommando `\onehalfspacing`, um auf 1,5 zeiligen Zeilenabstand umzuschalten. Außerdem gibt es `\doublespacing` und `\singlespacing`.

Tabelle 4.1: Einige nützliche Pakete

Zwischen diese beiden Befehle können die jeweiligen Objekte geschrieben werden. Um nun den Dokumentenheader vom Text des eigentlichen Dokumentes zu trennen wird die Umgebung *document* verwendet.

LaTeX erstellt automatisch (mit `\maketitle`) eine Titelseite für das Dokument, wenn man vorher Autor, Datum und den Titel (weitere Angaben sind möglich) definiert. Dies könnte wie folgt aussehen:

```
1 \begin{document}
2 \author{Albert Einstein}
3 \date{30.06.1905}
4 \title{Zur Elektrodynamik bewegter Körper}
5 \maketitle
6
7 \newpage
8 ...
9 \end{document}
```

Dabei erzeugt, wie schon erwähnt, `\maketitle` die Titelseite, `\newpage` erzeugt einfach einen Seitenumbruch. Optional kann man mit`\date{\today}` das jeweils aktuelle Datum der Systemzeit einsetzen lassen.

4.2.3 Eine Gliederung des Dokumentes

Damit der Text eines Dokumentes nicht wie ein riesiger Felsblock einfach nur uniform auf den Seiten steht, ist es sinnvoll, diesen durch Überschriften und Nummerierungen zu gliedern. Dies erfolgt unter LaTeX mit den Befehlen: `\section{Abschnittstitel}`, `\subsection{Unterabschnittstitel}` und `\subsubsection{Unterunterabschnittstitel}`. Beispiele für diese Nummerie-

rung sind die Über- und Unterüberschriften im vorliegenden Heftchen! Typischerweise sieht ein Dokument dann so aus:

```
      :
\begin{document}
      :
\section{Erster Abschnitt}
      :
\subsection{Unterabschnitt}
      :
\subsubsection*{Unterunterabschnitt}
...aber jetzt ohne Numerierung
      :
\section{Neuer Abschnitt}
      :
\end{document}
```

Der Stern direkt hinter „\(sub)section“ lässt die Nummerierung wegfallen, es bleibt nur eine Überschrift stehen. Weiterhin lässt sich eine sehr feine Unterteilung mit den untergeordneten `\paragraph{...}` und `\subparagraph{...}` vornehmen.

Durch Verwendung dieser Befehle lässt sich ein Inhaltsverzeichnis erstellen. Dazu ruft man das Kommando `\tableofcontents` einfach an der Stelle auf, an welcher das Inhaltsverzeichnis eingefügt werden soll.

Für die Dokumentenklassen *book* stehen zusätzliche Strukturierungsmöglichkeiten zur Verfügung: `\part{...}` und `\chapter{...}`.

4.2.4 Das Setzen von Text

Einfacher Text wird so umgesetzt, wie man ihn schreibt, allerdings werden die Leertaste und „neue Zeile“ anders behandelt: Ein einfacher Zeilenwechsel führt nur zur Trennung der Worte, ein zweifacher Zeilenwechsel führt zu einem neuen Absatz! Mehrere Leerzeichen werden grundsätzlich wie eines aufgefasst, mehrere Leerzeilen wie eine Leerzeile.

Ein weiteres Problem kann bei Befehlen auftreten, die nur aus Buchstaben bestehen und nicht mit einem Argument abschließen (also eine Klammer haben), wie z.B. `\grqq`, die deutschen Anführungszeichen unten. Nutzt man die Anführungszeichen innerhalb eines Satzes im Fließtext, so würde man intuitiv schreiben:

```
Wort1 \glqq Wort2 \grqq Wort3
```

Allerdings wird das Leerzeichen vor Wort3 als Ende des Befehles `\grqq` aufgefasst und das darauffolgende Wort3 ohne Abstand hinter die schließenden Anführungszeichen gesetzt (mehrere Leerzeichen würden hier wieder wie eines aufgefasst). Eine zuverlässige Möglichkeit zur Vermeidung dieser Problematik besteht darin, solche Befehle mit {}, also einem leeren obligatorischen Argument, zu schließen. Danach wird das Leerzeichen wieder als solches akzeptiert, also:

```
Wort1 \glqq{}Wort2\grqq{} Wort3
```

Die Größe der Schriftart lässt sich am Einfachsten durch die Befehle in Tabelle 4.2 ändern. Die Angaben sind relativ zur gewählten Standardschriftgröße. Wie man den Zeilenabstand verändert, steht in Tabelle 4.1.

`{\tiny winzig}`	winzig
`{\scriptsize Indexgröße}`	Indexgröße
`{\small klein}`	klein
`{\normalsize Standardgröße}`	Standardgröße
`{\large groß}`	groß
`{\Large sehr groß}`	sehr groß
`{\LARGE sehr sehr groß}`	sehr sehr groß
`{\huge riesig}`	riesig
`{\Huge gigantisch}`	gigantisch

Tabelle 4.2: Schriftgrößen

`\textrm{Serifenschrift}`	Serifenschrift
`\textsf{serifenlose Schrift}`	serifenlose Schrift
`\texttt{Schreibmaschinen-Schrift}`	`Schreibmaschinen-Schrift`
`\textbf{fettgedruckte Schrift}`	**fettgedruckte Schrift**
`\textit{Kursivschrift}`	*Kursivschrift*
`\textsc{Kapitälchen}`	KAPITÄLCHEN
`\emph{hervorgehobene Schrift}`	hervorgehobene Schrift

Tabelle 4.3: Schriftstile

Für die verschiedenen Schriftstile, also *kursiv*, **fett** usw. gibt es ebenfalls eine Liste von Befehlen, siehe hierzu Tabelle 4.3.

4.3 Spezielle Umgebungen

Im vorausgegangenen Kapitel wurden Umgebungen bereits kurz eingeführt. Umgebungen trennen Bereiche eines LaTeX-Dokumentes ab, in denen spezielle Bestandteile wie Formeln oder Tabellen erstellt werden können. Weitere wichtige Umgebungen sind:

tabular	Tabellenumgebung
figure	Grafikumgebung
itemize	Listenumgebung
enumerate	Aufzählumgebung
center	Zentrierungsumgebung

4.3.1 Erstellen mathematischer Formeln

Folgende Matheumgebungen gibt es unter LaTeX zum Erstellen einer Formel

- innerhalb eines Absatzes: `$ $`
- außerhalb des Absatzes **ohne** Nummer: `\begin{align*} \end{align*}`
- und außerhalb des Absatzes **mit** Nummer: `\begin{align} \end{align}`

Um zu referenzieren, schreibt man hinter die betreffende Gleichung `\label{eq:name}`, wobei für *name* ein Referenzname eingesetzt wird und das *eq:* optional zur besseren Übersicht hinzugefügt

`\\` `\newline`	Zeilenumbruch. Ergänzung: `\\[X]` die optionale Angabe in den eckigen Klammern führt zu einem entsprechend großen Abstand zwischen den beiden Absätzen. X besteht aus einer Zahl und einer Einheit (mm, cm, pt, m, in), also: 2mm, 0.002m, 0.123in o.ä.
`\hspace{X}` (`\vspace{X}`)	erzeugt einen horizontalen (vertikalen) Abstand der Größe X (Angabe wie oben).

`\smallskip`
`\medskip`
`\bigskip`
	erzeugt vertikale Abstände einer festen Größe

`\newpage`	erzwingt eine neue Seite
`\glqq` und `\grqq`	sind die deutschen Anführungszeichen (german left/right quotationmarks)
`\footnote{`Fußtext`}`	erzeugt eine Fußnote mit der Markierung an der Stelle, an der der Befehl im Text auftaucht. Die Nummerierung wird automatisch weitergezählt.
`\`	Ein Slash vor einem Leerzeichen schützt dieses Leerzeichen, es wird dann im Blocksatz nicht verbreitert (z.B. für Zahlen mit Einheiten).
`\,` und `\;`	erzeugen kleine Abstände (das Semikolon erzeugt einen wenig breiteren Abstand als das Komma). Der übliche Abstand zwischen Zahl und Einheit ist `\,`

Tabelle 4.4: Befehle für den Textsatz

`\frac{a}{b}`	$\frac{a}{b}$
`\sqrt{2}`	$\sqrt{2}$
`a_b`	a_b
`a_{max}`	a_{max}
`a^{min}`	a^{min}
`\left( \right)`	Der Größe des Inhaltes angepasste Klammern (funktioniert auch mit Betragsstrichen und allen anderen Klammersorten).

Tabelle 4.5: Einige wichtige mathematische Konstrukte unter LaTeX

werden sollte. Im Text ruft man diese Referenz dann später über „`... wie wir in Gleichung ~\ref{eq:name} leicht sehen, ...`“ auf und erhalten: „... wie wir in Gleichung 5 leicht sehen, ...“. Dabei wird durch das `~` bewirkt, dass *Gleichung* und *5* zusammenstehen und die Zeile hier nicht umgebrochen wird.

In Matheumgebungen werden alle Zeichen wie auch im normalen Text der Reihenfolge nach umgesetzt, man kann jedoch weitere Kommandos nutzen:
Wichtig ist nun, dass diese Kommandos verschachtelt werden können.
Z.B.: `\frac{\frac{a}{b}}{\frac{c}{d}}` wird zu

$$\frac{\frac{a}{b}}{\frac{c}{d}}$$

Ein weiteres Beispiel:
Quelltext:

```
\begin{align}
   \frac{1}{c} \phi_k = \frac{\mu}{c^2} \sum_{i=1}^n
   I_i \int_i \int_k \frac{{\rm d} \vec{r}_i \cdot
   {\rm d} \vec{r}_k} {\left|\vec{r}_i - \vec{r}_k
   \right|} = \sum_i L_{ki} I_i
   \label{eq:elektro}
\end{align}
```

kompiliert:

$$\frac{1}{c}\phi_k = \frac{\mu}{c^2}\sum_{i=1}^{n} I_i \int_i \int_k \frac{\mathrm{d}\vec{r}_i \cdot \mathrm{d}\vec{r}_k}{|\vec{r}_i - \vec{r}_k|} = \sum_i L_{ki} I_i \tag{4.1}$$

Um mehrere Gleichungen in eine Umgebung zu schreiben, trennt man die Zeilen mit `\\`. Mit `&` werden die folgenden Gleichheitszeichen untereinander ausgerichtet.

```
\begin{align}
  \vec{\nabla} \cdot \vec{D}  &= 4 \pi \varrho, \\ \label{max1}
  \vec{\nabla} \times \vec{E} &= -\frac{1}{c} \frac{\partial
  \vec{B}}{\partial t}, \\ \label{max2}
  \vec{\nabla} \cdot \vec{B}  &= 0, \\ \label{max3}
  \vec{\nabla} \times \vec{H} &= \frac{4 \pi}{c} \left( \vec{j} +
  \frac{1}{4 \pi} \frac{\partial \vec{D}}{\partial t} \right). \label{max4}
\end{align}
```

$$\vec{\nabla} \cdot \vec{D} = 4\pi\varrho, \tag{4.2}$$

$$\vec{\nabla} \times \vec{E} = -\frac{1}{c}\frac{\partial \vec{B}}{\partial t}, \tag{4.3}$$

$$\vec{\nabla} \cdot \vec{B} = 0, \tag{4.4}$$

$$\vec{\nabla} \times \vec{H} = \frac{4\pi}{c}\left(\vec{j} + \frac{1}{4\pi}\frac{\partial \vec{D}}{\partial t}\right). \tag{4.5}$$

Im Gegensatz zum Fließtext lassen sich in der Mathematikumgebung eine Vielzahl von Symbolen einfügen, darunter auch das vollständige griechische Alphabet (vgl. Tabellen 4.6 und 4.7).

Für komplexe Befehle können Abkürzungen definiert werden. Dies geschieht mit dem Befehl `\newcommand{\Abkürzung}{langer Befehl}`. Als Beispiel kann man das auf handgeschriebenen Übungsblättern manchmal zu findende Zeichen $\stackrel{!}{=}$ definieren:
`\newcommand{\musteq}{\stackrel{!}{=}}`

Im Mathemodus kann man auch für einzelne Zeichen andere Schriften verwenden. Es gibt z.B.:

- `\mathcal{ABC}` $\mathcal{ABC}$
- `\mathfrak{ABC}` $\mathfrak{ABC}$
- `\mathbb{ABC}` $\mathbb{ABC}$

Bindet man noch mit `\usepackage{bbm,mathrsfs}` die Pakete `bbm` und `mathrsfs` ein, so stehen auch folgende Schriften zur Verfügung:

- `\mathscr{ABC}` $\mathscr{ABC}$
- `\mathbbm{123}` $\mathbbm{123}$

$\pm$	`\pm`	$\mp$	`\mp`	$\circ$	`\circ`
$\cdot$	`\cdot`	$\leq$	`\leq`	$\geq$	`\geq`
$\equiv$	`\equiv`	$\sim$	`\sim`	$\rightarrow$	`\rightarrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Leftrightarrow$	`\Leftrightarrow`
∞	`\infty`	$\hbar$	`\hbar`	$\Re$	`\Re`
$\Im$	`\Im`	$\neq$	`\neq`	$\in$	`\in`
$\ll\gg$	`\ll, \gg`	$\doteq$	`\doteq`	$\bar{g}$	`\bar`
$\dot{g}\,\ddot{g}$	`\dot{g}, \ddot{g}`	$g'\,g''$	`g', g''`	$\sqrt{fg}$	`\sqrt{fg}`
$\{$	`\{`	$\}$	`\}`	$\int$	`\int`
∂	`\partial`	∇	`\nabla`	$\square$	`\square`
$\overrightarrow{f}$	`\overrightarrow{f}`	$\overleftarrow{f}$	`\overleftarrow{f}`	$\hat{=}$	`\hat{=}`
$\overleftrightarrow{f}$	`\overleftrightarrow{f}`				

Tabelle 4.6: Liste mathematischer Symbole

4.3.2 Gleitobjekte

Objekte wie Grafiken oder Tabellen, die nicht durch einen Seitenwechsel auseinandergerissen werden dürfen, jedoch aufgrund ihrer Größe auch nicht überall hinpassen, können in eine Gleitumgebung gesetzt werden. Damit kann man LaTeX die Freiheit geben, diese dorthin zu schieben, wo Platz ist; diese Objekte können dann „gleiten“. Die Umgebung, die dies bei Tabellen ermöglicht ist die *table* Umgebung:

α	\alpha	β	\beta	γ	\gamma	δ	\delta
ϵ	\epsilon	ε	\varepsilon	ζ	\zeta	η	\eta
θ	\theta	ϑ	\vartheta	ι	\iota	κ	\kappa
λ	\lambda	μ	\mu	ν	\nu	ξ	\xi
o	o	π	\pi	ϖ	\varpi	ρ	\rho
ϱ	\varrho	σ	\sigma	ς	\varsigma	τ	\tau
υ	\upsilon	ϕ	\phi	φ	\varphi	χ	\chi
ψ	\psi	ω	\omega				
Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
Ξ	\Xi	Π	\Pi	Σ	\Sigma	Υ	\Upsilon
Φ	\Phi	Ψ	\Psi	Ω	\Omega		

Tabelle 4.7: Das griechische Alphabet

```
\begin{table}[!htbp]
  \centering
       :
  \caption{Tabellenunterschrift}
  \label{tab:1}
\end{table}
```

`\centering` bewirkt hier lediglich eine Zentrierung des Objektes. Hier noch eine Grafik, welche analog zur Tabelle in eine *figure*-Umgebung eingebunden ist:

```
\begin{figure}[!htbp]
  \centering
       :
  \caption{Bildunterschrift}
  \label{fig:1}
\end{figure}
```

Die eckigen Klammern [!htbp] stellen Positionierungsbefehle dar (siehe Tabelle 4.8).

