Von DOS nach LINUX

Michael Kugler

Linux kommt als alternatives Betriebsystem immer mehr in Mode. Damit der Umstieg von DOS auf Linux nicht so schwer fällt, gibt es hier ein paar Tipps die aus der Dokumentation von Linux entnommen worden sind.

(SuSe-Distribution 6.2, `DE-DOS-nach-Linux-Howto.txt`)

0. Für die ganz ungeduldigen Leser eine kurze Übersicht in der nebenstehenden Tabelle

Nach dieser sehr kurzen und trockenen Gegenüberstellung einige wichtige Elemente dieses Betriebsystemes.

DOS	Linux	Bemerkung
`BACKUP`	`tar -Mcvf device dir/`	völlig anders
`CD dirname\`	`cd dirname/`	fast die gleiche Syntax
`COPY file1 file2`	`cp file1 file2`	fast die gleiche Syntax
`DEL file`	`rm file`	Vorsicht - kein undelete
`DELTREE dirname`	`rm -R dirname/`	Vorsicht - kein undelete
`DIR`	`ls`	nicht ganz die gleiche Syntax
`EDIT file`	`vi file`	sehr gewöhnungsbedürftig
	`emacs file`	etwas besser
	`jstar file`	etwa wie DOS' edit
`FORMAT`	`fdformat`	unterschiedliche Syntax
	`mount, umount`	
`HELP command`	`man command`	gleiches Schema
`MD dirname`	`mkdir dirname/`	fast gleich
`MOVE file1 file2`	`mv file1 file2`	fast gleich
`NUL`	`/dev/null`	fast gleich
`PRINT file`	`lpr file`	fast gleich
`PRN`	`/dev/lp0, /dev/lp1`	fast gleich
`RD dirname`	`rmdir dirname/`	fast gleich
`REN file1 file2`	`mv file1 file2`	nicht für mehrere Dateien
`RESTORE`	`tar -Mxpvf device`	andere Syntax
`TYPE file`	`less file`	viel besser
`IN`	`startx`	Dazwischen liegen Welten!

1. Dateien und Programme

1.1 Dateien

Die Struktur des Filesystems von Linux ist für den Benutzer nach außen hin der von DOS recht ähnlich. Mit Struktur des Filesystems ist hier die Anordnung von Verzeichnissen und der darin enthaltenen Dateien gemeint. Die Namen für Verzeichnisse und Dateien gehorchen bestimmten Regeln, Dateien werden in Verzeichnissen abgelegt, es gibt ausführbare Dateien und diese haben oft auch wie unter DOS Kommandozeilenparameter. Darüber hinaus kann man auch Platzhalter, Umlenkung und Piping verwenden. Es gibt jedoch gegenüber DOS ein paar Unterschiede:

Unter DOS sind die Dateinamen in der 8.3-Form, d.h. wie etwa `NICHGENG.TXT`. Unter Linux sind die Regeln für Dateinamen bei Benutzung des UMSDOS- oder EXT2-Filesystems wesentlich liberaler, vergleichbar etwa mit Win9 . Es können bis zu 2 Zeichen verwendet werden, und der Punkt kann beliebig oft auftreten. Ein Beispiel für einen Dateinamen unter Linux ist z.B. `Das_ist.ein.SEHR_langer.dateiname`. Man beachte, dass hier sowohl große als auch kleine Buchstaben verwendet werden, denn es wird auch hier zwischen großen und kleinen Buchstaben - im Gegensatz zu DOS - unterschieden. Das heißt, `FILENAME.tar.gz` und `filename.tar.gz` sind zwei unterschiedliche Dateien. So ist `ls` ein Kommando, LS dagegen wird höchst wahrscheinlich nur eine Fehlermeldung bringen.

Dateien, deren Name mit einem Punkt beginnt, werden als versteckte Dateien behandelt. Sie werden bei einem normalen Auflisten mit `ls` nicht angezeigt. Erst ein `ls -a` bringt sie zum Vorschein.

Optionen und Schalter werden unter DOS als `/schalter` angegeben, unter Linux mit `-schalter` oder `- -schalter`. Beispiel: `dir /s` wird zu `ls -R`.

1.2 Links

UNIX hat noch einen weiteren Dateityp, der bei DOS nicht existiert. Es ist der Link. Ein Link ist eigentlich keine richtige Datei, sondern nur eine Art Verweis auf eine andere, bereits existierende Datei oder Verzeichnis.Es gibt zwei Arten von Links, den Hardlink und den symbolischen Link. Meistens wird der symbolische Link verwendet, er ist am ehesten vergleichbar mit den Win9 Shortcuts.

