

# Vorkurs Mathematik

Sebastian Mai  
Kai Dannies

2011



Dieses Heft wurde vom Fachschaftsrat der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg für den Mathematik-Vorkurs 2011 produziert.

## **Inhaltsverzeichnis**

<b>1</b>	<b>UNIX</b>	<b>4</b>
<b>2</b>	<b>Das ist eine Kapitelüberschrift</b>	<b>7</b>
<b>3</b>	<b>Hello World!</b>	<b>8</b>

# 1 UNIX

Viele der Grundprinzipien von UNIX sind in heutigen Betriebssystemen noch vorhanden und mit einigen dieser Betriebssysteme werdet ihr im Laufe eures Studiums noch zu tun haben.

## 1.1 Verzeichnisstruktur

In Unix wird alles, egal ob es sich um eine Netzwerkschnittstelle, die Festplatte oder eine Musikdatei handelt, als Datei repräsentiert. Alle diese Dateien lassen sich im sogenannten "Verzeichnisbaum" des Betriebssystems finden. Das wohl wichtigste Verzeichnis in UNIX ist "/" – das Wurzelverzeichnis. In diesem Verzeichnis liegen alle anderen Verzeichnisse. An dieser Stelle lernen wir die ersten beiden Kommandos für das Terminal kennen: "ls" und "cd". Sie sind die wichtigsten Kommandos für die Navigation durch den Verzeichnisbaum von Unix. "cd" steht für "change directory" und lässt uns das Verzeichnis wechseln. Gibt man also "cd /" ins Terminal ein, so wechselt man ins Wurzelverzeichnis. "ls" steht für "list search", es liefert eine Auflistung der Dateien im aktuellen Ordner. Da in Unix alles eine Datei ist, sieht man auch die Unterverzeichnisse des aktuellen Verzeichnisses. Die Verzeichnisse im Wurzelverzeichnis sind bei den meisten Betriebssystemen (fast) gleich und jedes dieser Verzeichnisse erfüllt die selbe Bedeutung.

\* / - das Wurzelverzeichnis \* /bin – Ausführbare Programme, z.B. die Shell /bin/sh \* /boot - der Betriebssystemkern und was zum Hochfahren benötigt wird \* /dev - "Devices" zum Beispiel Festplatten, USB-Geräte, der Zufallszahlengenerator ... \* /etc - Konfigurationsdateien für das Betriebssystem \* /home - Das Elternverzeichnis aller Nutzer Verzeichnisse \* /lib - Bibliotheken für andere Programme \* /opt - manuell installierte Programme \* /proc - Systemressourcen \* /root - Das Heimatverzeichnis des "root" Nutzers \* /sbin - das "bin" für Programme die nur root ausführen darf \* /tmp - Temporäre Dateien \* /usr - "unix system resources" Dateien die für alle Nutzer relevant sind \* /var - Variabel

Außerdem gibt es noch Kurzschreibweisen für bestimmte Verzeichnisse. "." bezeichnet das aktuelle Verzeichnis, ".." das Elternverzeichnis und "-" das eigene Homeverzeichnis. Verzeichnisse die mit "." beginnen sind versteckt. Man kann sie anzeigen, wenn man ls mit dem Parameter "a" benutzt.

## 1.2 Arbeiten im Textmodus

Unix stammt aus Zeiten, in denen man einen Rechner mit mehreren "Terminals" bedient hat. Ist man an so einem Terminal angemeldet, sieht man erstmal die Ausgabe des sog. "Command Line Interpreters" – /bin/sh. Dieses Programm ist im wesentlichen dafür zuständig, andere Programme aufzurufen. Hinter dem Kommando "ls" versteckt sich zum Beispiel ein Aufruf des Programms /bin/ls.

Außerdem gibt es noch einige zusätzliche Kommandos, wie "help". Das wichtigste Kommando in Unix ist "man". Mit "man" lassen sich die sogenannten Manpages zu einem Programm anzeigen. In der Manpage stehen alle wichtigen Informationen, die man zu einem Programm braucht. "man man" gibt zum Beispiel eine Hilfeseite zur Benutzung der Manpages aus.

### 1.2.1 Wichtige Befehle

\* cat - gibt den Inhalt einer Datei aus \* cd - Verzeichnis wechseln \* cp - kopiert eine Datei \* date - zeigt das aktuelle Datum und die Uhrzeit an \* echo - gibt die Eingabe zurück \* exit - aktuelle Terminalsession beenden \* gedit - ruft einen Texteditor auf \* grep - Durchsuchen einer Datei \* javac - Javacompiler aufrufen \* java - ein Javaprogramm ausführen \* ls - zeigt den Inhalt des aktuellen Verzeichnisses \* man - zeigt eine Hilfeseite an \* mkdir - legt ein Verzeichnis an \* mv - verschiebt eine Datei \* passwd - eigenes Passwort ändern \* pwd - Zeigt das Verzeichnis an \* rm - Datei löschen \* ssh - eine Shell auf einem anderen Rechner öffnen \* tar - Dateiarhive packen und entpacken \* wget - herunterladen einer Datei \* yes - wiederholt die Eingabe

### 1.2.2 Befehlssyntax

Um einen Befehl richtig ausführen zu können benötigt dieser eine bestimmte Form. Ein typischer Unixbefehl könnte so aussehen: "ls -lBh /". ls ist der ausgeführte Befehl, -l, -B und -h sind Parameter für das Programm ls und "/" der Pfad zum Verzeichnis, dessen Dateien ls auflisten soll. -ist eine abgekürzte Schreibweise für das eigene home-Verzeichnis. Genau so werden die Parameter und der richtige Aufruf von ls auch in der Manpage beschrieben. Mehrere Parameter können oftmals auch zusammengefasst werden, so dass "ls -l -a ..." die selbe Bedeutung hat wie "ls -la .."