Abkürzung	Stelle an die das Floatingobjekt gesetzt werden darf
h	here: also genau dort im Text, wo angegeben. Sinnvoll für kleine Floats
t	top: Am Anfang der Seite
b	bottom: Am Ende der Seite
p	page: Auf einer speziellen Seite auf der nur Floats stehen
!	Ohne Rücksicht auf irgendwelche internen Beschränkungen
H	HERE: wie h nur in Verbindung mit `\float`-Paket

Tabelle 4.8: Mögliche Positionierungsvorgaben für Tabellen und Grafiken

Grafiken unter Latex

Das Einbinden einer Grafik erfolgt am besten mit einer `.pdf`-Datei. `pdflatex` erlaubt aber auch die Benutzung von `.png`- und `.jpg`-Grafiken. An der Stelle im Text, oder noch besser in der *figure*-Umgebung schreibt man dann:
`\includegraphics[Option=Wert,...]{pfad/datei}` Dabei kann *pfad* (Ein Pfad ist die „Adresse der Datei“ im Dateisystem) ein relativer Pfad (z.B. *Documents/*) oder der absolute Pfad (z.B. */home/mmuster/Documents/*) sein. *datei* ist der Name der Datei, etwa *plot1.pdf*.

Als Optionen kommen in Frage:

Option	Effekt
width	skaliert Grafik auf angegebene Weite
height	skaliert Grafik auf angegebene Höhe
angle	dreht Grafik um angegebenen Winkel
scale	skalierte Grafik

Insgesamt könnte dort also stehen:
`\includegraphics[angle=90,scale=1.2]{/home/mmuster/Documents/plot1.pdf}`
Das Referenzieren funktioniert hier ebenfalls: Wenn man die Grafik in eine *figure*-Umgebung eingebunden hat, dann kann mit dem Befehl `\caption{Bildunterschrift}` eine Bildunterschrift (wenn es unter dem `\includegraphics{...}` steht, wenn der Befehl davor kommt, dann setzt LaTeX eine Bildüberschrift) und mit dem Befehl `\label{fig:name}` einen Referenznamen erstellen (das Verfahren funktioniert wie im Abschnitt über mathematische Formeln beschrieben).
Das kann dann so aussehen:

```
1 \begin{figure}[!h]
2   \centering
3   \includegraphics[width=3.1in]{./plot.pdf}
4   \caption{Ein spannender Plot}
5   \label{fig:spec}
6 \end{figure}
```

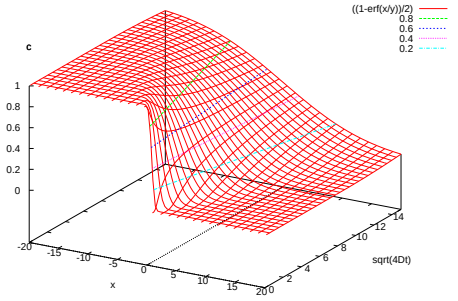


Abbildung 4.1: Ein spannender Plot

Tabellen

Tabellen werden in der Umgebung *tabular* erstellt. Diese wiederum sollte man in die Umgebung *table* einbetten, damit die Tabellen verschiebbar sind. Die Tabelle selbst ist für den Anfänger etwas ungewohnt zu schreiben: Die erste Zeile lautet: `\begin{tabular}{l r c}` Hier taucht also ein zweites obligatorisches Argument auf: In der zweiten geschweiften Klammer wird die Anzahl der Spalten und die Ausrichtung des Textes in den Spalten festgelegt. Dabei steht jeder Buchstabe für eine Spalte. `l` heißt, dass der Inhalt der Spalte linksbündig gesetzt wird, die Spalte mit `c` hat den Inhalt zentriert und `r` steht für rechtsbündig. Mit `p{X}` kann man einer Spalte eine feste Breite zuweisen, der Text ist linksbündig und wird ggf. umgebrochen. X besteht wie zuvor aus einer Zahl und direkt darauf folgend einer Maßangabe.

Die Zeilen starten jeweils mit dem Eintrag der ersten Spalte. Ist dieser fertig, so folgt das „kaufmännische Und“ (&), dann der Eintrag der zweiten Spalte und so weiter. Hinter der letzten Spalte kommt ein Zeilenumbruch `\\` und man verfährt ebenso mit den anderen Spalten. Nach der letzten Spalte benötigt man keinen Zeilenumbruch, sondern beendet die Tabellenumgebung mit einem `\end{tabular}` Mit `\hline` erstellt man eine waagerechte Trennlinie, die die gesamte Tabelle durchzieht. Senkrechte Trennlinien werden in der zweiten Klammer von `\begin{tabular}{l r c}` angegeben. Die Buchstaben stehen jeweils für eine Spalte. Entsprechend fügt man nun an der gewünschten Stelle eine sog. *Pipe* | (AltGr+<) ein. Möchte man also an den linken Rand der Tabelle eine Linie haben und zwischen der zweiten und dritten Spalte - dann sieht obige Zeile so aus: `\begin{tabular}{|l r|c}`.

Ein Beispiel für eine ganze Tabelle:

```
\begin{table}[!htbp]
  \centering
  \begin{tabular}{|c|l|}
    \hline
    $1{,}25$  & $2{,}34$ \\
    $2{,}50$  & $3{,}34$ \\
    \hline\hline
    $3{,}75$  & $4{,}34$ \\
    $5{,}00$  & $5{,}34$ \\
    $6{,}25$  & $6{,}34$ \\
    \hline
  \end{tabular}
  \caption{Mögliche Positionierungsvorgaben für Tabellen und Grafiken}
  \label{tab:position}
\end{table}
```

1,25	2,34
2,50	3,34
3,75	4,34
5,00	5,34
6,25	6,34

Tabelle 4.9: Mögliche Positionierungsvorgaben für Tabellen und Grafiken

Listen unter Latex

Listen, also Aufzählungen, können nummeriert oder unnummeriert sein. Eine unnummerierte Liste wird in die Umgebung *itemize* geschrieben, wohingegen eine nummerierte Liste von der *enumerate* Umgebung eingeschlossen wird. Dabei beginnt jeder Aufzählungspunkt mit dem Befehl `\item`.
Auch Aufzählungen können ineinander geschachtelt werden, indem man hinter einen `\item` einfach eine weitere Aufzählungsumgebung einfügt:

```
\begin{enumerate}
  \item Erster Punkt, erste Ebene
  \begin{enumerate}
    \item Erster Punkt, zweite Ebene
    \item Zweiter Punkt, zweite Ebene
      \begin{enumerate}
        \item Erster Punkt, dritte Ebene
      \end{enumerate}
    \end{enumerate}
  \item Zweiter Punkt, erste Ebene
\end{enumerate}
```

1. Erster Punkt, erste Ebene
    a) Erster Punkt, zweite Ebene
    b) Zweiter Punkt, zweite Ebene
        i. Erster Punkt, dritte Ebene
2. Zweiter Punkt, erste Ebene

Die beiden Umgebungen lassen sich beliebig austauschen – es ändern sich dadurch nur die Aufzählungszeichen!

4.4 Kompilieren

Nun fragt man sich noch, wie man aus diesen einfachen Editoreinträgen das gewünschte Dokument in DIN A4 bekommt. Man muss das Geschriebene zuerst kompilieren. Bei einigen Editoren kann man hierfür Tastenkombinationen oder auch Schaltflächen nutzen, die das Kompilieren automatisch übernehmen. Andernfalls muss dies über die Konsole geschehen. Dazu wechselt man zunächst in den Ordner, in welchem sich die Datei befindet. Das Kompilieren erfolgt dann über den Befehl
`pdflatex dateiname.tex`
Man erhält direkt eine pdf Datei mit dem Namen `dateiname.pdf`. Diese kann man sich nun mit dem Acrobat Reader, Xpdf oder Evince ansehen.

4.4.1 Shell Skripte

Wenn man innerhalb eines Dokumentes referenziert oder ein Inhaltsverzeichnis mit `\tableofcontents` erstellt, so wird es i.A. notwendig sein, LaTeX zweimal hintereinander auszuführen. Das liegt daran, dass der Compiler im ersten Durchgang eine Datei anlegt, in welcher die Informationen über die Struktur des Dokumentes gespeichert sind. Da er diese aber zum Erstellen der Referenzen benötigt, funktioniert dies bei nur einem Kompiliervorgang nicht. Beim zweiten

Durchgang liegen die Informationen aus dem ersten dann richtig vor und die gewünschten Funktionen werden (hoffentlich) richtig ausgeführt.

Bibliographien oder auch Stichwortverzeichnisse erfordern oftmals noch mehr Befehle, sodass es sich dort lohnt, ein Skript anzulegen, in welchem alle Befehle gespeichert sind. Zum Kompilieren muss dann nur noch das Skript aufgerufen werden, man benötigt damit dann nur noch ein Kommando. Ein Skript erstellt man, indem man zunächst mit einem Editor eine Textdatei in dem Verzeichnis des .tex-Dokumentes anlegt. Der Name kann dabei beliebig gewählt werden. In der ersten Zeile gibt man an, welche Shell das Skript benutzen soll. Anschließend schreibt man zeilenweise die Kommandos hinein, die ausgeführt werden sollen. Eine Datei, die unser Dokument zunächst zweimal kompiliert und anschließend anzeigt, sähe z.B. so aus

```
#!/bin/sh
pdflatex dateiname.tex
pdflatex dateiname.tex
xpdf dateiname.pdf
```

Anschließend müssen noch die Rechte der Datei (hier `script`) mit `chmod` geändert werden:

```
chmod +x script
```

Der Aufruf des Skriptes erfolgt anschließend über

```
./script
```

4.4.2 Fehlersuche

Häufig können kleine Tippfehler eine verheerende Wirkung haben. Ist eine Umgebung beispielsweise nicht richtig beendet worden, so kann der Text nach der vermeintlich beendeten Umgebung nicht mehr richtig interpretiert werden. Daher ein paar Hinweise, wie man solchen Pannen begegnen kann:

- Sind neue Befehle benutzt worden, die eventuell das Einbinden eines weiteren Paketes erfordern (`\usepackage{...}`)?
- Überprüfen, ob es Klammern gibt, die geöffnet, aber anschließend nicht wieder geschlossen worden.
- Überprüfen, ob Umgebungen nicht beendet worden (in dem Fall fehlt das `\end{Umgebung}`).
  Ein Hinweis auf diesen Fehler ist, wenn ein längeres Dokument mittendrin nach einer ganzen Seite abgebrochen wird.
- Überprüfen, ob vor irgendeinem Befehl ein `\` fehlt.
- Wenn es einen Bereich gibt, in welchem man den Fehler vermutet, so kann man diesen auskommentieren und anschließend schrittweise (zeilenweise) wieder „einkommentieren“.
- Die Fehlermeldungen des Compilers sind teilweise etwas kontraintuitiv - die Zeilenangabe („l. XX“) gibt aber häufig einen Hinweis darauf, wo ungefähr der Fehler liegt.

4.5 Präsentationen mit Latex

Eine Möglichkeit, Präsentationen mit Latex zu gestalten, besteht durch die Verwendung des Latex-Beamer Pakets. Im Internet kann man es unter `http://sf.net/projects/latex-beamer` herunterladen. Eine sehr gute Dokumentation wird gleich mit dem Paket installiert und ist unter `/usr/share/doc/latex-beamer/` zu finden. Im Folgenden werden die wichtigsten Elemente vorgestellt. Wollt ihr die fortgeschritteneren Möglichkeiten des Pakets kennenlernen und auch verwenden, so solltet ihr in der o.g. Dokumentation nachlesen.

4.5.1 Dokumentklasse und Header

Die Dokumentklasse, mit der man eine Beamer-Präsentation erstellt, wird mit

```
\documentclass{beamer}
```

eingebunden. Das Aussehen – d.h. den Aufbau, die Farben und die Schriften – bestimmen die sogenannten Themes. Ein Header mit dem Thema Berkley könnte z.B. so aussehen:

```
1 \mode<presentation>
2 {
3   \usetheme{Berkeley}
4   \usecolortheme{orchid}
5   \setbeamercovered{transparent}
6 }
```

Hierbei bestimmt das Thema Berkeley den Aufbau und orchid, ein sogenanntes Farbthema, überschreibt die Farbgestaltung, welche eigentlich durch das Thema vorgegeben wurde. Der dritte Befehl verändert das Verhalten von verdeckten Teilen einer Folie. Außer transparent gibt es noch `invisible`, `dynamic` und `highly dynamic`.