Beispiele für symbolische Links sind z.B. das Verzeichnis `/usr/X11`, welches ein Link auf `/usr/X11R6` ist und `/dev/modem`, welches entweder auf `/dev/cua0` oder `/dev/cua1` zeigt.

Um einen symbolischen Link anzulegen, gibt man ein:

```
$ ln -s <Datei_oder_Verzeichnis>
<Linkname>
```

Beispiele:

```
$ ln -s /usr/doc/g77/DOC g77manual.txt
```

Jetzt kann man sich auf `g77manual.txt` beziehen anstelle von `/usr/doc/g77/DOC.`

Links werden bei der Auflistung eines Verzeichnisses wie folgt angezeigt:

```
$ ls -l
... g77manual.txt -> /usr/doc/g77/DOC
```

1.3 echte und Eigentümer

Dos-Dateien haben folgende Attribute: `A` (archivieren), `H` (versteckt), `R` (nur lesbar) und `S` (System). Nur `H` und `R` sind unter Linux sinnvoll:

H sind Dateien die mit einem Punkt anfangen, und R wird später besprochen.

Unter UNIX besitzt jede Datei Rechte und einen Eigentümer, der wiederum zu einer Gruppe gehört. Hier ein Beispiel:

```
$ ls -l /bin/ls
-rwxr-xr-x 1 root bin 272 1 Aug 1  199
/bin/ls*
```

Das erste Feld enthält die Rechte der Datei `/bin/ls`, die `root` gehört, sowie der Gruppe bin. Die Zeichenfolge `-rwxr-xr-x` bedeutet von links nach rechts:

- ist der Dateityp (- normale Datei, d Verzeichnis, `l` Link, usw.);

`rwx` sind die Rechte für den Eigentümer der Datei (lesen,schreiben,ausführen);

`r-x` sind die Rechte für die Gruppe des Eigentümers (lesen,ausführen) - auf das Prinzip von Gruppen soll hier nicht weiter eingegangen werden, man kann als An-

fänger auch sehr gut ohne das auskommen ;-) - ;

r-x sind die Rechte für den Rest der Nutzer (lesen,ausführen).

Im Falle unseres **/bin/ls** kann man also die Datei nicht verändern, es sei denn, man ist **root**: alle anderen haben nicht die notwendigen Schreibrechte. Das Kommando, um die Rechte einer Datei zu ändern, ist:

```
$ chmod <werXrecht> <datei>
```

wobei wer für den steht, dessen Rechte geändert werden, also entweder **u** (user, der Eigentümer), **g** (group, die Gruppe), **o** (other,der Rest) oder **a** (all, alle Nutzer), **X** ist entweder **+**, **-** oder **=**, je nachdem, ob das Recht hinzugefügt oder weggenommen wird, bzw. auf den angegebenen Wert gesetzt wird, und recht ist das Recht, das geändert wird, also entweder **r** (read), **w** (write), oder **x** (execute).

Beispiele:

```
$ chmod u+x file
```

setzt die Ausführungsrechte für den Eigentümer.

```
$ chmod go-wx file
```

nimmt das Schreibrecht und das Ausführungsrecht für alle außer den Eigentümer weg.

```
$ chmod ugo+rwx file
```

setzt für alle Schreib-, Lese- und Ausführungsrechte. Man kann hier auch die Folge **ugo** einfach durch **a** ersetzen.

```
$ chmod u+s file
```

dieses setzt das sogenannte (oben nicht erwähnte) **setuid** oder suid Recht (meistens **setuid**-Bit genannt). Damit wird eine Datei, wenn sie ausführbar ist, automatisch beim Aufruf mit den Rechten des Eigentümers ausgeführt und nicht, wie sonst üblich mit den Rechten des Aufrufers. Gehört die Datei z.B. **root**, wird sie mit **root**-Rechten ausgeführt und hat somit vollen Zugriff auf das System (und kann bei einem Fehler auch entsprechenden Schaden anrichten). Also Vorsicht mit dem Setzen des **suid**-Bits.

