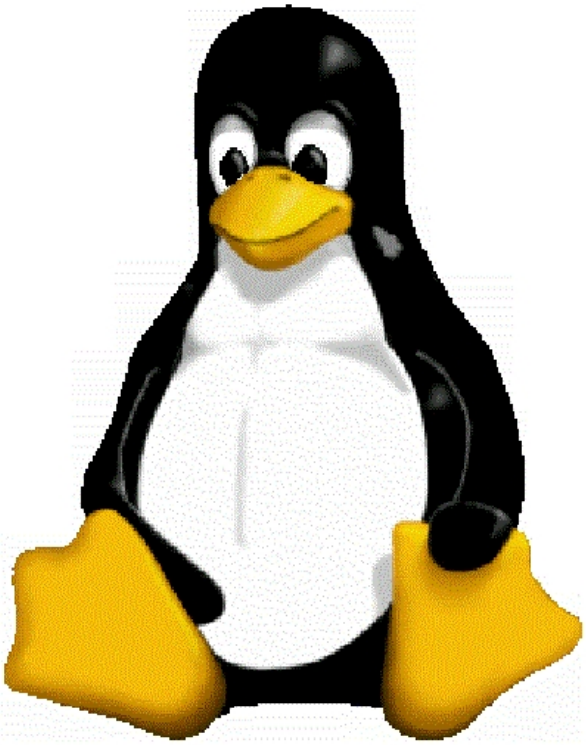
LINUX: KONZEPTE UND BEDIENUNG

Unterlagen zum Linux-Kurs

von
©Tobias Brinkert

eMail:
t.brinkert@15bit.de oder tbrinkert@web.de



Version 1.1
03. September 2001

Inhaltsverzeichnis

Kapitel 1

Einführung

1.1 Entstehung

1.1.1 Unix

Linux ist nur ein Teil in einer langen Kette von UNIX-Betriebssystemen. Die erste UNIX-Version, der Urahn von Linux, wurde 1969 von Ken Thompson auf einem DEC PDP-7 implementiert. 1971 wurde UNIX zusammen mit Dennis Ritchie in die Programmiersprache C umgeschrieben.

Unix ist im Grunde nur ein Oberbegriff für diverse abgeleitete Betriebssysteme, sogenannte Derivate von den einzelnen Herstellern, die meist ähnlich klingen und auf -ix enden, z.B. Sinix, Xenix, Ultrix, Irix ...

Herausragende Merkmale des UNIX-Betriebssystems sind :

- hierarchisches Dateisystem
- identische Schnittstelle für Daten-, Geräte- und Interprozeß- Ein-/Ausgabe
- Hintergrundprozesse
- Prozeßkonzept zur Realisierung synchroner und asynchroner Vorgänge
- Filtertechnik
- viele Werkzeuge
- ein hohes Maß an Portabilität

1.1.2 Linux

Der Vater und Entwickler von Linux ist Linus Benedict Torvalds, ein finnischer Student. Im September 1991 erschien die erste "öffentliche" Version von Linux. Seit dieser Zeit haben sich viele Studenten und andere Interessierte an der Weiterentwicklung beteiligt und ermöglichten im Frühjahr 1994 die Version 1.0. Der Vorteil von Linux gegenüber UNIX liegt in der größeren Palette von unterstützen Hardware-Komponenten und den effizienteren Code in vielen Bereichen.

Der Begriff Linux bezeichnet eigentlich nur den Kernel. Der Kernel ist der innerste Teil eines Betriebssystems mit elementaren Funktionen wie Speicherverwaltung, Prozeßverwaltung und Steuerung der Hardware.

Der Code des Linux-Kernels ist frei, daß bedeutet, dieser kann von jedem weiterentwickelt und angepaßt werden. Aus diesem Grund gibt es ständig neue Linux-Kernel-Versionen, die im Internet veröffentlicht werden und von jedem benutzt werden können. Man erhält nicht "den" Linux-Kernel, sondern nur den Quell-Code und muß sich seinen eigenen Kernel auf seinem Rechner kompilieren.

Die oberste Entscheidung, welche Eigenschaft in den Kernel mit hineingenommen wird und ab wann dieser als stabiler Anwender-Kernel deklariert wird, liegt allein bei Linus Torvalds. Anhand der Versionsnummern des Kernels kann erkannt werden, um welchen Typ von Kernel es sich handelt. Stabile Anwender-Kernel haben eine gerade Nummer nach dem ersten Punkt, Entwickler-Kernel eine ungerade. Z.B. Bezeichnen die Nummern 1.2.n, 2.0.n und 2.2.n stabilen Kernel, die Nummern 1.3.n, 2.1.n und 2.3.n die Entwickler-Kernel.

Aktuelle Kernel-Version ist zur Zeit 2.4.2 (wobei sich dieses jetzt schon wieder geändert haben könnte :-)).

1.2 Leistungsfähigkeit des Kernel 2.0

- Linux unterstützt Multitasking (gleichzeitige Abarbeitung mehrerer Prozesse), Multiuser-Betrieb (gleichzeitige Nutzung durch mehrere Anwender), Paging (Auslagerung von Speicher auf die Festplatte, wenn zuwenig RAM zur Verfügung steht), Shared Libraries (Bibliotheken mit Systemfunktionen werden nur einmal geladen, wenn sie von mehreren Prozessen benötigt werden), Interprocess Communication (IPC) und seit Version 2.0 sogar Symmetrix Multi Processing (SMP, die Nutzung mehrerer Prozessoren).

- Linux unterstützt praktisch die gesamte gängige PC-Hardware: alle Intel-386-kompatiblen Prozessoren; die Bussystem ISA, VLB, EISA, und PCI; die meisten (E)IDE- und SCSI-Festplatten-Controller und die daran angeschlossenen Festplatten, CD-ROM-Laufwerke und Streamer; die meisten marktüblichen CD-ROM-Laufwerke, die nicht am IDE- bzw. SCSI-Bus betrieben werden; die meisten Netzwerkkarten, Mäuse etc. (vollständige Liste in der Hardware-HOWTO).

- Linux unterstützt nicht nur INTEL-Prozessoren, sondern auch den DEC-Alpha-Prozessor, Sun Sparc, Mips, Motorola etc.

- Linux verwendet ein eigenes Dateisystem (ext2fs). Dateinamen dürfen bis zu 255 Zeichen haben, Dateien bis zu 16 GByte und Dateisystem bis zu 4 TByte groß werden. Es ist mit einer Menge von Sicherheitsmerkmalen ausgestattet.

- Unter Linux kann auf viele fremde Dateisystem zugegriffen werden: DOS, Win95 (mit langen Dateinamen), OS/2 (nur Lesezugriff), NTFS (ab 2.2.n), Minix, NFS (Netzwerkdateisystem) etc. Ein Zugriff auf komprimierte DOS/Windows-Partitionen ist allerdings nicht möglich (bzw. nur über den recht umständlichen Weg des DOS-Emulators).

- Linux unterstützt mehrere Binärformate für die Ausführung von Binärdateien: a.out (Standard bis Version 1.0), ELF (Standard seit Version 1.2) und iBCS2. iBCS2 ermöglicht es, viele kommerzielle Programme von SCO-Unix ohne eine Neuübersetzung bzw. Portierung auf Linux zu verwenden.

- Unter Linux steht eine ganze Palette von Netzwerkprotokollen zu Verfügung (TCP/IP, PPP, SLIP).

1.3 GNU und General Public License (GPL)

Linux ist frei und wird unter der GPL veröffentlicht. Dabei heißt frei nicht nur kostenlos, sondern bezieht sich aber auch und vor allem auf die Verfügbarkeit des gesamten Quellcodes. GNU-Programme sind (unter gewissen Einschränkungen) frei kopierbar. Programme, die mit Code aus GNU-Programmen geschrieben wurden, dürfen zwar verkauft werden, der Programmcode des neuen Systems muß abermals frei verfügbar sein. Diese Regelung stellt sicher, daß Erweiterungen am System allen Anwendern zugute kommen. Die Kernaussage der GPL liegt darin, daß zwar jeder den Code verändern und sogar die resultierenden Programme verkaufen darf, aber gleichzeitig der Anwender / Käufer das Recht auf den vollständigen Code hat, diesen ebenfalls verändern und wieder kostenlos oder kostenpflichtig weitergeben darf. Jedes GNU-Programm muß zusammen mit der vollständigen GPL weitergegeben werden.

Ziel der Entwickler von GNU und Linux war es also, ein System zu schaffen, dessen Quellen frei verfügbar sind und es auch bleiben. Um einen Mißbrauch auszuschließen, ist Software, die im Sinne von GNU entwickelt wurde und wird, durch die General Public License (GPL) geschützt. Hinter der GPL steht die Free Software Foundation (FSF). Diese Organisation wurde von Richard Stallmann (dem Autor des Editor Emacs) gegründet, um qualitativ hochwertige Software frei verfügbar zu machen.

Teil I

Linux-Konzepte

Kapitel 2

Der Systemstart

Jedes Computer- und Betriebssystem durchläuft nach dem Einschalten eine genau festgelegte Prozedur zum Starten des Systems. Nachdem der allgemeine Startvorgang eines PCs, der durch das BIOS festgelegt ist, abgeschlossen ist, wird im MBR-Record eines Datenträgers (Eingestellt durch das BIOS, auf welchem gesucht wird) ein Bootloader gesucht und gestartet. Bei Linux ist dies meist der Linux-Loader LILO. Dieser Bootloader lädt dann den Kernel, wobei ihm dabei Parameter übergeben werden können. Genauso können LILO einige Einstellungen, sei es über die Konfigurationsdatei oder am LILO-Prompt, angegeben werden, z. B. Kernel, Startplatte, Speichergröße u.v.a.

Der gestartete Kernel übernimmt alle weiteren Aktionen, wie sie in Abbildung 2.1 dargestellt sind. Hierbei regelt der Kernel lediglich den Zugriff auf die Hardware und stellt die Schnittstellen für den weiteren Betrieb zur Hardware bereit.

2.1 Der init-Prozeß

Nachdem der Kernel diese Aufgaben erledigt hat, wird der Prozeß `/sbin/init` gestartet. Dieses Programm mit der Prozeß-ID 1 übernimmt den weiteren Bootvorgang des Linuxsystems und ist der Ursprung aller Prozesse eines Unix- und Linuxsystems. Abgearbeitet wird die Datei `/etc/inittab`, die den weiteren Ablauf der Systeminitialisierung bestimmt.

2.1.1 /etc/inittab

Die Steuerungsdatei für den init-Prozeß ist die Datei `/etc/inittab`. Das Schema eines Eintrags hat die Form:

```
id-code:runlevel:action:command
```

wobei der `id-code` aus zwei Zeichen besteht und eindeutig in der Datei sein muß, um eine Identifizierung des Eintrags zu ermöglichen. `Runlevel` gibt an, für welchen Runlevel der Eintrag gilt, `action` ist eine Anweisung für `init` und `command` ein Linux-Kommando oder Programm, welches gestartet werden soll.

Die wichtigsten `action`-Schlüsselwörter sind (eine vollständige Beschreibung erhalten Sie mit `man inittab`):

`ctrlaltdel`: gibt an, wie init auf `Strg+Alt+Entf` reagieren soll
`initdefault`: definiert den Default-Runlevel

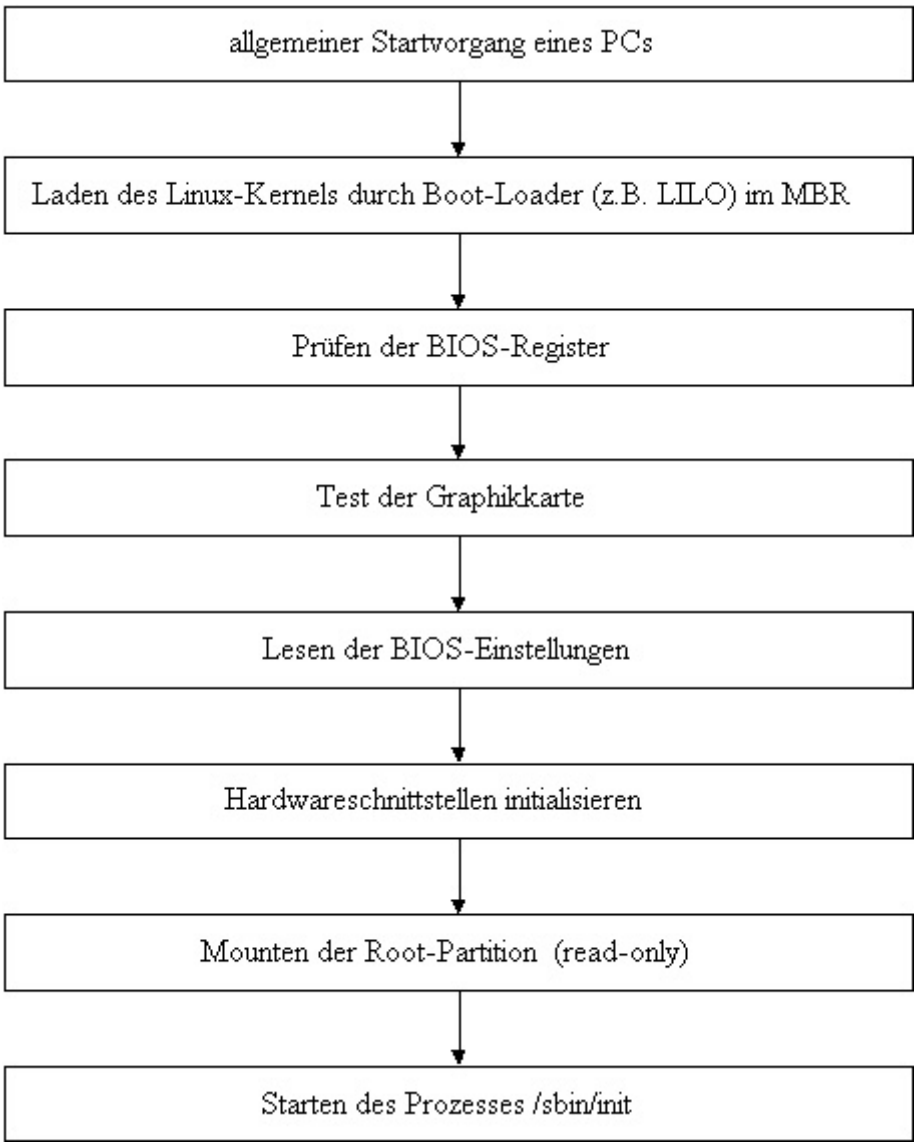


Abbildung 2.1: Der Systemstart

`once`: `init` startet das Kommando beim Wechsel in diesen Runlevel
`respawn`: `init` startet das Kommando nach seinem Ende wieder neu
`sysinit`: `init` startet das Kommando einmal während des Boot-Prozesses
`wait`: `init` wartet auf das Ende des nachfolgenden Kommandos

Das nachfolgende Listing gibt eine gekürzte `inittab`-Datei einer SuSE-Distribution wieder. Der Standardrunlevel ist 2. Daher startet `init` bei einem normalen Systemstart zuerst das Skript `/sbin/init.d/boot` und danach `/sbin/init.d/rc` mit dem Parameter "2". Desweiteren wird hier festgelegt, wie das System auf die Tastenkombination `Strg+Alt+Entf` reagieren soll sowie auf welchen Konsolen das Programm `mingetty` gestartet werden soll, um ein Login zu ermöglichen (hier auf den Konsolen 1 bis 6).

```
# default runlevel
id:2:initdefault:

# check system on startup
# first script to be executed if not booting in emergency (-
b) mode
si:I:bootwait:/sbin/init.d/boot

# /sbin/init.d/rc takes care of runlevel handling
#
l0:0:wait:/sbin/init.d/rc 0
l1:1:wait:/sbin/init.d/rc 1
l2:2:wait:/sbin/init.d/rc 2
l3:3:wait:/sbin/init.d/rc 3
#l4:4:wait:/sbin/init.d/rc 4
#l5:5:wait:/sbin/init.d/rc 5
l6:6:wait:/sbin/init.d/rc 6

# what to do in single-user mode
ls:S:wait:/sbin/init.d/rc S
~~:S:respawn:/sbin/sulogin

# what to do when CTRL-ALT-DEL is pressed
ca::ctrlaltdel:/sbin/shutdown -h -t 4 now

# getty-programs for the normal runlevels
# <id>:<runlevels>:<action>:<process>
# The "id" field  MUST be the same as the last
# characters of the device (after "tty").
1:123:respawn:/sbin/mingetty --noclear tty1
2:123:respawn:/sbin/mingetty tty2
3:123:respawn:/sbin/mingetty tty3
4:123:respawn:/sbin/mingetty tty4
5:123:respawn:/sbin/mingetty tty5
6:123:respawn:/sbin/mingetty tty6

# end of /etc/inittab
```

2.1.2 Runlevel

Der `init`-Prozeß kennt verschiedene Ausführungmodis, die sogenannten Runlevel. Runlevel sind numerische Bezeichnungen für Systemkonfigurationen. Abhängig vom Runlevel werden bestimmte Aktionen gestartet oder gestoppt. Z.B. gibt es Runlevel zum Starten im normalen, Multi-user und Netzwerk-Betrieb genauso wie es einen Runlevel für den Single-User-Modus, den Reboot-Modus oder Halt-Modus gibt. In der Datei `/etc/inittab` sind hierzu die Sollzustände für die jeweiligen Runlevel hinterlegt. Das erste Skript, welches durch `init` gestartet wird ist normalerweise das Skript `/etc/rc.d/rc.sysinit` (bei SuSE `/sbin/init.d/boot`). In Abbildung 2.2 wird aufgeführt, welche Aufgaben das Script ausführt.