4.5.2 Titelseite und Inhaltsverzeichnis

Die Angaben zur Titelseite werden ebenfalls im Header vorgenommen. Mögliche Angaben sind die Folgenden:

```
1   \author[kurze Version]{Autor1 \inst{1} \and Autor2 \inst{2}}
2   \institute[1]{Universität1}
3   \institute[2]{Universität2}
4   \title[kurze Version]{Titel der Präsentation}
5   \subtitle{Untertitel}
6   \date{Datum, oder etwas anderes}
```

Der Text in den eckigen Klammern (in dem obigen Beispiel kurze Version genannt) ersetzt den richtigen Titel bzw. die kompletten Autorennamen im Inhaltsverzeichnis, falls vorhanden. Der Befehl subtitle, sowie die Institutsangaben sind nicht zwingend notwendig.

In der `document`-Umgebung wird die Titelseite dann in einer frame-Umgebung eingebunden (ein Frame ist sozusagen eine Folie, welche aber aus verschiedenen sichtbaren und verdeckten Elementen bestehen kann):

```
  \begin{frame}
    \titlepage
  \end{frame}
```

Genauso verfährt man für das Inhaltsverzeichnis mit dem Befehl `\tableofcontents`.

4.5.3 Gliederung der Präsentation

Eine Latex-Beamer Präsentation wird wie ein normales Latex-Dokument mit den Befehlen `\section`, `\subsection` und `\subsubsection` gegliedert. Wichtiger Unterschied ist nur, dass diese Befehle beim Beamer-Paket **keine** Überschriften darstellen! Sie dienen nur für das Inhaltsverzeichnis, was bei den meisten Themen an bestimmten Stellen der Folien angezeigt wird. Die eigentliche Gliederung läuft über die schon erwähnten Frames, welche alle eigene Überschriften besitzen können:

```
\begin{frame}
  \frametitle{Folienüberschrift}
  \framesubtitle{weitere Überschrift}
  Text usw.
  ...
\end{frame}
```

4.5.4 Textkästen

In Präsentationen ist es manchmal wünschenswert, Text in Kästen anzuordnen. Hierfür bietet das Beamer Paket mehrere Möglichkeiten: die Allgemeinste ist die `block`-Umgebung. Ein solcher, allgemeiner Textkasten wird wie folgt eingebunden:

```
\begin{block}<2->{Beispielkasten}
  Hier steht Text...
  \begin{itemize}
  \item Quelltext dieses Kastens
    ...
  \end{itemize}
\end{block}
```

Das Argument in den spitzen Klammern wird nach dem nächsten Abschnitt verständlich, das zweite Argument in geschweiften Klammern ist der Text, welcher im Titelbalken des Kastens steht. Darüber hinaus gibt es weitere vordefinierte Kästen mit bestimmten Farben und Titeln für gewisse Zwecke. Mit englischen Titeln wären da: example, examples, definition, definitions, proof, theorem, fact, corollary und lemma; deutsche Titel (man muss dafür allerdings Pakete wie babel oder ngerman verwenden) haben dagegen die folgenden Textkästen: Problem, Lösung, Definition, Satz, Beweis, Folgerung, Lemma, Fakt, Beispiel und Beispiele. Die Syntax dieser Umgebungen ist ähnlich:

```
\begin{blockname}<action-spez>[weiterer Text im Titel]
...
\end{blockname}
```

4.5.5 Overlays

Oft möchte man z.B. nicht alle Stichwörter, die sich auf einer Folie befinden, von Anfang an dem Publikum offenbaren, sondern erst während des Vortragens der Folie aufdecken. Zu diesem Zweck gibt es bei Latex-Beamer die Overlay-Anweisungen, mit denen die Sichtbarkeit von Elementen kontrolliert werden kann.

Die erste Möglichkeit bietet der Befehl `\pause`. Er veranlasst, dass immer nur die Teile eines Frames bis zum nächsten `\pause` Befehl sichtbar sind. So ist es ziemlich einfach möglich z.B. Aufzählungen oder Stichpunkte nacheinander auftauchen zu lassen, indem man zwischen die `\item` Einträge die `\pause` Befehle einfügt.

Die zweite und etwas kompliziertere – aber auch mächtigere – Möglichkeit besteht über die Angabe von Befehlen und Ebenennummern in spitzen Klammern. Für einfachen Text gibt es die Befehle

- `\only<slides>{Text...}`: Der Text in geschweiften Klammern wird nur in den angegebenen Ebenen gezeigt. Da der Text in einem Frame immer zentriert ist, macht es einen wichtigen Unterschied, ob Platz für einen nicht sichtbaren Text reserviert wird oder nicht! In diesem Fall wird **kein** Platz reserviert.
- `\uncover<slides>{Text...}`: Dieser Befehl deckt den Text in geschweiften Klammern auf den, in spitzen Klammern angegebenen, Ebenen auf. Vorher ist der Text halbtransparent (wie genau das geschieht hängt von der Angabe von `\setbeamercovered{}` ab, s.o.).
- `\visible<slides>{Text...}`: `\visible` lässt den Text in geschweiften Klammern nur auf den angegebenen Ebenen erscheinen, wobei aber deren Platz auf der Folie schon vorher reserviert ist (im Gegensatz zu `\only`).

Die Ebenenangaben in spitzen Klammern können einzelne Nummern sein, oder auch Zahlenbereiche wie 1-3. Lässt man die hintere Zahl weg, so bleibt das betreffende Element bis zum „Ende“ des Frames sichtbar, z.B. lässt `<2->` im obigen Kastenbeispiel den Kasten auf der zweiten Ebene auftauchen und er bleibt dann sichtbar bis zur nächsten Folie gewechselt wird.

Die obigen Befehle können auch in die spitzen Klammern geschrieben werden, damit die Steuerung auch für Befehle wie `\item` oder ganze Umgebungen wie Textkästen funktioniert. Lässt man die Befehle weg, wird der als Standard gesetzte Mechanismus verwendet, wie beim obigen Kasten. Mit den Befehlen lautet die Syntax (wobei das Wort slides wieder durch Zahlenangaben ersetzt wird):

```
1  <only@slides>
2  <uncover@slides>
3  <visible@slides>
```

4.5.6 Eine Eigenheit von Latex-Beamer

Mit der Verwendung von Quelltext auf Folien durch `\verbatim`-Umgebungen oder aber bei `\itemize`-Umgebungen in einem Textkasten kommt Latex-Beamer nicht ohne Weiteres zurecht - es versucht ihre Größe anzupassen und versagt dabei vollständig. In solchen Fällen sollte man die Option fragile verwenden, damit diese Konstruktionen funktionieren:

```
1  \begin{frame}[fragile]
2   ...
3  \end{frame}
```

4.5.7 Erstellung eines Handouts zur Präsentation

Falls ein Handout zur Präsentation gewünscht ist, braucht man sich bei Verwendung des Latex-Beamer Pakets nicht die Mühe zu machen, ein separates Dokument zu schreiben, sondern verwendet das Paket beamerarticle. Um ein Handout aus den Folien zu produzieren, braucht man nun lediglich die Zeile `\documentclass{beamer}` durch z.B. folgende Zeilen ersetzen:

```
1 \documentclass[11pt,a4paper]{article}
2 \usepackage{a4}
3 \usepackage{beamerarticle}
```

Das beamerarticle Paket wandelt alle `frame`-Umgebungen in „Standard“ Latex-Befehle um und verwendet zur Formatierung im obigen Beispiel das Paket `article`, 11pt Schriftgröße und A4 als Papiergröße. Gewisse Elemente wie z.B. `\frametitle{}` werden vom vorherigen Text etwas abgesetzt und fett gedruckt, Textkästen werden ähnlich umformatiert.

Will man jetzt zusätzlich Text zum Handout hinzufügen, welcher nicht auf den Folien angezeigt werden soll, kann dies so erreicht werden:

```
1 \documentclass[ignorenonframetext]{beamer}
```

Alle Textteile, welche sich außerhalb einer `frame`-Umgebung befinden, werden dadurch ignoriert.

5 Das Plot- und Fitprogramm Gnuplot

5.1 Motivation

Bei Gnuplot handelt es sich um ein kommandozeilenorientiertes Programm zur graphischen Darstellung von Messwerten oder analytischen Zusammenhängen wie beispielsweise der Sinusfunktion. Diese Fähigkeit wird gemeinhin als Plotten bezeichnet. Der Plot ist dann die Zeichnung, die Gnuplot für uns anfertigt. Zusätzlich ist Gnuplot auch in der Lage einen analytischen Ausdruck mit gemessenen Datenpunkten zu vergleichen und gewählte Parameter (z.B. die Frequenz der Sinusfunktion) so lange zu verändern, bis die angenommene Kurve zu den Datenpunkten passt. Diesen Vorgang bezeichnet man als Fitten und er begegnet dem Leser im physikalischen Grundpraktikum am häufigsten bei Ausgleichsgeraden.

Warum ist es nun so wichtig, Daten anzupassen und graphisch darzustellen?

Die Triebfeder für die Beschäftigung mit Gnuplot ist zunächst einmal ein besseres Verständnis der physikalischen Gesetze. Ein Plot hilft einem Physiker häufig dabei, qualitative Aussagen über Extrema, Nullstellen oder den Verlauf einer Bahn zu gewinnen. Formal ließe sich dies auch anhand von Rechnungen und Fallunterscheidungen erzielen, aber eine Kurve ist in einigen Fällen die einzige Möglichkeit, den Kurvenverlauf zu untersuchen. Gnuplot ist über Umwege in der Lage, sogenannte implizite Funktionen $f(x) = 0$, die sich nicht nach einer Variablen auflösen lassen, zu plotten.

Das Fitten geht einher mit der sogenannten Modellbildung. Hinter dem schicken Begriff Modell verbirgt sich der theoretisch zu erwartende Kurvenverlauf. Der Theoretiker setzt sich also hin und überlegt sich eine geeignete Methode um diesen darzustellen. Man hat einen anderen Begriff dafür gewählt, weil das Modell nicht zwangsläufig eine einfache Funktion wie $f(x) = m \cdot x + b$ bei einer Geraden sein muss. Außerdem erhält man das Modell auf einem ähnlichen Wege, wie man auch ein Spielzeugmodell aus Lego baut, d.h. durch Wahl der richtigen Steine und durch Spielen mit den Kombinationsmöglichkeiten. Schließlich möchte man die Vorhersagekraft seiner Theorie testen. Hier kommt Gnuplot ins Spiel. Hat man ein Modell, versucht man die unbestimmten Parameter wie beispielsweise Geradensteigung und Schnitt mit der Ordinaten zu ermitteln. Gnuplot geht hierbei iterativ vor und variiert die Parameter bis eine Funktion (häufig das χ^2), die die Qualität der Anpassung beschreibt, minimal wird. Aus den Parametern und dem Wert der Funktion kann man sagen, ob man hier ein gutes Modell hat, oder nicht. Neben dem Test einer Theorie kann ein Fit auch einfach dazu verwendet werden, aus einer Messung unbekannte Parameter durch Anpassen an ein geeignetes Modell zu bestimmen. Häufig ist das eine elegante und genaue Messmethode.

Nun wollen wir uns den Anwendungsmöglichkeiten von Gnuplot zuwenden. Hoffnungsvollerweise versteht ihr jetzt, warum Plots und Fits für uns Physiker so wichtig sind.

5.2 Das Datenformat

Wenn ihr eine Datei mit Messdaten durch `plot 'dateiname.dat'` plotten wollt, benötigt Gnuplot die Daten in einem lesbaren Format. Grundsätzlich ist eine einfache Textdatei erforderlich, wobei in einer Zeile die Werte der Spalten getrennt durch einen Freiraum oder durch einen Tabulator enthalten sind. Für einen zweidimensionalen Plot nimmt man üblicherweise an, dass nur die Funktionswerte

fehlerbehaftet sind. Man hat also im Prinzip drei Spalten und so viele Zeilen wie Messpunkte genommen wurden. Kommentare werden mit der Raute # eingeleitet und von Gnuplot beim Einlesen der Daten ignoriert. Es ergibt sich also eine Datei mit zwei Datenpunkten wie z.B.

```
# x y Fehler von y
  1.0 2.3 0.1
  2.0 5.2 0.2
```

Nachkommastellen werden mit einem Punkt abgetrennt. Bei der Deklaration von Variablen ist dies sogar unumgänglich, da Gnuplot diese sonst nicht richtig verarbeitet. Die wissenschaftliche Schreibweise ist in Gnuplot mit einem nachgestellten e zugänglich (z.B `3.5e-22`). Wenn man zwei Datensätze in einer Datei hat und sich wünscht, dass die unterschiedlichen Datensätze nicht mit einer Linie verbunden werden sollen, sollte man die Datensätze mit zwei Zeilenumbrüchen voneinander trennen.

Häufig werden die Datensätze vorher oder nachher mit einer Tabellenkalkulation, wie z.B. dem freien LibreOffice Calc eingelesen und gespeichert. Hierbei sollte man den Datensatz als Text CSV (*.txt;*.csv) laden oder speichern. Wenn man unter Linux arbeitet, kann es hilfreich sein, aus zwei Dateien mit zwei Spalten eine Datei mit vier Spalten zu machen. Probiert dafür den Befehl `paste datei1.dat datei2.dat > mix.dat` aus. An dieser Stelle sei noch einmal auf den bereits zuvor erwähnten gewinnbringenden Einsatz des emacs oder des vims und der Arbeit mit Blöcken hingewiesen, die Zeit und Nerven vor der Abgabe spart.

5.3 Rahmendateien

Eine Rahmendatei[1] wird mit dem Befehl `load 'Dateiname.txt'` geladen und arbeitet alle Kommandos der Reihe nach ab. Auch in Rahmendateien lassen sich einzelne Befehle mit der Raute # auskommentieren. Wenn man ein Kommando auf mehrere Zeilen aufteilen möchte, verwendet man für den Zeilenumbruch den Befehl `\`.

5.4 Gnuplot – so geht's los

Gnuplot ist für verschiedene Betriebssysteme frei unter `http://www.gnuplot.info/` verfügbar. Auch die Windowsfassung lässt sich nur über die mitgelieferte Kommandozeile steuern. Unter Linux reicht der Aufruf `gnuplot`.

