Einführung in das Programmieren unter Unix

Ole Klein

Interdisziplinäres Zentrum für Wissenschaftliches Rechnen
Universität Heidelberg
Im Neuenheimer Feld 368
D-69120 Heidelberg
Telefon: 06221/54-8865
E-Mail: ole.klein@iwr.uni-heidelberg.de

(Bildquelle im folgenden: Wikipedia)

Wintersemester 2011/2012

Wozu dieser Vortrag?

In über 90% der Arbeitsgruppen in Mathematik, Physik und Informatik, die Computer einsetzen, kommt Linux zum Einsatz. Dadurch sind viele früher oder später gezwungen, sich mit Unix zu beschäftigen.

Linux ist mit Abstand das am häufigsten genutzte Unix. Über mit Linux betriebene Webserver werden nicht nur große Teile des Internet zur Verfügung gestellt, grob 95% der 500 schnellsten Rechner der Welt basieren auf Linux, und alle unter den zehn schnellsten. Wer mit Rechnern wissenschaftlich arbeiten will, kommt an Unix nicht vorbei.

Wer sich mit den eher praktischen Aspekten der Informatik beschäftigen will, erhält mit Linux ein kostengünstiges System, in dem praktisch alles offen ist und auf die Funktionsweise hin überprüft werden kann. Da man Programmieren am ehesten durch Übung und Beispiele lernt, ist eine große Sammlung von Code “aus dem wirklichen Leben” von unschätzbarem Wert.

Was ist Unix?

Im engeren Sinne Unix (1969), ein Betriebssystem, das viele Sachen popularisierte, die in heutigen Betriebssystemen selbstverständlich sind.

Ansonsten ein unixoides System, also ein Betriebssystem, das von Unix inspiriert ist und sich in weiten Teilen auch so verhält, z.B.:

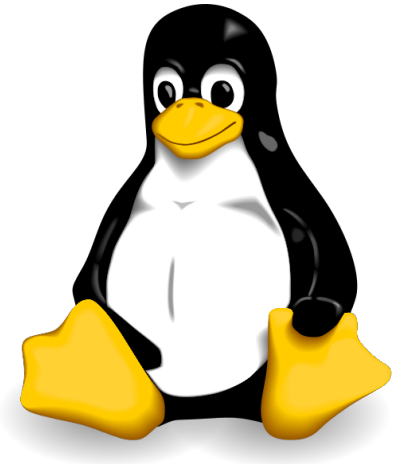
- {Free,Net,Open}BSD, basierend auf einer Unix-Variante von der UC Berkeley
- Mac OS X, das wiederum auf Teilen von FreeBSD und NetBSD basiert
- iOS (iPhone,iPod,iPad), aus Mac OS X hervorgegangen
- Linux (1991), streng genommen nicht mit Unix verwandt, ahmt das System jedoch in praktisch allen Belangen nach
- Android, basiert auf einem von Google stark modifizierten Linux-Kernel
- Zahllose weitere sogenannte Embedded Systems, z.B. Embedded Linux in Netzwerkhardware (Router, Access Points, Firewalls), Videorekordern (TiVo) und Industrierobotern.

Was ist Linux? (I)

Mit GNU (GNU's Not Unix) wurde von Richard Stallman (MIT) und anderen Hackern ab 1983 ein Freies Betriebssystem zu entwickelt, womit folgendes gemeint ist:

- **Freedom 0:** The freedom to run the program for any purpose.
- **Freedom 1:** The freedom to study how the program works, and change it to make it do what you wish.
- **Freedom 2:** The freedom to redistribute copies so you can help your neighbor.
- **Freedom 3:** The freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits.

Was ist Linux? (II)



Während das GNU Project ein praktisch vollständiges System entwickelt hatte, fehlte ausgerechnet der Kernel (Innerstes eines Betriebssystems).

Diese Lücke wurde von Linus Torvalds durch die Entwicklung des Linux-Kernels (1991) geschlossen. Durch Kombination der beiden Projekte entstand GNU/Linux, meist einfach als “Linux” bezeichnet.