Danach werden alle Skripte im Standardrunlevel gestartet. Leider benutzen nicht alle Distribu-

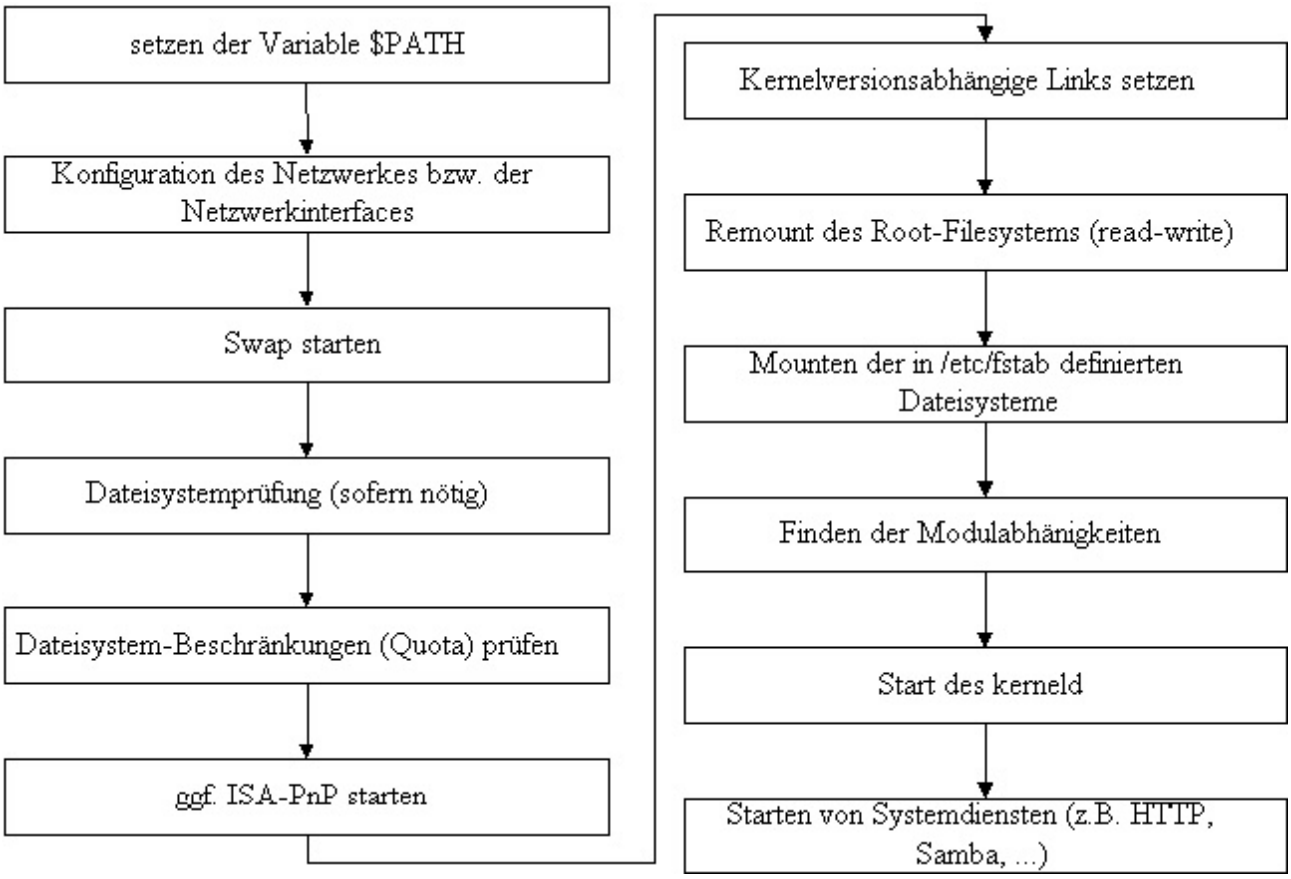


Abbildung 2.2: Das Script /etc/rc.d/rc.sysinit

tionen einheitliche Runlevel für gleiche Aufgaben. Die jeweilige Bedeutung der Runlevel ist in `/etc/inittab` dokumentiert (siehe auch Tabelle 2.1).

Im laufenden Betrieb kann der aktive Runlevel jederzeit geändert werden, indem an der Kommandozeile der Befehlt `init` gefolgt von dem Runlevel, in welchen gewechselt werden soll, eingegeben wird. Z.B. `init 0` um das System herunterzufahren. Beim Booten kann man LILO den gewünschten Runlevel übergeben, indem man nach dem Konfigurationsnamen den Runlevel als Parameter übergibt. Z.B. wenn eine Konfiguration `linux` heißt, kann über die Eingabe am LILO-Prompt von "`linux S`" ein SuSE-Linuxsystem in dem Single-User-Modus gestartet werden.

Runlevel	Bedeutung (allgemein)	Bedeutung bei SuSE
0	Halt	Halt
1	Single-User	Multi-User ohne Netz
2	Multi-User ohne Netz	Multi-User mit Netz (default)
3	Multi-User mit Netz (default)	Multi-User mit Netz und xdm
4	frei	frei
5	Multi-User mit Netz und xdm	frei
6	Reboot	Reboot
S	-	Single-User

Tabelle 2.1: Runlevel im Überblick

2.1.3 Skripte im Runlevel

Die Skripte für die Runlevel befinden sich im Verzeichnis `init.d` (unterhalb `/etc` oder `/etc/rc.d`, bei SuSE `/sbin`). In diesem Verzeichnis gibt es weitere Unterverzeichnisse für jeden Runlevel mit dem Namensschema `rc`*N*`.d`, wobei *N* den Runlevel bezeichnet. In diesen Unterverzeichnissen befinden sich Links auf die Skripte im Basisverzeichnis. Dabei sind die Namen nicht willkürlich gewählt, sondern bezeichnen die ausführenden Aktionen. Beginnt ein Link mit einen "S", so wird dem Skript der Parameter `Start` übergeben, beginnt er mit einem "K", so wird der Parameter `Stop` übergeben. Die nachfolgende Zahl, bestehend aus 2 Ziffern, gibt die Reihenfolge der Ausführung an, wobei zuerst die niedrigste Nummer ausgeführt wird. Hiermit können Abhängigkeiten zwischen verschiedenen Skripten aufgelöst werden. Zum Beispiel muß zuerst das Netzwerk gestartet werden, bevor das Routing gestartet werden kann.

Durch die Lösung der Links wird die Konfiguration und Wartung vereinfacht, da alle Skripte an einer zentralen Stelle gehalten werden und somit die Änderung an einem Skript allen Runleveln zugute kommt.

Natürlich können die Skripte auch von Hand gestartet werden (als `root`), indem das Skript mit einem Parameter aufgerufen wird. Hierfür gibt es je nach Skript zum Teil mehr Parameter als nur `start` und `stop`, z. B. `restart`, `reload` ...

Das nachfolgende Listing zeigt einen Ausschnitt aus dem Inhalt des Standard-Runlevel eines SuSE-Systems:

```
lrwxrwxrwx   1 root     root      7 Jul  1 17:52 K19cron -
> ../cron
lrwxrwxrwx   1 root     root      9 Jul  1 17:54 K19identd -
> ../identd
lrwxrwxrwx   1 root     root      6 Jul  1 17:55 K20gpm -> ../gpm
lrwxrwxrwx   1 root     root      8 Jul  1 17:52 K20inetd -
> ../inetd
lrwxrwxrwx   1 root     root      6 Jul  1 18:01 K20lpd -> ../lpd
lrwxrwxrwx   1 root     root      8 Jul  1 17:52 K20rwhod -
> ../rwhod
lrwxrwxrwx   1 root     root      9 Jul  1 17:52 K30random -
> ../random
lrwxrwxrwx   1 root     root      9 Jul  1 17:54 K35syslog -
> ../syslog
lrwxrwxrwx   1 root     root     11 Jul  1 18:02 K36scanlogd -
> ../scanlogd
lrwxrwxrwx   1 root     root      9 Jul  1 17:52 K50serial -
> ../serial
lrwxrwxrwx   1 root     root     11 Jul  1 17:57 K81svgatext -
> ../svgatext
lrwxrwxrwx   1 root     root     10 Jul  1 17:52 K99kerneld -
> ../kerneld
lrwxrwxrwx   1 root     root     10 Jul  1 17:52 S01kerneld -
> ../kerneld
lrwxrwxrwx   1 root     root      9 Jul  1 17:52 S02serial -
> ../serial
lrwxrwxrwx   1 root     root     11 Jul  1 18:02 S09scanlogd -
> ../scanlogd
```

```
lrwxrwxrwx   1 root     root      9 Jul  1 17:54 S09syslog -
> ../syslog
lrwxrwxrwx   1 root     root      9 Jul  1 17:52 S13random -
> ../random
lrwxrwxrwx   1 root     root     11 Jul  1 17:57 S19svgatext -
> ../svgatext
lrwxrwxrwx   1 root     root      6 Jul  1 17:55 S20gpm -> ../gpm
lrwxrwxrwx   1 root     root      8 Jul  1 17:52 S20inetd -
> ../inetd
lrwxrwxrwx   1 root     root      6 Jul  1 18:01 S20lpd -> ../lpd
lrwxrwxrwx   1 root     root      8 Jul  1 17:52 S20rwhod -
> ../rwhod
lrwxrwxrwx   1 root     root      7 Jul  1 17:52 S21cron -
> ../cron
lrwxrwxrwx   1 root     root      9 Jul  1 17:54 S21identd -
> ../identd
```

2.1.4 init-Bearbeitungstools

Für die Bearbeitung der Runlevel stehen verschiedene Tools zur Verfügung, z. B. `chkconfig`, `tksysv` und `ksysv`. In Abbildung 2.3 ist das Programm `ksysv` abgebildet. Bei diesem Programm aus der KDE-Sammlung stehen alle verfügbaren Dienste in dem linken Fenster und können per Drag and Drop in die Runlevelfenster gezogen werden. Durch Doppelklick auf einen Eintrag im Runlevel werden weitere Informationen angezeigt, unter anderem kann hier die Priorität des Eintrags geändert werden.

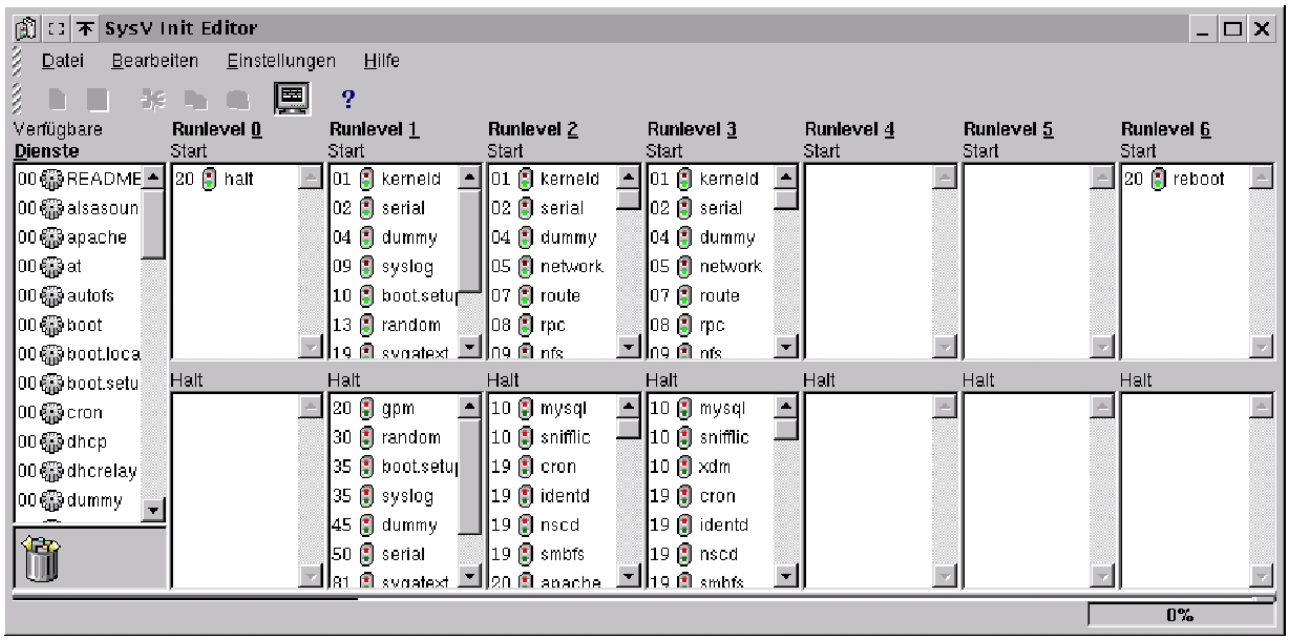


Abbildung 2.3: Das Programm “ksysv”

Kapitel 3

Das Linuxdateisystem (ext2)

3.1 Überblick

Das standardmäßige Dateisystem unter Linux ist das `ext2`-Dateisystem. Die Partitionstypnummer ist 83. Desweiteren gibt es für die `Swap`-Partition (die Auslagerungsdatei für MS Windows-Anwender) ein eigenes Dateisystem mit der Typ-Nr. 82 und für Disketten findet man noch häufig das Dateisystem `Minix` (siehe auch Kapitel 6.2 auf Seite 32).

Das Dateisystem `ext2` weist folgende Merkmale auf

- flache, baumartige Organisation des Dateisystems;
- Speichersystem (Festplatten, CD-ROM, Floppy, . . .) werden für den Anwender transparent ins Dateisystem eingebunden;
- Einarbeitung eines Rechtekonzeptes, mit den Rechten schreiben, lesen und ausführen; Rechte können für Besitzer, Gruppe und alle anderen vergeben werden;
- Aufbau nach dem Filesystem Hierachy Standard (FHS)
- die max. Dateigröße beträgt 16 GByte;
- es gibt keine Beschränkungen bei der max. Pfadlänge;

Die Dateiname können aus bis zu 255 Zeichen bestehen, wobei folgender Zeichenvorrat bereitsteht:

- Buchstaben: a . . . z, A . . . Z
- Zahlen: 0 . . . 9
- einige Sonderzeichen: u.a. -, _, ., . . .

Auf das Leerzeichen sollte wenn möglich verzichtet werden, da es als Trennzeichen bei der Eingabe von Kommandos verwendet werden. Deutsche Umlaute sowie das "ß" sollte gemieden werden. Sonderzeichen sind grundsätzlich möglich, sollten jedoch zur Vermeidung von Komplikationen nicht verwendet werden. Die Dateinamen dürfen mehrere Punkte enthalten.

Bei Linux, wie auch bei Unix, ist der Unterschied Groß- und Kleinschreibung signifikant, das bedeutet, die Datei "Linux.txt" und "linux.txt" sind zwei unterschiedliche Dateien.

In jedem Verzeichnis existieren zwei weitere Einträge, Verzeichnisse mit den Namen "." und "..". Dabei symbolisiert das Verzeichnis "." immer das Verzeichnis selbst, währen das Verzeichnis ".." stets das übergeordnete Verzeichnis repräsentiert.

Verzeichnis- und Dateinamen werden, wie unter Unix üblich, durch den einfachen Schrägstrich ("/") getrennt.

