EDV für Physiker

Martin Ueding
mu@uni-bonn.de
Martrikelnummer: 2439532
Tutor: Jörg Dabringhausen
Übungsgruppe Freitags 10:00 - 12:00 CIP-Pool Astronomie

2012-01-02

Zusammenfassung

Bericht für die „EDV für Physiker und Physikerinnen“ (physik130) Veranstaltung. Es wurden Einblicke in Linux, LaTeX, C++ sowie ROOT vermittelt. Dieser Bericht enthält die Berichtsaufgaben.

Dieser Bericht wurde mit LaTeX, pdflatex, Vim, TeX Maker, Ubuntu, make, qmake und git erstellt.

Inhaltsverzeichnis

Teil I

Linux

Kapitel 1

Übung 1

1.1 Befehle in der Konsole

1.1.1 Benutzerverwaltung und -rechte

`chmod` Ändert Dateirechte.

`hostname` Gibt den Rechnernamen aus.

`last` Letzte Anmeldungen aller Benutzer.

`ps` Gibt eine Prozessliste aus. Dieses Programm ist nicht interaktiv und eignet sich beispielsweise für Logdateien.

`top` Vorgänger von `htop`, eine interaktive Prozessverwaltung.

`uname` Gibt Kernelversion, Rechnername, ..., aus.

`uptime` Zeigt, wie viele Nächte der Rechner am Stück durchgemacht hat.

`whoami` Gibt den eigenen Benutzernamen aus.

`w` Wie `who`, nur ausführlicher.

1.1.2 Dateibehandlung

`bzip2` Komprimiert eine Datei.

`cd` Wechselt in das angegebene Verzeichnis. Dabei ist `..` das übergeordnete Verzeichnis. Wird als Verzeichnis `-` angegeben, kommt man in das vorherige Verzeichnis. `cd` ohne Verzeichnis wechselt in das Heimatverzeichnis[1].

`cp` Kopiert Dateien.

`df` Listet die Dateisysteme mit Belegungsangabe.

`du` Zeigt die Größe von Dateien auf dem Datenträger an. Diese muss nicht unbedingt mit der Größe überstimmen, die `ls` anzeigt, da die Dateien in Blöcken organisiert sind.

[1] Meistens `/home/<benutzername>`.

`find` Führt eine Dateisystemtraverse nach speziellen Suchvorgaben durch und zeigt standardmäßig alle Dateien und Ordner an.

`gzip` Komprimiert eine Datei.

`ls` Listet den Inhalt des angegebenen oder aktuellen Verzeichnisses auf. Dabei werden auch Informationen über Zugriffsrechte und Eigentümer, Größe und Änderungsdatum angezeigt, gibt man `-l` an.

`mkdir` Erstellt ein Verzeichnis.

`mv` Verschiebt Dateien, Umbenennen ist ein Spezialfall.

`rmdir` Löscht leere Verzeichnisse.

`rm` Löscht Dateien.

`scp` Kopiert über SSH.

`tar` Erstellt ein Archiv mehrerer Dateien. Mit `tar -xzf archiv.tar.gt datei1 datei2...`[4] erstellt man direkt ein komprimiertes Archiv mehrerer Dateien.

`touch` Setzt das Änderungsdatum einer Datei auf den aktuellen Zeitpunkt und erzeugt die Datei, falls sie nicht existiert.

1.1.3 Informationen zu Programmen

`apropos` Unscharfe Suche nach Befehlen.

`man` Zeigt Handbücher zu Programmen an.

`whatis` Zeigt die erste Zeile des Handbuchs an.

1.1.4 Textdateien

`diff` Vergleicht Dateien miteinander.

`emacs` Texteditor.

`grep` Filtert Zeilen nach einem Suchmuster.

`head` Zeigt die ersten n Zeilen einer Datei an.

`less` Zeigt eine Datei an und erlaubt scrollen, suchen, springen. Es können viele Befehle aus `vi` (gg, G, /, j, k) benutzt werden.

`pdflatex` Übersetzt ein LaTeX Dokument in ein PDF.

`sort` Sortiert Zeilen.

`tail` Gegenstück zu `head`.

`uniq` Spezialfall von `sort -u`

`vim` Texteditor für Programmierer.

`wc` Zählt Wörter, Zeilen, Buchstaben.

1.1.5 Bash built-ins

`clear` Fügt leere Zeilen ein, bis der Bildschirm leer ist.

`set` Zeigt das Environment der Shell an.

`echo` Gibt Text aus.

`history` Zeigt die letzten Kommandos an.

1.1.6 Diverses

`bc` Einfacher Taschenrechner. Man sollte in der Bash allerdings nicht mit `$(echo 5+4 | bc)` rechnen, sondern die neuen, von C übernommenen Funktionen zum direkten Rechnen, `$((5+4))`, benutzen.

`blkid` Zeigt die UUID der Partitionen an.

`e2label` Zeigt und vergibt Partitionslabel.

`ssh` Öffnete eine Shell auf einem anderen Computer.

`wget` Lädt Dateien per http oder ftp.

`cal` Zeigt einen Kalendermonat an.

`date` Zeigt das Datum in einem gewählten Format an.

`gv` Zeigt eine PDF oder PS Datei an. Normalerweise würde man okular (KDE) oder evince (Gnome) benutzen.

1.2 absoluter und relativer Pfad

Ein absoluter Pfad beginnt immer mit einem `/`, wie beispielsweise `/home/mu/Dokumente/Studium/EDV/Bericht` oder `/dev/null`. Ein relativer Pfad bezieht sich immer auf ein aktuelles Arbeitsverzeichnis. Beispielsweise beschreibt `datei.tex` die Datei `/tmp/datei.txt`, falls der Benutzer gerade `/tmp` als Arbeitsverzeichnis hat. Pfade können auch `..` enthalten, dies bezeichnet das übergeordnete Verzeichnis. Ist man gerade in `/etc/apache2`, so kann man mit `../passwd` auf die zentrale Passwortdatei verweisen.

Gemeinerweise können absolute Pfade auch `..` enthalten, so wäre `/etc/apache2/../passwd` ein legaler Pfad, sinnvoll ist es in vielen Fällen allerdings nicht.

1.3 grundlegende Emacs Steuerung

Aktion	Tasten
Cursor links	C-b
Cursor rauf	C-p
Cursor rechts	C-f
Cursor runter	C-n
Datei speichern	C-x C-s
Emacs beenden	C-x C-c
Hilfe aufrufen	C-h t
Seite rauf	M-v
Seite runter	C-v

Tabelle 1.1: Grundlegende Emacs Tastenkombinationen.

Kapitel 2

Übung 2

Neue Befehle sind in §1.1 mit den Befehlen aus vorherigen Übungen zusammen.

2.1 Das Unix-Hilfe-System – der `man` Befehl

Generell sollte jedes Programm eine Handbuchseite haben, die genauso wie das Programm selbst heißt. So kann man mit `man Programm` diese Seite aufrufen.

Bash Builtins haben keine solche Hilfeseite, sie werden mit `help Kommando` dokumentiert, oder können in `bash.1` nachgeschaut werden.

Darüber hinaus gibt es noch die info Dokumente.

Meistens haben Programme auch noch eine eigene Hilfe dabei, die mit `Programm -h` oder `Programm -help` aufgerufen werden kann.

2.2 Datums- und Kalenderangaben

2.3 Bash Variablen

Mit `echo $HOME` kann man das Heimatverzeichnis anzeigen lassen. Dabei ist `echo $HOME` eine der vielen Environmentvariablen, die in der Bash gesetzt sind. Man kann sie mit `set` anschauen. `echo $HOSTNAME` enthält den Rechnernamen.

Mit den geschweiften Klammern kann man mehrere Wörter aus einem erzeugen, dies ist praktisch für das erstellen von Sicherungskopien: `cp foo{,.bak}` erstellt eine Kopie der Datei `foo` nach `foo.bak`, ohne dass man foo zweimal schreiben muss.

Das Programm `cal` zeigt einen Kalendermonat an. Der September 1752 ist etwas anders, als die restlichen Monate, da hier in einigen, leider nicht allen, Ländern der Wechsel zwischen Kalendersystemen vollzogen worden ist.

2.4 Umgang mit Dateien und Verzeichnissen

Hier gibt es wenig zu beschreiben, man muss sein aktuelles Arbeitsverzeichnis im Auge behalten und beachten, dass `rmdir` nur leere Verzeichnisse löscht.

2.5 Absolute und relative Pfade

Auch hier muss man sein Arbeitsverzeichnis für die relativen Pfade im Kopf haben, ansonsten ist es alles recht logisch.

2.6 Wildcards

Die Dateien lassen sich recht einfach mit dem Code aus Listing 2.1 erstellen. Dabei werden die geschweiften Klammern benutzt, um mehrere Dateien, die Namensbestandteile gemeinsam haben, zu erzeugen.

