Universität Zürich[UZH]

Physik-Institut

Kurzeinführung in UNIX

Roland Bernet, Ulrich Straumann

28. Februar 2005

Inhaltsverzeichnis

1 Grundsätzliches

1.1 Einloggen

Beim Einloggen wird man von der KDE-Oberfläche zunächst nach seinem *Usernamen* und dann nach seinem *Passwort* gefragt. Darunter kann man noch auswählen, welchen Window-Manager man möchte. Standard ist derzeit die KDE-Oberfläche, es stehen jedoch auch andere Oberflächen zur Verfügung. Wer eine schnellere, jedoch auch einfachere Oberfläche haben möchte, kann den WindowMaker benutzen.
Man befindet sich automatisch bei jedem Terminal in seinem Home-Verzeichnis.

1.2 Ausloggen

Am Ende der Sitzung muss man sich aus der Grafikumgebung ausloggen. Bei jedem Window-Manager gibt es mehr oder weniger versteckt einen Knopf zum Ausloggen. Beim KDE ist er links unten im Applikationen Menü, beim andern Window-Managern manchmal der letzte Punkt eines Untermenüs beim Fenster-Knopf.
Bei manchen Linux-Installationen kommt man dann allerdings noch auf einen Textbildschirm, auf dem man dann **logout** oder **exit** eingeben muss.
Startet man bereits auf einem Textbildschirm statt in der X-Windows-Oberfläche, so muss man zum Start der Oberfläche entweder **startx** oder **xinit** eingeben.

1.3 Öffnen eines Terminals

Um Befehle eintippen zu können, muss man zuerst ein interaktives Befehlsfenster oder Terminal öffnen. Dazu klickt man am einfachsten auf das Symbol "Bildschirm mit Muschelünten auf der Symbolleiste. Dies öffnet ein Fenster, in dem nun Befehle eingetippt werden können. Das Programm, das diese Befehle übersetzt, heisst Shell (englisch Muschel). Deshalb das Symbol mit dem Bildschirm und der Muschel.

1.4 Passwort ändern

Der Befehl um sein Passwort zu ändern heisst

passwd

Am Physik-Institut der Universität Zürich haben wir einen zentralen **NIS+** Server, der die

Passwörter verwaltet. Daher muss man sein Passwort nach dem Befehl **passwd** zweimal eingeben und wird dann erst nach dem neuen Passwort gefragt. Aus Sicherheitsgründen muss danach das neue Passwort nochmals bestätigt werden.

1.5 Shell

Die Shell ist der UNIX-Befehlübersetzer, normalerweise von einem interaktiven Fenster. Es gibt verschiedene dieser Befehlübersetzer, die üblichsten sind die folgenden:

csh	Berkeley UNIX C Shell
tcsh	Enhanced Berkeley UNIX C Shell
bash	GNU Bourne Shell
ksh	Korn Shell
zsh	Enhanced Korn Shell

Die eigene Shell, die momentan für die Befehlinterpretation gebraucht wird, kann mit

echo $SHELL

oder

printenv SHELL

angezeigt werden.

1.6 Hilfe

Als Standardhilfe gibt es unter UNIX die Manual-Pages:

man *Befehl*

gibt meist eine sehr umfangreiche Hilfe zu dem entsprechenden Befehl aus. Für komplexere Befehle gibt es mehr Information mit dem Befehl

info *Befehl*

Um einen Begriff in den UNIX die Manual-Pages zu suchen, gibt es den Befehl

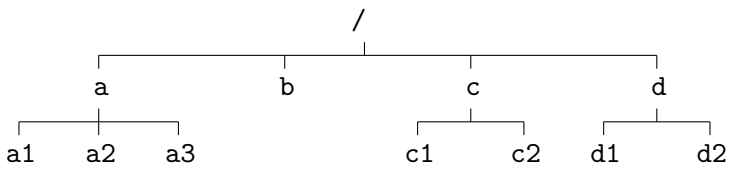
apropos *Begriff*

2 Das UNIX-Filesystem

Objekte wie Texte oder Programme werden auf der Festplatte als Dateien (Files) gespeichert. Das Filesystem ermöglicht die strukturierte Speicherung der Daten, so dass man sie später auch schnell wiederfinden kann.

1. Jede Datei hat einen Namen.
2. Dateien werden in Verzeichnissen (Directories) zusammengefasst.
3. Verzeichnisse können wiederum Verzeichnisse enthalten (so wie in einem Buch ein Kapitel mehrere Unterkapitel enthalten kann).
4. Das oberste Verzeichnis (das sogenannte Wurzel-Verzeichnis (root-directory)) hat immer den Namen `/`.

