Kurze Unix-Einführung

Version vom 8. Juni 1999

Keine Angst vor Unix! Hier ist das Wichtigste, um sofort loslegen zu können:

Dateiverwaltung

Eine **Datei** ist die kleinste Einheit in einem Computer, vergleichbar etwa mit einem Kapitel in einem Buch. Jede Datei hat einen eigenen Namen. Beispiel: `denzo.log`.

Bei Unix wird die Groß- und Kleinschreibung beachtet. Die Datei `denzo.log` ist also eine andere als `Denzo.log`.

Es gibt verschiedene Arten von Dateien: **normale Dateien**, **Programme** und **Verzeichnisse**.

In den **normalen Dateien** wird Information gespeichert: meist Texte oder Bilder.

Die **Programme** sind Dateien, die den Computer dazu bringen, etwas zu tun. Dazu muß der Name des Programms und die „Enter"-Taste eingetippt werden. Um zu zeigen, daß der Computer aufnahmebereit ist und auf eine Eingabe wartet, blendet er einen sogenannten Promt ein. Das ist oft ein `$` oder ein `%`. Manchmal wird auch zusätzliche Information angegeben, so z. B. der Name des jeweiligen Rechners.

Ein Beispiel für ein Programm: Um das aktuelle Datum zu erfahren, kann der Befehl `date` benutzt werden:

```
% date (Enter-Taste)
Fri Jun 4 16:17:42 MET DST 1999
%
```

In den **Verzeichnissen** sind andere Dateien, so wie in Bücherregalen Bücher stehen und in Büchern Kapitel zu finden sind. Und genau wie in einer Bibliothek die Bücher (und darin die Kapitel) sind in Verzeichnissen die Dateien hierarchisch geordnet. Als Beispiel könnte die Fundstelle eines bestimmten Kapitels so beschrieben werden:

`/Ausleihe/Viertes_Regal_links/Erstes_Buch/Zweites_Kapitel`

Genauso wird die Fundstelle einer Datei angegeben:

`/diska2/xprakt/Lysozym/images/1.img` gibt an, wo sich die Bild-Datei

namens `1.img` befindet. Diese Fundstellenangabe wird „der **Pfad** der Datei“ genannt. Bei Unix wird der normale Schrägstrich (`/`) verwendet, um die einzelnen Verzeichnisnamen voneinander zu trennen.

So wie die Leser durch eine Biblothek gehen, um sich Bücher zusammen zu suchen, können die User sich durch das Dateisystem (die Gesamtheit aller Pfade) bewegen: Mit **`cd Verzeichnisname`** (change directory) können sie in ein Unterverzeichnis gehen, und mit **`cd ..`** kommen sie wieder zurück.

Das Verzeichnis, in dem sie gerade arbeiten, wird ihr **Arbeitsverzeichnis** genannt. Der Pfad des aktuellen Verzeichnisses läßt sich mit **`pwd`** (print working directory) anzeigen.

Viele Befehle wirken nur auf Dateien, die in diesem Arbeitsverzeichnis sind. So läßt sich eine Übersicht über alle Dateien und Unterverzeichnisse in dem jeweiligen Arbeitsverzeichnis mit dem Befehl **`ls`** (list) erhalten.

Diesem Befehl kann aber noch noch zusätzlich angegeben werden, daß ein anderes Verzeichnis gemeint ist:

```
% ls /diska2/xprakt/
```

Diese Zusatzangaben werden **Argumente** genannt.

Ganz praktisch sind die Abkürzungen für häufig verwendete Pfadnamen: „`.`“ für das Arbeitsverzeichnis, „`..`“ für das übergeordnete Verzeichnis, und „`/`“ für das alleroberste Verzeichnis. Achtung! Deshalb ist die Eingabe „`diska2/xprakt`“ unterschiedlich von „`/diska2/xprakt`“.

Weitere Programme/Befehle, die sehr nützlich sind:

Kopieren einer Datei: **`cp alte_Datei neue_Datei`** (copy). Evtl. mit Pfadangaben: **`cp alte_Datei /Pfad/neue_Datei`**.

Löschen einer Datei: **`rm Dateiname`** (remove) — Vorsicht: die Datei wird unwiderruflich und meist ohne Nachfrage gelöscht!

Ein Verzeichnis löschen: **`rmdir Verzeichnisname`**

Ein neues Verzeichnis wird mit **`mkdir Verzeichnisname`** (make directory) erstellt.

Eine Textdatei kann mit `more` angezeigt werden. **`more Dateiname`**. Mit der Leertaste kann weitergeblättert werden und mit **`q`** (quit) wird das Programm verlassen. Komfortabler ist jedoch der Einsatz eines Editors (siehe unten).

Eine Datei drucken: Je nach Computer **`lp Dateiname`** oder **`lpr Dateiname`** Mit dem Zusatz `-PDruckername` kann ein bestimmter Drucker angegeben werden: **`lp(r) -PDruckername Dateiname`**. Solche Zusätze, die mit einem Minuszeichen beginnen, werden übrigens **Optionen** genannt.

Hilfe zu einem Programm: **`man Programm`** (manual). Dabei wird die Programmbeschreibung mit `more` (oder einem sehr ähnlichen Programm) geöffnet.

Es gibt viele Bücher mit einer ausführlichen Einführung in Unix, so z. B. „Unix — ein praktischer Einstieg“ von Grace Todino, John Strang und Jerry Peek, O'Reilly Verlag, Köln, 1. Aufl. 1996.

