Vorkurs Mathematik

Sebastian Mai
Kai Dannies

2011

Fachschaftsrat der Fakultät für Informatik der Otto-von-Guericke-Universität Magdeburg

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

Inhaltsverzeichnis

1 UNIX

Viele der Grundprinzipien von UNIX sind in heutigen Betriebssystem 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 "Verzeichnissbaum"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: "lsünd "cd". Sie sind die wichtigsten Kommandos für die Navigation durch den Verzeichnissbaum von Unix. "cdßteht für "change directoryünd lässt uns das Verzeichnis wechseln. Gibt man also "cd /ïns Terminal ein, so wechselt man ins Wurzelverzeichnis. "lsßteht 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 - "Devicesßum Beispiel Festplatten, USB-Geräte, der Zufallszahlengenerator ... * /etc - Konfigurationsdateien für das Betriebssystem * /home - Das Elternverzeichnis aller Nutzerverzeichnisse * /lib - Bibliotheken für andere Programme * /opt - manuell installierte Programme * /proc - Systemressourcen * /root - Das Heimatverzeichnis des "rootNutzers * /sbin - das "bin"für Programme die nur root ausführen darf * /tmp - Temporäre Dateien * /usr - ünix system resources"Datein 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 - Dateiarchive 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 Datein ls auflisten soll. -ïst eine abgekürzte Schreibweise für das eigene home-Verzeichnis.i 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 ¿ fileßchreibt 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 Unixoide 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 ßsh"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 ßsh -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 Kapital

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: ¡pre¿ ¡code¿public class Hello public static void main(String args[]) System.out.println("Hello World!"); // gibt 'Hello World' aus. ¡/pre¿ ¡/code¿

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[]) ïst 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.println̈st 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 aufsführt. 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.

4 Variablen und Datentypen

Computerprogramme weisen im Wesentlichen drei relativ unabhängige Komponenten auf: Eingabe, Verarbeitung und Ausgabe. Bis jetzt haben wir uns nur mit der Ausgabe beschäftigt. Die einfachste Form der Eingabe kennen wir auch schon: Wir setzen Werte im Quelltext. Ein Beispiel dafür ist unser "Hello WorldProramm, indem wir festlegen das genau "Hello Worldünd nicht "foobaräusgeben werden soll. Um komplexere Eingaben, Verarbeitung und die Ausgabe dieser veränderlichen Daten zu ermöglichen brauchen wir eine Möglichkeit Daten abzuspeichern und zu bearbeiten, die vor dem Programmaufruf noch nicht bekannt sind. Genau dies erlaubt das Konzept der Variablen. Eine Variable ist ein Platzhalter für einen dieser unbekannten Werte.

Ersetzen wir im "Hello WorldProgramm die Zeile ¡pre¿¡code¿System.out.println("Hello World"); // gibt "Hello Worldäus ¡/pre¿ ¡/code¿ durch ¡pre¿¡code¿String text; // legt eine Variable mit der Bezeichnung text an. text = "Hello World"; // weist der Variable text den Wert "Hello Worldßu. System.out.println(text); // gibt den Wert der Variablen text aus. ¡/pre¿ ¡/code¿ Erhalten wir ein Programm, dass das selbe tut wie das ursprüngliche "Hello WorldProramm. Dieses Programm ist allerding besser als das alte, denn die Ausgabe ist jetzt von der Eingabe (im Quelltext) getrennt.

4.1 Befehle

Wie wir sehen ist das "Hello WorldProgramm länger geworden. Um Variablen zu manipulieren braucht man nämlich Befehle. Diese Befehle kann man daran erkennen, dass sie mit einem Semikolon enden. Abgearbeitet werden sie in der Reihenfolge in der man sie liest. Im Quelltext oben stehen drei Befehlsaufrufe: Die Deklaration einer Variablen, eine Wertzuweisung und ein Funktionsaufruf. Das sind drei der wichtigsten Befehlsarten.

4.1.1 Deklaration und Definition

Um mit Variablen arbeiten zu können braucht man zwei Schritte: Deklaration und Definition.

SSei p eine Primzahlëntspricht in etwa dem, was man bei der Deklaration einer Variable tut. Um Werte digital abspeichern zu können muss man Wissen, um was für Daten es sich Handelt. Eine Zahl muss anders abgespeichtert werden, als zum Beispiel eine Zeichenkette wie "Hello World". Glücklicherweise müssen wir als Programmierer uns keine Gedanken mehr darum machen, wie die Daten intern repräsentiert werden. Um auszudrücken um was für Daten es sich handelt gibt es sogenannte Datentypen. Eine kleine Auswahl der verfügbaren Typen findest du in der Tabelle.

todo: Tabelle mit Datentypen hier einfügen.