Listing 2.1: Anlegen der Dateien

```
touch datei{1..9}.txt logfile{10,10a,11,11a,11b}.txt
```

Listing 2.2 zeigt, wie man diese Dateien ausgeben (Ausgabe in Listing 2.3) kann.

Listing 2.2: ls.sh

```
echo *.log
echo *[4-7]*
echo datei[2-78].txt
echo *[a-zA-Z].log
```

Listing 2.3: Ausgabe von ls.sh

```
logfile10a.log logfile10.log logfile11a.log logfile11b.log logfile11.log
datei4.txt datei5.txt datei6.txt datei7.txt
datei2.txt datei3.txt datei4.txt datei5.txt datei6.txt datei7.txt datei8.txt
logfile10a.log logfile11a.log logfile11b.log
```

Ich habe hier `echo` anstelle von `ls` benutzt, da die Wildcards so oder so von der Bash und nicht vom Befehl aufgelöst werden und `ls` jede Datei auf eine eigene Zeile schreibt, sofern `STDOUT` kein Terminal ist. Dies spart ein klein wenig Platz und erzeugt letztlich die gleiche Ausgabe.

2.7 Umgang mit Rechten unter Unix

Jede Datei hat drei grundlegende Rechte, lesen (r), schreiben (w) und ausführen (x). Darüber hinaus gibt es noch drei Kategorien: Eigentümer, Gruppe und Alle. Mit `chmod -w` entzieht man allen das Schreibrecht, auch sich selbst. Setzt man die Rechte auf `rw-r-r-`[1] darf nur der Eigentümer die Datei schreiben, alle sie aber lesen.

[1] mit `chmod 644`

Kapitel 3

Übung 3

3.1 Weitere Unix-Befehle

Siehe §1.1.

3.2 Eingabe- und Ausgabeumleitung

Listing 3.1: standardausgabe.txt

```
Autokennzeichen.txt
namen.dat
standardausgabe.txt
Uebung_03.pdf
zahlen.dat
```

3.3 Handling von Textdateien

Man kann es sich an dieser Stelle einfach machen und alle Kennzeichen rauswerfen, die mit einer Ziffer beginnen und es bleiben dann nur noch deutsche Kennzeichen übrig. Der Ansatz (Listing 3.2) ist nicht nennenswert robust, erfüllt aber in der kleinen Eingabemenge seinen Zweck (Ausgabe in Listing 3.3).

Listing 3.2: auto_einfach.sh

```
grep -v '^[0-9]' Autokennzeichen.txt
```

Listing 3.3: Ausgabe von auto_einfach.sh

```
K FG 1289
BN HG 212
SU RC 8370
MYK JJ 286
B-JH 125
K-AA 898
SU A 123
BN SF 214
```

```
MYK JJ 673
K KJ 201
K AB 1721
B TH 129
```

Allerdings ist es mit den Kennzeichen nicht so ganz einfach, möchte man wirklich nur deutsche Kennzeichen haben. Es gibt deutsche und Euro-Kennzeichen, alle Neuen sind Letztere. Die alten haben noch einen Bindestrich, die neuen nicht mehr. Außerdem dürfen in den alten die Buchstaben B, F, G, I, O und Q nicht vorkommen.[5]

Es gilt das Muster `XXX XX 0000`, in jeder Gruppe können auch weniger Zeichen sein. Allerdings dürfen es maximal 8 Zeichen sein, solange es kein Saisonkennzeichen ist.

Das führt dazu, dass man im regulären Ausdruck (RegEx) nicht einfach 3, 2 und 4 Zeichen suche lassen kann, ansonsten würden Zeichenketten zugelassen, die 9 Zeichen lang sind. Somit muss man eine Fallunterscheidung benutzten. Darüber hinaus dürfen bei einem alten Kennzeichen – erkennbar an dem Bindestrich – einige Buchstaben im mittleren Feld nicht auftauchen. Der komplexe reguläre Ausdruck ist in Listing 3.4 gezeigt.

Listing 3.4: auto.sh

```
grep -E \
    '^([A-Z]{1}( [A-Z]{1,2}|-[AC-EHJ-NPR-Z]{1,2}) [0-9]{1,4})$
^([A-Z]{2}( [A-Z]{1,2}|-[AC-EHJ-NPR-Z]{1,2}) [0-9]{1,4})$
^([A-Z]{3}( [A-Z]{1}|-[AC-EHJ-NPR-Z]{1}) [0-9]{1,4})$
^([A-Z]{3}( [A-Z]{2}|-[AC-EHJ-NPR-Z]{2}) [0-9]{1,3})$' \
    Autokennzeichen.txt
```

Listing 3.5: Ausgabe von auto.sh

```
K FG 1289
BN HG 212
SU RC 8370
MYK JJ 286
B-JH 125
K-AA 898
SU A 123
BN SF 214
MYK JJ 673
K KJ 201
K AB 1721
B TH 129
```

Man sieht, dass die gleiche Liste (Listing 3.5) erzeugt wird, jedoch fällt das zweite Skript nicht auf diverse Gemeinheiten (Listing 3.6) rein, die man sich ausdenken könnte.

Listing 3.6: Gemeinheiten für Listing 3.2

```
zKeinKennzeichen
NDH ND 2000
 1234 DC 75
@27
```

3.4 Pipelines

3.4.1 Namen

Listing 3.7: namen.dat

```
Herbert Meyer
Berta Meyer
Susanne Meier
Anton Meyer
Martin Meier
Zara Meier
```

Es ist recht sinnfrei `cat` zu benutzen, um den Inhalt zu in `sort` zu bekommen, da `sort` die Datei auch selbst laden kann.

Mit dem entsprechenden Befehl (Listing 3.8) bekommt man die gewünschte sortierte und gefiltere Ausgabe (Listing 3.9) der Originaldatei (Listing 3.7).

Listing 3.8: namen.sh

```
sort namen.dat | grep Meier
```

Listing 3.9: Ausgabe von namen.sh

```
Martin Meier
Susanne Meier
Zara Meier
```

3.4.2 Zahlen

Listing 3.10: zahlen.dat

```
1
30
11
2
8
9
44
635
```

Bei den Zahlen werden die Zahlen (Eingabedatei in Listing 3.10) exakt so sortiert, wie ich es erwarte. Und zwar nach ASCII Code Point. Es ist unnatürlich, dass eine 2 vor einer 1 steht, weil *nach* der 1 eine weitere Zahl folgt. Da die Zahlen in der Datei allerdings ohne führende Nullen stehen, muss man die Zahl zuerst interpretieren, damit sie nach Zahlenwert sortiert werden kann. Dafür ist das `-n` Flag da. Der komplette Befehl ist in 3.11 gezeigt. Die Ausgabe (Listing 3.12) ist wirklich nach Zahlenwert sortiert.

Listing 3.11: zahlen-sort.sh

```
sort -n zahlen.dat
```

Listing 3.12: Ausgabe von zahlen-sort.sh

```
1
2
8
9
11
30
44
635
```

3.5 Komprimierung und Archivierung

Leider war die Datei `cc++.tar` zum Bearbeitungszeitpunkt nicht verfügbar. So habe ich 55 MiB HTML Dateien zum Testen benutzt.

Programm	Option	Zeit [s]	Dateigröße [KiB]
tar + bzip2		4.29 + 11.02	12701
tar + bzip2	-1	4.29 + 11.81	12701
tar + bzip2	-9	4.29 + 11.00	12701
tar + gzip		4.29 + 2.37	14149
tar + gzip	-1	4.29 + 1.51	15514
tar + gzip	-9	4.29 + 4.30	14073
zip		2.29	16694

Tabelle 3.1: Zeiten und Dateigrößen verschiedener Kompressionsprogramme

Man sieht an den Daten in Tabelle 3.1, dass bzip2 etwas besser komprimiert, allerdings deutlich länger braucht. Zip ist am schnellsten, schafft allerdings auch am wenigsten Kompression.

Zip komprimiert jede Datei einzeln und kann so Passagen, die in mehreren Dateien vorkommen, nicht effizient komprimieren. Jedoch können einzelne Dateien aus dem Archiv genommen werden, bei einem komprimierten tar muss erst alles expandiert werden, damit eine einzelne Datei entnommen werden kann.

Das Skript, das die Zeiten misst, kann in §4.1 gefunden werden.

3.6 Verteiltes Arbeiten

Die Dateien auf den zwei verschiedenen Rechnern, die allerdings ihr Heimatverzeichnis teilen, ist fast gleich, bis auf die Datei, die auf dem anderen Rechner schon erstellt worden ist.