Der erste wichtige Befehl, den es sich zu merken lohnt, ist `help` und ruft die Hilfe auf. Ganz unten auf der Seite findet ihr die verfügbaren „Help topics“. Falls ihr bereits einen Begriff kennt, zu dem ihr Hilfe benötigt, dann könnt ihr ihn einfach hinter den Hilfebefehl schreiben, wie zum Beispiel `help plot`.

Die Hilfeseite könnt ihr wieder mit `q` verlassen. Ihr werdet dann evtl. noch einmal nach einem weiteren Hilfethema (`Help topic`) gefragt, was ihr durch Drücken der Eingabetaste umgehen könnt. Aussteigen könnt ihr mit den Befehlen `exit` oder `q`. Hoffentlich startet ihr Gnuplot gleich nochmal und plottet und fittet, was das Zeug hält.

[1] Siehe Kap. 5.7.

5.5 Plotten

5.5.1 Plot

Es gibt verschiedene Arten von Plots bei Gnuplot. Die wichtigsten sind `plot` für zweidimensionale und `splot` für dreidimensionale Plots. Haben wir nun unsere Messdaten z.B. in der Datei `daten.dat`, so können wir diese mit dem Aufruf `plot 'daten.dat'` graphisch auftragen. Die fehlerbedingten Abweichungen vom Funktionswert sollten mit Fehlerbalken in den Plot eingetragen werden. Dies lässt sich mit dem Zusatz `with errorbars` verwirklichen. Gnuplot nimmt im Aufruf `plot 'daten.dat' with errorbars` die Fehler aus der dritten Spalte.

Wenn die Spalten des Datensatzes anders organisiert vorliegen, kann man Gnuplot dies mitteilen, indem man die richtige Reihenfolge mit `using 2:3:4` angibt. Die Zahlen kennzeichnen die Nummer der Spalte von links nach rechts. Im vorliegenden Fall lägen die x-Werte in der zweiten Spalte, die y-Werte in der dritten Spalte und die Fehler der y-Werte in der vierten Spalte. Insgesamt startet man den Plot also mit `plot 'daten.dat' using 2:3:4 with errorbars`. Mit `using 1:(log10($2))` lassen sich dabei die einzelnen Werte auch umrechnen. Wichtig ist hier und bei allen anderen Umrechnungen, dass sowohl die Spalte mit z.B. `$2` angegeben, als auch die gesamte Umrechnung in Klammern gefasst werden muss. Beachtenswert wäre hier zusätzlich, dass Gnuplot den dekadischen Logarithmus mit `log10()` und den natürlichen Logarithmus mit `log()` kennzeichnet.

Ob Gnuplot nur Punkte zeichnet, wie bei den meisten Messungen empfohlen, oder eine Linie durchzeichnet, wie bei Spektren oder anderen hochaufgelösten Datensätzen üblich, kann durch `plot f(x) with lines` oder `plot f(x) with linespoints` gesetzt werden.

5.5.2 Legende und Gitternetzlinien

Um Messreihen besser vergleichen zu können, plottet man diese mit z.B. `plot 'daten1.dat', 'daten2.dat'` gemeinsam in ein Diagramm. Eine Legende dient der Übersichtlichkeit und hilft, die Messreihen wiederzufinden. Grundsätzlich ist die Legende eingeschaltet und beinhaltet den Dateinamen bzw. den Funktionsnamen. Man kann allerdings die Beschriftung mit `plot f(x) title 'Funktion'` setzen. Standardmäßig befindet sich die Legende innerhalb des Diagramms. Häufig verdeckt dies aber gerade einen interessanten Abschnitt oder sieht einfach weniger schön aus. Mit `unset key` schaltet man die Legende aus oder setzt sie mit `set key outside` oder ähnlichen Angaben an den gewünschten Ort.

Das Ablesen von Werten wird durch Gitternetzlinien deutlich erleichtert. Das Gitter wird mit `set grid` aktiviert.

5.5.3 Wertebereich

Gnuplot versucht automatisch den bestmöglichen Achsenbereich zu wählen. Manchmal möchte man allerdings einen anderen Bereich vergrößern oder hat andere Variablen als x, y und z. Gnuplot bietet zwei Möglichkeiten der Umsetzung an. Zum einen lässt sich der Bereich und ggf. die Variable direkt vor dem Plotbefehl angeben wie in `plot [t=0:10.0][10.0:*] f(t)`. Das Sternchen symbolisiert, dass man den Wert nicht kennt und wieder automatisch setzen lassen möchte. Alternativ kann der jeweilige Wertebereich auch mittels `xrange [0:1.0]; yrange [1.0:10.0]` angegeben werden.

5.5.4 Achsenbeschriftung

Für den Leser eurer Arbeit ist es von entscheidender Bedeutung, dass er oder sie versteht, wie die angefertigte Graphik zu lesen ist. Grundsätzlich gilt, dass die Achsenbeschriftung unter Angabe der Einheiten zu gestalten ist. Es gibt natürlich auch einheitenlose Darstellungen, wie z. B. die logarithmische Darstellung. Titel und Beschriftungen setzt man wie folgt:

```
set title 'Sinnvolle Bezeichnung des Plots'
set xlabel 't [s]'
set ylabel 'x [m]'
```

Zusätzliche Beschriftungen an einer Position in den Koordinaten des Plots sind durch `set label at 1.5,1.5` möglich.

Die Abstände der Markierungen und Gitternetzlinien, die sich mit `set grid` setzen lassen, können auch manuell verändert werden. Die Abstände der Markierungen werden mit `set xtics Schrittweite` gesetzt und die Anzahl der kleinen Unterteilungen legt man mit `set mxtics Zwischenschritte` fest. Man beachte, dass die Anzahl der Zwischenschritte unabhängig von der Schrittweite der Markierungen ist. Die obere x-Achse und die rechte y-Achse lassen sich beispielsweise mit `set x2tics 0.5` und `set y2tics 1.0` setzen.

5.5.5 splot

Dreidimensionale Plots bei denen $f(x, y) = z$ dargestellt wird, lassen sich mit `splot` realisieren. Auch hierbei können sowohl Datensätze als auch Funktionen geplottet werden. Ein Beispiel wäre `splot sin(x)*y`. Der Plot lässt sich mit der linken Maustaste drehen und mit der mittleren Maustaste zoomen. Gnuplot beinhaltet auch eine Farbdarstellung. Mit `set pm3d at s` wird gemäß dem Funktionswert ein zugehöriger Farbwert zugewiesen.

5.5.6 Multiplot

Immer wenn mehrere Plots ineinander verschachtelt, nebeinander oder übereinander gezeigt werden sollen, muss dieses kleine Feuerwerk von vielen Plotbefehlen mit `set multiplot` eingeleitet werden. Mit `set size` und `set origin` könnt ihr eurer kreativen Ader freien Lauf lassen und die einzelnen Plots unter ästhetischen Gesichtspunkten proportionieren. Damit das Ganze dann auch noch richtig bunt vor euren Augen explodiert, erfordert Gnuplot den Abschluss der `Multiplot`-Umgebung mit `unset multiplot`.

5.5.7 Abkürzungen

Wie häufig im Leben gibt es für vieles einen kürzeren Weg. Für einige Gnuplotbefehle reicht ein einzelner Buchstabe. Einige gebräuchliche Abkürzungen sind in Tabelle 5.1 zu finden.

5.5.8 Speichern

Um aus einem Plot eine fertige Grafikdatei zu bekommen, erfordert es in Gnuplot drei Zeilen. Die erste Zeile sagt, welches Format die Datei haben soll. Die gängigsten Graphikformate sind svg, png, postscript und PDF. Gesetzt wird das Format mit `set terminal svg`, `set terminal png`, `set terminal postscript` und `set terminal pdfcairo`. Da es sonst ja zu einfach wäre, gibt

Gnuplotausdruck	Abk.
plot	p
splot	sp
with	w
using	u
title	t
lines	l
points	p
linespoints	lp
errorbars	e
terminal	term

Tabelle 5.1: Gnuplot Abkürzungen

es bei praktisch allen Terminals noch Optionen. Der Zusatz `color` sorgt zum Beispiel meist dafür, dass ein farbiger Plot erzeugt wird.

Damit Gnuplot weiß, in welche Datei geschrieben werden soll, gibt man mit `set output 'datei.svg'` den Dateinamen und die jeweilige Endung an. Schließlich muss noch einmal geplottet werden, damit Gnuplot auch in die zugehörige Datei schreibt. Um den ganzen Plotbefehl nicht noch einmal eingeben zu müssen, reicht `replot` und schon ist das Bild fertig. Bei einigen Versionen kann es sein, dass nach Wechsel des Terminals und mehreren Plotbefehlen mehrere Seiten geplottet werden. Um wieder zurück in den normalen Plotmodus zu gelangen, verwendet man `set terminal wxt` unter Linux bzw. `set terminal windows` unter Windows. Insbesondere unter Windows schafft es Gnuplot häufig nicht richtig, die Datei zu schreiben, wenn man in ein anderes Terminal wechselt. Ein Neustart von Gnuplot ist dann hilfreich.

Für ein Bild auf einer Website bietet sich `png` an, für die manuelle Nachbearbeitung mit Inkscape (siehe Kap. 8.1.1) `svg` und um es direkt auszudrucken `postscript landscape enhanced color rounded`

Die Verbindung von Gnuplot mit Latex wird weiter unten noch einmal aufgegriffen.

```
 # Beispiel für das Speichern mit Gnuplot
set terminal png
set output 'plot.png'
replot
```

5.6 Fitten

Beim Fitten sind zwei Fälle zu unterscheiden: lineare und nichtlineare Fits. Ein linearer Fit ist nichts anderes als eine Ausgleichsgerade und funktioniert immer. Deswegen sollte man immer versuchen sein Problem zu linearisieren, z.B. durch Logarithmieren der Messwerte. Sollte dies nicht möglich sein, kann man auch einen beliebigen Ausdruck anpassen, was unter Umständen lange dauern kann oder gar zu einer unpassenden oder falschen Lösung führt. Dies geschieht meist, wenn man schlechte Startwerte vorgegeben hat.

Um einen Datensatz an eine Geradengleichung `f(x)=m*x+b` anzupassen, verwendet man den Befehl `fit f(x) 'daten.dat' via m,b`. Gnuplot versucht dann die Parameter so zu bestimmen, dass die Funktion möglichst gut an die Messwerte angepasst ist. Auch beim Fitten kann

man den Wertebereich einschränken und mit `using` die verwendeten Spalten und ggf. auch die zu berücksichtigenden Fehlerbalken angeben. Durch Verwendung des Befehls `set fit logfile 'datei.log'` wird ein Logbuch für den zugehörigen Fit angelegt und die Ergebnisse dort hineingeschrieben. Dort findet sich auch der Wert `reduced chisquare`, der ein Maß für die Güte der Anpassung ist. Liegt er nahe bei 1.0 so ist der Fit perfekt. Ist er viel größer, so wurden die Fehler unterschätzt, man verwendet ein falsches Modell oder die Statistik des Fehlers ist falsch. Ähnliches gilt für den Fall, dass der Wert sehr viel kleiner ist als 1.0, wobei die Fehler hierbei eher überschätzt wurden. Die Parameter werden stets mit Fehlern ausgegeben, die immer auch mit angegeben werden sollten.

Beim nichtlinearen Fitten benötigt man eine Vorstellung von der Größe der Parameter, die man anpasst. Ein Beispiel für einen nichtlinearen Fit wäre eine Exponentialfunktion. Gibt man z.B. `fit a*exp(b*x) 'daten.dat' via a,b` ein, so werden die Parameter `a` und `b` an die Messdaten angepasst. Es werden also die Parameter `a` und `b` gesucht, die optimal sind, um diese Messpunkte mit dieser Exponentialfunktion zu beschreiben. Es ist möglich, dass der Fit in einem lokalen Minimum „hängen bleibt“. Deswegen sollte man das Ergebnis auf Plausibilität prüfen und ggf. den Fit mit leicht veränderten Werten wiederholen. Die Parameterwerte voriger Generationen von Praktikanten oder die Literaturwerte können bei der Wahl der Startwerte für die Parameter hilfreich sein. Die Hilfedatei von Gnuplot meint dazu ganz richtig *Nonlinear fitting is an art*.

Um dem Betrachter eine faire Chance zu geben sich ein Bild von der Güte des Modells und der Messung zu machen, plottet man gerne (s. `Multiplot`) die sogenannten Residuen unter Messdaten und Modell. Hierbei wollen wir unter Residuen einfach nur die Differenz zwischen Modellfunktion und Messwerten verstehen. Wenn alles gut gelaufen ist, sind die Differenzen wie mit dem Salzstreuer um die Nulllinie verteilt. Verabschieden sich die Residuen von diesem Ebenmaß, ist es an der Zeit den Bleistift zu spitzen und sich Gedanken über Messung und Modell zu machen, denn mindestens eines von beiden ist nicht richtig - oder besser gesagt im Bereich der Abweichung nicht richtig gut.