3.2 Aufbau des Dateisystembaums

Der Linux-Dateisystembaum beginnt mit dem "root"-Verzeichnis "/" und enthält sowohl weitere Verzeichnisse als auch Dateien. Dabei können Verzeichnisse wieder beliebig viele Unterverzeichnisse und Dateien enthalten. Im Filesystem Hierachy Standard wird festgelegt, welche Verzeichnisse es standardmäßig gibt und welche Daten in welchen Verzeichnissen gespeichert werden. Hiermit soll eine distributionsübergreifende Ordnung erzielt werden, jedoch weichen einige Distributionen noch vereinzelt davon ab. Diese Ordnung spiegelt sich teilweise auch bei Programmverzeichnissen wieder, da die jeweiligen Unterverzeichnisse häufig nach dem FHS aufgebaut sind. In Tabelle 3.1 ist eine grobe Übersicht wiedergegeben, welche Verzeichnisse existieren und was man in ihnen finden kann.

/etc	Dateien zur Systemkonfiguration und Konfigurationsdateien, die von allen verwendet werden
/bin /usr/bin	Systemprogramme und Kommandos, Anwendungsprogramme
/sbin /usr/sbin	Programme zur Systemverwaltung
/lib /usr/lib	Programmbibliotheken
/var	Logbuchdateien, Drucker-Spool, Temp-Verzeichnis
/proc	Abbild des Prozeßsystems sowie des Betriebssystemkerns
/usr/man	Man-Seiten Info-SeitenDokumentationen
/usr/info /usr/doc	
/usr/src	Quellcode, u.a. Kernel-Quellcode
/usr/X11R6	X-Windows-System
/usr/include	Include-Dateien für die Programmierung
/home	Heimatverzeichnisse der einzelnen Benutzer
/dev	Gerätedateien
/opt	Kommerzielle Software
/root	Heimatverzeichnis des Administrators root
/	Hauptverzeichnis, sollte nur weitere Verzeichnisse enthalten;
/boot	Dateien, die zum Systemstart benötigt werden (z.B. Kernel)

Tabelle 3.1: Verzeichnisbaum

3.3 Dateiinformationen

Neben dem Dateinamen besitzt eine Datei noch weitere Informationen und Eigenschaften. Hierzu gehören die Dateigröße, das Erstellungs-, Änderungs- und Zugriffsdatum, der Besitzer, die

Gruppe, die Zugriffsrechte, die Lage der Daten auf der Festplatte (Dateinummer, Inode) und die Art der Datei.

Die Zugriffsrechte werden für den Eigentümer, die Gruppe und den Rest der Welt einzeln vergeben. Für jede gibt es drei Rechte: Lesen (r), Schreiben (w) und Ausführen (x).

Beispiel:

```
total 70
drwx--x--x  19 root     root         1024 Nov 14 13:48 .
drwxr-xr-x  22 root     root         1024 Nov 10 15:45 ..
drwx------   5 root     root         1024 Oct 30 15:28 Desktop
drwx------   2 root     root         1024 Oct 29 17:22 Mail
drwxr-xr-x  22 root     root         1024 Oct 30 02:53 Office51
-rw-r--r--   1 root     root         1944 Oct 29 17:20 ServerLog
-rw-rw-rw-   1 root     root         1047 Oct 29 17:21 StartLog
lrwxrwxrwx   1 root     root            3 Nov 14 13:48 binold -
> /bin
-rw-r--r--   1 root     root        12288 Oct 30 15:34 classes.db
-rwxr-xr-
x   1 root     root          206 Oct 30 18:24 knetmon_cleanup
```

Die erste Spalte gibt die Zugriffsrechte der jeweiligen Datei bzw. Verzeichnis an. Dabei kennzeichnet das erste Zeichen den Typ der Datei (siehe Tabelle 3.2), gefolgt von den Zugriffsrechten.

Zeichen	Typ
-	normale Datei
d	Verzeichnis
c	zeichenorientiertes Gerät (z.B. Modem, Maus, Drucker)
b	blockorientiertes Gerät (z.B. Festplatte, Diskettenlaufwerk)
l	Link (Verweis)
p	Pipe

Tabelle 3.2: Kennzeichnung der Dateitypen

3.4 Dateisystembereiche

Im `ext2`-Dateisystem wird eine Partition in Gruppen von 8192 Blöcken (oder einer variablen Anzahl) eingeteilt. Die Gruppen enthalten eine Kopie des Superblocks sowie einen Gruppenblock. Der Gruppenblock entspricht in seiner Funktion dem Superblock. Dabei werden die Gruppen wie eine gesamte Partition verwaltet, wobei die Unterteilung die Datensicherheit erhöht und die Geschwindigkeit steigert. Mehrere Kopien des Superblocks werden an genau definierten Stellen auf der Partition abgelegt, so daß die Chance besteht, bei einem defekten Superblock über die Kopien des Superblocks diesen zu reparieren. Beim Einbinden der ext2-Partition wird das valid-Flag im Superblock gelöscht und beim Ausbinden wieder gesetzt. Sollte die Partition nicht ordnungsgemäß ausgebunden werden, z. B. durch Absturz, einfach Ausschalten . . . , so erkennt der Kernel dieses durch das nicht gesetzte valid-Flag und startet ein Festplattenkontroll-Programm (`fsck`).

Bootblock der erste Block einer Festplatte oder Diskette kann hier ein Programm zum Laden des Betriebssystems, den Boot-Loader, enthalten;

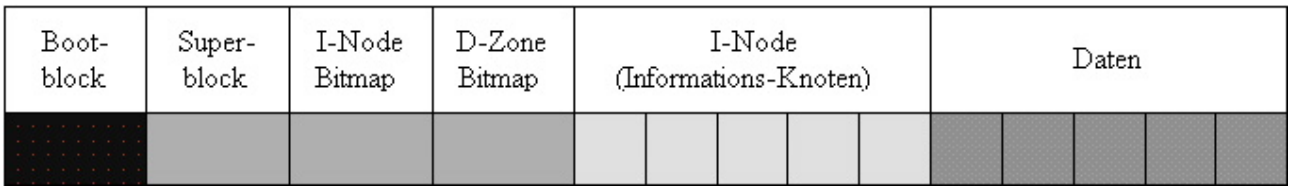


Abbildung 3.1: Dateisystembereiche

Superblock ist die Größe der Informationsknoten (I-Knoten)-Bitmaps, der Datenzonen-Bitmaps, der Informationsknoten, der Datenbereiche und der magischen Zahl abgelegt; die magische Zahl beinhaltet den Typ des Dateisystems;

I-Knoten-Bitmaps hinterlegt, welche I-Knoten belegt und welche frei sind;

D-Zone-Bitmaps hinterlegt, welche Sektoren im Datenbereich belegt oder frei sind;

I-Knoten in diesem Bereich sind die I-Knoten hinterlegt;

Daten in diesem Bereich sind die eigentlichen Daten hinterlegt;

3.5 Aufbau von Dateien im Dateisystem

Innerhalb des Linuxdateisystems werden die Informationen in Blöcken gespeichert. Dabei haben alle Blöcke innerhalb eines Dateisystems (Partition) die gleiche Größe. Das `ext2`-Dateisystem unterstützt Blockgrößen von 1024 Bytes, 2.048 Bytes, 4096 Bytes, 8192 Bytes, 16384 Bytes und 32768 Bytes. Die übliche Standardblockgröße ist 4096 Bytes.

Eine Datei wird demnach in mehreren Blöcken gespeichert. Ist eine Datei nun nicht durch die Blockgröße des Dateisystems dividierbar, entsteht ein Verschnitt. Dies ist darin begründet, daß das ext2-Dateisystem nur in ganzen Blöcken arbeitet. Der leere Teil des letzten benötigten Blocks ist dann der Verschnitt. Der Verschnitt bei einer Blockgröße von 1024 Bytes ist immer am Kleinsten. Eine große Blockgröße hat jedoch den Vorteil, daß das Einlesen und Bearbeiten der Datei performanter ist.

Vor dem Anlegen eines Dateisystems sollte man sich schon überlegen, welche Art von Daten und welche durchschnittliche Datengröße erwartet wird, damit man das beste Verhältnis aus kleinem Verschnitt und möglichst bester Performance wählen kann. Grundsätzlich gilt hierfür:

> Werden viele kleine Dateien erwartet, ist eine kleine Blockgröße von Vorteil, da ansonsten ein zu großer Verschnitt auftritt. Werden jedoch wenige große Dateien erwartet (z.B. bei Datenbanken), ist eine größere Blockgröße vorteilhafter, da damit die Performance gesteigert werden kann.

Kapitel 4

Management

4.1 Gerätemanagement und -dateien

Unter Linux werden alle Geräte über Devices angesprochen. Devices sind definierte Schnittstellen zu der Hardware, wobei der direkte Zugriff auf die Hardware nur dem Kernel vorbehalten ist. Im Grunde sind Devices Dateien mit einer Inode, jedoch ohne Datenteil und sind im Verzeichnis `/dev` gespeichert.

Sie werden durch drei Informationen beschrieben: der Major Nummer, der Minor Nummer und dem Zugriffstyp. Die Major Nummer bestimmt den Linuxkerneltreiber, der für dieses Device verantwortlich ist. Hiervon gibt es zur Zeit 25 Treiber. Die Minor Nummer spezifiziert die Geräteart, z. B. 720 kB Floppy oder 1,44 MB Floppy, bei Festplatten gibt sie die Partition an, die verwaltet werden soll. Beim Zugriffstyp gibt es zwei Arten, *blockorientiert* für gepufferte Geräte wie Festplatten und *zeichenorientiert* für ungepufferte Geräte wie serielle Schnittstellen.

Zum Teil finden Sie im /dev-Verzeichnis auch Links. Diese werden mit einem beschreibenden Namen für die Vereinfachung auf Standardschnittstellen gelegt, z. B. für das Modem oder die Maus. Eine Übersicht über die häufig benötigten Schnittstellen ist in Tabelle 4.1 aufgeführt.

Die Sternchen in der Tabelle sind Platzhalter für eine nähere Bezeichnung, welches Gerät angesprochen werden soll. Hierfür gibt es zwei Notationen. Bei zeichenorientierten Geräten wird numerisch von “0” an begonnen, die Geräte durchzuzählen. Z. B. ist die erste serielle Schnittstelle `/dev/ttyS0` (unter DOS `COM1`). Bei blockorientierten Geräten gibt es zwei Methoden. Zum einen wird z. B. beim Diskettenlaufwerk oder SCSI CD-ROM die Spezifizierung wie bei zeichenorientierten Geräten vorgenommen, `/dev/fd0` zeigt auf das erste Floppy. Bei Festplatten und CD-ROMs wird die Spezifizierung mittels Buchstaben vorgenommen, `/dev/sda` zeigt auf das erste SCSI Laufwerk, `/dev/hda` auf die Master-Festplatte am ersten IDE-Controller. Die einzelnen Partitionen einer Festplatte werden über eine weitere Zahl beschrieben, hier jedoch wird mit der Nummerierung mit “1” begonnen! Z. B. kennzeichnet `/dev/hdc5` die erste logische Partition der Festplatte, die als Master am zweiten IDE-Port hängt (siehe auch 7.2, Seite 38).

Der Systemverwalter kann weitere Devices anlegen, wenn die bereits durch die Installation vorhandenen Devices nicht ausreichen. Hierfür gibt es den Befehl `mknod`.

Syntax von `mknod`:

```
mknod [Optionen] Name Typ [Major Minor]

Typ: b          legt blockorientiertes Device an
     c, u legt  zeichenorientiertes Device an
     p          legt eine FIFO an
```

Devicename	Gerät
*bm	Bus Mäuse
console	aktive Konsole
fd*	Diskettenlaufwerk
ftape*	Floppystreamer (ohne Rückspulfunktion) (Link)
hd*	IDE / ATA Festplatten oder CDROM
lp*	Parallele Schnittstellen
mem	Arbeitsspeicher
modem	Standardschnittstelle für das Modem (Link)
mouse	Standardschnittstelle für die Maus (Link)
null	unendlich großes Loch (“schwarzes Loch”)
psaux	PS/2 Maus
nftape*	Floppystreamer mit Rückspulen (Link)
nrft*	Floppystreamer mit Rückspulen
nst*	SCSI Bandlaufwerk ohne Rückspulen
port	E/A-Schnittstellen
ptyp*	Terminalschnittstellen unter X (Master)
ram	RAM Disk
rft*	Floppystreamer ohne Rückspulen
rmt	Bandlaufwerk (ohne SCSI)
sd*	SCSI Laufwerk
scd*	SCSI CD-ROM
st*	SCSI Bandlaufwerk mit automatischer Rückspulfunktion
tape*	Standardstreamer (Link)
tty*	Virtuelle Textterminals
ttyp*	Terminalslaves unter X
ttyS*	Serielle Schnittstellen
zero	unendlich große Quelle von 0-Bytes

Tabelle 4.1: Devicenamen und ihre Bedeutung (Auswahl)

4.2 Prozeßmanagement

Das gesammte Management der Prozesse unterliegt dem Kernel. Er verteilt die Rechenzeit, überwacht und ermöglicht die saubere Kommunikation unter den Prozessen und kontrolliert die Rechte. Linux ist ein Multitasking-System, was bedeutet, daß mehrere Prozesse gleichzeitig abgearbeitet werden können. Dies kann aber nur auf Mehrprozessormaschinen wirklich erfolgen, auf Single-Prozessorsystemen wird ein System benötigt, daß dem Anwender scheinbar dir parallele Abarbeitung der Prozesse vorspielt. Hier kommt der Prozeß-Scheduler zum Einsatz. Er verteilt die Rechenzeit und Systemresourcen auf die einzelnen Prozesse für ein bestimmtes Zeitintervall.

> “Der *Mehrprogrammbetrieb* steigert den Informationsdurchfluß der Hardware. Stoppt z.B. Programm n, weil es einen E/A-Kanal aufruft, so verarbeitet der Prozessor das bereitstehende Programm m, bis dieses einen Stop erreicht usw. Ein Zuteilungsprogramm legt zuvor die Prioritäten fest.
>
> Steht jedem im Hauptspeicher befindlichen Programm eine feste Zeitspanne zu, so spricht man von **Zeitscheibenverfahren** (engl.: time slicing).” [Breuer, Seite 197]

Welcher Prozeß als nächster für die Bearbeitung durch die CPU ausgewählt wird, errechnet der Prozeß-Scheduler mit Hilfe von Prioritätsparametern, Prozeßzustand und Wartezeit. Auf die Priorität von Prozessen kann auch direkt Einfluß genommen werden über den Befehl `nice`. Normale Anwender können Prozesse herunterstufen, allein dem Systemadministrator `root` ist das Recht vorbehalten, Prozesse heraufzustufen. Der Aufruf erfolgt über:

```
nice -<nice-Wert> <Kommandofolge>
```

Für den `nice`-Wert kommt ein Zahlenbereich von -20 bis 20 in Frage, wobei ein niedriger `nice`-Wert eine höhere Priorität bedeutet. Der in der Syntax angegebene Bindestrich gehört zum Befehl dazu, so daß bei heraufsetzen der Priorität zwei Bindestriche einzugeben sind. Nachträglich, wenn der Prozeß schon gestartet ist, kann die Priorität mit dem Befehl `renice` geändert werden. Die Syntax hierfür lautet:

```
renice -<nice-Wert> <PID>
```

Der PID ist eine eindeutige Identifizierung eines Prozesses (Prozeß Identifier). Jedem Prozeß wird eine eindeutige Nummer zugewiesen. Wie schon in Kapitel 2.1 erwähnt, hat der Vater aller Prozesse, der `init`-Prozeß, die PID 1.

Ein Prozeß besteht immer aus einem Textsegment (Befehle), einen Datensegment und einem Stack-Bereich. Wird nun ein Befehl oder Kommando zwei oder mehrfach gestartet, so wird das Textsegment nicht noch einmal geladen, sondern im Speicher nur Bereiche für das Datensegment und den Stack-Bereich reserviert. Somit können die Speicherressourcen geschont werden.