Kopiert man die Dateien auf einen Rechner, kann man sie mit `diff -u` vergleichen. Die Unterschiede sind Listing 3.13 gezeigt.

Listing 3.13: Unterschied zwischen Ordnerinhalten

```
--- ls-lR.host1 2011-10-30 14:43:46.031286114 +0100
+++ ls-lR.host2 2011-10-30 14:43:49.275286307 +0100
@@ -2 +2 @@
-insgesamt 156
+insgesamt 184
```

```
@@ -9 +9,2 @@
--rw-r--r-- 1 ueding studis      0 30. Okt 2011  ls-lR.host1
+-rw-r--r-- 1 ueding studis  24614 30. Okt 2011  ls-lR.host1
+-rw-r--r-- 1 ueding studis      0 30. Okt 2011  ls-lR.host2
```

Die Datei, in die die Ausgabe umgeleitet wird, wird zwar direkt angelegt, weil bash sie öffnet, allerdings hat sie noch keine Größe. Erst nach dem Durchlauf ist die Datei komplett da. Zwischen Physik und Astro CIP-Pool sind natürlich alle Dateien anders, weil es komplett verschiedene Ordner sind. Nur Dateien wie meine `.bashrc` sind beispielsweise auf beiden Seiten vorhanden und haben die gleiche Größe, jedoch unterscheidet sich das Änderungsdatum.

3.7 Shell - Skript

Für die Farben kann man eine einfache `for`-Schleife benutzen, wie in Listing 3.14 gezeigt. Dies erzeugt die gewünschte Ausgabe (Listing 3.15).

Listing 3.14: farben.sh

```
for farbe in blau gelb gruen rot
do
    echo "Meine Lieblingsfarbe ist $farbe. Also fahre ich ${farbe}e Fahrraeder."
done
```

Listing 3.15: Ausgabe von farben.sh

```
Meine Lieblingsfarbe ist blau. Also fahre ich blaue Fahrraeder.
Meine Lieblingsfarbe ist gelb. Also fahre ich gelbe Fahrraeder.
Meine Lieblingsfarbe ist gruen. Also fahre ich gruene Fahrraeder.
Meine Lieblingsfarbe ist rot. Also fahre ich rote Fahrraeder.
```

3.7.1 Zahlen

Für die Zahlen kann man entweder `1..3` benutzen[1], oder eine C-artige for-Schleife benutzen (Listing 3.16, Ausgabe in Listing 3.17). Die Syntax, die auf dem Übungszettel steht, gibt es nicht, ich weiß nicht, wie Sie die Ausgabe auf dem Übungszettel damit erzeugt haben.

Listing 3.16: zahlen.sh

```
for (( i = 0; i <= 3; i++ ))
do
    echo $i
done
```

Listing 3.17: Ausgabe von zahlen.sh

```
0
1
2
3
```

[1] Ein Beispiel für die `1..3` finden Sie in §4.2

Soll von 2 bis 20 gezählt werden und der Vorname ausgegeben werden, wenn der Zähler auf 10 steht, sieht das Skript so aus wie in Listing 3.18. (Ausgabe in Listing 3.19.)

Listing 3.18: zahlen_name.sh

```
for (( i = 2; i <= 20; i++ ))
do
    echo $i

    if (( i == 10 ))
    then
        echo Martin
    fi
done
```

Listing 3.19: Ausgabe von zahlen_name.sh

```
2
3
4
5
6
7
8
9
10
Martin
11
12
13
14
15
16
17
18
19
20
```

3.8 Analyse von Pipelines

Der Befehl `find /uebung_03 -type f -exec du -k {} \;| sort -n -r` iteriert durch das Verzeichnis `uebung_03` und dessen Unterverzeichnisse, sucht Einträge heraus, die Dateien sind (und keine Ordner, Links, Fifos, ...). Auf jede dieser Dateien wird der Befehl `du -k` ausgeführt. Man erhält eine Liste mit Dateigrößen in kiB. Diese Liste wird dann numerisch und rückwärts sortiert, also die größten Dateien nach vorne.

3.8.1 Teilaufgabe a

In dieser Aufgabe ist wahrscheinlich ein `ps aux | grep ueding` gefragt. Dies ist allerdings gefährlich, falls jemand anderes ein Programm ausführt, das meinen Namen im Namen hat, oder

jemand anders so heißt wie ich, nur mit einer 2 dahinter. Außerdem wird `grep` auch sich selbst in der Ausgabe von `ps` finden, weil mein Name auch Teil des Kommandos ist und somit dieser wieder in der Prozessliste auftaucht. Daher wäre es mit einer genauen Suche nach der Position in der Zeile etwas sicherer (Listing 3.20). So hätte man beide Probleme aus dem Weg.

Allerdings kann man auch einfach das `a` weglassen und erhält direkt eine Liste mit seinen eigenen Prozessen und vermeidet an dieser Stelle die Benutzung von `grep` komplett.

Listing 3.20: Einschänkung des Suchbereichs

```
ps aux | grep '^ueding '
```

3.8.2 Teilaufgabe b

Für das Sortieren hat `ps` auch eine entsprechende Option: `O+p`. So lässt sich mit `ps ux O+p` direkt nach Prozess-ID sortieren.

3.8.3 Teilaufgabe c

Mit `ps ux O+p > myprocesses.txt` lässt sich die entsprechende Datei (Listing 3.21) erzeugen.

Listing 3.21: myprocesses.txt

```
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ueding    26464  0.0  0.1   9576  1784 ?        S    15:36   0:00 sshd: ueding@pts
    /0
ueding    26465  0.0  0.1   4760  2048 pts/0    Ss   15:36   0:00 -bash
ueding    26725  0.0  0.0   2700   816 pts/0    R+   15:47   0:00 ps ux O+p
```

Wenn man mag, kann man natürlich auch den Code aus Listing 3.22 benutzen.

Listing 3.22: verkettete Pipes

```
ps aux | grep ueding | sort -n > myprocesses.txt
```

3.9 weitere Linux Befehle

3.9.1 Super GAU

Der Unterschied zwischen `rm -rf ./uebung/*` und `rm -rf ./uebung/ *` ist, dass bei erstem alle Dateien und Unterordner des Verzeichnis `uebung` ohne Nachfrage gelöscht werden, im zweiten wird der Ordner selbst ebenfalls rekursiv gelöscht und alle anderen Dateien ebenfalls gelöscht. Das einzige, das dann noch bleibt, sind versteckte Dateien, deren Namen mit einem Punkt beginnt.

3.9.2 frage.txt

Der erste Befehl kopiert die Datei `/home/meinname/uebung/frage.txt` nach `/home/meinname/uebung`. Dies klappt allerdings nicht, weil es schon ein Verzeichnis mit dem gleichen Namen gibt.

Im zweiten Kommanto wird die Datei nach `/home/meinname/uebung/frage.txt` kopiert. Die Datei wird mit sich selbst überschrieben. Dies wird allerdings verhindert, das klappt also auch nicht.

Der dritte Befehl wird alle Dateien nach `/home/meinname/uebung` kopieren. Es sollte eine Fehlermeldung geben, die „keine reguläre Datei“ lautet, da man schlecht mehrere Dateien in eine kopieren kann.

Der vierte Befehl kopiert alle Dateien in den Ordner selbst. Da die Dateien mit sich selbst überschrieben werden würden, schlägt das fehl.

Verzeichnis existiert nicht Falls das Verzeichnis nicht existiert, kann die Datei `frage.txt` nicht existieren, womit die ersten Beiden Befehle daran scheitern werden, dass es die Datei nicht gibt.

Das zweite Kommando könnte sich auch noch darüber beschweren, dass es das Zielverzeichnis nicht gibt.

Die letzten beiden werden daran scheitern, dass die Datei `uebung/*` nicht existiert. Falls sie in Bash den Nullglob aktiviert haben, wird moniert, dass kein Ziel angegeben worden ist.

im Verzeichnis Falls man im Verzeichnis ist, so wird der erste Befehl im dortigen Verzeichnis eine neue Datei `uebung` erzeugen – kein Problem.

Der zweite Befehl wird wegen das wohl nicht vorhandene Verzeichnis scheitern.

Befehl drei wird nur funktionieren, wenn es exakt eine Datei gibt. Da mehrere Dateien drin sind, klappt es nicht.

Bei dem vierten Befehl würden die ganzen Dateien in den Unterordner `uebung` kopiert. Da es diesen nicht gibt, schlägt es fehl.

3.9.3 Backup

Um alle Dateien zu Packen, würde ich `tar -xzf backup.tar.gz .??* *` ausführen. Dies packt alle sichtbaren und unsichtbaren Dateien, allerdings nicht die impliziten `.` und `..` ein.

Das ganze bekommt man in den anderen Pool mit einem `scp backup.tar ciptux:`, falls man vorher in der `.ssh/config` entsprechend den kompletten Hostnamen des Servers eingetragen hat.