Durch diese Regeln wird eine Hierachie definiert. Zum Beispiel:



Eine Datei kann einerseits angesprochen werden, indem man den vollständigen Pfad von Wurzelverzeichnis über die Unterverzeichnisse bis zum Dateinamen angibt, jeweils getrennt durch das Zeichen `/`. Also spricht man in unseren Beispiel die Datei `a3` durch die Zeichenfolge `/a/a3` an. Will man das Verzeichnis `d` ansprechen, schreibt man `/d`. Diese Art der Datei-Adressierung nennt man **absolute Datei-Adressierung**.
Die zweite Art der Adressierung, ist die **relative Datei-Adressierung**: Alle UNIX-Befehle, die man eingibt, beziehen sich auf das sogenannte aktuelle Verzeichnis. Beim Start des Terminals ist das aktuelle Verzeichnis das Home-Verzeichnis, das heisst dasjenige Verzeichnis, auf das man selbst Schreib- und Lesezugiff hat. Das Home-Verzeichnis befindet sich irgendwo im Verzeichnisbaum. Damit man es schnell findet, gibt es eine abgekürzte Schreibung: das Zeichen `~` bezeichnet das Home-Verzeichnis.
Man kann mit UNIX-Befehlen ein anderes Verzeichnis zum aktuellen Verzeichnis machen und sich so im Verzeichnisbaum bewegen. Wenn man Daten oder Verzeichnisse im aktuellen Verzeichnis bearbeiten will, muss man keine absolute Adresse angeben. Es genügt, wenn man den Namen des Objektes angibt, das man bearbeiten will.
Am Physik-Institut der Universität Zürich hat man von allen Linux Rechnern Zugang zum gleichen Filesystem. Für Kursteilnehmer befindet sich das Home-Verzeichnis im Verzeichnis `/home/kurs/`*Username*.

3 Arbeiten mit Dateien und Verzeichnissen

3.1 Dateien ansehen

Den Inhalt einer Textdatei kann man ansehen mit

cat *Dateiname*.

Ist die Datei lang, rauscht der Text einfach nur durch. Seitenweise kann man die Datei darstellen mit

more *Dateiname*

oder

less *Dateiname*

Dabei kann man zum Beispiel bei more mit der Return-Taste zeilen- und mit der Leertaste seitenweise vorwärtsblättern.
Will man nur den Schluss einer Datei ansehen, so gibt man

tail *Dateiname*

ein. Die Anzahl der Zeilen lässt sich dabei steuern, zum Beispiel werden mit

tail -2 *Dateiname*

die letzten zwei Zeilen der Datei ausgegeben. Analog funktioniert der Befehl

head

für den Anfang der Datei.

3.2 Verzeichnisse erstellen, wechseln und löschen und den Inhalt anzeigen

Ein Verzeichnis bzw. Ordner bzw. Directory lässt sich mit

mkdir *Verzeichnisname* (mkdir für make directory)

erstellen. In dieses Verzeichnis wechselt man mit

cd *Verzeichnisname* (cd für change directory),

in das höhergelegene mit

cd ..

in das Home-Verzeichnis einfach mit

cd

und in das Root-Verzeichnis mit

cd /

Mit dem Befehl

pwd (pwd für print working directory)

kann man sich anzeigen lassen, in welchem Verzeichnis man sich gerade befindet. Meist sieht man dies allerdings auch am Prompt.

Ein Verzeichnis, das keine Dateien oder Unterverzeichnisse enthält, löscht man mit

rmdir *Verzeichnisname* (rmdir für remove directory).

Ist es nicht leer, so kann man das Verzeichnis inklusive aller Unterverzeichnisse und der darin enthaltenen Dateien mit

rm -rf *Verzeichnisname* (rm für remove)

löschen.

Den Inhalt eines Verzeichnisses kann man sich mit

ls (ls für list)

anzeigen lassen, in der Langform (mit Dateigrösse, Datum, ...) mit

ls -l

Die versteckten Dateien sieht man mit

ls -al

und rekursiv alle Dateien in den Unterverzeichnissen mit

ls -r

Achtung: rm und rmdir können nicht mehr rückgängig gemacht werden, es gibt kein undelete wie unter DOS und kein salvage wie unter Novell Netware!

3.3 Dateien umbenennen, kopieren und löschen

Umbenennen bzw. Verschieben einer Datei an eine andere Stelle im Verzeichnisbaum funktioniert mit

mv *AlterDateiname NeuerDateiname* (mv für move),

Kopieren einer Datei mit

cp *Dateiname1 Dateiname2* (cp für copy),

Löschen einer Datei mit

rm *Dateiname* (rm für remove).