Der Befehl für die Definition einer Variablen sieht so aus: "Typname variablenname;"Beachtet, das Variablennamen üblicherweise kleingeschrieben werden.

Jetzt kennt der Compiler den Typ der Variablen. Damit man auch etwas mit Variablen anfangen kann braucht die Variable noch einen Wert – die Wertzuweisung heißt Definition der Variablen. Eine Zuweisung sieht so aus: "variablenname = Wert;". Wichtig ist dabei vor allem eins: Das -älso der Zuweisungsoperator hat eine (Lese-)Richtung. Der Wert den die Variable bekommt steht immer rechts! Man kann in einer Zuweisung auch den (alten) Wert der Variable oder den einer anderen Variablen verwenden: "p = 2 * p + q;".

Außerdem gibt es noch zwei Kurzschriebweisen, die oft verwendet werden: "x += y;ëntspricht "x = x + y;"(das funktioniert auch mit "*", "/", x++;ënstpricht "x = x + 1;"(genauso funktioniert –)

Definition und Deklaration können auch in einer Anweisung gemacht werden: 'String text = "Hello World";'. Beachtet, das der Datentyp von Rechter und Linker Seite der Zuweisung übereinstimmen müssen. Es kann allerdings auch sein, dass ihr das gar nicht wollt. Um den Typ einer variablen zu ändern gibt es sogenannte Casts. Dafür gibt es verschiedene Methoden die vom Datentyp abhängen. Integrale Datentypen wie ïntöder "char"könnt ihr ineinander umwandeln, indem ihr den eingeklammerten Namen des Zieldatentyps vor die Variable schreibt: "char c = (char) intvariable;". Das "(char)ßorgt dann dafür, dass aus der Ganzzahl ein Buchstabe gemacht wird. Für andere Typänderungen gibt es Funktionen.

4.1.2 Funktionsaufrufe

Funktionen sind vorgefertigte Unterprogramme, die einen bestimmten Zweck erfüllen. Eine Funktion kennt ihr schon: SSystem.out.println()". Diese Funktion tut etwas, und gibt dann die Kontrolle wieder an den nächsten Befehl ab. Es gibt aber auch Funktionen, die einen Wert berechnen. Diesen Wert nennt man Rückgabewert. Die Funktion "Math.random()"hat als Rückgabewert einen zufälligen Wert. Wie packt man nun Funktionen in einen Befehl? Interessiert uns der Rückgabewert der Funktion, so benutzen wir die Funktion in einer Zuweisung: "rueckgabewert = funktionxy();Ïnteressiert uns der Rückgabewert nicht, so lassen wir die linke Seite der Zuweisung einfach weg: "funktionxy();". Viele Funktionen, beispielsweise SSystem.out.println()", sind in der Lage Daten weiterzuverarbeiten. Dafür muss der Funktion allerdings mitgeteilt werden, welche Daten gemeint sind. Um diese Daten an die Funktion zu übergeben gibt es die Klammern nach dem Funktionsnamen. Dort kann man die entsprechenden Werte eintragen. Sollen der Funktion mehrere Werte übergeben werden, so müssen diese durch Kommata getrennt werden – die Reihenfolge der Parameter entscheidet dabei darüber, welcher Wert wofür benutzt wird.

4.2 Einlesen von Werten

Um Werte einzulesen braucht man einen Variable vom Typ SScanner". Mit dem Scanner kann man nicht nur von der Standarteingabe, sondern auch aus Dateien, Netzwerkverbindungen und anderen Quellen Daten einlesen. Deswegen muss man Angeben, dass der Scanner die Daten aus SSystem.in"beziehen soll. Danach kann man mit scannervariable.next() die nächste Eingabe abholen. Im Quelltext könnte das zum Beispiel so aussehen: ¡pre¿¡code¿import java.io.*; import java.util.Scanner;

public class Hello public static void main(String args[]) Scanner scanner; // Beachtet den unterschied zwischen Datentyp('S') und Variable('s')! scanner = new Scanner(System.in); // Legt einen Scanner an, der System.in überwacht String name = scanner.next(); System.out.println("Hello "+ name + "!"); ¡/pre¿¡/code¿ Damit ist unser "Hello WorldProgramm, schon richtig gut, denn es kann nun auf Nutzereingaben reagieren.