5.7 Rahmendatei

Nun haben wir viele nützliche Befehle und Befehlskombinationen kennengelernt. Es wäre aber umständlich, jedes Mal alles neu hintereinander einzugeben. Deswegen ist es ratsam, ein kurzes „Skript“, eine *Rahmendatei* zu erstellen, in das man viele Befehle reinschreibt und diese bei Bedarf ändert, anpasst oder auskommentiert. Dies könnte z.B. wie der folgende Abschnitt aussehen. In der Datei `beispiel.plot` haben wir folgenden Text abgespeichert:

beispiel.plot

```
1  #Hier den gewünschten Namen für den Plot eingeben
2  set output 'dateiname.png'
3
4  # Hier steht im Wesentlichen, wie der plot ausgegeben werden soll,
5  # Größe, Farbe, Schrift usw.:
6  set terminal png large enhanced
7
8  # Hier kommt die logarithmische/nichtlogrithmische Skala, bei Bedarf
9  # bitte auskommentieren. Und andere Spielereien
10 #set logscale x
11 #set logscale y
12
13 set nologscale x
14 set nologscale y
15
16 #set xrange [0:100]
17 #set yrange [-10:10]
18
```

```
#set grid
#set nogrid


# Hier kann man die Achsenbeschriftung wählen:
set xlabel 'Zeit (in s)'
set ylabel 'Entfernung (in m)'

# Hier kann man sich eine Funktion definieren (Potenz wird z.B. mit
# x**3 ausgedrückt
#f1(x)=m*x+b
#f2(x)=sin(x**2)
f3(x)=a*exp(b*x)

# Hier können diese Funktionen an die Messdaten gefittet werden, wobei
# daten.dat die Datei ist, in der die Messwerte in Spalten eingetragen
# sind. Die Angabe (sqrt($3)) würde z.B bedeuten, dass von der dritten
# Spalte die Wurzel gebildet werden soll. Gleichzeitig ist es auch der
# Fehler im zweidimensionalen Plot, der in den Fit einbezogen wird
fit f3(x) 'daten.dat' using 1:2 via a, b

# Zum Schluss noch die Auftragung:

plot f3(x) title 'Entfernung der Rakete', \
'daten.dat' title 'messwerte' \
w errorbars

# Und eine erneute Ausgabe auf dem Bildschirm
set output
set terminal wxt

replot
```

beispiel.plot

5.8 Das Ergebnis

Die Messwerte für unser Beispiel Exponentialgesetz sind in der Datei `daten.dat` gespeichert und sehen dort so aus:

daten.dat

```
1       3       0.2
2       3.1     0.2
3       3.2     0.3
4       3.2     0.3
5       3.5     0.3
6       4       0.4
7       5       0.4
8       7       0.4
9       10      0.7
10      14      0.7
11      18      0.6
12      26      0.8
13      36      1.1
14      50      3
15      70      2
```

daten.dat

Unser Beispiel ergibt dann folgendes Ergebnis:

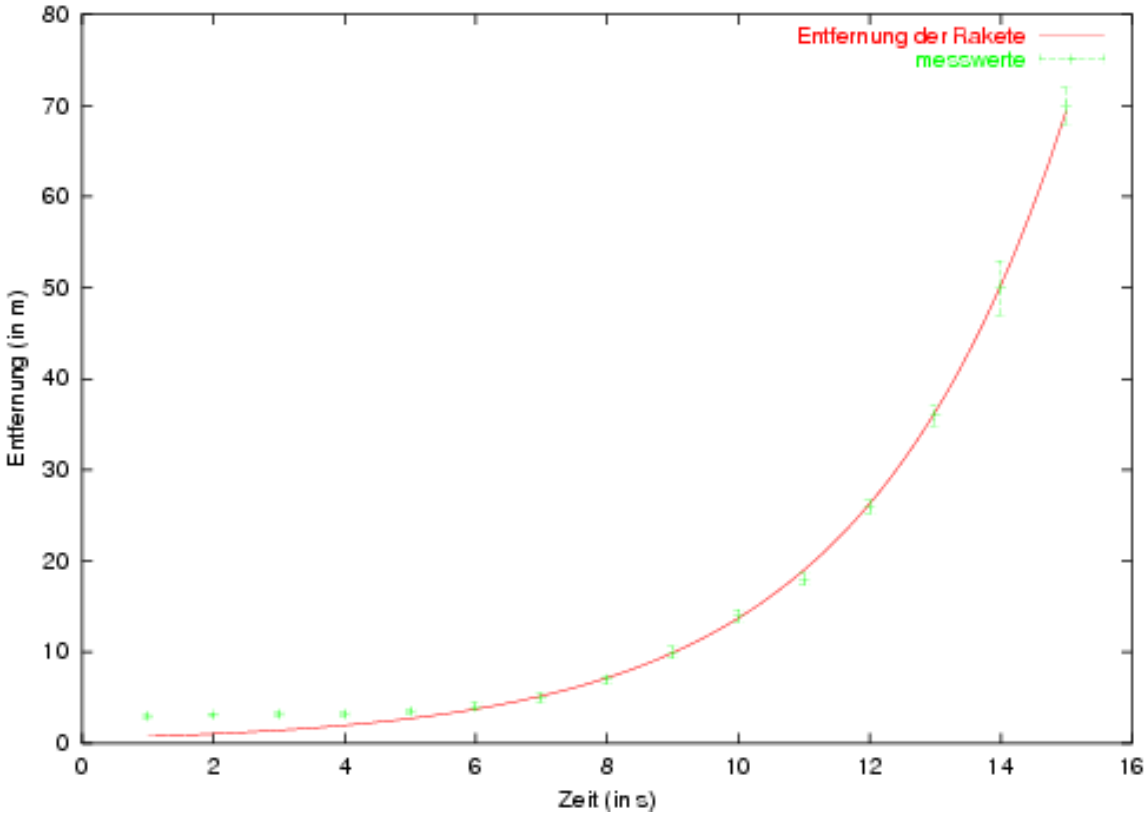


Abbildung 5.1: Das Ergebnis unserer Beispieldateien.

5.9 Verbindung mit Latex

Das kröhnende Lebensende eines Plots sollte ja möglichst das gerade erstellte Praktikumsprotokoll sein. Da wäre es doch schön, wenn der Plot die gleiche Schriftart verwenden würde, wenn man Latex-Formeln im Plot verwenden könnte und das ganze auch noch möglichst einfach wäre. Und das geht sogar. Dafür verwendet man am besten das Terminal `epslatex` mit der Option `color`. Bevor wir die kleinen Fallstricke erläutern, ganz kurz die Funktionsweise: Alle Texte werden von Gnuplot mit Postionsangaben in einer `.tex`-Datei gespeichert und nur der eigentliche Graph und die Achsen als Bild eingebunden. Daher werden auch zwei Dateien erzeugt. Und hier lauert das erste kleine Problem. Die zweite der beiden Dateien wird erst beim Beenden von Gnuplot erstellt. Plottet man nur mit den im vorherigen Abschnitt gezeigten Rahmendateien, stellt dies aber kein Problem dar.

Wie angedeutet, können in allen Textfeldern (also Funktionennamen, Achsenbeschriftungen usw.) Latex Befehle verwendet werden. Es ist nur zu beachten, dass man `\` als `\\` schreiben muss. Die letzte Hürde ist das Bildformat. `epslatex` erstellt die Grafik als `eps`-Datei, wir brauchen aber ein PDF. Das lässt sich aber schnell mit `epstopdf bild.eps` erledigen. Aber auch das kann man Gnuplot überlassen, indem man den Befehl mittels `!` als externes Kommando aufruft.

Nun steht der Einbindung des gerade geplotteten Bildes mittels `\input{bild}` (ohne Endung) nichts mehr im Wege. Im Folgenden ein Beispiel, dessen Ergebnis in Abb. 5.2 zu bewundern ist.

beispiel.plot

```
reset
set terminal epslatex color
set xlabel '$t$ [s]'
set ylabel '$|\vec{F}|$ [N]'
set key top right

set output 'beispiel.tex' # wie in C++ datei.open()
plot [0:5*pi] sin(20*x)*sin(x) title \
        '$A\sin(t)\cdot\sin(\omega t)$' lw 2 lt 1

set output              # wie in C++ datei.close()
!epstopdf beispiel.eps  # eps-Datei nach pdf umwandeln

# Einbindung in Latex am besten mit
# \begin{figure}[htb]
#     \centering
#     \input{beispiel}
#     \caption{Lustiger Beispielplot}
# \end{figure}
```

beispiel.plot

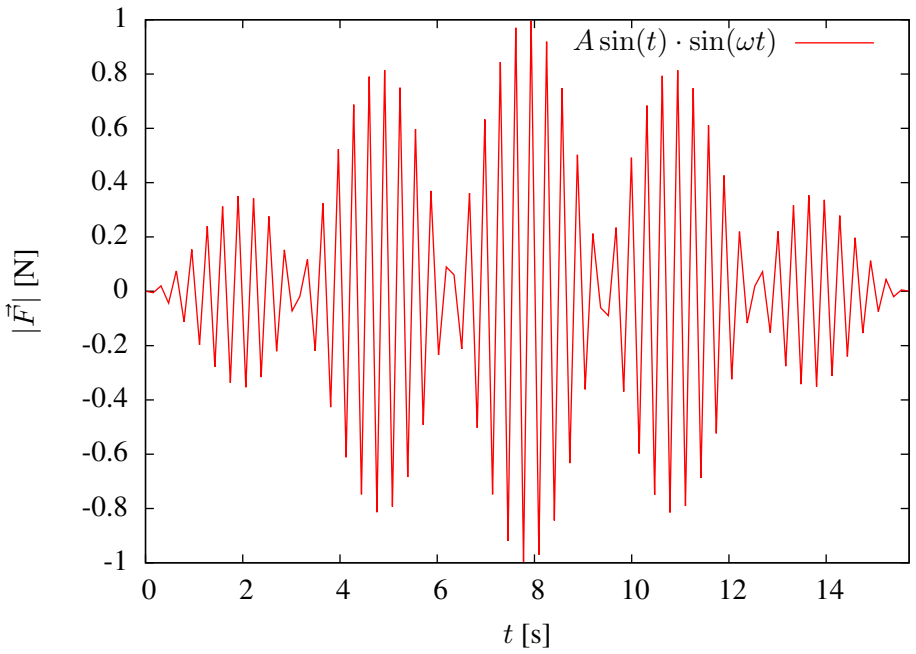


Abbildung 5.2: Lustiger Beispielplot

6 Subversion Revision Control

6.1 Motivation und Einführung

Teamarbeit an Quellcode-Projekten erfordert eine starke Koordinierung in der Gruppe. Sind die Aufgaben strikt verteilt und arbeitet jeder nur an einem Modul, so ist die Verteilung / Zusammenführung der Komponenten noch „zu Fuß“ zu bewerkstelligen. Doch wollen mehrere Personen an einer Quelldatei arbeiten, wird es schwierig. Entweder man arbeitet an verschiedenen Arbeitsumgebungen und tauscht die Datei regelmäßig aus – dies lässt aber keine gleichzeitige Änderung der gleichen Datei zu – oder man arbeitet über Netzwerk an einer Arbeitskopie des Quellcodes, hat dann jedoch das Problem, dass bei einem Fehler eines Teammitgliedes zum Beispiel der Code von anderen versehentlich gelöscht werden könnte.

6.1.1 Revision Control System – Was ist das?

Die Lösung für dieses Problem ist ein Revision Control System. Eine solche (auch Versionsverwaltung genannte) Software verwaltet auf einem zentralen Server ein Projektarchiv mit den Quellcodedateien. Das Wichtige ist, dass nicht nur die aktuellen Dateiinhalte gespeichert sind, sondern die gesamte Versionshistorie. Dadurch ist es möglich, Veränderungen zwischen der aktuellen und einer früheren Version (oder zwischen zwei früheren Versionen) zu betrachten und evtl. rückgängig zu machen. Dies ist ein sehr nützliches Feature, wenn man bei zu vorschneller Bug-Behebung nur noch mehr Fehler produziert hat.
Jeder Projekt-Mitarbeiter holt sich zu Beginn den aktuellen Stand der Serverkopie. Nun kann er an seiner Arbeitsversion beliebige Veränderungen vornehmen, ohne dadurch an der Hauptkopie etwas kaputt zu machen. Ein Standardverfahren ist, die Serverkopie immer kompilierfähig zu halten, damit jeder, der sich den Quellcode neu herunterlädt, diesen kompilieren und ausprobieren kann.
Die eigene Arbeitsversion wird verwendet, um dort Veränderungen und Neuerungen einzubauen. Anschließend beseitigt man alle Compilerfehler und kann dann nach dem Kompilieren seines Dokuments bzw. des Programms[1] nach Laufzeitfehlern suchen. Hat man die volle Funktionsfähigkeit hergestellt, schickt man dem Server seine Änderungen. Dieser erhöht die Versionszahl der Dateien (d.h. er speichert den aktuellen Stand als eine ältere Version) und pflegt die veränderten Teile in die aktuelle Version ein. Greifen nun andere auf die Projektdateien zu, so haben sie die (idealerweise auch bei ihnen voll lauffähige) erneuerte Version des Programms.
Dies klingt ja sehr toll, aber was kann alles schief gehen?

6.1.2 Konfliktmanagement

Es können an mehreren Stellen Konflikte auftreten, mit denen die Revision-Control-Software umgehen kann. Hier eine kleine Auflistung der Probleme:

- Sind neue Dateien und Ordner zu dem Projekt hinzugekommen, so ist dies kein Problem; das Projektarchiv wird einfach um diese Strukturen erweitert (mit leerer Historie).

[1] Versionskontrolle ist universell: Es kann, muss sich aber nicht um Latex-Dokumente oder C/C++-Programme handeln.

- Hat sich eine Datei verändert, so ist idealerweise die vom Server als aktuell angesehene Version gleich der ursprünglichen Version in der Arbeitskopie (also der vor dem Editieren). Dann können einfach alle Änderungen problemlos übernommen werden. Wäre dies immer der Fall, bräuchte man kein Versionsmanagement. Folgende Fälle sind wichtig:
  - Die Datei wurde zwischenzeitlich von jemand anderem verändert und aktualisiert. Sind seine Änderungen unabhängig von den eigenen, so muss dies die Versionsmanagement-Software erkennen und nur die Neuerungen an den richtigen Stellen einpflegen (Die Zeilennummerzuordnung ist ja evtl. verschoben worden).
  - Trickreicher ist der Fall, bei dem in der Zwischenzeit eine Änderung am gleichen Codefragment vorgenommen wurde. Erkennt dies die Software, bleibt ihr meist nichts anderes übrig, als die Neuerung zunächst zurückzuweisen und dem Entwickler von diesem sogenannten Versionskonflikt zu berichten[2].
- Es gibt noch eine riesige Menge weiterer Spezialfälle, beispielsweise wenn ein anderer Entwickler Code auf einen älteren Stand als den Stand der Arbeitskopie zurückgesetzt hat. Aber das wollen wir uns alles erstmal gar nicht ausmalen.

6.2 Subversion

Wir möchten euch an dieser Stelle Subversion (svn) vorstellen. Es ist ein zur Zeit beliebtes Versionsverwaltungssytem, ist Open-Source und wurde seit 2000 von CollabNet entwickelt und ist mittlerweile ein Projekt der Apache Software Foundation. Ihr findet das Projekt auf `http://subversion.apache.org`.
Unter Linux kann es praktisch immer direkt mit dem Paketmanager der Distribution installiert werden. Bei Ubuntu funktioniert es beispielsweise mittels `sudo apt-get install subversion`. Im CIP ist `subversion` bereits installiert.