3.9.4 Dateirechte

Im Allgemeinen werden Zugriffsrechte über die Werkzeuge `chmod` und `chown` sowie `chgrp` geändert.

Wenn ein Verzeichnis nicht leer ist, kann es von `rmdir` nicht gelöscht werden. Man kann es vorher mit `rm uebung02/*` leeren, wobei man dort bei Unterverzeichnissen das gleiche Problem hat. Die einfache Variante ist es, einfach `rm -rf uebung02` zu benutzen.

Wenn man unbedingt dem Kommilitonen Schreibrechte gewähren möchte, müssen die Rechte auf `664` oder `660` stehen. Ich würde lieber die Rechte auf `640` stellen und meinen Kommilitonen bitten seine Änderungen auf ähnliche Weise in seinem Heimatverzeichnis bereitzustellen. Dann kann ich mit `diff -u meinedatei seinedatei` schauen, was er verändert hat und es mit `cp seinedatei meinedatei` übernehmen.

Kapitel 4

Anhang

4.1 Kompressionsvergleich

Listing 4.1: compare.sh

```
#!/bin/bash
# Copyright (c) 2011 Martin Ueding <dev@martin-ueding.de>

# Compresses the files in the data folder with gzip, bzip2 and tar and displays
# time and file sizes.

set -e
set -u

data="data"

mytime="/usr/bin/time -f %E"

if [[ ! -d "$data" ]]
then
    echo "Please create a folder $data and fill it with test data"
    exit 1
fi

mkdir -p out
rm -f out/*

echo -ne "tar\t"
trap 'rm -f test.tar' EXIT
$mytime tar -cf test.tar "$data"

cp test.tar out/test-1.tar
cp test.tar out/test-9.tar
cp test.tar out/test.tar
echo -ne "gzip -1\t"
$mytime gzip -1 out/test-1.tar
echo -ne "gzip -9\t"
$mytime gzip -9 out/test-9.tar
echo -ne "gzip\t"
```

```
$mytime gzip out/test.tar


echo -ne "zip\t"
$mytime zip -rq out/test.zip "$data"

cp test.tar out/test-1.tar
cp test.tar out/test-9.tar
cp test.tar out/test.tar
echo -ne "bzip2 -1\t"
$mytime bzip2 out/test-1.tar
echo -ne "bzip2 -9\t"
$mytime bzip2 out/test-9.tar
echo -ne "bzip2\t"
$mytime bzip2 out/test.tar

ls -lhS out
```

4.2 Alternative Zählschleife

Listing 4.2: zahlen2.sh

```
for i in {1..3}
do
    echo $i
done
```

Listing 4.3: Ausgabe von zahlen2.sh

```
1
2
3
```

Teil II

LaTeX

Kapitel 5

Übung 4

Faust - Der Tragödie erster Teil, Johann Wolfgang von Goethe

Quelle: http://de.wikisource.org/

...

Faust.
Das also war des Pudels Kern!
Ein fahrender Scolast? Der Casus macht mich lachen.

Mephistopheles.
Ich salutire den gelehrten Herrn!
Ihr habt mich weidlich schwitzen machen.

Faust.
Wie nennst du dich?

Mephistopheles. *Die Frage scheint mir klein,*
Für einen, der das Wort so sehr verachtet,
Der, weit entfernt von allem Schein,
Nur in der Wesen Tiefe trachtet.

Faust.
Bey euch, ihr Herrn, kann man das Wesen
Gewöhnlich aus dem Namen lesen,
Wo es sich allzu deutlich weist,
Wenn man euch Fliegengott, Verderber, Lügner heit.
Nun gut wer bist du denn?

Mephistopheles.
Ein Theil von jener Kraft,
Die stets das Böse will und stets das Gute schafft.

...

Kapitel 6

Übung 5

Siehe `Uebung_05/mathe.pdf` für Matheübung.

6.1 Einbindung von Graphiken

Siehe `Uebung_05/Uebung05_Zubehoer/gesellschaft_vierte.pdf`.

6.2 Weiteres zur Textstrukturierung

Anstelle die Datei `gliederung.tex` zu formatieren, habe ich Verweise in diesen Bericht eingebracht, wo diese durchaus nützlich sein können.

Ein Anhang mit Code Listings ist auch schon vorhanden. Inzwischen habe ich die Anhänge in die jeweiligen Teile gepackt, somit habe ich momentan keine appendix-Umgebung am Ende.

Ein Inhaltsverzeichnis hat dieses Dokument natürlich auch.

Auf der Titelseite habe ich allerdings nur einen Autor. Würde man zwei haben wollen, kann man sie mit `\\` trennen, so wie überall sonst auch.

Das Literaturverzeichnis habe ich diesem Artikel hinzugefügt.

6.3 Berichtaufgaben

Die Linuxbefehle (§1.1) habe ich als `itemize` gehalten, was ich übersichtlicher finde. Eine Tabelle kann als Tabelle 3.1 auf Seite 13 gefunden werden. Diese Tabelle hat schon ein Label.

Teil III

C++

Kapitel 7

Übung 6

7.1 Das „Hello, World!“-Programm

Dies ist das einfache „Hello, World!“-Programm (Listing 7.1), das Kerningham und Ritchie zum Einstieg empfehlen um seine Entwicklungsumgebung entsprechend einzurichten.[1]

Listing 7.1: hello.cpp

```
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;

    return 0;
}
```

Um das ganze zu kompilieren braucht man einen C++-Compiler, in diesem Fall ist es `g++`. Man kann aber genauso gut `MSVC++` benutzen.

Ich möchte nicht immer den Kompilerbefehl erneut eintippen[1], so dass ich hier eine Makefile benutze um das Programm mit einem einfachen `make` kompilieren zu können. Außerdem wird hier auch direkt die Ausgabe des Programms in eine Datei gefügt, die dann wiederum in dieses LaTeX Dokument eingefügt wird. Somit sind die Ausgaben, die hier enthalten sind tatsächlich von den Programmen erzeugt worden.

Die Ausgabe des Programms 7.1 ist in Listing 7.2 zu sehen.

Listing 7.2: Ausgabe von hello

```
Hello, World!
```

Die Optionen `-Wall und -pedantic` zeigen deutlich mehr Unstimmigkeiten im Code auf und sind damit zu empfehlen. Das im Beispiel benutzte `int a;`, das allerdings nie verwendet wird, ist meistens ein Fehler und man hat an einer anderen Stelle nicht `a`, sondern versehentlich eine andere Zahl benutzt.

[1] Wobei in Bash der letzte Kompiliervorgang mit `!g++` noch recht schnell zu wiederholen ist.

7.2 Äthiopische Multiplikation

Das Programm (Listing 7.3) habe ich aus dem Skript kopiert und entsprechend wieder mit Makefile versehen. Damit es von der Makefile allerdings nicht-interaktiv kompiliert werden kann, übernehme ich die Eingabezahlen von der Kommandozeile.

Listing 7.3: ethiopian.cpp

```
#include <cstdio>
#include <cstdlib>
#include <iostream>

int main(int argc, char *argv[]) {
    if (argc != 3) {
        std::cout << "Usage: ethiopian NUMBER NUMBER" << std::endl;
        return 1;
    }

    int a = atoi(argv[1]);
    int b = atoi(argv[2]);

    int res = 0;

    while (a >= 1) {
        printf("%6d %6d\n", a, b);

        if (a % 2 == 1) {
            res += b;
        }

        a /= 2;
        b *= 2;
    }
    std::cout << "Result: " << res << std::endl;

    return 0;
}
```

Die Ausgabe für $14 \cdot 36$ ist in Listing 7.4 gezeigt.

Listing 7.4: Ausgabe von ethiopian

```
    14     36
     7     72
     3    144
     1    288
Result: 504
```

Möchte man nun eine Art Multiplikationstabelle von eins bis zehn erhalten, muss man zwei ineinander geschachtelte Schleifen bauen. `a` und `b` können nicht als Schleifenindizes verwendet werden, da sie vom Algorithmus verwendet werden. Entweder kapselt man den Algorithmus in eine Funktion, oder man benutzt einfach andere Variablen. Das geänderte Programm ist in Listing 7.5 zu sehen. Die Ausgabe ist in Listing 7.6) gezeigt.