Im Verzeichnis `/proc` befindet sich ein Abbild aller Prozesse sowie alle Informationen des Kernels. Es ist ein spezielles Dateisystem mit keinen echten Dateien und Verzeichnissen, sondern bildet die Schnittstelle zum Kernel. Hier kann der direkte Zugriff auf Informationen des Kernels realisiert werden. Bei Arbeiten in diesem Verzeichnis ist größte Sorgfalt angebracht, da Änderungen an den Dateien direkt den Kernel betreffen und das System in einen instabilen Zustand versetzen können, bis zum Absturz des Systems. Trotz allem ist dieses Verzeichnis eine der wichtigsten Informationsquellen zum System. Im nachfolgenden Listing ist ein Auszug aus dem Verzeichnis `/proc` wiedergegeben:

```
dr-xr-xr-x 3 root root         0 Jul 10 08:51 bus
-r--r--r-- 1 root root         0 Jul 10 08:51 cmdline
-r--r--r-- 1 root root         0 Jul 10 08:51 cpuinfo
-r--r--r-- 1 root root         0 Jul 10 08:51 devices
-r--r--r-- 1 root root         0 Jul 10 08:51 dma
-r--r--r-- 1 root root         0 Jul 10 08:51 filesystems
dr-xr-xr-x 2 root root         0 Jul 10 08:51 fs
dr-xr-xr-x 4 root root         0 Jul 10 08:51 ide
-r--r--r-- 1 root root         0 Jul 10 08:51 interrupts
-r--r--r-- 1 root root         0 Jul 10 08:51 ioports
-r-------- 1 root root 134221824 Jul 10 08:51 kcore
-r-------- 1 root root 134221824 Jul 10 08:51 kcore_elf
-r-------- 1 root root         0 Jul 10 08:40 kmsg
-r--r--r-- 1 root root         0 Jul 10 08:51 ksyms
-r--r--r-- 1 root root         0 Jul 10 08:51 loadavg
-r--r--r-- 1 root root         0 Jul 10 08:51 locks
-r--r--r-- 1 root root         0 Jul 10 08:51 meminfo
```

```
-r--r--r-- 1 root root         0 Jul 10 08:51 memstat
-r--r--r-- 1 root root         0 Jul 10 08:51 misc
-r--r--r-- 1 root root         0 Jul 10 08:51 modules
-r--r--r-- 1 root root         0 Jul 10 08:51 mounts
dr-xr-xr-x 3 root root         0 Jul 10 08:40 net
-r--r--r-- 1 root root         0 Jul 10 08:51 partitions
-r--r--r-- 1 root root         0 Jul 10 08:51 pci
dr-xr-xr-x 2 root root         0 Jul 10 08:51 scsi
-r--r--r-- 1 root root         0 Jul 10 08:51 sound
-r--r--r-- 1 root root         0 Jul 10 08:51 stat
-r--r--r-- 1 root root         0 Jul 10 08:51 swaps
dr-xr-xr-x 9 root root         0 Jul 10 08:51 sys
dr-xr-xr-x 4 root root         0 Jul 10 08:51 tty
-r--r--r-- 1 root root         0 Jul 10 08:51 uptime
-r--r--r-- 1 root root         0 Jul 10 08:51 version
```

Zum Beispiel enthält die Datei `/proc/cpuinfo` Informationen über die CPU, die Datei `/proc/modules` Informationen, welche Module geladen sind und die Datei `/proc/meminfo` Informationen über Speicherverbrauch und -ressourcen.

4.2.1 Prozesse anzeigen

Alle laufenden Prozesse können über die Befehle `ps`, `top` oder `pstree` angezeigt werden. Dabei gibt der Aufruf von `ps` viele wichtige Informationen über das System. Das nachfolgende Listing zeigt eine gekürzte Ausgabe des Befehls `ps -aux`, die Parameter werden in Kapitel B.5 beschrieben.

```
USER    PID %CPU %MEM   RSS TTY   STAT START TIME COMMAND
root      1  0.0  0.1   204 ?     S    12:28 0:04 init [2]
root      2  0.0  0.0     0 ?     SW   12:28 0:00 [kflushd]
root      3  0.0  0.0     0 ?     SW   12:28 0:00 [kupdate]
root      4  0.0  0.0     0 ?     SW   12:28 0:00 [kpiod]
root      5  0.0  0.0     0 ?     SW   12:28 0:00 [kswapd]
bin      94  0.0  0.3   400 ?     S    12:29 0:00 /sbin/portmap
root    101  0.0  0.2   360 ?     S    12:29 0:00 /usr/sbin/scanlogd
root    107  0.0  0.4   632 ?     S    12:29 0:00 /usr/sbin/syslogd
root    185  0.0  0.4   560 ?     S    12:29 0:00 /usr/sbin/inetd
root    200  0.0  0.4   572 ?     S    12:29 0:00 /usr/sbin/lpd
root    231  0.0  0.4   628 ?     S    12:29 0:00 /usr/sbin/cron
nobody  235  0.0  0.5   692 ?     S    12:29 0:00 /usr/sbin/in.identd
root    236  0.0  0.5   692 ?     S    12:29 0:00 /usr/sbin/in.identd
root    270  0.0  0.7  1016 tty1  S    12:29 0:00 login --
root
root    271  0.0  0.7  1016 tty2  S    12:29 0:00 login -- tobias
root    272  0.0  0.3   428 tty3  S    12:29 0:00 /sbin/mingetty tty3
root    273  0.0  0.3   428 tty4  S    12:29 0:00 /sbin/mingetty tty4
root    274  0.0  0.3   428 tty5  S    12:29 0:00 /sbin/mingetty tty5
root    275  0.0  0.3   428 tty6  S    12:29 0:00 /sbin/mingetty tty6
root    301  0.0  1.0  1328 tty1  S    12:30 0:00 -bash
```

```
tobias  345  0.0  1.0  1328 tty2  S    12:30 0:00 -bash
tobias 2784  0.0  1.5  1545 tty2  R    12:34 0:00 find
root   2785  0.0  0.7   928 pts/1 R    12:40 0:00 ps -aux
```

Dabei bedeuten die Spalten im einzelnen

USER: der aufrufende Benutzer des Prozesses;

PID: eindeutige ID eines Prozesses;

RSS: (Resident Set Size) Größe des Prozesses im Speicher in Kilobyte;

TTY: kontrollierender Terminal (ein ? bedeutet, daß diesem Prozeß kein Terminal zugeordnet ist);

Stat: Laufzeitzustand des Prozesses;

- R: Prozeß läuft im Moment;
- S: Prozeß “schläft”, befindet sich also im Wartezustand;
- D: Prozeß “schläft” und kann nicht zwingend unterbrochen werden;
- T: Prozeß ist gestoppt;
- Z: Prozeß ist ein Zombie;

Time: Bisher verbrauchte Prozessor-Zeit des Prozesses;

Command: Name des Kommandos, mit dem der Prozeß erzeugt wurde;

Der Befehl `top` gibt dieselben Informationen wie der Befehl `ps`, jedoch werden einige Zusammenfassungen und Gesamtspeicherinformationen zusätzlich angezeigt. `pstree` zeigt einen Prozeßbaum an, wobei gleiche Kommandos und Befehle zusammengefaßt werden und die Anzahl dieser als Multiplikator vorangestellt wird. Das nachfolgende Listing gibt eine gekürzte Ausgabe des Befehls `pstree` wieder:

```
init-+-atd
     |-cron
     |-httpd---httpd
     |-in.identd---in.identd---5*[in.identd]
     |-inetd
     |-kflushd
     |-klogd
     |-kpiod
     |-kswapd
     |-kupdate
     |-login---bash---pstree
     |-5*[mingetty]
     |-portmap
     |-scanlogd
     `-syslogd
```

4.2.2 Prozesse abbrechen

Um laufende Prozesse abzubrechen gibt es zum einen die Möglichkeit, mit der Tastenkombination STRG-C einen Prozeß abzubrechen. Leider funktioniert dies nicht immer, besonders, wenn die Prozesse abgestürzt sind oder als Zombies die Systemressourcen verbrauchen. Hier kommen die Kommandos `kill` und `killall` zu Hilfe.

Signal	ID	Funktion
SIGHUP	1	Hangup, zwingt viele Programme, ihre Konfigurationsdateien neu einzulesen
SIGINT	2	Interrupt, STRG+C
SIGQUIT	3	Abbruch
SIGKILL	9	Zwingender Abbruch, durch das Programm nicht abfangbar
SIGUSR1	10	Benutzerdefiniertes Signal
SIGTERM	15	Prozeß wird zum Beenden aufgefordert
SIGCONT	18	Ein gestoppter Prozeß wird fortgesetzt
SIGSTOP	19	Stoppt den Prozeß
SIGTSTP	20	Prozeß-Stop vom Benutzer gesteuert

Tabelle 4.2: Prozeß-Signale

Mit `kill` können Signale an einen Prozeß gesendet werden, nicht nur zum Abbrechen sondern z.B. auch um einen Prozeß anzuhalten und nachher wieder fortzusetzen. Beim normalen Aufruf von `kill` über `kill <PID>` wird dem Prozeß das Signal `SIGTERM` gesendet, welches den Prozeß zum Beenden auffordert. Reagiert der Prozeß auf dieses Signal nicht, kann die brachiale Methode verwendet werden, indem man über `kill -9 <PID>` oder `kill -SIGKILL <PID>` den zwingenden Abbruch des Prozesses fordert. Die Syntax des Befehls lautet:

`kill -Signal <PID>` oder `kill -Signal-ID <PID>`

Eine Auswahl der Signale ist in Tabelle 4.2 wiedergegeben.

Beim Befehl `killall` muß nur als Parameter der Name des Kommandos oder Befehls, welcher abgebrochen werden soll, angegeben werden und alle (!) Prozesse mit diesem Kommando- bzw. Befehlname werden abgebrochen.

4.2.3 Graphische Prozeß-Tools

Natürlich gibt es unter Linux mehrere graphische Tools, um die Arbeit und die Übersicht zu erleichtern. Das Programm `ktop` ist ein graphisches Front-End zum Befehl `top`. Mit diesem Programm können die Prozesse ausgewählt werden, welche anzeigt werden sollen (alle, System, Nutzer, eigene), ob als Liste oder als Prozeß-Baum und es können auch Prozesse abgebrochen werden. Zusätzlich kann man sich noch Informationen über die Systemauslastung anzeigen lassen (siehe Abbildung 4.1).

Das Programm `xosview` zeigt zwar nicht einzelne Prozesse an, bietet aber durch die Anzeige der Systemauslastung und der Arbeitsspeicherbelegung eine gute Übersicht über das System (siehe Abbildung 4.2).

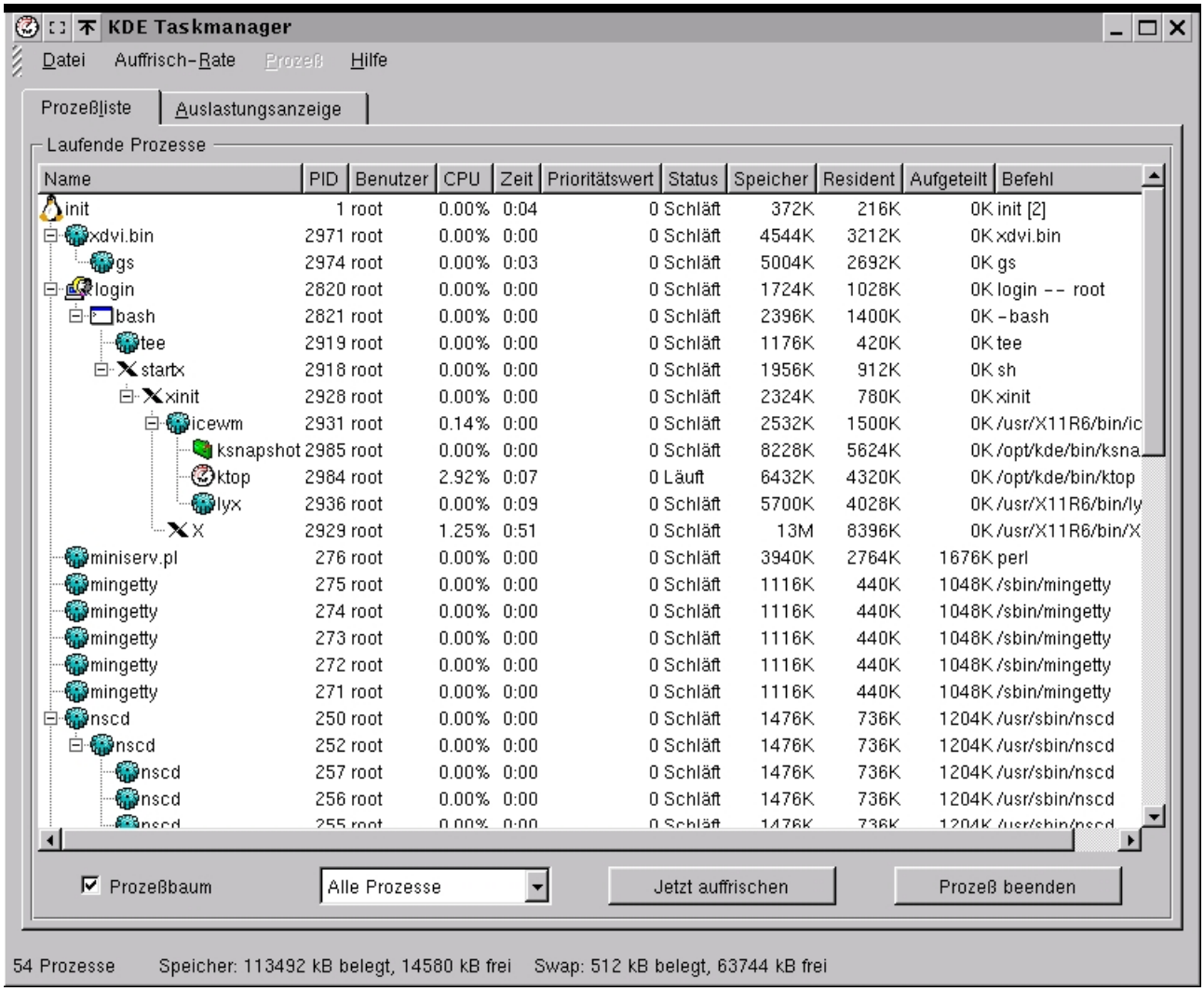


Abbildung 4.1: Das Programm `ktop`

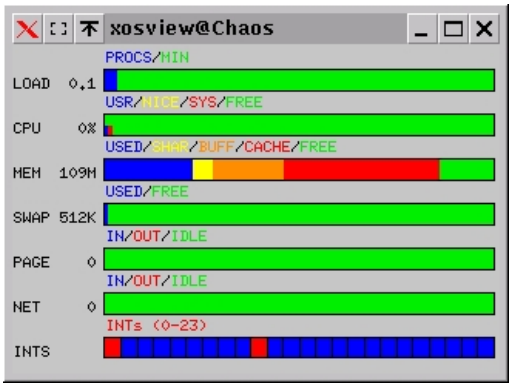


Abbildung 4.2: Das Programm `xosview`

Teil II

Linux-Grundlagen

Kapitel 5

Dokumentation und Hilfe

Unter Linux gibt es verschiedene Systeme für die Dokumentation und Hilfe. Einige davon sind schon vor der Installation unter DOS/Windows verfügbar, wie z.B. einige HOWTOs.

5.1 man - Das Online-Handbuch

Die wichtigste wird wohl das man-System sein. man-Seiten sind vor allem für Kommandos und Konfigurationsdateien verfügbar. Diese befinden sich in Verzeichnissen `/usr/man`, `/usr/X11R6-/man` und anderen. Schaut man sich mal das Verzeichnis `/usr/man` an, sieht man weitere Unterverzeichnisse, die unter anderem durchnumeriert sind. Diese Unterverzeichnisse spiegeln die Bereiche wieder, in welche die man-Seiten eingeordnet werden. Zur Zeit gibt es die folgenden 10 Bereiche:

1 Benutzerkommandos
2 Systemaufrufe
3 Funktionen der Programmiersprache C
4 Dateiformate, Device-Dateien
5 Konfigurationsdateien
6 Spiele
7 Diverses
8 Kommandos zur Systemadministration
9 Kernel-Funktionen
n neue Kommandos

Die Aufrufsyntax von man lautet:

```
man [optionen] [bereich] thema
```

Dabei sucht man die als Thema angegebene Manual-Datei in allen dem System bekannten man-Verzeichnissen (abgespeichert in der Variable `$MANPATH` in der Datei `/etc/profile`). Wird statt dem Thema eine Datei angegeben, wird diese angezeigt. Die optionale Angabe des Bereiches schränkt die Suche nach man-Texten auf einen Themenbereich ein.

Häufiger gibt es mehrere man-Texte zu einem Thema und im Normalfall wird nur die erste gefundene man-Seite angezeigt. Durch die Option "`-a`" werden alle gefundenen man-Seiten nacheinander angezeigt (zu den Optionen siehe auch Kapitel B.1 auf Seite 91).