3.4 Befehlsabkürzungen

Es gibt die Möglichkeit Befehle abzukürzen oder umzubenennen. Der Befehl dazu heisst

alias *Befehlsname*=’*Befehl*’

Bei cp umd mv besteht die Gefahr, dass eine Datei überschrieben werden könnte, nämlich wenn eine mit dem neuen Dateinamen bereits existierte. Deshalb bietet es sich an, als Alias jeweils **cp -i** für **cp**, **mv -i** für **mv** und **rm -i** für **rm** zu setzen.

alias cp=’cp -i’
alias mv=’mv -i’
alias rm=’rm -i’

Dann wird jeweils bei derartigen Gefahrensituationen nochmals eine Bestätigung verlangt.

4 Dateiattribute

4.1 Anzeige bei ls -al

Es sind stets 10 Attribute, die man mit **ls -al** mit anzeigt; am Anfang ist ein - für eine Datei oder ein *d* für ein Verzeichnis; dann folgen die Rechte dessen, dem die Datei gehört, nämlich *r (read)*, *w (write)* und gegebenenfalls *x (execute)*, dann die Rechte für die Gruppe, der der Besitzer der Datei angehört (meist die Arbeitsgruppe), dann nochmals dieselben Rechte für die ganze übrige Welt, also zum Beispiel

```
drwxr-x--- jsch cap 512 Apr 10 9:12 myprog
```

ist ein Unterverzeichnis namens myprog, das mir, jsch, gehört, 512 Bytes gross ist und zuletzt am 10. April um 9.12 Uhr verändert wurde. Ich habe alle Rechte, also *rwx*, wobei *x* bei einem Verzeichnis bedeutet, dass ich `cd myprog` machen kann. Die Gruppe cap hat nur die Rechte *r* und *x*, kann also in das Verzeichnis wechseln, den Verzeichnisnamen aber nicht ändern und auch keine neuen Dateien im Verzeichnis anlegen, weil ihnen das *w*-Recht fehlt; die übrige Welt hat keine Rechte.

```
-rwxr----- jsch cap 20000 Apr 11 10:13 beispiel
```

ist ein Programm namens beispiel, das mir gehört; ich kann alles damit machen, die Gruppe kann es sich nur ansehen und damit zum Beispiel kopieren, aber nicht aufrufen, die übrige Welt kann nichts.

```
-rw------- jsch cap 7193 Apr 11 9:12 beispiel.c
```

ist der C-Quellcode dafür, den nur ich lesen und beschreiben kann; eine C-Datei kann man ja nicht ausführen, deshalb ist *x* hier nicht angebracht. Bei

```
-rw-rw---- jsch cap 16385 Apr 11 9:12 javabeispiel.java
```

ist es dagegen so, dass sowohl ich als auch die Arbeitsgruppe den Java-Quelltext lesen und überschreiben können, die restliche Welt hat keinen Zugriff darauf.

4.2 Rechte ändern für User (u), Group (g) und Other (o)

chmod g+r beispiel.c

erlaubt der Gruppe, beispiel.c zu lesen.

chmod u-w beispiel.c

verbietet es mir, die Datei beispiel.c zu überschreiben oder zu löschen. (Nach einer zusätzlichen Abfrage kann die Datei dann doch gelöscht werden.)

chmod a+x beispiel

erlaubt es allen (`a=u+g+o`), das Programm beispiel auszuführen.
Will man von allen Dateien und Verzeichnissen die Rechte für Group und Other entfernen, so gibt man

chmod go-rwx *

ein. `*` bedeutet dasselbe wie unter DOS, hier werden alle Dateien und Directories gleichzeitig in den Rechten verändert. Damit man diesen Befehl nicht immer wieder eingeben muss, weil man ja immer wieder neue Dateien anlegt, schreibt man in eine Einlogg-Datei:

umask 077

5 Kompilieren und Ausführen

5.1 Kompilieren

Die grundlegende Syntax ist bei allen Compilern gleich:

```
Compilername [Optionen] Quellcode [Optionen]
```

Beispiele:

Sprache	Compiler	Extension
Fortran 77	g77	.f
C	gcc	.c
C++	g++	.cc, .CC, .C, .cxx, .cpp
Java	javac	.java

Auf dem Spinor Cluster gibt es keinen Pascal Compiler. Um Pascal Code zu kompilieren, gibt es einen Pascal zu C Konverter:

p2c *PascalDatei*

Anschliessend muss der Code mit einem C Compiler kompiliert werden.