Sehr empfehlenswert sind auch die http-WWW-Seiten `www.uni-koeln.de/RRZK/Systeme/Unix/` und `tech.ilp.physik.uni-essen.de/linux/LDP/` (dort besonders: „The Linux Users' Guide“ von Larry Greenfield), sowie der „Crash-Kurs Linux“ von Winfried Trümper (`www.guug.de/~winni/linux/`)

Ein Allround-Talent der Dateiverwaltung ist der Midnight Commander. Er wird mit dem Kommando **`mc`** gestartet. Für Gelegenheitstäter unbedingt zu empfehlen! Leider ist er auf vielen Unix-Rechnern nicht installiert. Wer das Progamm hat, sollte **`mc`** aufrufen und mit der `F1`-Taste die Hilfe starten.

Editoren

Editoren sind Programme, mit denen Texte geschrieben und verändert werden. Es gibt auf jedem Unix-Rechner **`vi`**, und auf fast jedem **`emacs`**. Dazu existieren noch eine ganze Reihe sehr einfacher Editoren, z. B. **`ae`**, **`ee`**, **`joe`**, **`jot`** und **`pico`**. Sie sind selbsterklärend, aber nicht auf allen Unix-Rechnern installiert. Es ist Geschmackssache, welches Programm bevorzugt wird. Allerdings sollten alle `vi` und `emacs` kennen.

vi: Der `vi` hat zwei verschiedene Zustände: In dem einen kann geschrieben werden, und jede Eingabe wird Text (Eingabemodus). In dem anderen werden Kommandos ausgeführt, z. B. Buchstaben gelöscht, der Text abgespeichert oder das Programm verlassen. Schritt für Schritt sieht das so aus:

1. **`vi Dateiname`**
2. Der `vi` ist zuerst im Kommandomodus. Durch das Drücken der Tasten `i` oder `a` wird in den Eingabemodus gewechselt (insert bzw. append).
3. Tippen. An vielen Unix-Rechnern funktionieren die Cursortasten auch im Eingabemodus.
4. Für alle anderen Kommandos wird durch Drücken der `ESC`-Taste in den Kommandomodus gewechselt. Hier ist es möglich, Zeichen zu löschen (`x`), sich im Text zu bewegen (`h`=links, `j`=runter, `k`=rauf, `l`=rechts), abzuspeichern (`:w Dateiname`) und das Programm zubeenden (`:q`).

5. Die Tastenfolge `ESC:q!` beendet den `vi` **ohne** Abspeichern.

Das Buch zum `vi`: „Learning the vi Editor“ von Linda Lamb, O'Reilly & Associates, Sebastopol, CA, USA, 5. Aufl. 1990.

emacs:

1. **`emacs Dateiname`**
2. Tippen. Wenn die Cursortasten und `Backspace/Delete` nicht ordentlich funktionieren, können auch die Tasten `Control-b`, `Control-f`, `Control-n` und `Control-p` zum Bewegen im Text verwendet werden und `Control-d` zum Löschen (Die Tasten sind leicht zu merken: Sie stehen für backwards, forwards, next line, previous line und delete).
3. Ein Kommando (z. B. `Control-h` für Hilfe) kann mit `Control-g` abgebrochen werden.
4. Rückgängig machen: `Control-/`. Mehrere dieser Rückgängig-Befehle hintereinander machen entsprechend viele der letzten Eingaben rückgängig.
5. Den Text wird mit `Control-x Control-s` abgespeichert.
6. Beenden mit `Control-x Control-c`.

Wer mehr wissen will, sollte **`emacs`** aufrufen und dann das Tutorial mit `Control-h t` starten. Außerdem gibt es natürlich ein Buch: „Learning GNU Emacs“ von Debra Cameron, Bill Rosenblatt und Eric Raymond, O'Reilly & Associates, Sebastopol, CA, USA, 2. Aufl. 1996.

Fehlermeldungen

Die meisten Unix-Programme belästigen die User nicht mit unötigen Meldungen. Wenn das aufgerufene Programm das erreicht hat, was es sollte, bekommen die User eine erneute Eingabeaufforderung und der Computer wartet auf den nächsten Befehl. Ein Beispiel zum Ausprobieren: **`cd /usr`**. `/usr` wird neues Arbeitsverzeichnis, und der Promt erscheint wieder. Rückmeldungen kommen aber dann, wenn **Fehler** gemacht wurden. Diese Fehlermeldungen sind meist recht informativ, so daß sofort klar ist, was falsch gelaufen ist. Häufige Anfängerfehler sind neben „normalen“ Tippfehlern vergessene Leerzeichen oder Schrägstriche. Hilfreich ist es, sich anzusehen, wie die entsprechen Fehlermeldungen lauten. Zum Abtippen: **`dare`** (statt `date`), **`cd/usr`** und **`/usr`** (statt `cd /usr`) und **`/etc/passwd`** (statt `ls /etc/passwd`).

Prozeßkontrolle

Unter Unix können mehrere Prozesse gleichzeitig laufen. Dazu wird ein „&“ hinter den jeweiligen Befehl getippt, und dieses Programm arbeit im Hintergrund. Der Computer gibt dabei zwei Zahlen an: eine laufende Nummer in eckigen Klammern, die Jobnummer, und eine ohne Klammern, die PID (process identification). Ein Job im Hintergrund wird mit **`fg Jobnummer`** wieder an die Oberfläche geholt. Hier werden Jobs mit `Control-z` gestoppt, und **`bg Jobnummer`** bringt sie wieder zurück in den Hintergrund.

Das Programm **`jobs`** gibt die Jobnummern aus, und **`ps`** zeigt eine Liste aller laufenden Prozesse mit ihren PIDs. Soll ein Prozeß vorzeitig beendet werden? Einfach eintippen: **`kill -TERM`** und die entsprechenden PID. Nur ganz hartnäckige Prozesse werden mit **`kill -KILL PID`** beendet.