Listing 7.5: ethiopian-ng.cpp

```
#include <cstdio>
#include <cstdlib>
#include <iostream>

int main(int argc, char *argv[]) {
    int a;
    int b;
    int res;

    int i, j;
    for (i = 1; i <= 10; i++) {
        printf("%2d:", i);
        for (j = 1; j < i; j++) {
            printf("    ");
        }
        for (j = i; j <= 10; j++) {
            a = i;
            b = j;
            res = 0;


            while (a >= 1) {
                if (a % 2 == 1) {
                    res += b;
                }

                a /= 2;
                b *= 2;
            }
            printf(" %3d", res);
        }
        std::cout << std::endl;
    }

    return 0;
}
```

Listing 7.6: Ausgabe von ethiopian-ng

```
 1:   1   2   3   4   5   6   7   8   9  10
 2:       4   6   8  10  12  14  16  18  20
 3:           9  12  15  18  21  24  27  30
 4:              16  20  24  28  32  36  40
 5:                  25  30  35  40  45  50
 6:                      36  42  48  54  60
 7:                          49  56  63  70
 8:                              64  72  80
 9:                                  81  90
10:                                     100
```

Möchte man alle Multiplikationen haben, lässt man die Indizes von 1 bis 10 laufen. Um kommutativ äquivalente Rechnungen zu unterdrücken, lässt man den inneren Index beim äußeren

Index starten.

7.3 Heron

Der Algorithmus ist recht einfach umzusetzen, man benötigt die Formel, die auf dem Übungsblatt angegeben ist.

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \tag{7.1}$$

Dabei ist a die Zahl, von der die Wurzel bestimmt werden soll. Diese Formel braucht noch ein x_0, damit sie funktioniert. Hier kann man einfach jede beliebige Zahl benutzen. Allerdings ist $x_0 = a$ keine allzu schlechte Wahl.

Eine Abbruchbedingung kann verschieden aussehen. Man kann nach einer gewissen Anzahl von Iterationen abbrechen. Ich habe mich dafür entschieden, abzubrechen, sobald sich die Zahl nur noch wenig verändert. Dazu muss man sich merken, welchen Wert die Zahl vorher hatte, um vergleichen zu können.

Das ganze Programm könnte man rekursiv implementieren, allerdings spricht hier nichts gegen eine iterative Variante, daher habe ich mich für letzteres entschieden. Mein Programm ist in Listing 7.7 zu sehen.

Der Quellcode ist in Englisch dokumentiert, wie in [2] und [3] empfohlen.

Listing 7.7: heron.cpp

```
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <iostream>

/**
 * Iterates one step with the heron algorithm.
 *
 * @param x Current best guess.
 * @param a Input value of which the root should be calculated.
 * @return Better guess.
 */
double iterate(double x, double a) {
    return (x + a / x) / 2;
}

/**
 * The main function.
 *
 * Parses a float from the second command line argument and calculates the
 * square root of that number. The result is printed on the screen.
 *
 * @param argc Number of command line arguments.
 * @param argv Command line arguments.
 * @return 0 on success, 1 on negative number, 2 if not enough arguments.
 */
```

```
int main(int argc, char **argv) {
    // Abort if the user did not a number on the command line.
    if (argc != 2) {
        std::cout << "Usage: heron number" << std::endl;
        return 2;
    }

    // Trust the user that he entered something that can be parsed as a float.
    // Parse the input into an automatic variable.
    double input = atof(argv[1]);

    // If the input is negative, abort. This algorithm would find that the
    // root of -4 is -2, which is not the case.
    if (input < 0) {
        std::cout << "Complex numbers do not exist." << std::endl;
        return 1;
    }

    // If the number changes by less then this amount (measured as ratio from
    // the input number), the result is probably somewhat accurate.
    double smallest_change = 1e-5;

    // In order to find out the change, the program needs to remember the last
    // state.
    double current, previous;

    // Set the previous value to something negative so that the program does
    // not abort right away.
    previous = -1 - smallest_change;

    // Start with the user supplied value.
    current = input;

    // Iterate while the difference between the current and previous step are
    // bigger than the set amount of the user's input value.
    while (fabs(current - previous) > smallest_change * input) {
        // Save the current value for later comparison.
        previous = current;

        // Print the current state of the calculation.
        printf("%f\n", current);

        // Apply the heron algorithm and go one step further.
        current = iterate(current, input);
    }

    // Sometimes, the algorithm steps over into the negative values.
    current = fabs(current);

    // Print the answer.
    printf("\n");
    printf("The square root of %f is %f.\n", input, current);
```

```
    // Return with a zero return value so that the calling shell knows that
    // everything went fine.
    return 0;
}
```

Ich lasse $\sqrt{2} \approx 1.41421356237309504880168872421$[2] berechnen, die Ausgabe ist in Listing 7.8 zu sehen. Man kann sehen, dass das Programm schnell zu einem brauchbaren Ergebnis kommt, der Algorithmus also leistungsfähig ist.

Listing 7.8: Ausgabe von heron für 2

```
2.000000
1.500000
1.416667
1.414216

The square root of 2.000000 is 1.414214.
```

Listing 7.9: Ausgabe von heron für 4

```
4.000000
2.500000
2.050000
2.000610
2.000000

The square root of 4.000000 is 2.000000.
```

Listing 7.10: Ausgabe von heron für 10

```
10.000000
5.500000
3.659091
3.196005
3.162456
3.162278

The square root of 10.000000 is 3.162278.
```

Für die CppDoc Kommentare habe ich in [6] nachgeschaut.

[2] Mit `N[Sqrt[2], 30]` in Mathematica berechnet.

Kapitel 8

Übung 7

Mein Programm, dass die Daten einließt und wieder speichert ist in Listing 8.1 zu finden. Die Aufgabe ist in Listing 8.2.

Listing 8.1: arrays.cpp

```
// Copyright (c) 2011 Martin Ueding <dev@martin-ueding.de>

#include <cstdio>
#include <fstream>
#include <iostream>

#define LENGTH 50

double mult(double a, double b) {
    return a * b;
}

double add(double a, double b) {
    return a + b;
}

double div(double a, double b) {
    return a / b;
}

int main() {
    std::ifstream infile;
    infile.open("data2.dat");

    std::ofstream outfile;
    outfile.open("out.dat", std::ofstream::out);

    double x[LENGTH], y[LENGTH];
    double results_add[LENGTH];
    double results_mult[LENGTH];
    double results_div[LENGTH];

    for (int n = 0; n < LENGTH && infile.good(); n++) {
        infile >> x[n] >> y[n];
```

```
        if (!infile.good()) {
            break;
        }

        results_add[n] = add(x[n], y[n]);
        results_mult[n] = mult(x[n], y[n]);
        results_div[n] = div(x[n], y[n]);

        outfile << x[n] << " + " << y[n] << " = " << results_add[n]
            << std::endl;
        outfile << x[n] << " * " << y[n] << " = " << results_mult[n]
            << std::endl;
        outfile << x[n] << " / " << y[n] << " = " << results_div[n]
            << std::endl;
    }

    infile.close();
    outfile.close();

    return 0;
}
```

Listing 8.2: Ausgabe von arrays

```
1 + 2 = 3
1 * 2 = 2
1 / 2 = 0.5
2 + 4.1 = 6.1
2 * 4.1 = 8.2
2 / 4.1 = 0.487805
3 + 5.8 = 8.8
3 * 5.8 = 17.4
3 / 5.8 = 0.517241
4 + 8.1 = 12.1
4 * 8.1 = 32.4
4 / 8.1 = 0.493827
5 + 9.7 = 14.7
5 * 9.7 = 48.5
5 / 9.7 = 0.515464
6 + 12 = 18
6 * 12 = 72
6 / 12 = 0.5
7 + 14.5 = 21.5
7 * 14.5 = 101.5
7 / 14.5 = 0.482759
8 + 15.9 = 23.9
8 * 15.9 = 127.2
8 / 15.9 = 0.503145
9 + 18.1 = 27.1
9 * 18.1 = 162.9
9 / 18.1 = 0.497238
10 + 20.4 = 30.4
10 * 20.4 = 204
10 / 20.4 = 0.490196
```

8.1 Versuchsergebnisse

Dies ist das Programm, das die Spannungs- und Stromdaten einließt und entsprechend auswertet. Die Widerstandstabelle ist in Listing 8.4, die statistischen Größen in Listing 8.5.