Wichtige Optionen für den Compiler und den Linker (dies gilt nicht für Java):

Option	Beschreibung
-O	Der Code wird dann vom Compiler **optimiert**, wodurch **wesentlich weniger Rechenzeit** benötigt wird. Es gibt im übrigen mehrere Optimierungsstufen, die man mit **-O1, -O2, -O3, ...** einstellen kann.
-o *Dateiname*	Wird diese Option nicht angegeben, so heisst das ausführbare Programm `a.out`. So wird es automatisch in `Dateiname` umbenannt.
-g	Diese Option dient zum Debuggen des Programms: stürzt das Programm unter Erzeugung eines `core`-Files ab, lässt sich mit **dbx** (Solaris) bzw. **gdb** (Linux, GNU-Compiler) der Fehler rasch eingrenzen: Es wird automatisch die Programmzeile ausgegeben, bei der das Programm abgestürzt ist. Mit **print** `Variable` lässt sich der Inhalt einer Variablen ausgeben. Mit **quit** verlässt man den Debugger wieder.
-I*Include-Pfad*	Zusätzlicher Suchpfad für `include`-Dateien, in dem zum Beispiel für `C++-include`-Dateien gesucht wird.
-L*Bibliothekspfad*	Zusätzlicher Suchpfad für Bibliotheksdateien, in dem für Bibliotheken gesucht wird, die mit der Option `-l` angegeben sind.
-l*Bibliothek*	Damit wird eine Bibliothek hinzugelinkt. Am gebräuchlichsten ist **-lm** für die Mathe-Library.
-D *Definition*	Hat man im C-Quelltext die Abfrage `#ifdef`, so kann man das entsprechende Schlüsselwort auch von aussen definieren, ohne im C-Text ein entsprechendes `#define` eingeben zu müssen.

So gibt man beispielsweise zum Kompilieren eines C-Programms

gcc -O beispiel.c -o beispiel -lm

und eines `C++`-Programms

g++ -O beispiel.cc -o beispiel -lm

ein.

Zum Kompilieren eines Java-Programms muss zunächst der Pfad erweitern werden, damit der Compiler gefunden werden kann. Unter der csh oder tcsh geschieht dies zum Beispiel mit

```
setenv PATH /usr/local/j2sdk/bin:$PATH
```

oder unter der zsh, ksh oder bash mit

```
export PATH=/usr/local/j2sdk/bin:$PATH
```

Beim Kompilieren eines Java-Programms durch

javac -O javabeispiel.java

wird eine Reihe von `.class`-Dateien angelegt, eine pro Klasse im Java-Quelltext.

5.2 Programm laufen lassen

Starten kann man ein Programm normalerweise durch die Eingabe des Namens der ausführbaren Datei, also einfach zum Beispiel durch:

beispiel

Bei Java ist dies allerdings anders: Java-Applikationen werden durch

java javabeispiel

gestartet, Java-Applets durch

appletviewer javabeispiel.html

bzw. unter dem Webbrowser.

Allerdings sollte man nur kurze Programme derart im Vordergrund laufen lassen. Oftmals hat man auch Programme, die über Nacht oder sogar tage- bzw. wochenlang laufen. In diesem Fall muss man die Programme im Hintergrund starten (also mit **&**), derart, dass sie sich nicht aufhängen, wenn man den Rechner verlässt (also mit **nohup** für nohangup), und so, dass man nett zu dem ist, der am nächsten Tag am Rechner sitzt, und für ihn die Priorität des Jobs mit **nice** heruntersetzt. Dabei müssen Standard-Eingabe und Standard-Ausgabe umgelenkt werden:

nice +10 nohup beispiel < *eingabe* > *ausgabe* **&**

Damit startet man das Programm im Hintergrund, es läuft von selbst auch nach dem Ausloggen weiter. Das, was man normalerweise an der Tastatur eingeben würde, muss in der richtigen Reihenfolge in der Datei *eingabe* stehen, und das, was am Bildschirm erscheinen würde, schreibt das Programm nun in die Datei *ausgabe*. Will man diese Datei nicht überschreiben sondern an ihr Ende den Programmoutput anhängen, so verwendet man >> statt >.

5.3 Job-Verwaltung

Programme, die im Vordergrund gestartet wurden, kann man mit

Ctrl-C

abbrechen. Will man sie dagegen in den Hintergrund schieben, so muss man sie zunächst mit

Ctrl-Z

stoppen und kann dann mit

bg

sie in den Hintergrund abschicken oder mit

fg

weiter im Vordergrund laufen lassen.

Die Prozesse, die Rechenzeit verbrauchen, kann man sich je nach Shell folgendermassen anzeigen lassen:

ps auxr

Eine laufend aktualisierte Ausgabe erhält man mit

top

Alle eigenen Prozesse filtert man einfach mit

ps aux | **grep** *Username*

heraus.

Manchmal hat man sich beim Starten eines Hintergrundprogramms vertippt oder erkennt an der Ausgabe, dass ein Fehler vorhanden ist. In diesem Fall merkt man sich die mit **ps** bzw. mit **top** angezeigte Prozess-Nummer und stoppt das Programm mit

kill *Prozess-Nummer*.