1.4 Übertragen von Kommandos von DOS nach Linux

Auf der linken Seite ist das DOS Kommando aufgeführt, auf der rechten das Linux-Pendent

COPY	**cp**
DEL	**rm**
MOVE	**mv**
REN	**mv**
TYPE	**less, cat**

Umleitungs- und Pipingoperatoren:

Beispiele

DOS	Linux
`C:\GUIDO>copy joe.txt joe.doc`	`$ cp joe.txt joe.doc`
`C:\GUIDO>copy *.* total`	`$ cat * > total`
`C:\GUIDO>copy fractals.doc prn`	`$ lpr fractals.doc`
`C:\GUIDO>del temp`	`$ rm temp`
`C:\GUIDO>del *.bak`	`$ rm *~`
`C:\GUIDO>move paper.txt tmp\`	`$ mv paper.txt tmp/`
`C:\GUIDO>ren paper.txt paper.asc`	`$ mv paper.txt paper.asc`
`C:\GUIDO>print letter.txt`	`$ lpr letter.txt`
`C:\GUIDO>type letter.txt`	`$ more letter.txt`
`C:\GUIDO>type letter.txt`	`$ less letter.txt`
`C:\GUIDO>type letter.txt > nul`	`$ cat letter.txt > /dev/null`
`nicht vorhanden`	`$ more *.txt *.asc`
`nicht vorhanden`	`$ cat section*.txt | less`

< > >> |	**< > >> |**

Platzhalter:

*** ?**	*** ?**
nul:	**/dev/null**
prn, lpt1:	**/dev/lp0 or /dev/lp1; lpr**

Bemerkungen

Der * ist unter Linux intelligenter: * passt auf alle Dateien, außer auf die versteckten, ***.*** passt nur auf solche Dateien, die ein **'.'** in der Mitte oder am Ende haben, **p*r** passt auf **'peter'** und **'pfeiffer'** (mit 3 f ;-), ***c*** paßt auf 'picken', **'pack.txt'**, **'mac'** und **'c'** selbst.

Es gibt kein UNDELETE, also zweimal überlegen bevor man etwas löscht.

1.5 Die Pipeoperationen:

Zusätzlich zu den DOS-üblichen **< > >>** hat Linux noch weitere Möglichkeiten.

2> um Fehlermeldungen umzulenken (**stderr**) und

2>&1 lenkt **stderr** nach **stdout** um und

1>&2 lenkt **stdout** nach **stderr**.

1.6 Programme starten: Multitasking und Sessions

Um ein Programm auszuführen, gibt man einfach den Namen wie unter DOS ein. Falls das Verzeichnis (Abschnitt "Verzeichnisse"), in dem sich das Programm befindet, im Pfad **PATH** (Abschnitt "Der Systemstart") ist, wird das Programm starten. Unterschied zu DOS: ein Programm, das sich im aktuellen Verzeichnis befindet, wird nicht gefunden - es sei denn, das aktuelle Verzeichnis ist als '.' explizit im Pfad enthalten.(Aus Sicherheitsgründen sollte der Punkt in der Pfadangabe nie enthalten sein!!)

Um ein Programm im aktuellen Verzeichnis zu starten, welches nicht im Pfad eingetragen ist, hilft **./programm**. Wenn man z.B. ein kleines Prgrämmchen geschrieben hat und es test nennt, wird, wenn man es mit **test** aufruft und nicht mit **./test**, zuerst das UNIX-Kommando test selbigen Namens gefunden (oder die Shell-interne Funktion, je nach Shell) und ausgeführt und nicht das eigene Programm im aktuellen Verzeichnis. Das führt oft zu langem Grübeln, bis man endlich merkt, dass das falsche Programm aufgerufen wurde, denn test ohne Parameter gibt keinerlei Meldungen o.ä. aus. Auf diesen "Trick" sind schon Generationen von Einsteigern hereingefallen und werden wahrscheinlich auch noch weitere Generationen hereinfallen.

Hier das Aussehen eines typischen Kommandos:

```
$ kommando -s1 -s2 ... -sn par1 par2
... parn < input > output
```

wobei **-s1, ..., -sn** die Programmschalter sind und **par1, ..., parn** die Parameter. Der Rest sind die Umlenkungen, d.h. das Programm erhält seine Eingaben aus der Datei **input** und schreibt die Ausgaben in die Datei **output**. Es müssen natürlich nicht immer alle Teile enthalten sein. Mehrere Kommandos hintereinander können so eingegeben werden:

```
$ kommando1 ; kommando2 ; ... ;
kommandon
```

Das ist alles, was man braucht, um ein Kommando aufzurufen. Es gibt jedoch darüber hinaus Möglichkeiten, die Linux zusätzlich zu den von DOS bekannten bietet. Einer der Gründe die für Linux sprechen ist es, dass es ein Betriebsystem mit Multitasking ist, d.h. es kann mehrere Programme (ab jetzt Prozesse genannt) gleichzeitig ausführen. Man kann einen Prozess im Hintergrund starten und mit einem anderen weiterarbeiten. Darüber hinaus bietet Linux auch mehrere Sitzungen (Sessions) gleichzeitig an. Es ist so, als ob man an mehreren Rechnern arbeiten würde.