Die zentrale Steuerungsdatei für man ist `/etc/man.config`, wo Änderungen von Pfadangaben, z.B. für die Cache-Verzeichnisse (bei SuSE `/var/cache/man`) oder das Anzeigeprogramm eingestellt werden können. Die Cache-Verzeichnisse sind nötig, da die man-Seiten normalerweise im

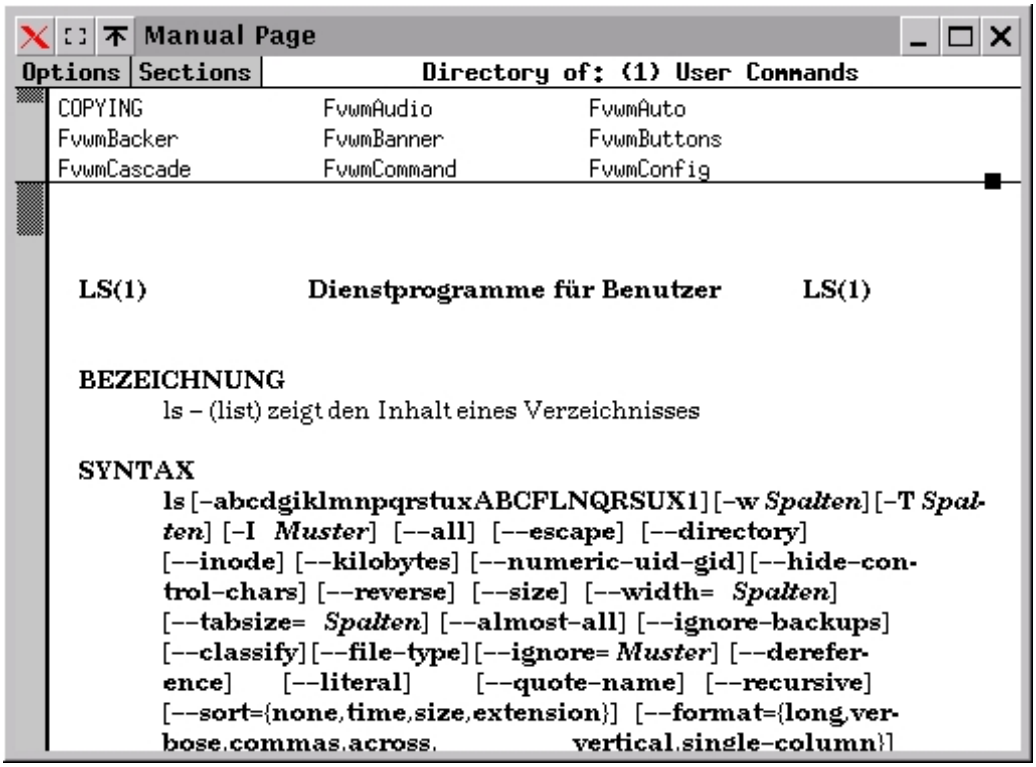


Abbildung 5.1: Das Programm `xman`

Quelltext vorliegen und vor dem ersten Aufruf mit `groff` ins ASCII-Format übersetzt werden. Damit dieser Vorgang nicht jedesmal wiederholt werden muß, werden die formatierten Quelltexte in den Cache-Verzeichnissen abgespeichert. Für weitere Möglichkeiten der Formatierung von `man`-Seiten möchte ich auf [Kofler, Seite 108ff] verweisen.

Zu den `man`-Seiten gibt es noch zwei weitere Kommandos. Der Befehl `apropos Schlüsselwort` zeigt alle Manualkurzbeschreibungen zum Schlüsselwort und der Befehl `whatis Kommando/Datei` durchsucht die Indexdatenbank nach Manualkurzbeschreibungen. Sollte der Befehl `whatis` nicht funktionieren, besteht die Möglichkeit, das die Indexdatenbank nicht aufgebaut wurde. Dieses kann über den Befehl `makewhatis` nachgeholt werden.

Unter dem X-Windows System stehen unter anderem die Befehle `xman` (siehe Abbildung 5.1) und `tkman` zur Verfügung, die eine graphische Oberfläche zu `man` bieten.

5.2 info und weitere Hilfe-Texte

Das Programm `info` ist wesentlich leistungstärker als der `man`-Befehl. Er erlaubt Querverweise, was vor allem für umfangreichere Hilfetexte sehr hilfreich ist. Aufgerufen wird `info` über:

```
info [optionen] [thema]
```

Zum Teil liegen die `info`-Dateien auch als HTML-Datei vor, so zum Beispiel bei SuSE, was das Lesen mit einem Browser ermöglicht. Ein `man`-Text zu `info` beschreibt die Grundzüge der Bedienung des Programms, worauf ich hier verweisen möchte.

Desweiteren gibt es im Verzeichnis `/usr/doc` weitere Hilfetexte und Dokumentationen. Unter anderem die Frequently Asked Questions (FAQ) im Unterverzeichnis `./FAQ` oder `./faq` und

die HOWTOs in Unterverzeichnis `./howto`. Bei der SuSE-Distribution gibt es zusätzlich das SuSE-Hilfesystem und die Support-Datenbank. Diese liegen zumeist in den Formaten ASCII, HTML, PostScript oder DVI vor. DVI-Dateien können per `dvips` zu PostScript umgewandelt und ausgedruckt werden oder weiter per `ps2pdf` in das PDF-Format konvertiert werden, wo sie mit dem Acrobat Reader gelesen und ausgedruckt werden können[1].

[1] Es gibt auch die Programme `xdvi` und `kdvi`, die DVI-Dateien anzeigen können sowie Programme wie `ghostview`, `gs`, `kghostview` ... um PostScript-Dateien anzuzeigen.

Kapitel 6

Installation vorbereiten

Bevor Sie mit der Installation eines Linux-Systems beginnen, sollten Sie so viele Informationen über die Rechnerhardware sammeln wie nur irgendwie möglich. Dazu gehört vor allem der Typ der Graphikkarte, damit bei der Installation der richtige X-Server ausgewählt werden kann. Genauso wichtig sind Informationen über die Festplatte, welche Partitionen sind vorhanden, wie groß ist die Festplatte und hab ich noch Plattenplatz frei, um neue Linux-Partitionen anlegen zu können. Sollte der gesammte Festplattenplatz durch Partitionen belegt sein, muß das System umpartioniert werden. Entweder durch Löschen einiger Paritionen oder über das Tool `fips` (meinst auf der CD-Rom im Verzeichnis `dosutils\fips`). Dieses Tool verkleinert bestehende Partitionen ohne Datenverlust. Bevor das Programm gestartet wird, muß die Partition defragmentiert werden, damit alle Dateien am Anfang der Platte stehen. Hierfür gibt es aber weitere Dokumentationen im Programmverzeichnis von `fips` oder in der beiligenden Linux-Dokumentation. Vor der Installation sollte man sich auch überlegen, welche Partitionen für Linux eingerichtet werden sollen und wie groß sie sind. Eine kleine Übersicht über Partitionierungsvorschläge finden Sie im Angang C auf Seite 104.

Das Starten der Installation kann über mehrere Wege erfolgen. Wenn das CD-ROM bootfähig ist, kann direkt über die CD-ROM gebootet werden, ansonsten kann eine Boot-Diskette erzeugt werden. Einige Distributionen bieten auch ein Setup-Programm an, welches von DOS, teilweise auch von Windows aus gestartet werden kann. Die SuSE-Distribution bietet zum Beispiel ein solches Setup-Tool für DOS.

6.1 Startdisketten erstellen

Das Erstellen einer Startdiskette erfolgt mit einem Programm, welches den Inhalt einer Image-Datei (bei SuSE im Verzeichnis “`images`” der CD-ROM) Sektor für Sektor vornimmt. Hierfür können Programme wie `RAWRITE.EXE` (im Verzeichnis `dosutils` der CD-ROM) oder das Programm `NCDD.EXE` aus den Norton Utilities verwendet werden. Das SuSE Setup-Programm kann auch eine solche Boot-Diskette erstellen.

Falls man schon ein lauffähiges Linux-System sein eigen nennen kann, kann die Boot-Diskette auch hierüber mit dem Befehl `dd` erstellt werden. Die Syntax für die Erstellung lautet:

```
dd if=/cdrom/images/boot.img of=/dev/fd0 bs=8192
```

Dabei ist der Pfad nach “`if=`” die Pfad- und Dateiangabe des Image und `/dev/fd0` das erste Floppy.

6.2 Festplatte partitonieren mit fdisk

Das Linux-Partitionierungsprogramm wird über den Befehl `fdisk` aufgerufen. Hier können dann primäre und erweiterte Partitionen sowie logische Laufwerke erstellt werden. Desweiteren gibt es die Möglichkeit, den Typ des Dateisystems festzulegen und die aktive Partition zu markieren. In der Tabelle 6.1 auf Seite 32 sind einige unterstützte Dateisystem mit ihren IDs aufgeführt.

ID	Bezeichnung	ID	Bezeichnung
0	Empty	65	Novell Netware
1	FAT12	**80**	**Old Minix**
4	FAT16 <32M	**81**	**Minix / old Linux**
5	Extended	**82**	**Linux swap**
6	FAT15	**83**	**Linux**
7	HPFS/NTFS	**85**	**Linux extended**
a	OS/2 Boot Manager	86	NTFS volume set
b	Win95 FAT32	87	NTFS volume set
c	Win95 FAT32 (LBA)	a6	Open BSD
e	Win95 FAT16 (LBA)	c1	DRDOS/sec (FAT12)
f	Win95 Extended (LBA)	c4	DRDOS/sec (FAT16 < 32M)
12	Compaq diagnostic	c6	DRDOS/sec (FAT16)
24	NEC DOS	eb	BeOS fs
3c	Partition Magic	f2	DOS secondary
55	EZ-Drive	**fd**	**Linux raid auto**
64	Novell Netware	fe	LANstep

Tabelle 6.1: Von “fdisk” unterstützte Dateisysteme (Auswahl der Wichtigsten)

6.3 Swap-Partition oder Swap-Datei einrichten

Die Swap-Partition oder Swap-Datei ist ein zusätzlicher Arbeitsspeicher auf der Festplatte (die Auslagerungsdatei unter Windows). Hier wird der Arbeitsspeicher ausgelagert, wenn er zu klein ist. Normalerweise wird die Swap-Partition bei der Installation eingerichtet, ohne daß man sich mit den nachfolgenden Befehlen beschäftigen muß. Manchmal ist es jedoch nötig, selbst Hand anzulegen. Dies kann dann der Fall sein, wenn man nachträglich merkt, daß der Swap-Bereich zu klein ist oder für die Installation zu wenig RAM zur Verfügung steht (bei SuSE 6.3 benötigt die Installation schon eine RAM-Disk von etwa 14 MB!). Dabei sollte der Swap-Partition den Vorzug gegeben werden.

Wenn eine neue Swap-Partition verwendet werden soll, muß diese zuvor mit `fdisk` und dem Dateisystemtyp 82 (Linux-Swap) angelegt werden. Der Befehl:

```
mkswap -c <device>
```

“formatiert” die Swap-Partition und der Befehl:

```
swapon <device>
```

aktiviert die Swap-Partition. Soll diese Swap-Partition bei jedem Systemstart aktiviert werden, sollte sie in die Datei `/etc/fstab` eingetragen werden (siehe Kapitel 6.2, Seite 32).

Für die Anlegung einer Swap-Datei muß folgende Befehlsfolge angegeben werden:

```
root@serverI: # dd if=/dev/zero of=<datei> bs=1024 count=<blocks>
root@serverI: # sync
root@serverI: # mkswap -c <datei>
root@serverI: # sync
root@serverI: # swapon <datei>
```

Dabei wird zuerst eine Datei, gefüllt mit Null-Bytes, der Blockgröße 1024 und der Anzahl der Blocks, angelegt. Anschließend wird sie wieder "formatiert" und aktiviert.

6.4 LILO oder Loadlin einrichten

Zum Starten eines Linux-System stehen zwei Möglichkeiten zur Verfügung, Loadlin oder ein Boot-Manager. Als Boot-Manager kommt normalerweise LILO zum Einsatz, der **Li**nux **Lo**ader.

6.4.1 Loadlin

Loadlin ist ein Programm, welches unter DOS läuft und den Linux-Kernel laden kann, so das hiermit ein Linux-System gestartet werden kann. Beim Starten von `LOADLIN.EXE` muß der zu ladene Kernel, die Linux Root-Partition und bei Bedarf weitere Kernelparameter angegeben werden. Die Verwendung von Loadlin hat einige Vorteile gegenüber einem Boot-Manager: es werden keine Änderungen am MBR vorgenommen und es kann gut in ein DOS / Windows 9x Startmenü mit eingebunden werden. Der Nachteil ist natürlich, daß eine Partition mit DOS / Windows vorhanden sein muß, so das es häufiger für ein System mit mehreren Betriebssystemen eingesetzt wird. Sollte unter Linux ein neuer Kernel kopiliert werden, muß dieser in eine Partition verschoben werden, die von DOS lesbar ist. Im Setup-Programm von SuSE kann man Loadlin auf die Festplatte kopieren.

6.4.2 LILO

Der am häufigsten benutzte Boot-Manager ist der Linux Loader LILO. Viele Administrationstools, wie zum Beispiel YaST unter SuSE, erleichtern die Konfiguration mit einer graphischen Oberfläche. Zentrale Konfigurationsdatei für LILO ist die Datei `/etc/lilo.conf`. Hier werden alle Einträge für LILO vorgenommen. Das nachfolgende Listing zeigt eine einfache LILO-Konfigurationsdatei für ein Computersystem, wo nur Linux installiert ist:

```
append="mem=0x8000000"
boot=/dev/hda
vga=normal
message=/boot/lilo.msg
read-only
prompt
timeout=50
# die eigentlichen Kernel-Eintragungen
image = /boot/vmlinuz.2.2.13
  root = /dev/hda3
  label = 1
```

Die erste Zeile übergibt dem Kernel weitere Parameter, hier die Größe des Arbeitsspeichers in hexadezimaler Form (bei manchen Computersystemen ist diese Angabe notwendig, da sonst eine falsche Größe für den Arbeitsspeicher erkannt wird), die zweite Zeile gibt den Installationsort von LILO an, z.B. wie hier den MBR der ersten Festplatte. Die Option "`vga`" gibt den Videomodus an, in dem gestartet werden soll, dabei kann unterschieden werden zwischen 80x25 oder 80x50, "`normal`" bezeichntet den ersten Modus. Um nicht nur beim Starten von LILO den LILO-Prompt zu erhalten, kann mit der Option "`message`" eine Textdatei angegeben werden, die beim Starten angezeigt wird. "`Read-Only`" sorgt dafür, daß die Root-Partition zuerst im Read-Only-Modus gemountet wird, damit die Dateisystemprüfung vorgenommen werden kann, "`prompt`" gibt den LILO-Prompt aus. Die Einstellung "`timeout`" gibt die Wartezeit in zehntel-Sekunden an, bis der erste Kerneleintrag als Default-Kernel geladen wird wenn keine anderen Eingaben am LILO-Prompt vorgenommen werden. Danach folgt ein Kernel-Eintrag zum booten. Zu einem Kernel-Eintrag gehören 3 Werte: `image`, der die Kernel-Datei benennt, `root` gibt die Root-Partition an und `label` den Eintrag, der am LILO-Prompt einzugeben ist, damit dieser Kernel geladen wird.

Sollte keine LILO-Meldung eingestellt sein, die die `label`-Namen für die Kernel-Eintragungen angibt, kann mit der `Tab`-Taste am LILO-Prompt eine Liste mit Kernel-Eintragungen angezeigt werden.

Zu beachten sind noch zwei Dinge. Bei jeder Änderung an der Kerneldatei, z.B. verschieben, neuen Kernel mit gleichen Namen in das Verzeichnis /boot hineinkopieren, Änderung der Konfigurationsdatei usw., muß LILO neu installiert werden. Hierfür muß nur das Kommando `lilo` ausgeführt werden. Sollten sich Fehler eingeschlichen haben, werden diese angezeigt. Zum anderen muß LILO sowie die Partition für das Verzeichnis `/boot` in den ersten 1024-Zylindern liegen, damit LILO die benötigten Dateien lesen kann. Für weitere Konfigurationseinstellungen und besonders Systeme, mit anderen Boot-Managern (z.B. Windows NT Boot-Manager) möchte ich auf das Linux-Handbuch, welches mit der Distribution mitgeliefert wird, auf [Kofler, Seite 222ff] und auf die LILO-Dokumentation (z.B. `man lilo` oder `man lilo.conf`) verweisen.