Listing 8.3: bericht.cpp

```
// Copyright (c) 2011 Martin Ueding <dev@martin-ueding.de>

#include <cmath>
#include <fstream>
#include <iostream>

#define LENGTH 50

struct measurement {
    double voltage;
    double current;
};

int main() {
    std::ifstream infile;
    infile.open("data_bericht.dat");

    std::ofstream outfile;
    outfile.open("out.dat", std::ofstream::out);

    struct measurement cur, sum, avg;
    sum.voltage = 0.0;
    sum.current = 0.0;

    double sum_power = 0.0;
    double sum_voltage_squared = 0.0;

    int n = 0;

    for (; n < LENGTH && infile.good();) {
        infile >> cur.voltage >> cur.current;

        if (!infile.good()) {
            break;
        }

        n++;

        sum.voltage += cur.voltage;
        sum.current += cur.current;

        sum_power += cur.voltage * cur.current;
        sum_voltage_squared += pow(cur.voltage, 2);
```

```
        outfile << cur.voltage << " " << cur.current << " "
                << cur.voltage / cur.current << std::endl;
    }

    infile.close();
    outfile.close();

    avg.voltage = sum.voltage / n;
    avg.current = sum.current / n;

    double avg_power = sum_power / n;
    double avg_voltage_squared = sum_voltage_squared / n;

    double m = (avg_power - avg.voltage * avg.current)
               / (avg_voltage_squared - pow(avg.voltage, 2));
    double c = avg.current - m * avg.current;

    std::cout << "Means: " << avg.voltage << " V, " << avg.current << " A"
              << std::endl;
    std::cout << "Means: " << avg_power << " V^2, " << avg_voltage_squared
              << " W" << std::endl ;
    std::cout << "m: " << m << ", c: " << c << std::endl ;

    std::ofstream means;
    means.open("means.dat", std::ofstream::out);

    means << "Means: " << avg.voltage << " V, " << avg.current << " A"
          << std::endl;
    means << "Means: " << avg_power << " V^2, " << avg_voltage_squared
          << " W" << std::endl ;
    means << "m: " << m << ", c: " << c << std::endl ;

    means.close();

    return 0;
}
```

Listing 8.4: Ausgabedatei von bericht

```
Means: 5.5 V, 11.0632 A
Means: 76.0632 V^2, 37.75 W
m: 2.02877, c: -11.3815
7 0.5
4 8.1 0.493827
4.5 8.8 0.511364
5 9.7 0.515464
5.5 11.2 0.491071
6 12 0.5
6.5 12.8 0.507812
7 14.5 0.482759
7.5 15.3 0.490196
8 15.9 0.503145
8.5 17 0.5
```

```
9 18.1 0.497238
9.5 19.3 0.492228
10 20.4 0.490196
```

Listing 8.5: Ausgabe von bericht

```
Means: 5.5 V, 11.0632 A
Means: 76.0632 V^2, 37.75 W
m: 2.02877, c: -11.3815
```

Kapitel 9

Übung 8

9.1 Strings

Listing 9.1: String main.cpp

```
#include <iostream>
#include <string>

int main() {
    std::string s1 = "abc";
    std::string s2 = "def";
    std::string s;

    std::cout << s1.length() << std::endl;

    s = s1 + s2;

    std::cout << "s = " << s << std::endl;

    s1.append(s2);
    std::cout << "s1 = " << s1 << std::endl;

    std::string s_sub = s.substr(1, 4);
    std::cout << s_sub << std::endl;

    // Test whether the stings are equal.
    if (s.compare(s1) == 0) {
        std::cout << "s and s1 are equal" << std::endl;
    }

    if (s1.compare(s2) == 0) {
        std::cout << "s1 and s2 are equal" << std::endl;
    }

    if (s.compare(s2) == 0) {
        std::cout << "s and s2 are equal" << std::endl;
    }

    // Position of substring "ef".
```

```
    size_t pos_ef = s.find_first_of("ef");
    std::cout << "Position of \"ef\" is: " << pos_ef << std::endl;

    // Replace "cd" with "ZZ".
    size_t pos_cd = s.find_first_of("cd");
    s = s.replace(pos_cd, 2, "ZZ");

    std::cout << "s is now " << s << std::endl;

}
```

9.2 Klassen

Listing 9.2: String main.cpp

```
#include "student.h"
#include <iostream>
#include <string>

using namespace std;

int main(void) {
    Student student;
    string temp_vorname;

    cout << "Standardvorname und -nachname des Studenten: "
         << student.getVorname() << " " <<  student.getNachname() << endl;

    cout << "Setze einen Vornamen: ";
    cin >> temp_vorname;
    student.setVorname(temp_vorname);
    cout << endl << "Neuer Voname des Studenten: " << student.getVorname()
         << endl;

    cout << "Setze einen Nachnamen: ";
    cin >> temp_vorname;
    student.setNachname(temp_vorname);
    cout << endl << "Neuer Nachnamen des Studenten: " << student.getNachname()
         << endl;

    cout << "Neuer Name des Studenten: " << student.getVorname() << " "
         << student.getNachname() << endl;

    int tag, monat, jahr;
    cout << "Hier bitte die Geburtsdaten im Format <Tag Monat Jahr> eingeben (ab
       1970)."
         << endl;
    cin >> tag >> monat >> jahr;
    cout << "Das Geburtsdatum lautet: " << tag << " " << monat << " " << jahr
         << endl;
    student.setGeburtstag(tag, monat, jahr);
    cout <<  "Das Alter des Studenten ist: " << student.getAlter() << endl;
    cout <<  "Der Geburtstag des Studenten ist (YYYYMMDD): "
```

```
         << student.getGeburtstag() << endl;

    return 0;
}
```

Listing 9.3: String student.cpp

```
#include "student.h"

using namespace std;

void Student::setVorname(string vorname) {
    this->vorname = vorname;
}

void Student::setNachname(string nachname) {
    this->nachname = nachname;
}

string Student::getVorname() {
    return vorname;
}

string Student::getNachname() {
    return nachname;
}

int Student::getAlter() {
    struct tm *timeinfo;
    time_t rawtime;

    time(&rawtime);
    timeinfo = localtime(&rawtime);

    struct tm *timeset;
    timeset = localtime(&rawtime);

    timeset->tm_year = geb_jahr - 1900;
    timeset->tm_mon = geb_monat - 1;
    timeset->tm_mday = geb_tag;
    time_t settime = mktime(timeset);

    int diff = (int)difftime(rawtime, settime) / 60 / 60 / 24 / 365;

    return diff;
}

void Student::setGeburtstag(int tag, int monat, int jahr) {
    geb_tag = tag;
    geb_monat = monat;
    geb_jahr = jahr;
}

Student::Student() {
```

```
    vorname = "Vorname";
    nachname = "Nachname";

    geb_tag = 1;
    geb_monat = 1;
    geb_jahr = 1950;

    matrikelnummer = 0;
    studienfach = 0;
}

Student::~Student() {
}

int Student::getGeburtstag() {
    return geb_tag + geb_monat * 100 + geb_jahr * 10000;
}
```

Listing 9.4: String student.h

```
#ifndef STUDENT_H
#define STUDENT_H

#include <iostream>
#include <string>
#include <time.h>

using namespace std;

class Student {
private:
    string vorname;
    string nachname;

    int studienfach;
    int matrikelnummer;

    int geb_tag;
    int geb_monat;
    int geb_jahr;

public:
    Student();
    ~Student();

    void setVorname(string vorname);
    string getVorname();
    void setNachname(string nachname);
    string getNachname();

    int getAlter();
    void setGeburtstag(int geb_tag, int geb_monat, int geb_jahr);

    int getGeburtstag();
```

```
};

#endif
```

9.3 Berichtsaufgabe

Listing 9.5: Consolidator main.cpp

```
#include <fstream>

int main(int argc, char *argv[]) {
    std::ofstream outfile;
    outfile.open(argv[1]);

    for (int i = 2; i < argc; ++i) {
        outfile << argv[i] << std::endl;
    }

    outfile.close();
}
```

Kapitel 10

Übung 9

10.1 Einfache Zeiger

Die Programmzeile wird die Zahl 5 und die Adresse von i, die auf dem Aufgabenzettel angegeben ist, ausgeben.

Die Ausgabe von `&i` ist die Speicheradresse, `*(&i)` wird die Zahl 5 ausgeben.

10.2 Überprüfungsaufgabe

Fast alles davon geht. Nur kann man keinem Array einen Pointer zuweisen. Das Array ist zwar ein Pointer, allerdings ein Speziallfall davon. Klassische Polymorphie.

Bei der zweiten Teilaufgabe sollte der Compiler nichts davon akzeptieren, da `int` und `int*` unterschiedliche Datentypen sind. Da es allerdings nur Zahlen sind, kann man trotzdem irgendwie mit ihnen rechnen.

Auf den Pointer kann man anscheinend ints addieren, allerdings kann man dem int fast keine Pointerwerte zuweisen. Nur eine Differenz ist möglich. Das ist interessant. Letztlich ist nur eine Differenz interessant, da dies die Lange eines Arrays darstellen kann.

10.3 Zeiger und Funktionen

Die Lösung ist in Listing 10.1 gezeigt. In der Übung schreiben Sie `int **`, allerdings weiß ich nicht so recht, wo ich das dann einbauen soll. Die Ausgabe ist in 10.2 gezeigt.