Obwohl die Entwicklung von Linux in tausenden unabhängigen und teilweise konkurrierenden Projekten stattfindet, ist ein Betriebssystem entstanden, das praktisch nur in zwei Bereichen Wünsche offen lässt (Digitale Bildbearbeitung und Computerspiele).

Die Shell (I)

Linux lässt sich heutzutage problemlos graphisch bedienen und verhält sich dann in weiten Teilen wie die bekannten Desktops von Windows und Mac OS. Dabei hat man die Wahl zwischen verschiedenen Oberflächen, die Windows nachahmen, Mac OS nachahmen oder gänzlich neue Wege gehen.

Es gibt jedoch trotzdem Gründe, sich mit der bekannten schwarz-weißen Texteingabe, der Shell, zu befassen:

- Man ist effizienter und produktiver, sobald man die Syntax beherrscht (was zugegebenermaßen dauert)
- Langwierige und monotone Arbeiten können automatisiert werden, statt tausend Mal zu klicken
- Einige extrem praktische Programme werden per Texteingabe gesteuert (locate, grep, ...)

Die Shell (II)

Zwei wichtige Konzepte erleichtern die Eingabe sehr:

- Über Platzhalter kann man oft Worte zusammenfassen, so steht zum Beispiel a*.txt für alle Textdateien, die mit “a” beginnen, b{0..9}.tex für die zehn Dateien “b0.tex” bis “b9.tex” und {c,d}.jpg für “c.jpg d.jpg”. Das ist vor allem dann nützlich, wenn die Dateinamen sehr lang und unhandlich sind.
- Die sogenannte Tab Completion erlaubt eine überraschend schnelle Eingabe gerade von besonders langen Zeichenketten. Wenn man die Tabulator-Taste drückt, wird die Zeichenkette automatisch so weit ergänzt, wie sie eindeutig ist. Statt

  /home/foo/studium/wintersemester2011/hausarbeit/quellen/

  vollständig abzutippen, kann bereits

  /h⟨TAB⟩f⟨TAB⟩s⟨TAB⟩w⟨TAB⟩h⟨TAB⟩q⟨TAB⟩

  ausreichen, was die Eingabe von 13 statt 57 Zeichen (23%) erfordert.

Einige wichtige Befehle: man

Der wichtigste Befehl ist eindeutig man, denn er steht für das Wort “Manual”, also Anleitung.
Man erhält Anleitungen zu allen möglichen Befehlen, indem man in der Shell

man ⟨Name des Befehls⟩

eingibt. man man liefert entsprechend eine Anleitung zum Lesen von Anleitungen, quasi eine Meta-Anleitung. Man muss aber eigentlich nur wissen, dass man mit einem kleinen q zurück zur Shell kommt.

Wer zu viel Zeit hat, kann die Anleitung zur Standardshell von Linux, der bash lesen. Empfehlenswerter ist jedoch die für ddate, einem der nützlichsten Befehle (vgl. `http://en.wikipedia.org/wiki/Ddate`).

Einige wichtige Befehle: ls und cd

Um sich den Inhalt von Verzeichnissen anzeigen zu lassen, benutzt man ls (“list”). Leicht zu merken ist zum Beispiel

ls -halt

Dabei stehen die Buchstaben für “human readable”, “all”, “long” und “time”. Es werden also alle, auch versteckte, Dateien in einem besonders ausführlichen Format aufgelistet, und zwar nach Zeit sortiert (nicht lexikographisch) und leicht lesbar (und daher nicht zur automatischen Weiterverarbeitung geeignet).

Um in ein anderes Verzeichnis zu kommen, benutzt man cd (“change directory”). Hinter dem Befehl gibt man das neue Verzeichnis an. Falls man danach direkt wieder in das ursprüngliche Verzeichnis möchte, muss man nicht den ganzen Namen erneut eintippen, sondern benutzt den Platzhalter -.