Kapitel 7

Arbeiten auf der Kommandozeile

Zwar können immer mehr Arbeiten an einem Linux System mit Tools, die unter dem X-Windows System arbeiten, erledigt werden und die Bedienung lehnt sich mehr und mehr an den Klick-Mechanismus an, den wir von Windows, dank KDE, Gnome und Co, gewöhnt sind. Trotzdem ist es nicht nur den Unix-Cracks vorbehalten, auf der Kommandozeile zu arbeiten. In diesem Abschnitt erfolgt eine kurze Einführung zur BASH-Shell, die Standard-Kommandozeile unter Linux. Ausführlicher werden die Shells im Kapitel 10 ab Seite 53 besprochen.

In Abbildung 7.1 sehen Sie eine `Telnet`-Einwahl auf einem Linux-Rechner. Bei jedem Linux-System erfolgt zuerst die Anmeldung mit Name und Paßwort. Linux gibt weitere Informationen, z.B. welcher Kernel auf dem System läuft (hier *i386-Kernel* mit Version 2.2.13), auf welchem Terminal man sich gerade befindet (hier *pts/7*) und wann der letzte Login war. Sollten seit der letzten Anmeldung fehlgeschlagene Logins erfolgt sein, wird dies mit ausgegeben. Nach der Anmeldung werden die benutzerspezifischen Einstellungen gelesen und der Prompt ausgegeben (hier ~ #). Dabei steht die Tilde (~) für das Heimatverzeichnis des Benutzers.

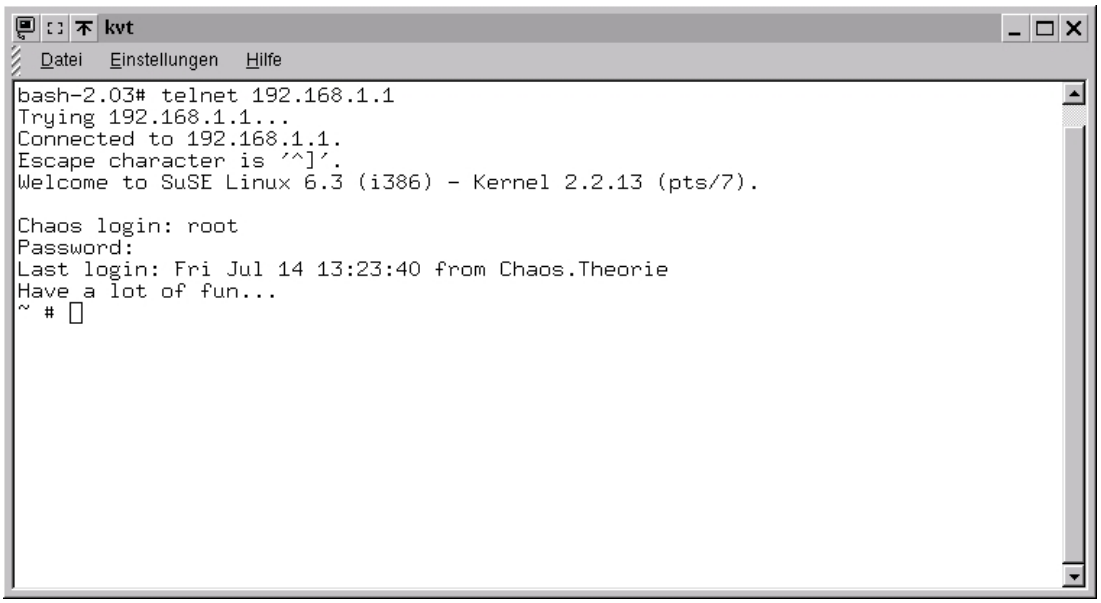


Abbildung 7.1: Einwahl auf einem Linux-System per Telnet

Nun steht die Shell für Ihre Eingaben bereit. Wenn Sie nun den Inhalt des aktuellen Verzeichnisses angeben lassen wollen, geben sie den Befehl `ls` ein und es erfolgt folgende Ausgabe:

```
~ # ls

Desktop   Office51   StarOffice   bin
Mail      ServerLog  StartLog     fte-new.cnf
```

Der Befehl `ls -la` gibt die Ausgabe im Langformat sowie alle versteckten Dateien (das sind Dateien, die mit einem Punkt beginnen) aus:

```
~ # ls -la
insgesamt 289
drwx--x--x  23 root     root         1024 Jul 17 16:47 .
drwxr-xr-x  24 root     root         1024 Jul  1 22:28 ..
-rw-r--r--   1 root     root         2908 Jul 18 13:35 .X.err
-rw-------
1 root     root         5890 Jul 18 10:30 .bash_history
-rw-r--r--   1 root     root       106692 Jul  1 21:01 .fterc
drwxr-xr-x   2 root     root         1024 Jul  8 16:30 .icewm
drwxr-xr-x   3 root     root         1024 Jul  1 21:37 .kde
-rw-r--r--   1 root     root          486 Jul 17 11:32 .kderc
-rw-r--r--   1 root     root        11666 Jul  8 12:45 .pinerc
drwxr-xr-x   2 root     root         1024 Jul  1 21:37 .skel
drwx------   5 root     root         1024 Jul  1 21:37 Desktop
drwx------   2 root     root         1024 Jul 14 13:51 Mail
drwxr-xr-x  22 root     root         1024 Jul  2 09:21 Office51
-rw-r--r--   1 root     root         1993 Jul 13 20:35 ServerLog
-rwxr-xr-x   1 root     root          573 Jul  2 09:21 StarOffice
-rw-rw-rw-   1 root     root         1189 Jul 13 20:36 StartLog
drwxr-xr-x   2 root     root         1024 Jul  1 22:29 bin
-rw-r--r--   1 root     root       106692 Jul  1 21:00 fte-
new.cnf
```

Dabei gibt die erste Spalte die Zugriffsrechte an, die zweite die Anzahl der Verweise, danach folgen Besitzer, Gruppe, Größe, letztes Änderungsdatum und der Name. Die ersten beiden Einträge sind spezielle Verzeichnisse. Das Verzeichnis "`.`" kennzeichnet das aktuelle Verzeichnis und "`..`" das übergeordnete Verzeichnis.

Das Verzeichnisse wechseln können Sie über `cd <Verzeichnis>`. Wird `cd` ohne Verzeichnis aufgerufen, wird ins Home-Verzeichnis des Benutzers gewechselt, das selbe erfüllt `cd ~`. Dateien können mit `cat <Dateiname>` angezeigt werden und seitenweise über `cat <Dateiname> | more` bzw. `cat <Dateiname> | less`.

Navigieren innerhalb des Kommandozeile können Sie mit den Pfeiltasten, links und rechts navigieren innerhalb des Textes um z.B. Änderungen vorzunehmen und die Hoch- und Runter-Taste in der History-Liste, um einen alten Befehl nocheinmal aufzurufen (diese Funktion bieten nicht alle Shells, aber z.B. die `bash`). Ein weiteres Feature ist die `TAB`-Taste. Hiermit können Kommando- und Dateinamen expandiert werden. Hierfür wird in allen Verzeichnissen, die in der Variablen Pfad gespeichert sind, nach einem entsprechenden Namen gesucht bzw. bei der Angabe eines Pfades in dem jeweiligen Verzeichnis. Sollten mehrere Namen expandiert werden können, ertönt ein Signalton und durch dochmaliges Drücken der `TAB`-Taste wird eine Liste von möglichen Namen angezeigt.

Im Kapitel 2.1.1 auf Seite 10 haben Sie schon gehört, daß mehrere Konsolen gestartet werden können und zum `login` bereitstehen. Standardmäßig befinden Sie sich in der Konsole 1. Über die ALT-Taste in Verbindung mit den Funktionstasten können Sie in andere Konsolen wechseln, sich anmelden und Kommandos ausführen. Sollten Sie keine Änderungen an `/etc/inittab` diesbezüglich vorgenommen haben, stehen mit `ALT-F1` bis `ALT-F6` die sechs Konsolen zur Verfügung, mit `ALT-F10` wechseln Sie in die System-Konsole, wo der Kernel Meldungen ausgibt und diese in der Datei `/var/log/messages` abspeichert. Sollten Sie sich im X-Windows System befinden, gelangen Sie in die Konsolen über die Tastenkombination `STRG-ALT-F`*n*. Möchten Sie wieder zurück ins X-Windows System, wechseln Sie auf Konsole 7 (`ALT-F7`).

Zu jeder Datei und jedem Verzeichnis werden Besitzer, Gruppe und Zugriffsrechte gespeichert. Die Zugriffsrechte sind in drei Gruppen eingeteilt: Besitzer (u), Gruppe (g) und alle anderen (o). Hierfür stehen dann jeweils 3 Zugriffsrechte bereit: read (r), write (w) und execute (x). Mit den drei Befehlen `chown`, `chgrp` und `chmod` können diese Informationen verändert werden (siehe Anhang B.10, Seite 103).

7.1 Links

Links sind Verweise auf Dateien um z.B. eine Datei über mehrere Verzeichnisse verfügbar zu machen. Dies geschieht völlig transparent, so daß der Benutzer gar nicht merkt, daß er auf einen Link zugreift. Hierbei muß zwischen zwei verschiedenen Link-Arten unterschieden werden: feste und symbolische.

Feste Links sind ein neuer Verweis auf den Speicherbereich einer Datei. Daher besitzen sie auch dieselbe I-Node wie die Ursprungsdatei. Bei jedem Anlegen eines neuen festen Links wird ein Zähler um eins erhöht um beim Löschen um eins dekrementiert. Wenn der Zähler auf Null steht, gilt die Datei als gelöscht und der Speicherbereich wird freigegeben. Da der I-Node gespeichert wird, sind nur feste Links auf der selben Partition möglich (jede Partition besitzt ihre eigene I-Node-Tabelle). Einen festen Link können Sie durch den Befehl:

```
ln <Quelle> <Ziel>
```

anlegen. Dagegen wird bei symbolischen Links der Pfad und Dateiname gespeichert. Hierdurch verändert sich beim Löschen der Ursprungsdatei der Link zwar nicht, jedoch zeigt er dann ins Leere. Wird der Link gelöscht, hat dies keinen Einfluß auf die Ursprungsdatei. Symbolische Links können Partitionsübergreifend eingesetzt werden und auch auf Verzeichnisse Links gesetzt werden. Bei symbolischen Links sollten relative Pfadangaben benutzt werden, da dies Probleme z.B. beim Verschieben von ganzen Verzeichnisbäumen verhindern kann. Symbolische Links werden durch Angabe der Option `-s` des `ln`-Befehls angelegt:

```
ln -s <Quelle> <Ziel>
```

Das nachfolgende Listing zeigt ein Verzeichnis mit festen und symbolischen Links. Dabei ist `ks.eps` ein fester Link auf `ksysv.eps` und `ksysv2.eps` ein symbolischer. Sie sehen, bei den festen Links ist die in der ersten Spalte angegebene I-Node-Nummer gleich (durch die Option `-i` bei `ls`) und beim symbolischen Link wird der Pfad angegeben.

```
~/bilder # ls -li
insgesamt 5223
   6165 -rw-r--r--   2 root  root  1496097 Jul 16 10:48 ks.eps
   6165 -rw-r--r--   2 root  root  1496097 Jul 16 10:48 ksysv.eps
   6162 lrwxrwxr-
wx   1 root  root        9 Jul 18 14:28 ksysv2.eps -> ksysv.eps
```

7.2 Einbinden von Dateisystemen

Unter Linux werden die einzelnen Partitionen, logische Laufwerke und externe Geräte (CD-ROM, Diskettenlaufwerk) nicht jeweils über eigene Laufwerksbuchstaben, sondern in einen gesamten Verzeichnisbaum eingebunden. Das Laufwerk, welches alle anderen Dateisysteme aufnimmt und an oberster Stelle steht, wird Root-Verzeichnis (Wurzel) genannt (nicht zu verwechseln mit dem Home-Verzeichnis des Benutzers "`root`").

Bevor Sie ein Dateisystem benutzen können, müssen Sie dieses formatieren (mit dem Befehl "`mkfs`") und danach in den Verzeichnisbaum einbinden. Dabei spielt es keine Rolle, ob es sich um eine lokale Festplattenpartition, einen externen Datenspeicher (z. B. CD-ROM) oder um ein Netzwerkdateisystem, womöglich von einem Rechner Tausende Kilometer entfernt, handelt. Der Befehl hierfür ist "`mount`" mit folgendem Syntax:

```
mount -t [dateisystemtyp] [Gerätedatei] [Mount-Punkt]
```

Beispiel:

```
mount -t msdos /dev/hda1 /mnt
```

Anstatt der Gerätedatei muß bei Netzwerkdateisystemen der Rechner- und Freigabename angegeben werden[1].

Die Gerätedateien symbolisieren das Gerät, auf das sich bezogen wird. Die einzelnen IDE-Controller-Ports haben die Bezeichnung "hd + Buchstabe", wobei von "a" an durchbuchstabiert wird, für den SCSI-Controller die Bezeichnung "sd + Buchstabe" und für SCSI-CD-ROMs “scd + Nummer”.

Device	Beschreibung
hda	Primärer IDE-Controller, Master
hdb	Primärer IDE-Controller, Slave
hdc	Sekundärer IDE-Controller, Master
hdd	Sekundärer IDE-Controller, Slave
sda	erstes SCSI-Gerät
sdb	zweites SCSI-Gerät
scd1	erstes SCSI CD-Rom
scd2	zweites SCSI CD-Rom
fd0	erstes Floppy-Laufwerk
fd1	zweites Floppy-Laufwerk

Tabelle 7.1: einige Gerätedateien und Beschreibung

[1] Form: `rechnername:/verzeichnis`

Die einzelnen Partitionen auf dem Gerät werden mit Nummern symbolisiert, d. h. die primären und erweiterten Partitionen auf dem Gerät werden von 1 - 4 nummeriert, die logischen Laufwerke bekommen Nummern ab 5.

Anstatt ständig selber die gesamten Dateisysteme per Hand zu mounten (so nennt man das Einbinden), kann dieses das System selbstständig erledigen. Zumal nur der Administrator (root) dieses darf. Das kann über die Datei `/etc/fstab` erreicht werden. In dieser Datei werden die Gerätedateien, das dazugehörige Dateisystem und der Mount-Point eingetragen, so daß das Betriebssystem alle Informationen besitzt, um diese Aufgabe selbstständig zu erledigen (siehe dazu auch Kapitel 17 auf Seite 81).

Nun ist es unpraktisch, ein CD-ROM Laufwerk oder eine Floppy beim Boot-Vorgang ins Dateisystem einzubinden, da diese im Betrieb häufiger gewechselt werden müssen. Vor allem, weil man eine eingebundene CD nicht aus dem CD-ROM Laufwerk hinausbekommt, da Linux den Öffnen-Knopf sperrt. Andererseit wäre es auch nicht praktisch, jedesmal den vollen Mountbefehl mit allen Parametern anzugeben, da dies auch nur "`root`" darf. Auch hierfür gibt es eine Lösung. In der `/etc/fstab`-Datei kann jede Gerätedatei über die Option `noauto` so eingetragen werden, daß sie nicht beim booten automatisch eingebunden wird, sondern daß man für den Mount-Befehl eine verkürzte Schreibweise anwenden kann. Z.B. wenn man "`mount /dev/hdc`" eingibt, wird in der Datei /etc/fstab nachgeschaut, ob es einen Eintrag mit der betreffenden Gerätedatei gibt und wenn ja, dann werden die restlichen Parameter hinzugefügt. Genauso kann man in `/etc/fstab` auch die Option "`user`" eintragen, so daß jeder Benutzer den Mount-Befehl auf diese Gerätedatei anwenden darf.

Dieses Vorgehen hat noch einen großen Vorteil. Dateisysteme, die selten benötigt werden können später einfach eingebunden werden und ein nicht eingebundenes Dateisystem kann im Grunde nicht zerstört werden, z.B. durch löschen oder Formatierung.

Um ein Dateisystem auszubinden gibt es den Befehl "`umount [Gerät]`". Dabei wird überprüft, ob noch Schreiboptionen auf das Dateisystem ausgeführt werden müssen, die Daten synchronisiert und ob noch jemand auf Daten von diesem Dateisystem zugreift. Linux ist ja ein Multi-User Betriebssystem, woauf mehrere Benutzer gleichzeitig arbeiten können und da wäre es ziemlich gefährlich, wenn jemand ein Dateisystem ausbindet, wenn jemand anderes noch darauf zugreift.