Um zu den Sessions 1..6 zu wechseln:

$ ALT F1 ... ALT F6

Wenn man gerade unter X-Windows ist, benutzt man statt dessen CTRL ALT Fn .

Michael Kugler michael.kugler@tgm.ac.at

2. Verzeichnisse

2.1 Allgemeines

Wir haben bereits die Unterschiede zwischen Dateien unter DOS/Windows und Linux besprochen. Was Verzeichnisse angeht, ist das \ Wurzelverzeichnis unter DOS während es unter Linux / ist. Ebenso werden Verzeichnisse in Pfadnamen unter DOS durch ein \ voneinander getrennt, unter Linux durch ein /. Darüber hinaus gibt es unter Linux keine Laufwerke, wie z.B. **C:** unter DOS. Wenn man mehrere Festplatten oder Partitionen hat, werden diese unter Linux zu einem einzigen Verzeichnisbaum zusammengefasst. Beispiele für Dateipfade:

```
DOS: C:\PAPERS\GEOLOGY\MID_EOC.TEX
Linux:
/home/guido/papers/geology/mid_eocene.tex
```

Wie üblich ist **..** das übergeordnete Verzeichnis und **.** das aktuelle. Man beachte, dass man nicht überall in jedes Verzeichnis mit **cd** wechseln kann, oder mit **rmdir** oder **mkdir** Verzeichnisse löschen bzw. anlegen kann, in Abhängigkeit davon, ob man die entsprechenden Rechte hat oder nicht. Jeder Nutzer befindet sich nach dem Einloggen in seinem eigenen Verzeichnis, üblicherweise als HOME-Verzeichnis bezeichnet. Auf meinem PC ist mein Home-Verzeichnis z.B. **/home/guido**.

2.2 Zugriffsrechte von Verzeichnissen

Verzeichnisse haben natürlich auch Zugriffsrechte wie Dateien. Die Rechte, die im Abschnitt "Rechte und Eigentümer" für Dateien besprochen wurden, existieren auch analog bei Verzeichnissen (**user**, **group**, **other**). Für ein Verzeichnis bedeutet **rx**, dass man in das Verzeichnis wechseln kann und sich den Inhalt auflisten lassen kann. **w** dagegen bedeutet, dass man eine Datei im Verzeichnis löschen kann oder eine neue anlegen. Man kann eine Datei in einem Verzeichnis, für das man Schreibrechte besitzt, auch dann löschen, wenn die Datei für einen eigentlich schreibgeschützt ist. Das System fragt zwar an, ob man die Datei wirklich löschen will, aber wenn man ja sagt verschwindet die Datei. Der Schreibschutz für Dateien bezieht sich nur auf das Verändern der Datei selber. Das Löschen einer Datei ist aber eine Änderung am Verzeichnis, denn die Datei selbst wird nicht verändert, sondern nur ihr Eintrag aus dem Verzeichnis entfernt.

Um z.B. zu verhindern, dass andere Nutzer im Verzeichnis **/home/guido/text** herumschnüffeln, gibt man ein:

```
$ chmod o-rwx /home/guido/text
```

2.3 Übertragen von Kommandos von DOS nach Linux

DIR	**ls, find, du**
CD	**cd, pwd**
MD	**mkdir**
RD	**rmdir**
DELTREE	**rm -R**
MOVE	**mv**

Beispiele

siehe Kasten auf der nächsten Seite

Bemerkungen

1. Bei der Benutzung von **rmdir** muss das zu löschende Verzeichnis leer sein. Will man ein Verzeichnis mitsamt seinem Inhalt auf einmal loswerden hilft ein **rm -R**. VORSICHT! Mit diesem Befehl kann man ganze Verzeichnisbäume auf Nimmerwiedersehen verschwinden lasen. Also, zweimal überlegen.

2. Das Zeichen '~' ist eine Abkürzung für den Namen des eigenen Home-Verzeichnisses, wenn es am Anfang des Pfad/Dateinamens steht. Das Kommando cd oder cd ~ bringen einen sofort ins eigene Home-Verzeichnis, egal wo man gerade ist. cd **~/tmp** bringt einen demnach ins Verzeichnis **/Pfad_zum_Homeverzeichnis/tmp**.

3. **cd -** ist ein "undo" für das letzte **cd**.

3. Floppies, Harddisks und Ähnliches

Wenn man unter DOS ein **FORMAT A:** ausführt, geschehen drei Dinge: 1. Die Diskette wird physisch formatiert, d.h. es werden Spuren und Sektoren darauf angelegt; 2. es wird das **A:**-Verzeichnis angelegt und damit ein DOS-Filesystem; 3. die Diskette wird dem Nutzer zum Zugriff zur Verfügung gestellt.