Als weiteres Versionsverwaltungssystem ist noch git zu nennen; es verfolgt im Gegensatz zu svn das etwas anderes Konzept einer verteilten Versionsverwaltung (ohne dominanten Zentralserver). Es hat zudem in den Augen vielen Nutzer weitere Vorteile, die sich aber durch eine kompliziertere Bedienung erkauft werden. Das Projekt wird unter der Leitung von Linus Torvalds für die Arbeit am Linux-Kernel entwickelt. Neben dem Linux-Kernel werden viele weitere Projekte mit Hilfe von git entwickelt.

6.2.1 Repositories

Ein von Subversion koordiniertes Projekt wird Repository genannt. Dies ist das Archiv aller erstellten, veränderten, auch überschriebenen und gelöschten Dateien und Verzeichnisse.

6.2.2 Arbeiten mit svn

Beziehen einer Arbeitskopie

Wir gehen der Einfachheit halber davon aus, dass das Repository auf einem Webserver unter `https://domain.de/svn/repo/` liegt[3]. Wollt ihr eine Arbeitskopie ins aktuelle Verzeichnis herunterladen, müsst ihr einen *checkout* des Repositories machen:

[2] Nähere Details zur Konfliktroutine findet ihr in Abschnitt 6.2.3.
[3] Wie es dort hinkommt, sei erstmal egal. Wie ihr so etwas ohne großen Aufwand bekommt, steht in Abschnitt 6.3.

`svn checkout https://user@domain.de/svn/repo/` oder kürzer
`svn co https://user@domain.de/svn/repo/`

Beachtet das `user@`: Da sollte euer Benutzername stehen. Ihr werdet nach dem zugehörigen Passwort gefragt[4] und dann beginnt das Herunterladen des aktuellen Stands.
Nun könnt ihr nach Belieben an den Daten herumdoktern.

Veränderungen hochladen

Nach einer sinnvollen Änderung schickt man diese mittels

`svn commit` oder `svn ci`

an den Server. Es ist auch möglich, nur einzelne Dateien hochzuladen: `svn commit datei.ext`.

Bei jedem Einchecken von Inhalten wird `svn` einen Editor[5] öffnen. Der Text, den man hier schreiben soll, ist als Hinweis für andere Entwickler gedacht und landet im Revisionslog (siehe Kap. 6.2.4). Der Text sollte folglich kurz und knapp, aber verständlich die gemachten Änderungen beschreiben.

Hinzufügen von Dateien Neu im Ordner angelegte Dateien werden nicht automatisch in das Repository übernommen. Das ist in vielen Fällen auch erwünscht, da Kompilate und Zwischendateien nur für zusätzlichen Traffic und belegten Plattenplatz auf dem Server sorgen, aber jederzeit aus den Quellen generiert werden können. Daher synchronisiert man üblicherweise nur die Quelldateien. Ein explizites Hinzufügen einer Datei in das Archiv lässt sich mit

`svn add <dateien>`

bewerkstelligen. `<dateien>` kann eine Datei, eine durch Leerzeichen getrennte Liste von Dateien oder ein Platzhalter, wie `*.c` sein. Beachtet: Sie sind jetzt nur zum Einchecken markiert. Tatsächlich auf dem Server landen die Dateien erst nach einem `commit` (s.o.).

Umsortieren vorhandener Dateien Wir haben gesehen, dass die Ordnerstruktur nicht identisch mit der Archivstruktur ist. Damit `svn` Umstrukturierungen innerhalb der "working copy" mitbekommt, muss man beim Verschieben und Kopieren von Dateien im Repository nun `svn` statt der üblichen Konsolenbefehle verwenden. Das geht aber quasi genauso:

```
svn move foo.c bar.c
svn copy original.tex kopie.tex
svn delete brauch-man-nich.bakup
```

Arbeitsverzeichnis synchronisieren

Wollt ihr eure Arbeitskopie an den Serverstand anpassen (weil andere dort etwas verändert haben könnten), so tut dies folgender Befehl:

`svn update` oder `svn up`

Vorausgesetzt ihr seid in einem Verzeichnis eurer Kopie des Repositories (`svn` merkt sich die Quelle

[4] Evtl. müsst ihr vorher noch das Zertifikat des Servers verifizieren. Am besten vertraut ihr dem Zertifikat permanent.
[5] Meist ist dies ein Konsoleneditor wie `emacs`, `vim` oder `nano`. Man kann ihn z.B. über die Shell-Variable EDITOR ändern.

in den Konfigurationsdateien, daher ist keine weitere Angabe des Servers nötig.), werden die Daten des aktuellen Verzeichnisses und aller Unterverzeichnisse mit denen des Servers verglichen und wenn nötig neu heruntergeladen.

6.2.3 Versionskonflikte

Grundsätzliches Vorgehen

Nimmt man Änderungen an einer working copy mit dem Ziel vor, diese wieder an andere zu verteilen, so ist das Vorgehen grundsätzlich wie folgt:

- Aktuelle Version holen (`svn update`)
- Änderungen vornehmen
- Holen eventueller Veränderungen (`svn update`)
  - Konfliktbehebung, falls nötig (siehe unten)
  - Danach `svn` Konfliktfreiheit mitteilen (`svn resolved`)
- Hochladen der Neuerungen (`svn commit`)

Umgang mit Versionskonflikten

Betrachten wir einmal eine beispielhafte Ausgabe von `svn update`:

```
> svn update
U  INSTALL
G  README
C  foobar.c
Updated to revision 46.
```

Wir haben uns Version 46 geholt. Dabei wurden `INSTALL` erneuert (`U`), Änderungen in `README` erfolgreich zusammengefügt (`G` wie merge) und `foobar.c` mit einem Konflikt (`C`) zusammengefügt. Letzteres passiert im Grunde nur, wenn zwei Personen die gleichen Quelltextzeilen bearbeitet haben. Arbeitet man an verschiedenen Bereichen eines Dokuments, klappt der merge praktisch immer reibungslos.

Lösen von Konflikten Im Falle eines Konflikts fragt das Tool `svn update` nach dem weiteren Vorgehen. In der entsprechenden Datei wird Subversion den merge so gut es geht durchführen. Nun kann man dies von Hand anpassen und den Konflikt gleich freigeben, oder man stellt die Konfliktlösung zurück.
In diesem Fall werden mit dem Update drei zusätzliche Dateien angelegt:

foobar.c.mine : Datei aus eurer “working copy” mit euren Änderungen.

foobar.c.r45 : Die alte Serverversion (vorherige Revision 45, allgemein die Revision der letzten Änderung).

foobar.c.r46 : Die neue Serverversion (aktuelle Revision 46).

Nun kann man alle vier Dateien in einem Editor seiner Wahl öffnen und die nach eigenen Vorstellungen zusammengeführte Datei nach `foobar.c` schieben. Hilfreich ist hierbei unter Umständen das Programm `meld`. Abschließend ruft man `svn resolved foobar.c` auf, um `svn` die Lösung des Konflikts anzuzeigen. Ein abschließender `commit` schickt den neuen Inhalt zum Server.

6.2.4 Informationen über das Archiv

Log

Das Log mit allen Änderungstexten (also den Freitexten, die man beim `commit` mitgegeben hat) ist mit dem Befehl `svn log` wiederzufinden. Mit dem Switch `-v` werden alle modifizierten Dateien zu jedem Eintrag angezeigt. Man kann das Log auf eine Auswahl von Revisionen (`svn log -r 5:19`) oder auf einzelne Dateien (`svn log foo.c`) beschränken.

Dateistatus

Als letzten sehr nützlichen Befehl wollen wir `svn status` anführen. Mit diesem Befehl kann man sich anzeigen lassen, welche Dateien des aktuellen Verzeichnisses wirklich zum Archiv gehören und welchen Modifikationsstatus diese besitzen. Da die Liste mit dem Statusmarkern sehr lang ist, beschränken wir uns hier auf zwei Beispiele:

```
> svn status
M   abc.c           # lokale Aenderungen vorgenommen
C   conflict.tex    # diese Datei ist konfliktbehaftet
```

Die anderen Marker können mit `svn help status` nachgelesen werden. Diese nützliche Hilfe gibt es übrigens für alle Befehle. Mit `svn diff abc.c` kann man sich die lokalen Änderungen (gegenüber der letzten Serverversion) anzeigen lassen.

6.3 Kollaborationsunterstützung an der Physik-Fakultät

Das Lehrportal (`lp.uni-goettingen.de`) bietet für jeden Physik-Studierenden (bzw. Teams, die sich aus solchen zusammensetzen) die Einrichtung eines svn-Repository und eines Wikis an. Wie man sich das einrichten lassen kann, steht unter `http://lp.uni-goettingen.de/blogs/lp/category/lp-collaboration`.
Für das Repository lassen sich mehrere Benutzernamen mit Zugriff auf das System anlegen. Der Lehrportal-Server ist von überall erreichbar. Die Verbindung ist verschlüsselt und nutzt HTTPS mit Adressen der Form `https://lp.uni-goettingen.de/collaboration/svn/<name>/`.

6.4 Einen eigenen SVN-Server anlegen

Anlegen eines Archivs

Hat man Zugriff auf einen Rootserver[6], kann man sich so ein Repository selbst anlegen. Man ruft einfach `svnadmin` mit dem Verzeichnis auf, in dem die Serverkopie verwaltet werden soll:

```
svnadmin create --fs-type fsfs /path/to/repos
```

`/path/to/repos` ist danach nicht leer, da es nicht nur die Projektdateien, sondern auch Konfigurations- und Verwaltungsdateien enthält. Es ist nicht zum Arbeiten gedacht. Auch Benutzer, die auf dem Rechner, der als Server dient, direkt arbeiten, müssen sich immer eine Arbeitskopie beschaffen[7].

[6] Oder man möchte ein privates Repository für die eigene Arbeit pflegen, was ab und an extrem praktisch ist. Der *Server* kann dabei durchaus der eigene Arbeitsrechner sein.
[7] Wie das geht, steht in Abschnitt 6.2.2.

Mit Inhalt füllen Nun legt man sich in einem temporären Ordner (beispielsweise dem Ordner der späteren Arbeitsversion) die Ordnerstruktur des Projektes an. Das kann zum Beispiel so aussehen:

```
mkdir project
mkdir project/code
mkdir project/interface
mkdir project/doc
```

Anschließend wird diese Struktur in das svn-Repository importiert via

```
svn import . file:///path/to/repos --message 'Initial layout'
```

Beachtet den Zugriff auf das Repository mittels `file://...`. Dies funktioniert nur direkt auf dem Server. Will man von außen zugreifen, muss man einen Server konfigurieren.

Konfiguration des Servers Um via `svn://...` auf das Repository zuzugreifen, muss man den Dämon `svnserve` konfigurieren. Das ist mühsam und fehleranfällig. Alternativ lässt sich ein Webaccess (d.h. Zugriff via `http://...` oder `https://...`) über einen Apache-HTTP-Server einrichten. Das ist einfacher, aber immer noch etwas umständlich. Hat man Zugriff auf den Server via ssh, dann bedient man sich einfach `svn+ssh:///path/to/repos`, was sicher der bevorzugte Weg ist. Da ihr von der Physik aber ein Repository gestellt bekommt (siehe Kap. 6.3), entfallen die nötigen Schritte zur Einrichtung an dieser Stelle und ihr könntet direkt loslegen. Für den Umgang mit eigenen Repositories empfiehlt sich ein gutes Internet-HowTo zum Thema, beispielsweise das SVNBook (`http://svnbook.red-bean.com`).

6.4.1 Subversion unter Windows – TortoiseSVN

Für den Windows-Explorer gibt es die Erweiterung TortoiseSVN. Zu beziehen ist diese unter `http://tortoisesvn.net`. Es wird ein zusätzliches svn-Menü in das Kontextmenü von Ordnern und Dateien integriert, aus dem man die svn-Optionen aufrufen kann. Alle Befehle sind mit einer grafischen Benutzeroberfläche umgesetzt, so dass das Programm für jemanden, der mit `svn` umgehen kann, intuitiv zu bedienen ist.

7 Rechenprogramme

7.1 Qalculate! - mehr als nur ein Taschenrechner

Ein Taschenrechner? Der ist doch bei Windows extrem simpel und meist nutzlos. Die Oberfläche von Qalculate ist tatsächlich etwas nüchtern, und ein zaghaftes `42/23` liefert dann auch nur eine Zahl. Aber was erwartet man auch sonst?

Nun tippen wir aber mal `42m/23h` und sehen immerhin, dass er Einheiten nicht sofort verbannt. Es folgt ein Rechtsklick auf das Ergebnis „Convert to Base Unit“ et voila - eine vernünftige Einheit. Noch nicht überzeugt, etwas mehr damit zu spielen?

Okay - wie wäre es `20mmHg` einzutippen, Rechtsklick→Convert→„Pa“→Enter? Immer noch nicht? Gut, dann die harte Variante: `20mg*c^2` → Convert „eV“.

Qalculate kann dies alles und noch viel mehr in verschiedenen Zahlsystemen, mit komplexen Zahlen, mit Variablen... Es stehen extrem viele fertige Funktionen (auch gewichtetes Mittel) bereit und er kann in Maßen sogar integrieren und differenzieren - ja sogar Währungen nach aktuellen Kursen umrechnen. Und er kennt so ziemlich jede wichtige physkalische Einheit und Konstante (der geneigte Leser tippe z.B. mal `newtonian_constant`). Man kann schließlich auch noch redundante Klammern weglassen - ein Ausdruck wie `2+2)/(5+3` wird korrekt erkannt. Auch praktisch: mit *Exact* werden Brüche nicht in Fließkommazahlen umgerechnet.

Es bleibt hier wie an vielen Stellen des Skripts nur auf eigene Spielversuche und die ausführliche Hilfe zu verweisen, da eine komplette Einführung den Rahmen bei Weitem sprengen würde.

7.2 Gnumeric, Calc und ein wenig Excel

Die meisten kennen es aus der Schule: `Excel`. Nun wollen wir politisch korrekt bleiben und es eine Tabellenkalkulation nennen und im gleichen Atemzug die beiden wichtigsten Linux Alternativen `LibreOffice Calc` und `Gnumeric` erwähnen. Die meisten Teilnehmer des Anfängerpraktikums verwenden so ein Programm für die wesentlichen Schritte der Datenauswertung, also das Ausrechnen einer Funktion mit 84 verschiedenen Eingaben.