Falls das nicht funktioniert, mit
kill -9 *Prozess-Nummer*.

6 Editoren

6.1 Editor: emacs

emacs bzw. die X-Window Variante **xemacs** ist ein sehr mächtiger Editor, der fast schon eine Betriebssystemumgebung bietet.

Einige wichtige Befehle sind unten aufgeführt. Dabei bedeutet **C-**, dass die "Ctrl"-Taste und die folgende Taste zusammen gedrückt werden. **M-** bedeutet, dass die "Esc"- bzw. "Meta"-Taste zuerst gedrückt wird und dabach die folgende Taste.

Cursor verschieben:

C-e	Ans Ende einer Zeile springen.
C-a	An den Anfang einer Zeile springen.
M->	Ans Ende des Files springen.
M-<	An den Beginn eines Files springen.

Suchen und ersetzen:

C-s	Suchen (vorwärts).
C-r	Suchen (rückwärts).
M-%	Suchen und ersetzen.

Cut und Paste (Auschneiden und Einsetzen):

C-d	Ein Buchstaben löschen und speichern.
C-k	Ganze Zeile löschen und speichern.
C-y	Paste (Speicher einsetzen).
	Kopieren: Mit Maus und linker Taste hervorheben, dann mit Maus an neuer Position und mittlere Maustaste drücken.

File einlesen, abspeichern usw.:

C-x C-f	Neues File editieren.
C-x i	File einfügen.
C-x s	File abspeichern.
C-x C-c	Emacs verlassen.

Befehle rückgängig machen:

C-_	Editierbefehle rückgängig machen (undo).
C-g	Angefangener Befehl löschen.

Eine Referenzkarte mit den mehr Befehlen ist dem Script beigelegt.

Zusätzliche Informationen können mit Hilfe des Info Befehl erhalten werden.
info emacs

6.2 Weitere Editoren

Daneben gibt es selbstverständlich noch weitere Editoren, es gibt wahre Glaubenskriege, was nun der beste Editor ist. (Dasselbe gilt im übrigen auch für Programmiersprachen.)

joe ist ein WORDSTAR-kompatibler Editor. Mit Ctrl-K-X kann man ihn mit Abspeichern verlassen. Mit Ctrl-K-Q wird er ohne Abspeichern beendet.

kwrite ist ein einfacher Editor mit Pulldown-Menus, der Teil der KDE Oberfläche ist.

nedit ist ein einfacher Editor mit Pulldown-Menus.

6.3 Standard-Editor vi

Der Standard-Editor unter UNIX ist der **vi**:

vi *Dateiname*

lädt eine Datei namens Dateiname, falls sie schon existiert, bzw. startet mit dem Aufbau einer neuen Datei. Normalerweise kann man mit den Cursor-Tasten den Cursor in die verschiedenen Richtungen bewegen.
Dieser Editor ist auf allen UNIX Systemen vorhanden, ist jedoch nicht sehr anwenderfreundlich. Er wird meistens nur gebraucht, wenn kein anderer Editor vorhanden ist.
Beim **vi** unterscheidet man 2 Modi, den Befehlsmodus und den Einfügemodus. Anfangs befindet man sich im Befehlsmodus. Mit den Buchstaben

i für Einfügen (insert)
a für Anhängen (append)
o für neue Linie (open new line)

kommt man in den Einfügemodus. Zurück in den Befehlsmodus geht es mit der Esc-Taste. (Bei manchen Installationen sollte man im Einfügemodus die Cursortasten nicht benutzen.)

Einige wichtige Befehle sind unten aufgeführt. All diese können natürlich nur im Befehlsmodus ausgeführt werden.

x löscht ein Zeichen.
dd löscht eine ganze Zeile.
dw löscht ein Wort (delete word).
7dd löscht 7 Zeilen und kopiert den gelöschten Abschnitt in einen Puffer.
5yy (yank) kopiert die nächsten 5 Zeilen in einen Puffer.
p (paste) fügt den Puffer unterhalb der aktuellen Zeile ein.
r (replace) ersetzt ein Zeichen.
3s ersetzt einen String aus 3 Zeichen.
/Text sucht nach der nächsten Stelle, wo `Text` steht.
:g/alterText/s//neuerText/g ersetzt überall in der Datei `alterText` durch `neuerText`.
:g/alterText/s//neuerText/gc dasselbe, fragt aber an jeder Stelle zusätzlich ab.
:100 springt zur Zeile mit der Nummer 100.
:set nu zeigt die Zeilennummern an,
:set nonu schaltet dies wieder ab.
:r Dateiname kopiert die angegebene Datei an die aktuelle Cursorposition.
:w führt eine Zwischenabspeicherung durch.
ZZ oder **:wq** dient zum Abspeichern und Beenden.
:q verlässt den vi, ohne abzuspeichern.
:q! erzwingt das Beenden des vi, wobei die Veränderungen nicht abgespeichert werden.