Diese drei Schritte werden unter Linux separat behandelt. Man kann Disketten im MSDOS-Format verwenden, es gibt aber auch andere Formate die z.B. lange Dateinamen und auch Zugriffsrechte unterstützen. Hier die Schrittfolge zum Vorbereiten einer Diskette (man muss dazu **root** sein):

Formatieren einer Standard 1,44 MB Floppydisk (A:):

```
# fdformat /dev/fd0H1440
```

Filesystem erstellen:

```
# mkfs -t ext2 -c /dev/fd0H1440
```

Anstelle von /dev/fd0H1440 muss auf manchen Systemen auch /dev/fd0h1440 verwendet werden. Hiermit wird das Standard-Linuxfilesystem, das ext2-Filesystem, auf der Diskette angelegt. Dieses Filesystem ist jedoch für kleine Datenträger wie Disketten eigentlich nicht so gut geeignet, da es für die Verwaltung großer Festplatten entwickelt wurde. Daher wird für Disketten, auf denen man ein UNIX-artiges Filesystem haben will, das Minix-Filesystem eingesetzt.

```
# mkfs -t minix -c /dev/fd0H1440
```

Um ein MS-DOS Filesystem zu erstellen, benutzt man folgendes Kommando:

```
# mformat a:
```

oder

```
# mkfs -t msdos -c /dev/fd0H1440
```

Vor der Benutzung der Diskette muss sie gemountet werden:

```
# mount -t <typ> /dev/fd0H1440 /mnt
```

wobei **<typ>** der Typ des Filesystems ist, mit dem die Diskette formatiert wurde, also **ext2**, **minix** oder **msdos**. Jetzt kann man auf die Floppy zugreifen. Der Inhalt der Floppy steht jetzt als Inhalt des Verzeichnisses **/mnt** zur Verfügung, d.h. das Verzeichnis **/mnt** entspricht jetzt der Diskette. Wenn man fertig ist, muss man die Diskette wieder unmounten.

Zum Unmounten der Disk:

```
# umount /mnt
```

Jetzt kann man die Diskette aus dem Laufwerk entnehmen. Natürlich muss man Disketten nur dann formatieren und ein Filesystem anlegen, wenn dieses auf den Disketten noch nicht geschehen ist. Für Laufwerk **B:** ersetzt man einfach **fd0H1440** durch **fd1H1440** in der obigen Anleitung.

Alles was man unter DOS mit **A:** und **B:** gemacht hat, wird jetzt mit dem Verzeichnis **/mnt** gemacht.

Beispiele

DOS	Linux
C:\GUIDO>dir a:	$ ls /mnt
C:\GUIDO>copy a:*.*	$ cp /mnt/* /docs/temp
C:\GUIDO>copy *.zip a:	$ cp *.zip /mnt/zip
C:\GUIDO>a:	$ cd /mnt

Wenn Sie dieses ganze mounten/unmounten von Disketten nicht mögen, benutzen Sie die **mtools**-Suite: dieses ist ein Satz von Kommandos, die wie ihr jeweiliges DOS-Gegenstück arbeiten, jedoch jeweils mit einem '**m**' anfangen, also **mformat**, mdir, mdel und so weiter. Sie können damit sogar lange Dateinamen erhalten, jedoch keine Rechte. Man benutze diese Kommandos einfach wie die DOS-Kommandos.

Was für Disketten gilt, gilt natürlich auch für andere Arten von Laufwerken, wie z.B. Zip-Disks, CD-ROM's, neue Festplatten usw. Für Zip-Laufwerke gibt es übri-

gens ein eigenes HOWTO, das ZIP Drive Mini-HOWTO , bzw. eine deutsche Version als Deutsches ZIP-HOWTO. Hier das Verfahren zum Mounten eines CD-ROM:

```
# mount -t iso9660 /dev/cdrom /mnt
```

Das ist der "offizielle" Weg einen Datenträger ins Dateisystem des Rechners einzubinden (zu mounten). Da es jedoch etwas lästig ist, jedesmal zum mounten **root** sein zu müssen, gibt es eine "Abkürzung", die es jedem Nutzer erlaubt, die Datenträger einzubinden und der außerdem Schreibarbeit spart:

Als **root** legt man zuerst die Verzeichnisse **/mnt/a**, **/mnt/a:**, und **/mnt/cdrom** an.