Bevor wir auf die grundsätzliche Bedienung eingehen, noch zwei Worte zu den Alternativen: Gnumeric ist etwas leichtgewichtiger, schneller und hübscher als Calc, welches dagegen einige Funktionen mehr hat. Die Gnumeric Programmierer haben mehr Wert auf Benutzerfreundlichkeit gelegt. Beiden Programmen fehlen ein paar der sehr fortgeschrittenen Funktionen von Excel. Aber um die Kirche im Dorf zu lassen: Der Autor ist selbst mit einer noch eingeschränkteren alten Version von Gnumeric locker durch das Anfängerpraktikum gekommen. In der grundsätzlichen Bedienung sind alle drei identisch und alle drei können `.xls`-Dateien lesen und schreiben.

Also ganz grob: *Wie geht man vor?* Zunächst tippt man recht nervig alle Messwerte ab[1], und zwar zusammengehörende Messwerte nebeneinander. Das könnte dann z.B. so aussehen:

[1] Findige Studenten machen das z.T. schon während der Messung.

t [in s]	s [in m]
0	0,0
1	2,0
2	3,9
4	8,2

Sehr gut! Und nun kann man in die Zelle hinter `0.0` gehen und dort mal `=b2/a2` eingeben und mit Enter bestätigen. Es sollte das Ergebnis erscheinen. Das muss man jetzt nicht für jede Zelle machen, denn, wenn man mit der Maus in die rechte untere Ecke des gerade verwendeten Kästchens geht, ändert sich der Zeiger. Nun klickt man und zieht die Maus bis unter das Kästchen hinter `8.2`. Und das Wunder geschieht, wir haben viele tolle Werte. Ab und an, mag man das automatische Erhöhen der Ziffern bei diesem Aufziehen nicht. Manchmal hat man z.B. eine Konstante in der Zelle `c4` und möchte, dass sich eben dieses c4 bei dieser Operation nicht ändert. Dann stellt man vor den Ausdruck, der sich nicht ändern soll, ein $. Also wenn man keine Änderung beim Runterziehen mag, verwendet man `c$4`, wenn man beim horizontal Aufziehen keine Änderung mag `$c4`. Oder halt gleich zwei $.

Es gibt natürlich auch komplexere Funktionen wie `sin`, `ln`, `exp`, `sqrt` und das Potentieren `2^3`. Oft verwendet wird auch `sum(a4:a42)` bzw. `sum(a4:h4)`, das alle Zahlen zwischen den beiden Grenzen aufsummiert. Im gleichen Sinne kann auch `average()` für den Mittelwert verwendet werden. Diese und viele weitere Funktionen sind ausführlich in der jeweiligen Hilfe der Programme erklärt.

Eine nette Eigenheit von Calc und Gnumeric hilft uns sehr bei der Arbeit mit Gnuplot. Markiert man die Spalten, die man plotten mag, klickt auf Bearbeiten→Kopieren und fügt nun in einem Texteditor seiner Wahl ein (z.B. `gedit`), so erhält man direkt die nötigen Tab-getrennten Werte. Anders herum klappt das auch sehr gut: Man kopiere Tab-, Leerzeichen- oder Komma- getrennte Zahlenkolonnen, drücke in der Tabellenkalkulation auf Einfügen und nach ein paar mal *Weiter* klicken sind sie schon fertig für die weitere Bearbeitung.

7.3 Das Computeralgebrasystem Maple

Ihr wollt mal eben schnell ein Integral ausrechnen? Einen komplizierten Ausdruck differenzieren? Einen hässlichen Term vereinfachen? Oft kann euch hierbei ein Computeralgebrasystem wie Maple helfen. Maple rechnet mit komplexen, irrationalen ($\sqrt{2}$,...) und transzendenten (π,...) Zahlen, Brüchen, Funktionen und vielem mehr. Viele Probleme werden nach einem bisschen Tipparbeit „hübsch“, bei manchem Problem muss man allerdings noch ein bisschen mehr reinstecken, damit wirklich das Optimum herauskommt. Wie das geht, werden wir später sehen.

Normalerweise benutzt man Maple mit grafischer Oberfläche. Das geht zum Beispeil mit dem Befehl `xmaple` in einer Konsole an einem CIP-Pool-Rechner. Oder, einen hinreichend schnellen Internetzugang[2] vorausgesetzt, über `ssh -X` von zu Hause aus. Nach dem Programmstart öffnet Maple eine Arbeitsoberfläche im Document Mode, man kann hier einfach seine Rechenanweisungen eingeben. Als Alternative bietet sich bei einem langsamen Internet der Konsolenmodus von Maple an, den man über `maple` erreicht.

Wenn man eine neue Berechnung anfängt, sollte man als erstes `restart;` eingeben. Dies sorgt dafür, dass wie bei einem Neustart von Maple alle Variablen, Funktionen, Annahmen usw. gelöscht werden. Wie im ersten Beispiel erkennbar, muss jeder Befehl in Maple von einem Semikolon abgeschlossen werden, allerdings ergänzt das grafische Interface die meisten vergessenen Semikolons.

[2] Mit einem DSL-Anschluss wartet man ewig auf das Laden der grafischen Oberfläche.

π	`Pi`
i	`I`
∞	`infinity`
e^x	`exp(x)`
$\sqrt{z}$	`sqrt(z)`
$\|x\|$	`abs(x)`
$\Re(x)$	`Re(x)`
$\Im(x)$	`Im(x)`

Tabelle 7.1: Einige wichtige Symbole und Funktionen

7.3.1 Variablen

Manchmal möchte man einen größeren Ausdruck nicht immer wieder schreiben. In diesem Fall bieten sich Variablen an. Variablen weist man in Maple mit `a:=simplify(cos(x)^2+sin(x)^2);` einen Wert zu, deklarieren muss man sie nicht. Anschließend kann man mit ihnen ganz normal weiterrechnen, zum Beispiel: `a+a;`. Möchte man die Zwischenergebnisse zum Beispiel bei einer größeren Variablenzuweisung nicht sehen, so kann man den Befehl anstatt mit `;` mit `:` abschließen. Um `a` wieder zu löschen, gibt man `a:='a';` ein.

Eine praktische Variable sei nicht vergessen: `%` enthält das Ergebnis der letzten Berechnung, `%%` das, der vorletzten, und `%%%` das, der drittletzten. Eine beliebte Anwendung ist `simplify(%);`, die das Ergebnis der letzten Berechnung algebraisch vereinfacht.

7.3.2 Funktionen und Differentiation

Funktionen deklariert man mit `f:=x->sin(x);`. Den Wert an der Stelle $\frac{\pi}{3}$ berechnet dann das Kommando `f(Pi/3);`. Das Ergebnis $\frac{\sqrt{3}}{2}$ ist exakt, aber meist möchte man ganz am Ende dann doch eine Dezimalzahl haben. Dafür gibt es `evalf(<expression>);`, das man natürlich auch mit dem Argument `%` aufrufen kann. So praktisch das erscheinen mag, die numerischen Werte bringen immer Rundungsfehler mit, weswegen man nur als letztes Mittel zur numerischen Auswertung greifen sollte.

Möchte man sich die Ableitung berechnen lassen, so kann man `diff(f(x),x)` (und natürlich auch mit jeder anderen Funktion wie `diff(sin(x),x)`) benutzen. Um höhere Ableitungen zu berechnen, muss man `diff` nicht schachteln, denn man kann einfach `diff(f(x),x$2)` für die zweite und `diff(f(x),x$n)` für die n-te Ableitung benutzen. Diese Ableitungsanweisung funktioniert auch mit Funktionen mehrerer Veränderlicher, wie zum Beispiel in `diff(h(x,y),x,y$2)`.

7.3.3 Integration

Integrale und Stammfunktionen berechnet Maple mit dem `int`-Befehl. Um eine Stammfunktion zu erhalten, ruft man `int(g(x),x)` auf, wobei g eine beliebige Funktion und x die Integrationsvariable ist. Besonders bei komplizierten Integranden kommt es vor, dass die Stammfunktion eher unbekannte Funktionen enthält. Ein bestimmtes Integral berechnet Maple, wenn man zusätzlich Grenzen angibt: `int(sin(x^4)^3,x=0..2*Pi)`. Dieses Beispielintegral kann Maple nicht analytisch berechnen, hier hilft wieder ein `evalf(%)`, um das Integral numerisch auszuwerten.

Das Integral `int(x^n,x=a..b)` kann Maple nicht direkt berechnen, da es zu wenig über n „weiß“. Wir möchten eine Lösung für $n > -1$ und teilen Maple diese Annahme mit dem Befehl

`assume(n>-1);` mit. Anschließend werden die Variablen, für die Maple Annahmen getroffen hat, mit einer Tilde markiert. Solche Annahmen sind häufig nötig, wenn Maple scheinbar grundlos die Arbeit verweigert. Leider teilt es uns meist auch nicht mit, warum eine Rechnung nicht durchgeführt werden kann. Möchte man ausschließlich im Reellen arbeiten, so ist es sinnvoll nicht jede Variable mit einem `assume` zu versehen, sondern mit `with(RealDomain)` ein hierfür zuständiges Paket einzubinden.

7.3.4 Matrizen

Maple kann natürlich auch mit Matrizen rechnen. Um die meisten Funktionen nutzen zu können muss man das Paket für lineare Algebra laden: `with(linalg)`. Die Matrix bekommt man entwerder mit dem Befehl `A:=matrix(2,2,[a11,a12 , a21,a22]);` in Maple. Oder, was bei großen Matrizen schöner ist, man benutzt die „Matrix erstellen“'-Funktion aus der Maple-Werkzeugleiste. Nun kann man allerhand mit den Matrizen machen. Leider funktionieren Operationen wie + nicht so einfach, man muss dafür die Matrizenbefehle `matadd(A,A)` für Addition oder `multiply(A,A)` für Multiplikation verwenden. Desweiteren gibt es natürlich noch Funktionen für invertieren (`inverse(A)`), transponieren (`transpose(A)`) und das Berechnen von Eigenwerten (`eigenvals(A)`). Die meisten dieser Funktionen sind auch über das Kontextmenü des angeklickten Ausdrucks erreichbar.

7.3.5 Grenzwerte und Summen

Mit dem Befehl `limit(sin(x)/x,x=0);` lassen sich Grenzwerte und mit `sum(1/q^n,n=1..infinity);` Summen berechnen. Auch hier ist es oft notwendig, mit `assume();` Einschränkungen zu machen.

7.3.6 (Differential-) Gleichungen

Gleichungen löst Maple mit dem Befehl `solve(x^2+3x-2=0,x);`. Das funktioniert auch mit Gleichungssystemen, dann trägt man als erstes Argument die „Menge“ der Gleichungen und als zweites die „Menge“ der Variablen, nach denen aufgelöst werden soll, ein: `solve({x+3y=5,4x+7y=9},{x,y});`. Bei vielen Gleichungssystemen gibt Maple die Lösung in der `RootOf`-Notation an. Hier hilft `allvalues(%[i]);` weiter, wobei i für das i-te Element der Lösung steht.

`fsolve();` löst Gleichungen numerisch, verliert aber häufig Lösungen. Um andere Lösungen zu finden, gibt man den Variablen in der Berechnung Startwerte mit: `fsolve({x^2+2/y^2=2,y^2-x^2=1},{x=0,y=0.1});`.

Für Differentialgleichungen gibt es den Befehl `dsolve();`. Ein Aufruf von `dsolve(diff(y(x),x)=y(x),y(x))` gibt als Lösung $y(x) = \_C1\, e^x$ zurück. Die in den Lösungen enthaltenen $\_Ci$ sind Integrationskonstanten, die man mittels Anfangsbedingungen festlegen kann: `dsolve({D(y)(x)=y(x),y(0)=1},y(x))`. Hierbei wurde der Differentialoperator `D` benutzt. `(D@@m)(f)(x)` berechnet die m-te Ableitung von f an der Stelle x und `D(f@sin)(x)` die Ableitung von $f(\sin(x))$ an der Stelle x.

Möchte man ein System von DGLn lösen, benutzt man wieder die Mengenklammern ({ }) und den `dsolve`-Befehl: `dsolve({dgls,anfb},{funkt})`. Hier sieht man schon, dass der Befehl schnell unübersichtlich wird, es ist daher ratsam mit Variablen zu arbeiten. Hier werden die Variablen

`dgl`, `anfb` und `funkt` benutzt. In `dgls` steht das zu lösende DGL-System, die Eingabe erfolgt so wie bei den normalen Gleichungen, aber in DGL-Notation. `anfb` legt die Anfangsbedingungen fest. In `funkt` müssen die Funktionen angegeben werden nach denen gelöst werden soll, also so etwas wie `x1(t),...,xn(t)`.

7.3.7 Plotten

Weiter vorne wurde schon mal Gnuplot vorgestellt. Gnuplot selbst ist schon ziemlich mächtig, aber manche Sachen, wie Vektorfelder, kann man nur mit viel Aufwand plotten. Maple kann so etwas recht einfach – und noch viele andere lustige Plotvarianten. Aber zuerst schauen wir uns die Standardsachen zuerst an.

Der einfache 2D Plotbefehl ist `plot(f,x,y,options)`. `f` ist der zu plottende Ausdruck, das kann eine Funktion, ein Vektor oder eine Menge von Punkten sein. `x` und `y` geben den Darstellungsbereich an. In `options` kann man seiner kreativen Ader freien Lauf lassen. Hier kann alles eingestellt werden: Farbe, Linientyp, Linendicke, Koordinatensysteme... Ein Beispiel ist:

```
plot(x*x, x = -1 .. 1, y = -2 .. 2, color='green')
```

Bei vielen Optionen empfiehlt es sich wieder hier mit Variablen zu arbeiten. In der Maple Hilfe ist eine Übersicht über alle verfügbaren Optionen mit Beispielen zu finden. Möchte man eine parametrischen Plot machen, so ändert sich der Befehl zu: `plot([x(t),y(t),t],options)`. Hier sind `x(t)` und `y(t)` die Koordinaten in Abhängigkeit des Parameters `t`. Für `t` muss wieder angegeben werden, welche Werte `t` annehmen kann.