### 1.2.3 Befehle verknüpfen

In Unix können auch mehrere Befehle verknüpft werden. Dazu gibt es mehrere nützliche Operatoren. \* & - "command &" führt das Kommando im Hintergrund aus \* && - mit "command1 && command2" kann man mehrere Kommandos hintereinander ausführen z.B. "mkdir foo && ls -l && cd foo/" \* — - mit "command1 — command2" kann man die Ausgabe von Kommando1 in Kommando2 verwenden z.B. "ls -l — grep foo"

Man kann mit diesen Verknüpfungen aber auch sehr viel Schabernack treiben, man sollte davon nicht mehr benutzen als man benötigt, da man so unnötige Fehler vermeiden kann. Weitere nützliche Operatoren sind: \* & - "command &" file schreibt die Ausgabe der Befehls in die angegebene Datei \* && - wie >, hängt die Ausgabe hinten an die Datei an

#### 1.2.4 Vom Textmodus zum Graphikmodus und wieder zurück

Wie man zum Beispiel in den Rechnerpools der Fakultät sieht, besitzen moderne Unix-ide Betriebssysteme nicht mehr ausschließlich den Textmodus. Man kann auch wie gewohnt in Fenstern bunte Bildchen anschauen. Im Gegensatz zu Windows, kann man hier allerdings Programme nicht wirklich von der Shell losgelöst starten – die Ein- und Ausgaben sind nur für den User nicht mehr sichtbar. Besonders deutlich wird wenn man zum Beispiel "firefox" über ein Terminal startet – ob Webbrowser, Spiel oder Programmierumgebung – jedes Programm hängt an der Shell. Die Wichtigkeit der Shell und für viele Entwickler der Grund überhaupt eine Shell zu benutzen heißt Secure Shell. Mit dem Befehl `ssh` kann man nämlich auch eine Shell auf einem anderen Unix-Rechner öffnen und sich beispielsweise von Zuhause im Rechnerpool der FIN einloggen und dort arbeiten – sogar mit Programmen die man lokal gar nicht installiert hat. Mit `ssh -X` kann man sogar Graphikmodus-Programme benutzen. Im Gegensatz zum Remotedesktop werden hier allerdings wesentlich weniger Daten übertragen und der Computer bleibt (einigermaßen) schnell.

## 2 Das ist eine Kapitelüberschrift

### 2.1 Das ist eine Überschrift in einem Kapitel

#### 2.1.1 und so weiter...

Das ist ein wenig einfacher Text, nichts einfacher als das. Für mathematische Ausdrücke wie  $a^b$  benutzen wir diese Dollarzeichen.

- hier
- sind
- Stichpunkte

Und auch eine Tabelle macht oft Sinn:

a	b	c	d
e	f	g	h

Und auch eine Grafik 2.1 will ich dir nicht vorenthalten. Sie wird nicht zwingend genau da auftauchen wo du sie hinschreibst.



Abbildung 2.1: Das ist eine Bildunterschrift

## 3 Hello World!

### 3.1 Wozu "Hello World!"?

Fast alle Bücher, Tutorials, Vorkurshefte und sonstige Programmierbeispiele beginnen mit einem einfachen Programm, das nichts tut als "Hello World!äuf der Konsole auszugeben.

Im Grund behandelt man mit dem "Hello World!Programm zwei Fragen: \* Was und an welchen Stellen kann ich im Quelltext ändern, um ein tolleres Programm zu erhalten? \* Wie komme ich von der Quelltext-Datei zum laufenden Programm?

In den meisten Programmiersprachen gibt es eine Art "Rahmen"der die eigentliche Programmlogik umgibt. In Java sieht das Hello-World-Programm wie folgt aus: 

```
i/pre¿i/code¿public class Hello { public static void main(String args[]) { System.out.println("Hello World!"); } }
```

 // gibt 'Hello World' aus.

Die erste Zeile enthält unter anderem den Namen des Programms nämlich "Hello". Die Datei mit dem Quelltext muss demzufolge "Hello.java" heißen. "public static void main(String args[])" ist der Anfang der Main-Funktion des Programmes. Hier stehen die Befehle, die nach Programmstart als erstes abgearbeitet werden. System.out.println("Hello World!"); – Das ist der Befehl der ausgeführt wird. SSystem.out.printlnist der Befehl der Text ausgibt, "Hello World!"der Text den wir ausgeben wollen. Die beiden Schließenden Klammern sind das Ende der Main-Funktion und das Ende des Programmes. Wenn wir also ein anderes Programm schreiben wollen, müssen wir lediglich den Befehl SSystem.out.println ..."durch andere Befehle ersetzen.

Um das Programm auszuführen, müssen wir zuerst den Quelltext des Programms in Bytecode umwandeln lassen und dann die Java-Virtual-Maschine aufrufen, die den Bytecode aufführt. Für das Übersetzen des Quelltextes in Bytecode ist der Java-Compiler zuständig. Unter UNIX ruft man den Java-Compiler aus dem Terminal mit "javac Hello.javaäuf. Der Compiler hat nun, sofern das Programm keine Fehler enthält eine Datei mit dem Namen "Hello.classërzeugt (oder eine ältere Version überschrieben) – diese Datei enthält den Bytecode. Die JVM kann man mit dem Befehl "javaäufrufen. "java Hello"führt den Bytecode in Hello.class aus.