Danach fügt man in **/etc/fstab** die folgenden Zeilen hinzu:

```
/dev/cdrom /mnt/cdrom iso9660
ro,user,noauto          0        0
/dev/fd0H1440 /mnt/a: msdos
user,noauto,umask=000 0          0
/dev/fd0H1440 /mnt/a minix user,noauto
0    0
```

Um jetzt eine DOS-Diskette, eine Minix-Diskette und ein CDROM zu mounten, reicht nun folgendes:

```
$ mount /mnt/a:
$ mount /mnt/a
$ mount /mnt/cdrom
```

Auf **/mnt/a**, **/mnt/a:**, und **/mnt/cdrom** kann jetzt von jedem Nutzer zugegriffen werden. Man sollte beachten, dass man für ein frisch angelegtes Minix oder ext2-Filesystem auf der Diskette erst die entsprechenden Rechte vergeben muss, damit alle Nutzer darauf schreiben können:

```
# mount /mnt/a
# chmod 777 /mnt/a
# umount /mnt/a
```

Man sollte sich natürlich im Klaren darüber sein, dass dies ein dickes Sicherheitsloch ist. Wenn Sicherheit also von Bedeutung ist, z.B. in einer Firma, sollte man etwas vorsichtiger sein.

4. Einrichten des Systems

4.1 Der Systemstart

Zwei wichtige Dateien unter DOS sind **AUTOEXEC.BAT** und **CONFIG.SYS**, welche beim Booten zur Initialisierung des Systems, zum Setzen von Umgebungsvariablen wie **PATH** oder **FILES** und zum Starten von Treibern und Programmen verwendet werden. Unter Linux gibt es mehrere Dateien für das Starten des Systems, wobei man eine Reihe von ihnen besser unangetastet läßt, bis man genau weiß, was man tut. Hier aber trotzdem eine Liste der wichtigsten Dateien:

Dateien	Bemerkungen
/etc/inittab	Hände weg fürs erste!
/etc/rc.d/* bzw. **/sbin/int.d/***	dito

Falls man nur die **PATH**-Variable oder eine andere Umgebungsvariable setzen möchte, die Login-Meldung ändern oder automatisch nach dem Login ein Programm starten möchte, kann man dieses in den folgenden Dateien tun:

Dateien	Bemerkungen
/etc/issue	setzt Pre-Login Meldung
/etc/motd	setzt Post-Login Meldung
/etc/profile	setzt u.a. **PATH** und andere Variablen
/etc/bashrc	setzt u.a. Aliase und Funktionen(s.u.)
/ihr_home_Verz/.bashrc	setzt Ihre Aliase und Funktionen
/ihr_home_Verz/.bash_profile	setzt Umgebung + startet Ihre Programme
/ihr_home_Verz/.profile	setzt Umgebung + startet Ihre Programme

Wenn letztere Datei existiert (man beachte, dass sie eine versteckte Datei ist), wird sie nach dem Login gelesen und die Anweisungen darin ausgeführt.

Beispiel eines .profile:

```
# Ich bin ein Kommentar

echo Umgebung:

printenv | less # entspricht Kommando
SET unter DOS
alias d='ls -l' # d bedeutet ab jetzt
ls -l
alias up='cd ..'
echo "Der Pfad ist jetzt "$PATH
echo "Heutiges Datum: 'date'" # Ausgabe
des Kommandos 'date' verwenden
echo "Schönen Tag noch, "$LOGNAME
```

$PATH und **$LOGNAME** sind Umgebungsvariablen; es gibt noch viele andere, mit denen man experimentieren kann.

4.2 Dateien zur Programm-Initialisierung

Unter Linux kann fast alles den eigenen Bedürfnissen angepasst werden. Die meisten Programme haben ein oder mehrere Initialisierungsdateien, an denen man herumbasteln kann, oft mit dem Namen **.Programmnamerc** in Ihrem Home-Verzeichnis. Die ersten, an denen man üblicherweise etwas verändert sind:

.inputrc

benutzt von **bash**, um Tastenzuordnungen festzulegen.

.xinitrc

benutzt von **startx**, um das X Windows System zu initialisieren.

.fvwmrc

benutzt vom Windowmanager **fvwm**. Ein Beispiel ist in:

/usr/lib/X11/fvwm/system.fvwmrc
.Xdefault

benutzt z.B. von **rxvt**, einem Terminalemulator für X und anderen Programmen.