Kapitel 8

Daemonen

Daemonen sind spezielle Hintergrundprozesse zur Systemverwaltung, die normalerweise über `init` beim booten gestartet werden. Wenn Sie mit `ps -aux` sich alle Prozesse anzeigen lassen, werden Sie feststellen, daß bei einigen Einträgen in der Terminal-Spalte ein Fragezeichen steht. Dies sind Daemonen, da sie keinem Terminal zugeordnet werden. Meistens enden die Namen eines Daemons mit einem "d". Wichtige Daemonen sind:

crond — Batchdaemon, der routinemäßige Aufgaben erledigen kann; die Steuerung erfolgt über `/etc/crontab` und `/var/spool/cron/crontabs`;

inetd — Startet weitere Netzwerk-Daemonen; Steuerung über `/etc/inetd.conf`;

klogd — Protokoliert Kernel-Meldungen in `/var/adm/*` bzw. `/var/log/*`;

syslogd — Protokoliert Systemmeldungen in `/var/adm/*` bzw. `/var/log/*`; gesteuert über `/etc/syslog.conf`;

lpd — Druckerspooler;

smbd — der Samba-Daemon (zusätzlich muß noch der `nmbd`-Daemon gestartet werden), um Verzeichnisse und Drucker für Windows-Rechner verfügbar zu machen; die Konfigurationsdatei lautet `/etc/smb.conf`, Logbücher befinden sich in den Dateien `/var/log/log.smb`, `/var/log/log.nmb` und `/var/log/smb.log` wenn in der Konfigurationsdatei nichts anderes eingestellt wurde;

kerneld — Kernel-Daemon zum automatischen Laden von Modulen; Modulkonfiguration über `/etc/modules.conf`

named — Nameserver-Daemon des BIND-Paketes[1]; die Konfigurationsdatei lautet `/etc/named.boot`, die Zonendateien befinden sich meistens in `/var/named/*`;

dhcpd — Daemon des DHCP-Servers (DHCP = Dynamic Host Configuration Protocol), Konfiguration über `/etc/dhcpd.conf`;

[1]Das BIND-Paket (Berkeley Internet Name Domain) ist ein Domain Name Server (DNS), der auf den meisten UNIX-Systemem verfügar ist. Derzeit gibt es zwei Versionen, v4 und v8, wobei die Syntax der beiden Versionen unterschiedlich ist. Für ein sicheres Netz ist die BIND-Version 8 zu empfehlen.

sendmail Server-Programm für den eMail-Verkehr, der auch als Daemon gestartet werden kann; `sendmail` ist der am weitesten verbreitete Mail-Server und auf vielen UNIX- und Linux-Rechnern implementiert; die Konfigurationsdatei für `sendmail` heißt `/etc/sendmail.cf`[2].

httpd Daemon des Web-Servers, am weitesten verbreitet ist der Apache Web-Server, den es für (fast) alle Rechner-Plattformen konstenlos erhältlich ist; die Konfigurationsdateien für Apache lieben bei der SuSE-Distribution im Verzeichnis `/etc/httpd/*`[3];

[2] Sie kann jedoch auch in den Verzeichnissen `/etc/mail` und `/usr/lib` sich befinden.
[3] Die Hauptkonfigurationsdatei des Apache-Servers lautet `httpd.conf`.

Kapitel 9

Paketinstallation und Kompilierung

Im Laufe der Linux-Systemadministration werden Sie irgendwann vor dem Problem stehen, Pakete, die Sie aus dem Internet heruntergeladen haben oder auf irgendeiner CD-ROM finden, zu installieren. Dabei kann es sich um Dateien mit dem Dateisuffix `.rpm`, `.deb` oder `.tar.gz` bzw. `.tgz` handeln. Das Paketformat `rpm` wurde von RedHat entwickelt und wird unter anderem von den Distributionen RedHat, Caldera, DLD und SuSE verwendet, die Distribution Debian benutzt das Paketformat `deb`. Beide Paketformat haben einige Vorteile gegenüber `.tar.gz` bzw `.tgz`. Unter anderem lösen sie Paketabhängigkeiten auf, z.B. wird für Paket x auch noch das Paket y zur Ausführung benötigt. Dateien können bei diesen Formaten Paketen zugeordnet werden und Updates sind unproblematischer, da Informationen über den Zweck der Dateien im Paket gespeichert werden und Konfigurationsdateien nicht ohne weitere überschrieben werden (z.T. werden alte Konfigurationsdateien bei SuSE dann gesichert). Vorteile werden wie immer mit Nachteilen erkauft. So sind diese Paketformate nicht so portabel wie `tar.gz` bzw. `tgz`, zum anderen wird der `rpm` Paket-Standard ständig weiterentwickelt und die einzelnen Versionen sind teilweise inkompatibel, so daß Sie möglicherweise zuerst die richtige `rpm`-Version herunterladen müssen.

Eigentlich jede Distribution hat ein entsprechendes Tool, um `rpm`- bzw. `deb`-Pakete zu installieren. Bei RedHat und Caldera ist es das graphische Programm `glint`, unter DLD `viper`, bei SuSE `YaST` und bei Debian `dselect` und `dpkg`. Dazu gibt es noch die distributionsunabhängigen Tools wie `xrpm`, `kpackage` `kPackViewer` ...

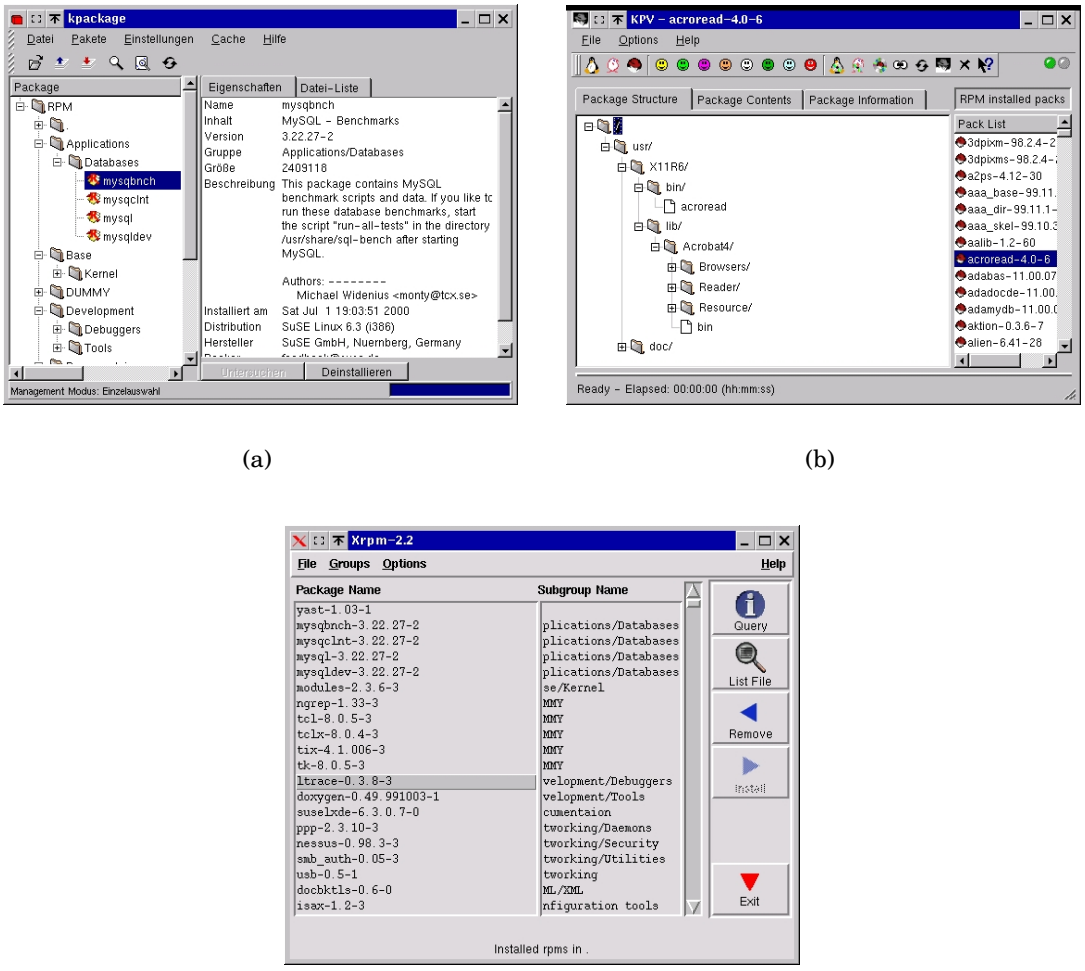
9.1 Pakete installieren

9.1.1 `rpm`-Pakete installieren

Bei der Installation von externen `rpm`-Paketen, also solche, die nicht von der laufenden Distribution stammen, sollte man vorsichtig vorgehen. Wird zum Beispiel ein aktuelleres SysV-Paket einer RedHat-Distribution über eine SuSE-Distribution installiert, ist das System nicht mehr lauffähig (siehe dazu Kapitel 2.1 auf Seite 10). Aus diesem Grund sollten nur `rpm`-Pakete nachträglich installiert werden, die zur Distribution gehören.

Der Name eines rpm-Paketes gibt schon viele Informationen über das Paket an sich aus. Sie sind nach dem Schema

```
<name>-<versions-nr.>-<rpm-release-nr>-<rechner-architektur>.rpm
```



(a) (b)

(c)

Abbildung 9.1: Die Paketprogramme (a) `kpackage`, (b) `kPackViewer` und (c) `xrpm`

bezeichnet. Dabei gibt eine `rpm`-Release-Nummer größer 1 eine Änderung der Zusammenstellung an. Für `<rechner-architektur>` werden ihnen am häufigsten die Bezeichnungen `i386` und `src` begegnen. `i386` bezeichnet ein Paket mit Binärdateien für die Intel-Architektur und `src` Quellcodepakete. Über alle installierten Binärpakete wird eine Datenbank verwaltet, sie befindet sich im Verzeichnis `/var/lib/rpm`, vor allem, um Paketabhängigkeiten zu überprüfen und aufzulösen.

Installiert wird ein Paket über die Befehlsfolge:

```
rpm -i [optionen] dateiname
```

Die Parameter `--test` und `--root verzeichnis` können mit angegeben werden, um eine Dummyausführung (Testausführung ohne Änderungen) bzw. den Installationsort zu verändern. Soll der Installationsort eines Quellcodepaketes verändert werden, muß die Datei `/etc/rpmrc` modifiziert werden.

Eine Aktualisierung eines installierten Paketes erreicht man durch

```
rpm -u [optionen] dateiname
```

und die Entfernung über

```
rpm -e [optionen] paketname
```

Manchmal ist es auch nötig festzustellen, zu welchem Paket eine Datei gehört. Hierfür kann die Befehlsfolge

```
rpm -qf Dateipfad und -name
```

verwendet werden. Als Dateiname kann bei der Installation und Aktualisierung auch der komplette Pfad zu einem FTP-Server angegeben werden. Für weitere Möglichkeiten mit dem `rpm`-Paket möchte ich auf [Kofler, Seite 253ff], den Anhang B.6, Seite 99 und die Online-Dokumentation verweisen. Die oben genannten Tools können häufig diese Aufgabe erledigen und erleichtern die Installation und Wartung von `rpm`-Paketen.

9.1.2 `tar.gz`- bzw. `tgz`-Pakete installieren

Sollten Sie Software aus dem Internet herunterladen, wird Ihnen am häufigsten dieses Paketformat begegnen. Hierbei handelt es sich um `tar`-Achive (steht für **T**ape **Ar**chive) die mit `gzip` komprimiert wurden. Dies ist deswegen nötig, da `tar` nicht selber komprimieren kann. Weitere Komprimierungsverfahren, die zusammen mit `tar` benutzt werden, sind `compress` (Endung `.Z`) und `bzip2` (Endung `.BZ2`).

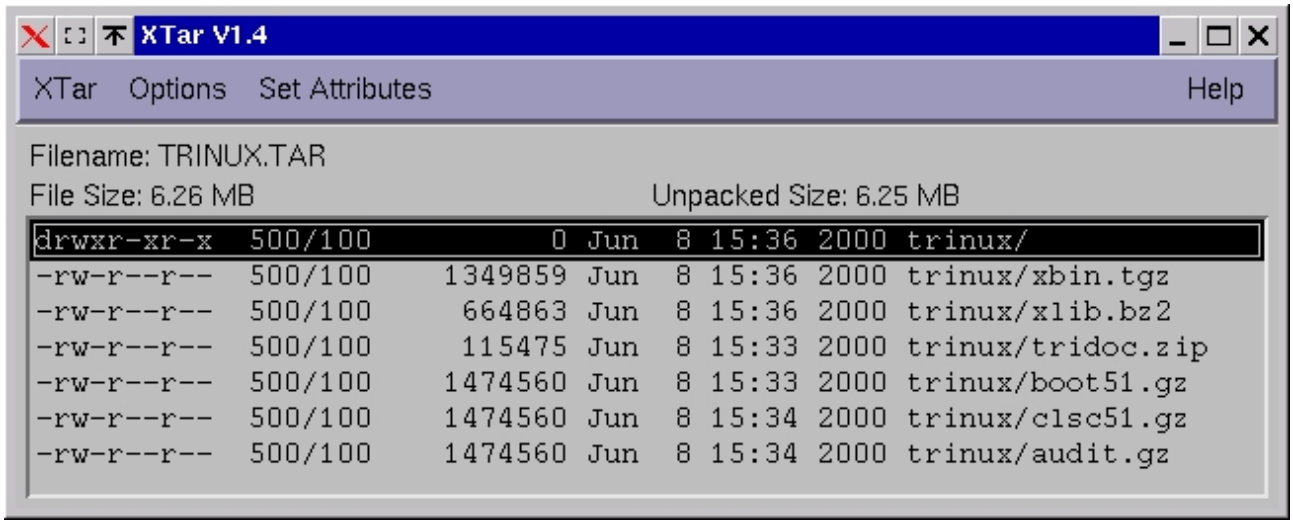


Abbildung 9.2: Das Archivprogramm `xtar`

Den Inhalt eines gezipten Tar-Archives kann mit

```
tar -tzf <archivname>
```

angezeigt werden. Relativ zum aktuellen Verzeichnis wird mittels

```
tar -xzf <archivname>
```

entpackt und über

```
tar -xzf <archivname> -C <verzeichnis>
```

in das angegebene Verzeichnis. Soll nur ein Teil der Dateien, z.B. alle `.tex`-Dateien, extrahiert werden, kann dies nach dem Paketnamen angegeben werden.

```
tar -xzf <archivname> "*.tex"
```

Sollte es sich um ein mit `compress` gepacktes Archiv handeln, muß der Options-Parameter "z" durch "Z" und bei einer `bzip2`-Komprimierung "z" durch "I" ersetzt werden. Natürlich ist es auch möglich, die beiden Schritte (Dekomprimierung und Archivextrahierung) nacheinander durchzuführen. Eine solche Befehlsfolge lautet dann für ein `gzip`-komprimiertes Archiv:

```
gzip -d <archivname>.tar.gz
tar -xvf <archivname>.tar
```

Die Option "v" gibt jede extrahierte Datei mit Pfadnamen zusätzlich aus (weitere Optionen siehe Anhang B.6 auf Seite 98).