Eine Referenzkarte mit den wichtigsten Befehlen ist dem Script beigelegt.

7 Weitere wichtige Befehle

7.1 Plattenplatz

Der eigene Plattenplatz ist oftmals durch Quota beschränkt. Mit

quota -v (v für verbose)

kann man sich ansehen, wieviel Platz belegt ist bzw. wie weit man den Plattenplatz bereits überschritten hat. In diesem Fall hat man Dateien zu löschen.

Auf dem Spinor Cluster gibt es momentan keine Diskquota.

Eine Auflistung, wieviel Platz die Dateien in den verschiedenen Verzeichnissen einnehmen, liefert der Befehl

du -k (Diskusage in Kilobyte).

7.2 Zeitmessung und Datum

time java javabeispiel

zeigt in Standarderror auf dem Bildschirm an, wieviel Zeit **javabeispiel** verbraten hat. Die Umlenkung des Standarderrors auf eine Datei (zum Beispiel `zeitmessung`) erfolgt je nach Shell durch

time java javabeispiel >& zeitmessung

oder durch

time java javabeispiel !> zeitmessung

date

zeigt das aktuelle Datum an,

cal

den aktuellen Monat,

cal 2 2000

den Februar des Jahres 2000. Dieser Kalender gilt vom Jahr 1 n.Chr. bis 9999 n.Chr. und berücksichtigt, dass die USA ihre Zeitrechnung erst im September 1752 auf den Gregorianischen Kalender umgestellt haben.

7.3 Drucken

Mit

lpr *Dateiname*

kann man eine PostScript-Datei oder Text-Datei drucken. Mit

lpq

sieht man den Drucker-Status.
Es empfiehlt sich, Programmtexte zweispaltig zu drucken, wofür es den Befehl

a2ps *Dateiname*

gibt. Um das File als PostScript File abzuspeichern anstatt direkt zu drucken, muss noch die folgende Option angegeben werden:

a2ps -o *PostScriptDateiname TextDateiname*

8 E-mail

Man kann auch unter UNIX mailen:

mail roland.bernet@physik.unizh.ch

schickt eine Mail an mich. Mails müssen immer mit einem einzelnen Punkt am Anfang einer Zeile abgeschlossen werden. Beim Schreiben einer Mail kann man mit

~r *Dateiname*

eine Datei einfügen (Binärdateien mit **uuencode** verschlüsseln!) und mit

~vi

in den vi-Modus umschalten, die E-mail mit dem vi editieren, mit **:wq** abspeichern und mit einem **.** die E-mail abschicken. Mit

mail

kann man die eingegangenen E-mails lesen, indem man dann einfach die entsprechende Nummer eintippt. Unter dieser Mail-Umgebung gibt es zusätzlich folgende wichtige Kommandos:

r (für reply): man schickt eine Antwortmail.

d (für delete): man löscht die soeben gelesene Mail.

x (für exit): man beendet **mail** ohne Abspeichern, alle alten und neuen E-mails bleiben in dem ursprünglichen Zustand erhalten.

q (für quit): man beendet **mail** mit Abspeichern.

Alle gelesenen E-mails werden in einer Datei namens `mbox` abgespeichert.

Selbstverständlich gibt es komfortablere textbasierende Mail-Programme, zum Beispiel **pine**. Ausserdem kann man nat2urlich auch all die verschiedenen graphischen Mail Programme wie **mozilla** oder **kmail** zum Lesen seiner E-mail benutzen.

Mit folgende Einstellung müssen diese Mail-Programme konfiguriert werden um am Physik-Institut der Universität Zürich bzw. von einem Internet Provider Mail zu lesen:

Paramter	Einstellung (Physik-Institut)	Einstellung (Provider)
E-mail Address	*username*@physik.unizh.ch	*username*@*domain*
Incoming Mail Server	mail.physik.unizh.ch	Mail Server Adresse
Mail Server Protokol	IMAP	POP oder IMAP
Incoming Username	*username*	*username*
Outgoing Mail Server	smtp.physik.unizh.ch	SMTP Server Adresse
Outgoing Username	kein Username nötig	*username*
zusätzlich für **pine**		
User Domain	physik.unizh.ch	
Inbox Path	{mail.physik.unizh.ch/ssl}INBOX	

9 Remote-Login und Datentransfer

9.1 Remote-Login

Mit

telnet *remote-machine*

kann man sich in eine andere Maschine einloggen. Man wird nach dem dortigen Login-Namen und Passwort befragt.

slogin *remote-machine*

bzw.

ssh *remote-machine*

verlangen nur das Passwort, gleicher Login-Name wird vorausgesetzt. Mit

slogin *remote-machine* **-l** *remote-username*

oder

slogin *remote-username***@***remote-machine*

kann der Login-Name abgeändert werden, es ist dann das Passwort des anderen Users an der

dortigen Maschine einzugeben. Analog funktioniert dies für **ssh**.