5. Das verbleibende 1%

5.1 Arbeiten mit tar & gzip

Unter UNIX gibt es einige weit verbreitete Programme zum Archivieren und Komprimieren von Dateien. Um Archive anzulegen wird **tar** benutzt, ähnlich dem PKZIP, aber ohne Kompression. Um ein neues Archiv anzulegen:

```
$ tar -cvf <archiv_name.tar> <Datei>
[Datei...]
```

Um eine Datei aus einem Archiv zu extrahieren:

```
$ tar -xpvf <archiv_name.tar>
[Datei...]
```

Zum Auflisten des Inhaltes eines Archives:

```
$ tar -tf <archiv_name.tar> | less
```

Man kann Dateien (und damit auch **tar**-Archive) mit **compress** komprimieren. Es ist jedoch eigentlich veraltet und sollte daher nur noch in Ausnahmefällen verwandt werden. Das gebräuchlichere Programm ist **gzip**. Aufruf:

```
$ compress <Datei>
$ gzip <Datei>
```

Das erzeugt eine Datei mit der Endung **.Z** (**compress**) oder **.gz** (**gzip**). Diese Programme können nur jeweils eine Datei gleichzeitig komprimieren. Zum Entkomprimieren:

```
$ compress -d <Datei.Z>
$ gzip -d <Datei.gz>
```

Die Programme **unarj**, **zip** und **unzip** (**PK??ZIP** kompatibel) existieren ebenfalls. Dateien mit der Endung **.tar.gz** oder **.tgz** (archiviert mit **tar**, dann komprimiert mit **gzip**) sind in der Unixwelt so verbreitet wie **.ZIP** Dateien unter DOS. So listet man den Inhalt eines **.tar.gz** Archivs auf:

```
$ gzip -dc <Datei.tar.gz> | tar tf - |
less
```

5.2 Nützliche Tipps

Kommandovervollständigung: drücken der TAB Taste während der Eingabe eines Befehls auf der Kommandozeile vervollständigt den angefangenen Befehl. Beispiel: Man will die Zeile **gcc dies_ist_ein_sehr_langer_Name.c** eingeben. Eintippen von **gcc** die TAB veranlasst die Shell, automatisch den langen Dateinamen zu ergänzen, falls die angegebenen Buchstaben ausreichen, um die Datei eindeutig zu identifizieren.

Michael Kugler michael.kugler@tgm.ac.at

Beispiele

DOS	Linux
C:\GUIDO>dir	$ ls
C:\GUIDO>dir file.txt	$ ls file.txt
C:\GUIDO>dir *.h *.c	$ ls *.h *.c
C:\GUIDO>dir/p	$ ls | more
C:\GUIDO>dir/a	$ ls -l
C:\GUIDO>dir *.tmp /s	$ find / -name "*.tmp"
C:\GUIDO>cd	$ pwd
nicht vorhanden - siehe Bemerkung	$ cd
dito	$ cd ~
dito	$ cd ~/temp
C:\GUIDO>cd \other	$ cd /other
C:\GUIDO>cd ..\temp\trash	$ cd ../temp/trash
C:\GUIDO>md newprogs	$ mkdir newprogs
C:\GUIDO>move prog ..	$ mv prog ..
C:\GUIDO>md \progs\turbo	$ mkdir /progs/turbo
C:\GUIDO>deltree temp\trash	$ rm -R temp/trash
C:\GUIDO>rd newprogs	$ rmdir newprogs
C:\GUIDO>rd \progs\turbo	$ rmdir /progs/turbo

Zurückscrollen: drücken von SHIFT BildHoch (PageUp) ermöglicht es, ein paar Seiten des Bildschirminhaltes zurückzuholen, je nachdem wie viel Videospeicher man hat.

Reset für den Bildschirm: Wenn man mit **more** oder **cat** eine Binärdatei auf den Bildschirm ausgegeben hat, kann es passieren, dass der Bildschirm danach völlig unlesbar wird. Um das wieder gerade zu biegen, gibt man blind ein **reset** ein oder diese Folge von Zeichen: echo CTRL V ESC C RETURN.

Text einfügen: auf der Konsole siehe unten; unter X klickt man in das (z.B. **xterm**) Fenster und zieht die Maus bei gedrückter linker Maustaste über den Text. Dann in dem Fenster, wo der Text hin soll, die mittlere Maustaste drücken, oder wenn man eine Zweitastenmaus hat, beide Tasten gleichzeitig. Es gibt auch das xclipboard (leider nur für Text) als Alternative.

Maus nutzen: Zunächst muss man den **gpm** installieren, einen Maustreiber für die Konsole. Text selektieren wie unter X und dann die rechte Maustaste zum Einfügen drücken (oder auch die mittlere). Dieser Mechanismus funktioniert über mehrere virtuelle Terminals hinweg.

Kernelmeldungen: man kann als **root** in **/var/adm/messages** oder **/var/log/messages** nachschauen, was der Kernel so an Meldungen produziert. Hier stehen auch die Meldungen vom Booten des Systems. Das Kommando **dmesg** ist hier auch hilfreich.