Listing 10.1: Swap

```
#include <iostream>

using namespace std;

void swap(int *x, int *y) {
    int z;
    z = *x;
    *x = *y;
```

```
    *y = z;
}

int main() {
    int x = 2, y = 3;
    cout << x << " " << y << endl;
    swap(&x, &y);
    cout << x << " " << y << endl;
    return 0;
}
```

Listing 10.2: Ausgabe von Swap

```
2 3
3 2
```

10.4 Versuchsergebnisse, die Zweite

Dies ist das Programm aus der 7. Übung, entsprechend für die Berichtsaufgabe angepasst.

Bei der Bestimmung des Fehlers von R weiß ich nicht genau, was δR sein soll. Ich rechne jetzt mit

$$\Delta R = \sqrt{(\Delta U)^2 + (\Delta I)^2} \tag{10.1}$$

Listing 10.3: bericht.cpp

```
#include <cmath>
#include <fstream>
#include <iostream>

// Encapsulate the maximum length of the data sets in a define statement.
#define LENGTH 50

/**
 * Tuple of measurement data.
 */
struct measurement {
    double voltage;
    double current;
    double e_voltage;
    double e_current;
};

/**
 * Main function.
 *
 * Iterates through the data files, calculates some statistic values and writes
 * it to the output files.
 *
 * @return 0 on success, 1 if not enough arguments.
```

```
 */
int main(int argc, char **argv) {
    // Abort if there are no two file names in the command line options.
    if (argc != 3) {
        std::cout << "Usage: bericht in out" << std::endl;
        return 1;
    }

    // Open the input file.
    std::ifstream infile;
    infile.open(argv[1]);

    // Open the output file.
    std::ofstream outfile;
    outfile.open(argv[2], std::ofstream::out);

    // Create variables that hold the accumulated data.
    struct measurement cur, sum, avg;

    // Initialize these data field since C++ does not do this for us.
    sum.voltage = 0.0;
    sum.current = 0.0;

    // Create some scalar values for other statistic values.
    double sum_power = 0.0;
    double sum_voltage_squared = 0.0;

    // The counter for the for loop is outside of the for loop so that it's
    // value is not removed from the automatic memory once the loop is over.
    // After the loop, it contains the number of data sets.
    int n = 0;

    // Iterate through all the lines in the data file until end of file (EOF)
    // is reached.
    for (; n < LENGTH && infile.good();) {
        // Parse the line from the input file.
        infile >> cur.voltage >> cur.current >> cur.e_voltage >> cur.e_current;

        // In case the last line was read, abort right here.
        if (!infile.good()) {
            break;
        }

        // Increase n to show that another line has been read.
        n++;

        // Sum the just read values to the accumulator variables.
        sum.voltage += cur.voltage;
        sum.current += cur.current;

        // Sum the just read values to the other statistic variables.
        sum_power += cur.voltage * cur.current;
        sum_voltage_squared += pow(cur.voltage, 2);
```

```
    // Calculate the resistance.
    double resistance = cur.voltage / cur.current;

    // Calculate the combined error.
    double e_resistance = sqrt(pow(cur.e_voltage, 2) + pow(cur.e_current, 2))
        ;

    // Write the set into the output file.
    outfile << cur.voltage << "\t" << cur.current << "\t" << cur.e_voltage <<
        "\t" << cur.e_current << "\t" << resistance << "\t" << e_resistance
           <<
        std::endl;
}

// Close the input and output file.
infile.close();
outfile.close();

// Calculate the average voltage and current.
avg.voltage = sum.voltage / n;
avg.current = sum.current / n;

// Calculate average power and voltage^2.
double avg_power = sum_power / n;
double avg_voltage_squared = sum_voltage_squared / n;

// Calculate the linear fit for this data.
double m = (avg_power - avg.voltage * avg.current)
          / (avg_voltage_squared - pow(avg.voltage, 2));
double c = avg.current - m * avg.current;

// Print the means and slope.
std::cout << "Means: " << avg.voltage << " V, " << avg.current << " A"
          << std::endl;
std::cout << "Means: " << avg_power << " V^2, " << avg_voltage_squared
          << " W" << std::endl ;
std::cout << "m: " << m << ", c: " << c << std::endl ;

// Write the same data to the means.dat output file.
std::ofstream means;
means.open("means.dat", std::ofstream::out);

means << "Means: " << avg.voltage << " V, " << avg.current << " A"
      << std::endl;
means << "Means: " << avg_power << " V^2, " << avg_voltage_squared << " W"
      << std::endl;
means << "m: " << m << ", c: " << c << std::endl ;

means.close();

// Return with a success.
return 0;
```

```
}
```

Listing 10.4: Ausgabedatei von bericht

```
1   2   0.35    0.25    0.5 0.430116
1.5 2.9 0.35    0.35    0.517241    0.494975
2   4.5 0.35    0.23    0.444444    0.418808
2.5 5.5 0.35    0.3 0.454545    0.460977
3   5.8 0.35    0.4 0.517241    0.531507
3.5 7   0.35    0.05    0.5 0.353553
4   8.3 0.35    0.2 0.481928    0.403113
4.5 8.7 0.35    0.3 0.517241    0.460977
5   9.7 0.35    0.353   0.515464    0.497101
5.5 11.5    0.35    0.23    0.478261    0.418808
6   12  0.35    0.351   0.5 0.495682
6.5 12.8    0.35    0.352   0.507812    0.496391
7   14.6    0.35    0.01    0.479452    0.350143
7.5 15.3    0.35    0.359   0.490196    0.501379
8   15.9    0.35    0.26    0.503145    0.436005
8.5 17  0.35    0.359   0.5 0.501379
9   18.5    0.35    0.07    0.486486    0.356931
9.5 19.3    0.35    0.358   0.492228    0.500664
10  20.2    0.35    0.02    0.49505 0.350571
10.5    21  0.35    0.05    0.5 0.353553
11  21.9    0.35    0.23    0.502283    0.418808
11.5    23.1    0.35    0.24    0.497835    0.424382
12  24  0.35    0.31    0.5 0.467547
12.5    24.6    0.35    0.357   0.50813 0.499949
13  25.9    0.35    0.01    0.501931    0.350143
13.5    27.3    0.35    0.08    0.494505    0.359026
14  28  0.35    0.23    0.5 0.418808
14.5    29.5    0.35    0.26    0.491525    0.436005
15  29.8    0.35    0.41    0.503356    0.539073
15.5    31.1    0.35    0.35    0.498392    0.494975
16  32.1    0.35    0.351   0.498442    0.495682
16.5    33  0.35    0.05    0.5 0.353553
17  33.7    0.35    0.24    0.504451    0.424382
17.5    35.2    0.35    0.08    0.497159    0.359026
18  36  0.35    0.353   0.5 0.497101
18.5    36.9    0.35    0.26    0.501355    0.436005
19  38.5    0.35    0.359   0.493506    0.501379
19.5    39.1    0.35    0.22    0.498721    0.413401
20  39.9    0.35    0.09    0.501253    0.361386
```

Listing 10.5: Ausgabe von bericht

```
Means: 10.5 V, 21.0795 A
Means: 284.538 V^2, 141.917 W
m: 1.99591, c: -20.9933
```

Teil IV

ROOT

Kapitel 11

Übung 10

Listing 11.1: Code für Histogram

```
/**
 * Creates a histogram with given data.
 */
{
    // Create a new canvas.
    TCanvas *c1 = new TCanvas("c1", "mein canvas", 800, 500);

    // Create a new histogram.
    TH1F *h1 = new TH1F("h1", "mein histo", 10, 0, 10);

    // Fill the latter with data.
    h1->Fill(3.2);
    h1->Fill(7);
    h1->Fill(3.6);
    h1->Fill(4.2);

    // Label the axes.
    h1->SetXTitle("x");
    h1->SetYTitle("Anzahl");

    // Draw it.
    h1->Draw();
}
```

Man kann mit `c1->SetLogx(1)` eine logarithmische Achse aktivieren und mit `c1->SetLogx(0)` wieder zurück zur linearen Achse wechseln.

11.1 Berichtsaufgabe

Man kann hier einfach den entsprechenden Konstruktor benutzen, da die Datei freundlichweise schon passend formatiert ist.