Einige wichtige Befehle: cp und mv

Unter Unix verwendet man den Befehl cp (“copy”) zum Kopieren von Dateien. Will man einen ganzen Ordner samt Inhalt kopieren, so gibt man hinter zwischen dem Befehl und den zu kopierenden Dateien noch ein -r (“recursive”) an:

cp -r ⟨Name des Verzeichnisses⟩ ⟨Zielort⟩

Mit mv (“move”) erreicht man, dass die Datei verschoben statt kopiert wird.

Im Gegensatz zu einigen anderen Sachen lassen sich Kopieren und Verschieben normalerweise auch unter der graphischen Oberfläche schnell erledigen.

Einige wichtige Befehle: cat und less

Der Befehl cat gibt Textdateien auf dem Bildschirm aus. Dabei kommt der Name von “concatenate”, weil das Programm auch dazu benutzt werden kann, mehrere Dateien aneinander zu hängen.

Meist besser geeignet ist das Programm less. Es heißt so, weil es im Gegensatz zu seinem Vorgänger more den Text in beide Richtungen scrollen kann...

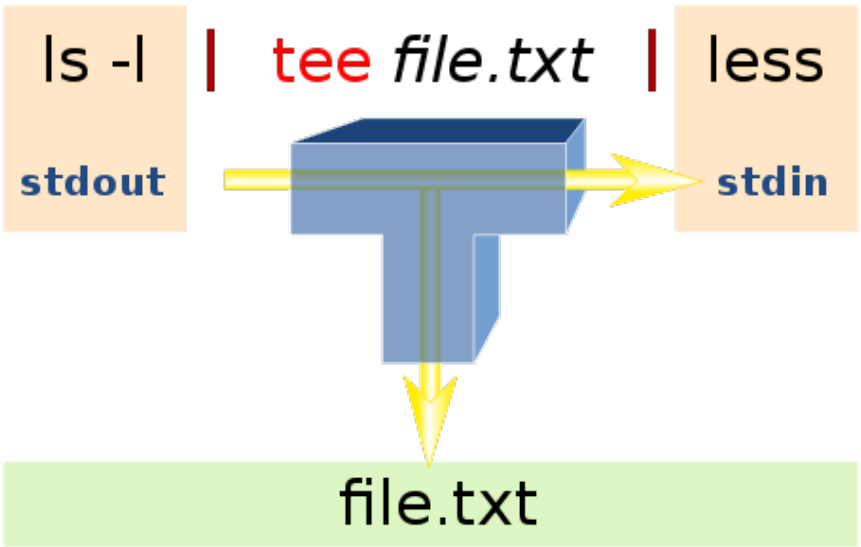
less ist das Programm, das von man zur Anzeige benutzt wird. Entsprechend ist die Bedienung gleich: mit / kann man das Dokument durchsuchen, mit n kommt man zur nächsten Fundstelle, mit q wird die Anzeige beendet.

Pipes und Redirections

Oft möchte man die Ausgabe von Programmen (nicht nur bereits vorhandenen, auch den selbst geschriebenen) weiterverwenden zu können. Dazu gibt es die Möglichkeit, die Ausgabe an ein anderes Programm oder in eine Datei weiterzuleiten. Dazu schreibt man

⟨Erstes Programm⟩ | ⟨Zweites Programm⟩ > ⟨Dateiname⟩

Man kann sich diese Verbindungen als Röhren für Daten vorstellen, darum heißen sie auch Pipes. Entsprechend kann man auch ein T-Stück anflanschen, dafür benutzt man ein Programm namens tee.