5.3 Nützliche Programme und Kommandos

Diese Liste ist natürlich nur eine persönliche und richtet sich nach meinen Bedürfnissen und Vorstellungen. Zunächst einmal, wo man sie finden kann: Da sich sicher alle auskennen mit dem Internet und Dinge wie **archie** und **ftp** beherrschen, hier nur die drei wichtigsten Adressen für Linux: **sunsite.unc.edu**, **tsx-11.mit.edu**, und **nic.funet.fi**. Man benutze soweit möglich den nächstgelegenen Mirror.

at erlaubt das automatische Ausführen von Programmen zu bestimmten Zeiten;

cron ist ein nützliches Programm, welches zum periodischen Ausführen von Kommandos zu einem bestimmten Datum und einer Zeit dient;

df gibt Informationen über die gemounteten Disks (freier Platz, usw.);

file <Dateiname> gibt darüber Auskunft, was Dateiname für ein Typ ist (ASCII-Text, ausführbar, Archiv, etc.);

find ist eines der mächtigsten und nützlichsten Kommandos. Es wird dazu benutzt, Dateien zu finden, die gewissen Bedingungen entsprechen und auf diese dann bestimmte Kommandos anzuwenden. Allgemeine Form des Kommandos find :

```
$ find <Verzeichnis> <Ausdruck>
```

wobei **<Ausdruck>** die Suchkriterien und Kommandos enthält.

Beispiele:

```
$ find . -type l -exec ls -l {} \;
```

sucht alle Dateien, die symbolische Links sind und gibt aus, auf was sie ein Link sind.

```
$ find / -name "*.old" -ok rm {} \;
```

sucht alle Dateien, die auf **.old** enden und löscht diese nach vorheriger Rückfrage.

```
$ find . -perm +111
```

sucht alle Dateien, deren Rechte **111** sind (ausführbar).

```
$ find . -user root
```

sucht alle Dateien, die **root** gehören. Es gibt noch viele weitere Möglichkeiten.

grep sucht Textmuster in Dateien. z.B.:

```
$ grep -l "Geologie" *.tex
```

listet alle Dateien ***.tex**, die das Wort Geologie enthalten. Das Programm **zgrep** arbeitet mit gezippten Dateien.

joe guter Editor. Wenn man ihn mit **jstar** aufruft, kennt er die selben Tastaturkombinationen wie WordStar et al., einschließlich DOS und Borlands Turbo-Editoren;

less wahrscheinlich der beste Textbrowser; wenn korrekt konfiguriert, kann man damit auch **gzip-**, **tar-** und **zip**-Dateien anzeigen;

lpr <Datei> druckt eine Datei im Hintergrund aus. Zum Abfragen des Status des Druckauftrages gibt es **lpq**; um eine Datei aus der Warteschlange des Druckers zu entfernen, gibt es **lprm**;

mc ist ein guter Dateimanager (ähnlich Norton Commander);

6.0 Copyright

Da ich einige Teile aus der mitgelieferten Dokumentation verwendet habe, folgt an dieser Stelle das entsprechende Copyright:

Falls nicht anders vermerkt, unterliegen Linux HOWTO's dem Copyright ihrer Autoren. Linux HOWTO's dürfen teilweise oder im Ganzen vervielfältigt und vertrieben werden, mittels jedweden Mediums, ob physisch oder elektronisch, so lange wie diese Copyrightnotiz in allen Exemplaren enthalten ist. Kommerzielle Distribution ist erlaubt und erwünscht; der Autor (der englischen Originalversion) würde jedoch gern davon in Kenntnis gesetzt werden.

Alle Übersetzungen, abgeleitete Werke sowie zusammenfassende Arbeiten, die ein Linux HOWTO enthalten, müssen diesem Copyright unterliegen. D.h., es ist nicht gestattet, eine von einem HOWTO abgeleitete Arbeit zusätzlichen Restriktionen zu unterwerfen. Ausnahmen können unter bestimmten Bedingungen gewährt werden. Näheres ist dazu beim jeweiligen Koordinator der HOWTO's zu erfahren. Die Adressen sind weiter unten angegeben.

Kurz gesagt, eine möglichst weite Verbreitung der HOWTO's über viele Kanäle ist von uns gewünscht, wir behalten uns jedoch das Copyright vor und möchten von Plänen zu weiteren Verteilung der HOWTO's in Kenntnis gesetzt werden.

Falls Sie Fragen dazu haben, wenden Sie sich an den Koordinator der englischen HOWTO's Greg Hankins (gregh@sunsite.unc.edu), oder an den Koordinator der deutschen HOWTO's Marco Budde (*Budde@tu-harburg.d400.de*).