Auf vielen Systemen ist **telnet** nicht mehr verfügbar, da beim Einloggen das Passwort im Klartext übertragen wird und dies als Sicherheitsproblem betrachtet wird. **slogin** (secure login) und **ssh** (secure shell) sind die sicheren Varianten zum Einloggen, da das Passwort verschlüsselt wird bevor es übertragen wird.

9.2 ftp, sftp und scp

Mit dem File-Transfer-Protocol (**ftp**) lassen sich Dateien zwischen verschiedenen UNIX Maschinen bzw. zwischen UNIX und Windows übertragen.

ftp *remote-machine*

öffnet eine ftp-Verbindung zu einer anderen Maschine.
Man sitzt also lokal an einem Rechner und hat eine Verbindung zu einem entfernten Rechner aufgebaut. Mit

get *Dateiname*

holt man sich eine Datei, mit

put *Dateiname*

kopiert man eine Datei auf den entfernten Rechner.
Mehrere Dateien kann man sich gleichzeitig mit **mget** holen bzw. mit **mput** rüberkopieren, als Wildcard dient der *.
Dabei hat man darauf zu achten, dass der richtige Übertragungsmodus eingestellt ist. Mit

bin

wechselt man in den Binärmodus, mit

ascii

in den ASCII-, sprich in den Textmodus.
Im ASCII-Modus überträgt man Textdateien, wie Java-Quellcode. Dabei werden zwischen UNIX und MSDOS automatisch die richtigen Konvertierungen vorgenommen (zum Beispiel Zeilenendezeichen!). Im Binärmodus kann man zum Beispiel Grafiken (ausser PS, EPS, ... → dies sind normale Textdateien!) oder Java-Class-Files übertragen.

Achtung: EXE-Dateien von DOS bzw. Windows laufen nicht direkt unter UNIX und UNIX-Programme nicht direkt unter DOS!

Nützlich sind auch die Befehle

dir

bzw.

ls,

mit denen man auf dem entfernten Rechner nachsehen kann, wie die Dateien heissen. Die Befehle **mkdir**, **cd**, **rmdir** und **del** (statt rm) funktionieren ebenfalls.
Mit **prompt** kann man die Abfragen bei **mput** und **mget** ausschalten, mit **hash** werden bei der Übertragung # gezeichnet. Dies empfiehlt sich beim Transfer von grösseren Dateien.
Mit

bye

oder

quit

verlässt man die ftp-Verbindung.
Zwischen verschiedenen UNIX-Systemen gibt es noch eine einfachere Möglichkeit der Dateiübertragung, nämlich **scp**. Auf vielen Systemen ist **ftp** wie **telnet** nicht mehr verfügbar, da beim Einloggen das Passwort ebenfalls im Klartext übertragen wird. Bei **sftp** und **scp**

wird das Passwort, wie bei **slogin** und **ssh**, verschlüsselt bevor es übertragen wird.
sftp ist die sichere Version von **ftp**, bei der des Passwort verschlüsselt übertragen wird. Die Syntax ist die selbe.
scp hat folgende Syntax:

scp *remote-login*@*remote-machine*:*remote-directory*/*remote-filename* .

kopiert die angegebene Datei vom entfernten Rechner ins aktuelle Verzeichnis.

scp *local-filename remote-login*@*remote-machine*:*remote-directory*

kopiert die angegebene Datei vom lokalen Rechner in ein bestimmtes Verzeichnis auf dem entfernten Rechner. Dabei muss jeweils das Passwort vom entfernten Rechner angegeben werden.

9.3 MTOOLS

Unter Linux gibt es die sogenannten MTOOLS, die einen MSDOS-ähnlichen Zugriff auf das Diskettenlaufwerk gestatten. Mit

mdir

erhält man den Inhalt des aktuellen Verzeichnisses auf der Diskette, mit

mcd *Verzeichnisname*

wechselt man das Verzeichnis, mit

mmd *Verzeichnisname*

erzeugt man ein neues Unterverzeichnis auf der Diskette, mit

mcopy -t *Dateiname* **a:**

kopiert man eine Textdatei auf die Diskette, mit

mcopy -t a:*Dateiname* **.**

kopiert man eine Textdatei von der Diskette ins aktuelle Verzeichnis, mit

mdel *Dateiname*

löscht man eine Datei auf der Diskette.
Dabei hat man darauf zu achten, dass man bei der gleichzeitigen Übertragung mehrerer Dateien unter Verwendung von `*` die Dateinamen zusätzlich in Anführungszeichen zu setzen hat.