Einige wichtige Befehle: grep

Ein besonders wichtiger Befehl ist grep. Mit diesem Programm kann man sehr schnell nach Textstellen in Dateien suchen, was nicht nur beim Programmieren hilfreich ist. Zum Beispiel kann man in Sekunden alle Mails der vergangenen Jahre nach Schlüsselwörtern durchsuchen, oder die Ausgabe eines anderen Programms in Echtzeit filtern, um aus einer Flut an Statusmeldungen nur die Information angezeigt zu bekommen, an der man gerade interessiert ist.

grep ⟨Begriff⟩ ⟨Dateien⟩ bzw. ⟨Programm⟩ | grep ⟨Begriff⟩

Es werden alle Zeilen ausgegeben, die den Begriff enthalten.

Interessant sind vor allem die Optionen -i (Ignoriert Groß-/Kleinschreibung) und -v (invertiert die Auswahl, gibt also alles aus, was den Begriff nicht enthält).

Editieren von Programmcode

Unter Unix gibt es sehr mächtige Editoren, mit denen man mit zwei, drei kurzen Befehlen praktisch alles Vorstellbare machen kann. Die beiden bekanntesten sind vim und Emacs.

Die Befehle sind teilweise kryptisch, so kann man unter vim einen ganzen Text passend einrücken, indem man gg=G eingibt. Wer Spaß an so etwas hat, sollte es sich näher anschauen, es gibt jedoch auch einfachere Editoren. Beliebt sind zum Beispiel Gedit (GNOME) und Kate (KDE).

(Hier würde normalerweise der Teil beginnen, in dem erklärt wird, was man denn nun mit diesem Editor schreibt, aber da das Teil der Vorlesung bzw. des Programmierkurses ist, lassen wir das.)

Compilieren von Programmen

Wenn man ein Programm geschrieben hat, muss man es in eine Form bringen, die der Rechner versteht. Diesen Vorgang nennt man Kompilieren, und entsprechend wird er von einem Compiler ausgeführt.

Ein guter Editor (vim, Emacs) bietet die Möglichkeit, dies automatisch zu machen und bei Fehlern sogar direkt in die Zeile zu springen, in der Probleme beim Übersetzen auftraten.

Man kann ein Programm jedoch auch von Hand übersetzen, und im einfachsten Fall geht das so:

g++ -o ⟨Zielprogramm⟩ ⟨Dateien, die den Code enthalten⟩

Ausführen von selbstgeschriebenen Programmen

Im Gegensatz zu den bisher vorgestellten Programmen muss man bei den eigenen Programmen eine Kleinigkeit beachten, zum Ausführen benutzt man

$$./\langle\text{Programmname}\rangle \text{ statt } \langle\text{Programmname}\rangle$$

Das hat sicherheitstechnische Gründe, man muss sich nur merken, dass man den gesamten Pfad angeben muss (der Punkt steht dabei für das aktuelle Verzeichnis).

Wubi

Wir empfehlen, die Programmieraufgaben unter Unix durchzuführen. Da die meisten einen Windows-Rechner besitzen und nicht jeder sich ein Linux installieren möchte, weisen wir kurz auf den Windows Ubuntu Installer (Wubi) hin.

Wubi kann man wie jedes andere Windows-Programm installieren und deinstallieren, es richtet ein Linux in einer "künstlichen" Festplatte ein, die als gewöhnliche Datei unter Windows liegt.

Weitere Informationen zu Wubi finden sich unter `www.ubuntu.com/download/ubuntu/windows-installer`.

Weiterführende Informationen

Wer sich das ganze etwas genauer anschauen möchte, aber keine Lust hat, all die Befehle auswendig zu lernen, ist mit einem sogenannten Cheat Sheet gut beraten. Darauf kann man die wichtigsten Befehle schnell nachschlagen, bis man sie oft genug benutzt hat, um die richtigen Verknüpfungen im Kopf zu haben.

Diese Hilfestellungen findet man mit den Suchbegriffen

- cheat sheet
- refcard
- oder Referenzkarte

im Internet, oft als geTEXtes PDF, das man sich ausdrucken kann, oder als großes Bild. Dabei kann man auch gezielt nach Themengebieten suchen, also zum Beispiel bash (die Standard-Shell), vim, Emacs, aber auch LaTeX.