Für 3D-Plots schreibt man einfach `plot3d(f,x,y,options)`. Im 3D sieht der parametrische Plotbefehl ein wenig anders aus: `plot3d([x(s,t),y(s,t),z(s,t)],s,t,options)`. Die Koordinaten können jetzt noch von einem zusätzlichen Parameter `s` abhängen. In `s` muss immer ein Parameterbereich für s angegeben werden, auch wenn die Koordinaten nicht von diesem abhängen.

Nun zu den fortgeschrittenen Techniken. Um die extravaganteren Plotbefehle nutzen zu können, muss man mit `with(plots)` das Plotpaket nachladen. Ein Vektorfeld plottet man mit `fieldplot3d([x,y,z],xr,yr,zr,options)`. In den eckigen Klammern stehen die Komponenten des Vektorfeldes, für `xr` etc. werden wieder die Intervalle eingetragen. Bei den Optionen kann man alles wieder so einstellen, wie man es mag. Besonders interessant ist die Möglichkeit in anderen Koordinatensystemen zu plotten.

Es gibt noch viele andere Dinge, die man mit den Plotfunktionen anstellen kann. Ich möchte noch kurz die Möglichkeit erwähnen Plots zu animieren. Das geschieht mit dem Befehl `animate(Befehl,[args],t,options)`. Für `Befehl` kann jeder der vorgestellten Plotbefehle eingesetzt werden (nur der Name des Befehls!). In args stehen die Argumente des Befehls und die verwendeten Intervalle. Der Parameter `t` gibt an wie der Plot über die Animation verändert wird. In `options` stehen wie immer die gewünschten Optionen drin. Wichtig ist bei den animierten Plots die folgende Option: `frames=n`, mit $n \in \mathbb{N}$.

7.3.8 Ausblick

Maple kann noch viel mehr. Deswegen folgen hier noch einige Anregungen für den Spieltrieb. Manches findet ihr hoffentlich irgendwann mal nützlich.

- `factor(); expand(); simplify();`
- `collect(); combine(); convert();`

- `with(inttrans); fourier(); laplace();`
- `with(linalg);` – ein Paket für lineare Algebra
- `with(DEtools);` – ein Paket für Differentialgleichungen

Ausführliche Erläuterungen zu den einzelnen Befehlen gibt es in der Maple-Hilfe, die ihr unkompliziert im Programm mit `?befehl` aufrufen könnt.

7.3.9 Literatur

Wem das noch nicht genug ist dem seien diese zwei Adressen nahe gelegt:

- Einführungskurs der Uni-Regensburg in fünf Teilen:
  `http://www.physik.uni-regensburg.de/studium/edverg/maple/`
- Ein 300 Seiten starkes Buch, was die Verwendung von Maple anhand von physikalischen Beispielen zeigt. Die Worksheets können auch runtergeladen werden:
  `http://mikomma.de/fh/embuch.html`

Zum Schluss noch ein Wort der Warnung: Wenn euch ein Ergebnis komisch vorkommt, rechnet selbst. **Auch ein Computer-Algebra-System macht unter gewissen Umständen Fehler**.

8 Künstlerisches

Im Folgenden sollen einige freie Grafikprogramme vorgestellt werden, die nicht nur im Praktikum sehr nützlich sein werden. Ein genauerer Blick lohnt sich also, damit man daran denkt, wenn man die Funktion einmal braucht[1].

8.1 Vektorgrafikprogramme

Es gibt zwei gängige Möglichkeiten Bilder im Computer zu speichern. Die eine ist, das Bild aus einer festen Anzahl von Bildpunkten zusammenzusetzen. Dabei merkt man sich für jeden Bildpunkt die Farbe. Dieses, z.B. von Digitalkameras und Paint verwendete Verfahren, eignet sich sehr gut für Fotos. Es hat aber den Nachteil, dass man beim Reinzoomen natürlich irgendwann die einzelnen Pixel als Klötzchen zu sehen bekommt.

Eine andere Möglichkeit ist, das Bild aus geometrischen Formen (Kreise, Quadrate, Linien mit definierten Krümmungen...) aufzubauen. Dann kann man beliebig zoomen und der Computer berechnet immer wieder ein scharfes Bild. Naturgemäß eignet sich dieses Verfahren weniger für echte Fotos, dafür aber um so mehr für Skizzen, Schaltpläne, Logos usw.

Gängige Formate für Vektorgrafiken sind .svg, .cdr, .eps, .pdf und Flash. Man kann in PDF-Dateien aber auch Pixelgrafiken unterbringen. Die bekanntesten Pixelgrafikformate sind sicherlich .bmp, .png, .gif und .jpg.

8.1.1 Inkscape

Inkscape ist ein Vektorgrafikprogramm im Stile von Corel Draw oder Adobe Freehand. Da es zur Zeit aktiv entwickelt wird, kommen monatlich neue Funktionen hinzu. Bereits jetzt können mit ihm Grafiken, Poster, Flyer u.ä. sehr komfortabel erstellt werden. Als Besonderheit benutzt Inkscape das freie Format `svg`, das mittlerweile so etwas wie ein Standard ist und auch von den großen Windowsprogrammen importiert wird. Selbst `gnuplot` kann svg ausgeben, was die nachträgliche Bearbeitung mit Inkscape ermöglicht. Als letzter Leckerbissen sei die gute PDF-Ausgabe erwähnt, die als ideale Druckvorstufe dient. Jede Druckerei kann heutzutage .pdf lesen.

Das Bild 8.1 zeigt Inkscape in Aktion. Mit kurzer Eingewöhnung kommt man recht schnell klar. Als hilfreiche Tipps seien aber folgende Punkte erwähnt:

- Die Rechteckauswahl wählt nur komplett im Gummiband befindliche Objekte aus.
- Mit Strg + g kann man mehrere ausgewählte Objekte zu einem gruppieren. Mit der zusätzlich gedrückten Shift-Taste macht man dies wieder rückgängig.
- Strg + d dupliziert die gerade ausgewählten Objekte und fügt sie an geeigneter Stelle ein.

[1] Der Autor weißt darauf hin, dass man z.B. ein Poster erst in etwa 1,5 Jahren für das Projektpraktikum anfertigen muss - aber auch dann bleibt Scribus ideal für diesen Job.

- `Stil einfügen` (Strg + Shift + v) ist von größerem Nutzen, als man zuerst meinen würde. Man überträgt damit die Schriftart, Linienbreite, Farbe. . . des vorher kopierten Objektes auf das gerade gewählte.
- Zieht man mit gedrückter Maustaste vom Lineal aus in das Zeichenfeld, so erhält man eine Hilfslinie. Außerdem kann man sich unter Datei→ Dokumenteigenschaften ein Raster und ein automatisches Einschnappen an diesem und/ oder den Hilfslinien aktivieren.
- Objekt→Ausrichtung bietet die schnelle Möglichkeit, mehrere Objekte auf die gleiche Höhe zu bringen.
- Inkscape besitzt ein integriertes Whiteboard, mit dem man über das Internet live und zusammen auf dem gleichen Zeichenfeld malen kann - siehe den Menüpunkt Whiteboard.
- Ein Tutorial zu einem Vektorgrafikprogramm (z.B. in der Inkscape-Hilfe), ist extrem hilfreich.
- Auf `http://screencasters.heathenx.org/` gibt es Schritt-für-Schritt Videoanleitungen, die sehr sehenswert sind.
- Inkscape kann PDF-Dateien auch lesen und bearbeiten.

Inkscape und Latex

Auch an dieser Stelle wollen wir wieder eine einfache, aber schöne Möglichkeit für die Verwendung von Latex-Formeln in Bilddokumenten vorstellen. Für Inkscape gibt es im Moment leider keinen brauchbaren Latex-Export[2]. Aber das Inkscape Plugin `Textext` von [3] hilft uns, TeX-Formeln direkt

[2] Ein vielversprechender Ansatz mit dem Namen `inkscape2tikz` ist auf `http://code.google.com/p/inkscape2tikz/` zu finden.

[3] `http://pav.iki.fi/software/textext/` und im CIP schon installiert

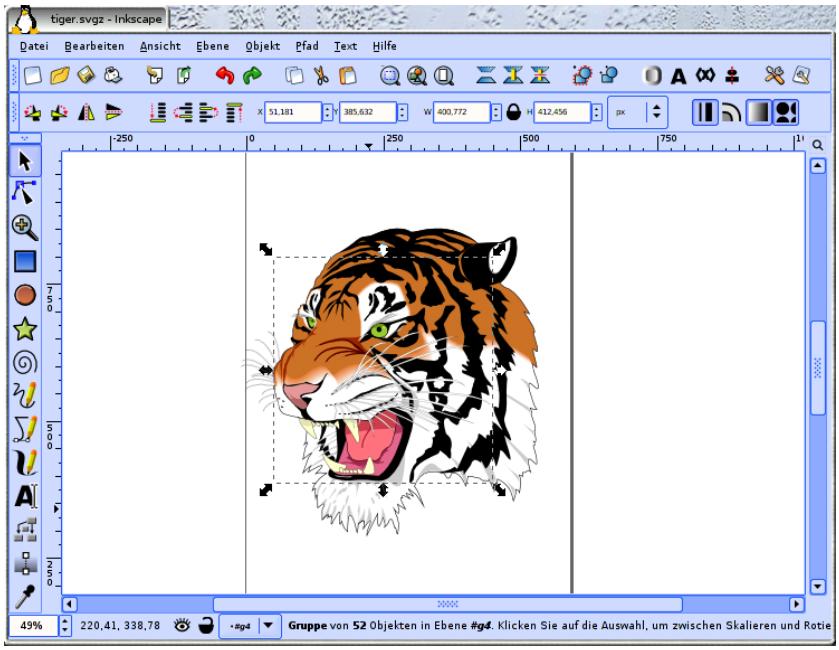


Abbildung 8.1: Inkscape in Aktion

in Inkscape zu verwenden. Man findet es unter Effekte→TexText[4]. Dort kann man auch eine Header-Datei einbinden, damit die gleiche Schriftart verwendet wird, man weitere Pakete verwenden kann usw. Ebenfalls kann hier direkt die Größe verändert werden. Dies ist aber auch später in Inkscape durch einfaches Großziehen möglich. Der Text lässt sich nachträglich durch Auswählen und einen anschließenden Klick auf den selben Menüpunkt ändern.

Die Verbindung mit Latex passiert dann über Datei→Kopie speichern unter→PDF via Cairo. Zuvor sollte man noch mit Strg+a, Datei→Dokumenteinstellungen→Seite in Auswahl einpassen die Größe der exportieren PDF-Datei tunen. Schließlich fügt man die Grafik mittels `\includegraphics{}` ein. ACHTUNG: Das Bild immer auch als .svg-Datei speichern (dem Standard), da nur diese weiterbearbeitet werden können, wenn man später kleine Fehler findet.

8.1.2 Dia

`dia` kann als gesunde Mischung aus Inkscape und XFig betrachtet werden. Wie letzteres bringt es einige Bibliotheken mit fertigen Symbolen mit, wie ersteres ist die Bedienung eher standardkonform. Ab und an zickt dia noch herum, aber die Entwicklung schreitet stetig voran. Dia stellt mit `LaTeX PGF Makros` einen vernünften Latex-Export bereit.

8.2 Layoutprogramm

Ein Layoutprogramm (neudeutsch DeskTop Publishing - DTP) steht unter Linux mit `Scribus` zur Verfügung. Es zeichnet sich durch einfache Handhabung von Fließtext und Text- und Grafikpositionierung aus, sodass sich z.B. auch Poster einfach und schnell erstellen lassen. Der PDF- Export ist hervorragend und bietet sich auch an, wenn man mal eben ein Foto oder gescanntes Dokument zu einem PDF konvertieren muss.

8.3 Schaltplaneditor

`gEDA` ist eigentlich eine komplette Enwicklungsumgebung für Elektroniker. Man kann damit Schaltpläne entwerfen, Bauteile einfügen und diese sogleich am Rechner durchmessen und virtuell den Strom fließen lassen. Das wollen wir alles nicht, aber der enthaltene Schaltplaneditor `gschem` kann sich sehen lassen. In seiner Bibliothek unter *analog* finden sich praktisch alle Bauelemente, die man für die Elektroversuche so verwendet. In Latex bekommt man die Zeichnungen, indem man sie in eine Datei *druckt* und die entstehende Postscript Datei dann mit ps2pdf zu einem PDF konvertiert und sogleich einbindet.

8.4 Pixelgrafik und Fotobearbeitung

`gimp` ist sicherlich das Grafikprogramm schlechthin. Im Funktionsumfang ähnlich zu Photoshop, kann man es auch als Ersatz für Paintshop Pro, Corel Painter u.ä. ansehen. Viele Filter helfen Digitalfotos zu verbessern, Grafiken für das Web zu bauen usw. Gimp kann praktisch jedes Grafikformat auf diesem Planeten lesen und in den meisten Formaten abspeichern. Vektorgrafiken werden dabei aber umgewandelt in Pixelgrafiken und verlieren ihre tollen Eigenschaften.

[4] Inkscape bietet serienmäßig unter Effekte→Rendern→LaTex-Formel ein ähnliches Plugin, aber ohne weitere Header und vor allem nicht nachträglich änderbar.

8.5 Openclipart.org

Dies ist kein Programm sondern eine Website, aber eine sehr nützliche. Man findet hier unzählige frei verwendbare Cliparts und Bilder, die sich eventuell gut als Lückenfüller in Protokollen oder zur Auflockerung von Präsentationen verwenden lassen.

8.6 TikZ

Auch dies ist kein Programm (man beachte die rekursive Abkürzung: “TikZ ist kein Zeichenprogramm”) sondern eine Erweiterung von Latex. Damit kann man Bilder als TEX-Quellcode schreiben. Informationen zum Projekt gibt es unter `http://sourceforge.net/projects/pgf/`. Die vielfältige Verwendbarkeit von TikZ fürs Plotten, Zeichnen, Designen, Dekorieren von Text und Mindmaps lässt sich am Besten unter `http://www.texample.net/tikz/examples/` erahnen. Dort sind viele gute Beispiele dokumentiert, die ein sehr breites Spektrum an Verwendungsmöglichkeiten bieten.

Viel Spass beim Ausprobieren!