10 Packen

Um Plattenplatz zu sparen, kann man seine Dateien auch zusammenpacken:

tar cvf archiv.tar *

kopiert alle Dateien in einem Verzeichnisbaum in ein Archiv zusammen. Mit

gzip archiv.tar

komprimiert man das Archiv, die Originaldateien kann man dann löschen. Mit

gzip -d archiv.tar.gz

und

tar xvf archiv.tar

kann man die Originaldateien zurückerhalten.

11 Nachschlagewerke

- Linux in a nutshell: A desktop quick reference
  Ellen Siever and the staff of O'Reilly & Associates, Inc.
  O'Reilly 1999.

- `http://unixhelp.ed.ac.uk`
  Helpful information for users of the UNIX operating system

Developed at the University of Edinburgh.

12 Anhang

dez	Abkürzung	Beschreibung
0	NUL	Null (Nil)
1	SOH	Start of Heading (Anfang des Kopfes)
2	STX	Start of Text (Anfang des Textes)
3	ETX	End of Text (Ende des Textes)
4	EOT	End of Transmission (Ende der Übertragung)
5	ENQ	Enquiry (Stationsaufforderung)
6	ACK	Acknowledge (Positive Rückmeldung)
7	BEL	Bell (Klingel)
8	BS	Backspace (Rčkwärtsschritt)
9	HT	Horizontal Tabulation (Horizontal-Tabulator)
10	LF	Line Feed (Zeilenvorschub)
11	VT	Vertical Tabulation (Vertikal-Tabulator)
12	FF	Form Feed (Formularvorschub)
13	CR	Carriage Return (Wagenrücklauf)
14	SO	Shift-Out (Dauerumschaltung)
15	SI	Shift-In (Rückschaltung)
16	DLE	Data Link Escape (Datenübertragungsumschaltung)
17	DC1	Device Control (Gerätesteuerung)
18	DC2	Device Control (Gerätesteuerung)
19	DC3	Device Control (Gerätesteuerung)
20	DC4	Device Control (Gerätesteuerung)
21	NAK	Negative Acknowledge (Negative Rückmeldung)
22	SYN	Synchronous Idle (Synchronisierung)
23	ETB	End of Transmission (Ende des Datenüebertragungsblock)
24	CAN	Cancel (Ungültig)
25	EM	End of Medium (Ende der Aufzeichnung)
26	SUB	Substitute Character (Substitution)
27	ESC	Escape (Umschaltung)
28	FS	File Separator (Hauptgruppentrennung)
29	GS	Group Separator (Gruppentrennung)
30	RS	Record Separator (Untergruppentrennung)
31	US	Unit Separator (Teilgruppentrennung)
127	DEL	Delete (Löschen)

Die Steuerzeichen des ASCII-Zeichensatzes

dez	char	dez	char	dez	char	dez	char	dez	char
32		51	3	70	F	89	Y	108	l
33	!	52	4	71	G	90	Z	109	m
34	"	53	5	72	H	91	[	110	n
35	#	54	6	73	I	92	\	111	o
36	$	55	7	74	J	93	]	112	p
37	%	56	8	75	K	94	^	113	q
38	&	57	9	76	L	95	_	114	r
39	’	58	:	77	M	96	‘	115	s
40	(	59	;	78	N	97	a	116	t
41	)	60	<	79	O	98	b	117	u
42	*	61	=	80	P	99	c	118	v
43	+	62	>	81	Q	100	d	119	w
44	,	63	?	82	R	101	e	110	x
45	-	64	@	83	S	102	f	121	y
46	.	65	A	84	T	103	g	122	z
47	/	66	B	85	U	104	h	123	{
48	0	67	C	86	V	105	i	124	|
49	1	68	D	87	W	106	j	125	}
50	2	69	E	88	X	107	k	126	~

Die Schriftzeichen der ASCII-Tabelle

13 Schlussbemerkung

Dieses UNIX-Skript ist nur als Kurzeinführung für die Bedienung von Linux im Spinor Cluster der Universität Zürich gedacht und daher höchst unvollständig. Ausserdem dürfte es wohl in ein paar Abschnitten nicht für alle Ewigkeit Geltung besitzen.

Ein Danke geht an Johannes Schneider für seine Vorlage zu diesem Script.

Korrekturvorschläge und Erweiterungsvorschläge sind immer willkommen.