Listing 11.2: Code für Plot

```
/**
 * ROOT macro to parse voltage and current.
```

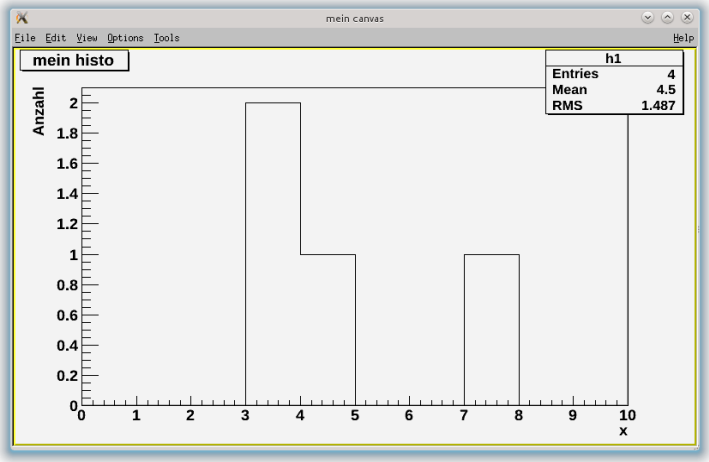

Abbildung 11.1: Standardplot

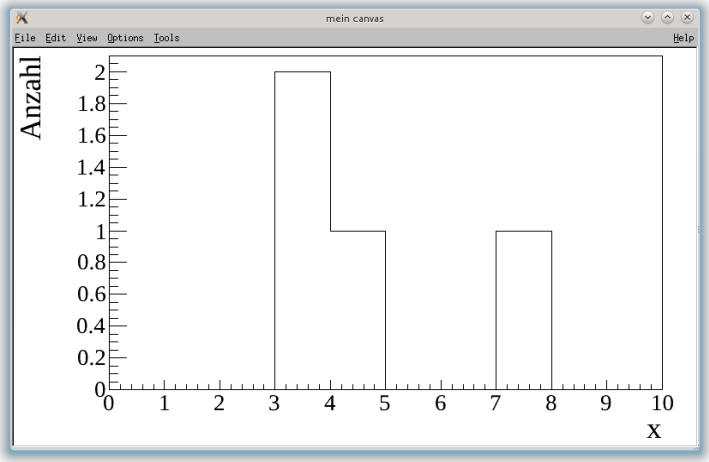

Abbildung 11.2: BABAR Plot

```
 */
{
    // Generate a new Canvas element with the given size.
    TCanvas *c1 = new TCanvas("c1", "c1", 800, 600);

    // Change to the first part of this.
    c1->cd(1);

    // Create a new graph with error bars from the data file.
    TGraphErrors *tge = new TGraphErrors("../Finale/data_abschlussbericht.dat");

    // Draw the graph with axes and points.
    tge->Draw("ap");

    // Apply a linear fit, first order polynomial.
    tge->Fit("pol1");

    // Label the axes.
    tge->GetXaxis()->SetTitle("Spannung/[V]");
```

```
    tge->GetYaxis()->SetTitle("Strom/[A]");

    // Update the canvas to accommodate the changes of the plot.
    c1->Update();

    // Save the whole canvas to a PDF file.
    c1->Print("histogram.pdf");
}
```

ROOT gibt dann folgende Parameter für den Fit aus (Tablle 11.1).

Chi2	=	48.4851		
NDf	=	17		
p0	=	−0.153455	±	0.0434887
p1	=	2.04459	±	0.00707916

Tabelle 11.1: Parameter des Fits

ROOT zeichnet auch direkt die Linie ein. Man kann ihr Erscheinungsbild in der GUI dann auch noch verändern.

Abbildung 11.3: Messwerte mit linearem Fit

11.2 Kugel im Öltank

Für $\alpha \neq 0$ kann man die Funktionen einfach plotten. Für die kleine Masse habe ich blaue Farbe benutzt, für $m = 1{,}5\,\mathrm{kg}$ rote Farbe. Das kleine α ist mit einer dünnen Linie, das große α mit einer dicken Linie dargestellt. Die ganz dünne schwarze Linie zeigt $\alpha = 0$.

Dabei braucht man gar nicht die Taylorreihe zu entwickeln, sondern auf den Aufgabenzettel der Physik zu schauen. Dort hat man bereits die Formel

$$\dot{\vec{v}} = \vec{g} - \frac{\alpha}{m}\vec{v}$$

hergeleitet. Fällt das α weg, so hat man einfach nur eine ganz normale Fallbewegung. Dann braucht man nur noch $v(t) = g * t$ einzutragen.

Abbildung 11.4: Zeit-Geschwindigkeits Plot für die Kugel im Öltank

Listing 11.3: Code für Kugelaufgabe

```
{
    // Create an empty image.
    TCanvas *c1 = new TCanvas("c1", "c1",30,113,800,600);
    c1->Range(-6.200942,-66.29705,56.27944,587.529);
    c1->SetBorderSize(2);
    c1->SetFrameFillColor(0);

    // Plot the linear acceleration function as a reference. Since it grows the
```

```
    // most, it should be set first, in order to get the axes right.
    TF1 *f5 = new TF1("f5","10*x",0,50);
    f5->SetTitle("Kugel");
    f5->SetFillColor(19);
    f5->SetFillStyle(0);
    f5->SetLineWidth(1);

    // Label the axes.
    f5->GetXaxis()->SetTitle("Zeit/[s]");
    f5->GetYaxis()->SetTitle("Geschwindigkeit/[m/s]");
    f5->Draw("");

    // Now draw a=0.1, m=0.5.
    TF1 *f1 = new TF1("f1","0.5*10/0.1*(1-exp(-0.1*x/0.5))",0,50);
    f1->SetFillColor(19);
    f1->SetFillStyle(0);
    f1->SetLineColor(2);
    f1->SetLineWidth(2);
    f1->Draw("same");

    // Now draw a=0.1, m=1.5.
    TF1 *f2 = new TF1("f2","1.5*10/0.1*(1-exp(-0.1*x/1.5))",0,50);
    f2->SetFillColor(19);
    f2->SetFillStyle(0);
    f2->SetLineColor(4);
    f2->SetLineWidth(2);
    f2->Draw("same");

    // Now draw a=0.2, m=0.5.
    TF1 *f3 = new TF1("f3","0.5*10/0.2*(1-exp(-0.2*x/0.5))",0,50);
    f3->SetFillColor(19);
    f3->SetFillStyle(0);
    f3->SetLineColor(2);
    f3->SetLineWidth(3);
    f3->Draw("same");

    // Now draw a=0.2, m=1.5.
    TF1 *f4 = new TF1("f4","1.5*10/0.2*(1-exp(-0.2*x/1.5))",0,50);
    f4->SetFillColor(19);
    f4->SetFillStyle(0);
    f4->SetLineColor(4);
    f4->SetLineWidth(3);
    f4->Draw("same");

    c1->Modified();
    c1->cd();
    c1->SetSelected(c1);

    c1->Update();
    c1->Print("kugel.pdf");
}
```

Teil V

Anhang

Listings

Literaturverzeichnis

[1] Kerningham und Ritchie: „Programmieren in C“

[2] McConnel, Steve: „Code Complete 2“

[3] McConnel, Steve: „Rapid Development“

[4] Manual Page von `tar`.

[5] https://secure.wikimedia.org/wikipedia/de/wiki/Kfz-Kennzeichen_%28Deutschland%29
(Shortlink: http://bit.ly/uaX1nm)

[6] http://www.cppdoc.com/cppdoc_help.html
(Shortlink: http://bit.ly/s5ZAS2)

[7] hadron.physics.fsu.edu/~skpark/document/ROOT/root_beginers/ROOT_for_beginners_Day1.pdf
(Shortlink: http://bit.ly/sezD7a)

[8] http://en.wikibooks.org/wiki/LaTeX/Tables
(Shortlink: http://bit.ly/G5bEH)

[9] http://phacker.org/2009/02/20/minimal-program-to-create-histograms-from-a-file-w
(Shortlink: http://bit.ly/vdvkxp)

[10] http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/WhiteSpace.html
(Shortlink: http://bit.ly/6RH2B1)

[11] http://www.tug.dk/FontCatalogue/
(Shortlink: http://www.tug.dk/FontCatalogue/)

[12] http://www.ijon.de/comp/tutorials/makefile.html
(Shortlink: http://bit.ly/4G6C5K)

[13] http://regexpal.com/

[14] http://kkovacs.eu/cool-but-obscure-unix-tools
(Shortlink: http://bit.ly/ldfI0j)

[15] http://mywiki.wooledge.org/BashPitfalls
(Shortlink: http://bit.ly/VRI2T)

[16] http://www.gnu.org/savannah-checkouts/gnu/make/manual/html_node/Automatic-Variables.html
(Shortlink: http://bit.ly/uDDQgy)

[17] http://docs.oracle.com/javase/1.4.2/docs/api/java/util/regex/Pattern.html
(Shortlink: http://bit.ly/uFNrG6)

[18] http://root.cern.ch/drupal/content/users-guide
(Shortlink: http://bit.ly/6I1KWy)

Erklärung

Hiermit versichere ich, dass ich den vorliegenden Bericht selbstständig angefertigt, und nur die angegebenen Hilfsmittel benutzt habe.

Bonn,	2439532	
Ort/Datum	Martrikelnummer	Unterschrift