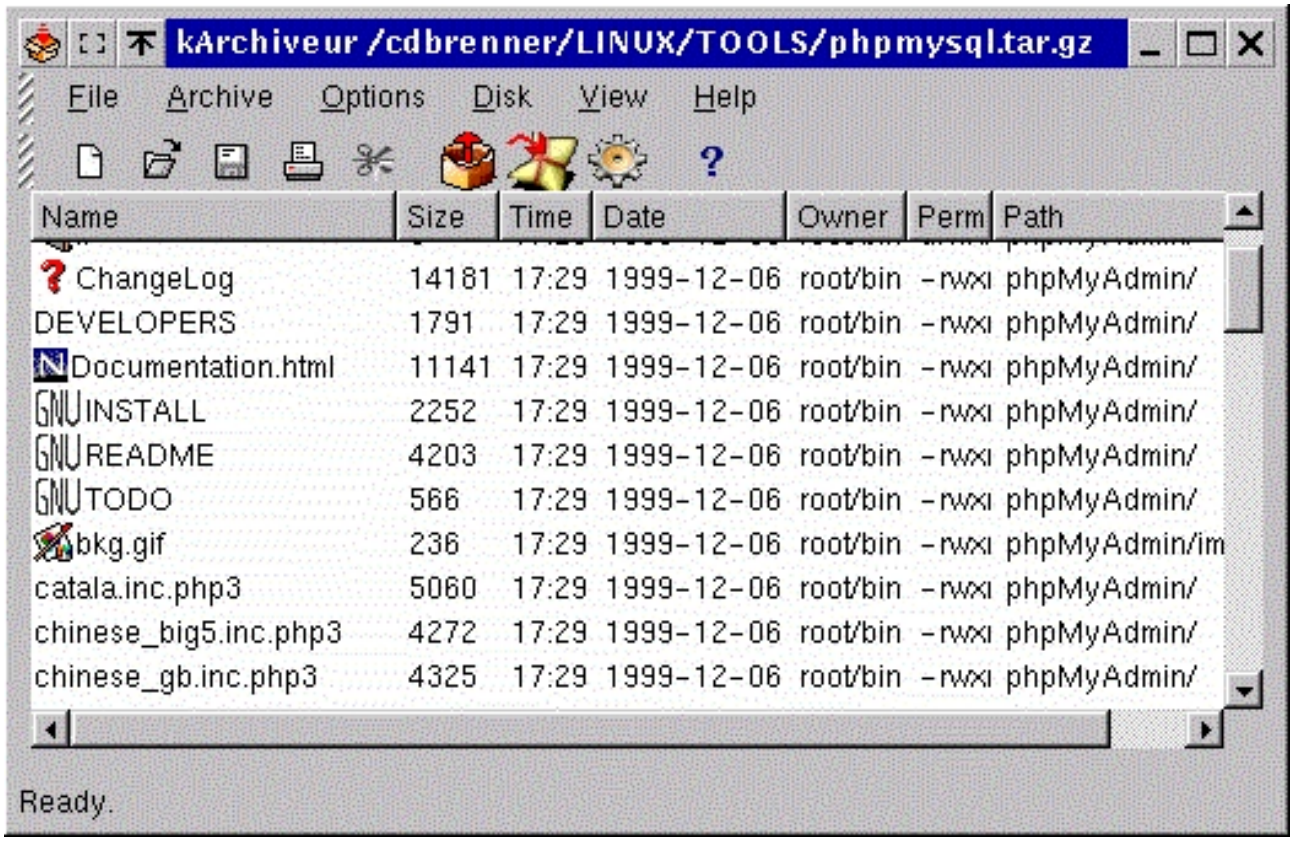


Abbildung 9.3: Das Archivprogramm `kArchiveur`

9.2 Pakete kompilieren

Was aber nun tun, wenn sich in einem Archiv nur der Quellcode des Programmes befindet? Der erste Anlaufpunkt sind die (hoffentlich vorhandenen) Hilfedateien, meist `readme` und `install` – hier finden Sie Hinweise, die zum erfolgreichen kompilieren nötig sind und mögliche

Fehlerquellen aufzeigen. Im klassischen Fall müßte nun das `Makefile` an das eigene System angepaßt werden, die Pfade zu den Bibliotheken und Include-Dateien ... Jedoch werden immer mehr Quellcode-Pakete mit einem Shell-Script mitgeliefert, welches viele Anpassungen für Sie vornimmt. Diese von `autoconf` erzeugten Skripte werden im allgemeinen mit

```
./configure
```

aufgerufen. Die Angabe von "`./`" ist wichtig, damit auch wirklich das Shellskript im aktuellen Verzeichnis aufgerufen wird. Geprüft werden die Voraussetzungen für die erfolgreiche Kompilierung des Paketes und die Pfade angepaßt. Sollten Komponenten fehlen, z.B. eine Include-Datei, wird das in einer Fehlermeldung ausgegeben. Dazu später mehr.

Wird das Konfigurations-Skript erfolgreich durchlaufen, kann durch die Angabe des Befehls `make` in dem Verzeichnis, wo sich die Datei `Makefile` befindet, die Kompilierung begonnen werden. Danach kann häufig mit `make install` oder einem speziellen Installations-Skript die Installation abgeschlossen werden, indem die Dateien in die richtigen Verzeichnisse verschoben werden. Ansonsten bleibt Ihnen nichts weiter übrig, als die Dateien per Hand in die richtigen Verzeichnisse zu kopieren und gegebenenfalls die Dateiberechtigungen zu verändern.

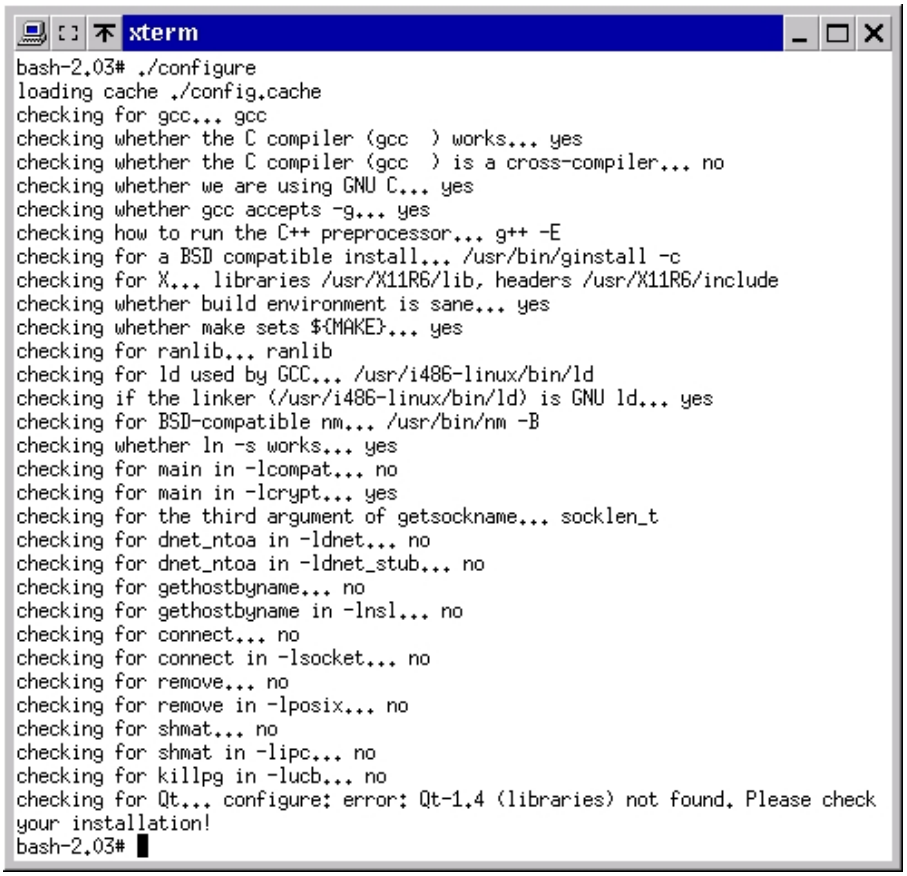


Abbildung 9.4: Konfigurations eines Makefiles per `./configure`

Doch alle Theorie kann in der Praxis versagen. Häufige Fehlermeldungen sind fehlende Bibliotheken oder falsche Versionen derselben. Beliebte Probleme gibt es neben den KDE-Bibliotheken mit den `libc`-Versionen. Sollte Ihr System mit dem `egcs`-Compiler arbeiten (Standard bei SuSE), kommt es zur Meldung "`configure: error: installation or configuration problem: C++ compiler cannot create executables`", wenn das Paket `gpp` nicht installiert ist.

In Abbildung 9.4 ist ein `configure`-Lauf dargestellt, der mit der Fehlermeldung abbricht, daß eine benötigte Bibliothek (hier `Qt-1.4`) nicht vorhanden ist. Da hilft nur die Installation der fehlenden Bestandteile. Sollte es mit der `libc6` bzw. dem Äquivalent `glibc2` Probleme geben, ist die Glibc2-HOWTO zu empfehlen.

Nach der Installation der fehlenden Bestandteile, dem `./configure`-Aufruf und dem anschließenden `make` wollen Sie nun das soeben erzeugte Programm starten. Nichts war mit der Freude, wenn sie die folgende beispielhafte Fehlermeldung beim Starten des Programmes `xfoo` erhalten

```
xfoo: can't load library /lib/libdl.so.1
```

Bei der Kontrolle des Verzeichnisses `/lib` fällt Ihnen auf, daß sie die Bibliothek `libdl.so.1.9` besitzten, jedoch nicht `libdl.so.1`. Nun müssen Sie sich aber nicht diese Bibliothek besorgen, sondern können mit "`ln -s libdl.so.1 libdl.so.1.9`" einen symbolischen Link auf die Version 1.9 setzen, da diese abwärtskompatibel zur Version 1.0 sein sollte. Ein praktisches Beispiel bietet die SuSE 6.3-Distribution. Hier funktioniert das Acrobat-PlugIn in Netscape nicht, da die Bibliothek `libc.so.5` fehlt. Ein symbolischer Link auf die Version 6 bringt hier Abhilfe.

9.3 Kernel kompilieren

Linux währe nicht das, was es ist, wenn es nicht den Quellcode der Kernels mitliefern würde. Im Grunde gibt es zwei Gründe, den Kernel neu zu kompilieren:

1. der Kernel soll optimal an die Hard- und Software-Anforderungen angepaßt werden, z.B. möchten Sie nur die vorhandene SCSI- und Netzwerkkarte in den Kernel integrieren;
2. die Kernelversion soll aktualisiert werden, entweder durch den kompletten Quellcode des neuen Kernels oder über einen Kernel-Patch;

Die Linux-Kernelquellen werden im Verzeichnis `/usr/src` installiert. Hier finden Sie normalerweise ein Verzeichnis `linux` und z.B. `linux-2.2.13`. Dabei ist das Verzeichnis `linux` ein Link auf `linux-2.2.13`. Bei SuSE ist zu beachten, daß die Kernel-Quellen standardmäßig nicht mit installiert werden, sondern explizit ausgewählt werden müssen (ansonsten werden nur die Include-Dateien installiert, die für die Kompilierung anderer Programmpakete benötigt werden).

Das Ziel einer Neukompilierung ist die Erstellung eines möglichst kompakten Kernels, der exakt auf Ihre Hardware zugeschnitten ist. Was möchten Sie beispielsweise mit einer SCSI-Unterstützung im Kernel, wenn Sie kein SCSI-System besitzten? Hierfür brauchen Sie aber keine Programmierkenntnisse, sondern müssen nur Fragen zu Ihrer Hardware beantworten und die benötigten Software-Schnittstellen kennen. Ein Risiko in der Systemsicherheit stellt die Neukompilierung des Kernels auch nicht dar. Durch einen kompakteren Kernel, der auf das jeweilige System zugeschnitten ist, steigt die Systemperformace und die Systemresourcen werden geschont. Informationen und die Kernel-Dokumentation finden Sie in `/usr/src/linux/README` und im Verzeichnis `/usr/src/linux/Documentation`.

9.3.1 Module

Während der Konfiguration des Kernels stehen Sie häufiger vor der Frage, den betreffenden Baustein direkt in den Kernel einzufügen oder als Modul verfügbar zu halten. Module sind

sozusagen Kernel-Treiber, die dynamisch zur Laufzeit hinzugeladen oder entladen werden können. Dadurch können nicht so häufig gebrauchte Hard- und Software-Schnittstellen dann geladen werden, wenn sie gebraucht werden und belasten ansonsten nicht die Systemresourcen. Hardware-Hersteller können dann auch Module zur Unterstützung Ihrer Hardware zur Verfügung stellen, ohne den Quellcode freigeben zu müssen.

Diese Kernel-Module werden normalerweise vollautomatisch bei Bedarf durch den Kernel-Daemon `kerneld` hinzugeladen und Abhängigkeiten zwischen den Modulen aufgelöst (Modul x benötigt Modul y, was dann automatisch hinzugeladen wird). `kerneld` liest hierfür die Modulabhängigkeitsdatei (siehe unten) und die optionale Steuerungsdatei `/etc/modules.conf`. `/etc/modules.conf` kann unter anderem dafür verwendet werden, Optionen an das Modul anzugeben. Z.B. wenn zwei Drucker-Ports vorhanden sind, muß dieses in dieser Datei bekannt gegeben werden, in dem die zweite E/A-Adresse angegeben wird. Natürlich besteht auch die Möglichkeit, manuell in die Modulverwaltung einzugreifen.

Der Befehl

```
insmod modulname [option=wert ]
```

integriert das angegebene Modul in den Kernel. Dabei wird der Modulname ohne Pfadangabe und ohne Suffix angegeben. `insmod` sucht nach dem Modul im Verzeichnis `/lib/modules/<kernel-version>/` und kann dem Modul diverse Optionen übergeben. Der Aufruf von

```
modprobe modulname [option=wert ]
```

funktioniert ähnlich wie `insmod`, jedoch werden die Abhängigkeiten des Moduls getestet. Voraussetzung hierfür ist, daß vorher mittels

```
depmod -a
```

die Abhängigkeitsdatei `/lib/modules/<kernel-version>/modules.dep` erstellt wurde. Sollte diese Abhängigkeitsdatei beim Starten des Linux-System fehlen, wird sie automatisch angelegt. Kernel-Module werden wieder entladen mit

```
rmmod modulname
```

und eine Liste aller geladenen Module mit den Abhängigkeiten kann mit

```
lsmod
```

angezeigt werden.

9.3.2 Kernel konfigurieren

Der Kernel wird über die Datei `.config` im Kernel-Basisverzeichnis konfiguriert. Jedoch wird diese Datei nicht manuell konfiguriert, sondern mit einem Konfigurationstool. Hierfür stehen drei verschiedene zur Verfügung, die folgendermaßen aufgerufen werden:

```
root@ServerI # cd /usr/src/linux
root@ServerI # make config          oder
root@ServerI # make menuconfig      oder
root@ServerI # make xconfig
```

Das Tool `make config` weist am wenigsten Komfort auf, hier muß eine endlose Liste von Fragen der Reihe nach beantwortet werden. Wesentlich komfortabler ist `make menuconfig`, ein im Text-Modus grafisches Konfigurationstool (die `ncurses`-Bibliothek muß installiert sein). Ein unter dem X-Server laufendes Konfigurationsprogramm ist `make xconfig` (`Tcl/Tk` muß installiert sein). In den Abbildungen 9.5 und 9.6 sind die beiden grafischen Konfigurationstools dargestellt. Sie bieten zusätzlich eine Hilfe zu den einzelnen Punkten an, bei SuSE sogar teilweise auf Deutsch, wenn die Kernel-Konfigurationssprache auf Deutsch eingestellt wird und anschließend das Tool neu gestartet wird. Weitere Informationen zu den einzelnen Optionen liefert das Distributions-Handbuch (SuSE z.B. hat in seinem Handbuch viele Optionen beschrieben).

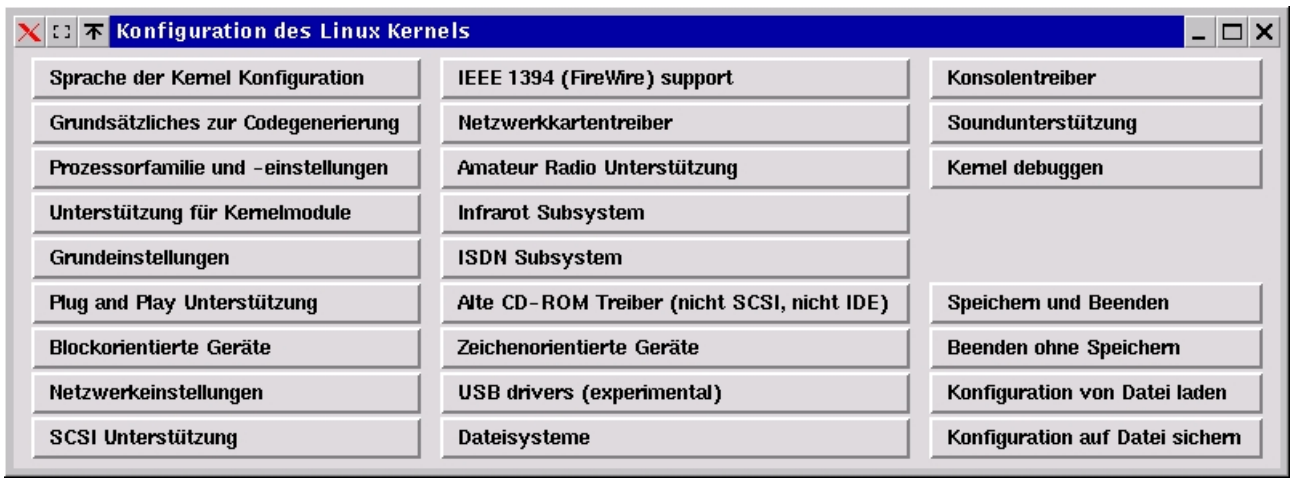


Abbildung 9.5: Kernel-Konfiguration über `make menuconfig`

9.3.3 Kernel kompilieren und installieren

Nachdem die Konfiguration abgeschlossen ist, kann nun der Kernel kompiliert werden. Sollten Sie das erste mal einen Kernel auf Ihrem System kompilieren, müssen zuerst die Abhängigkeiten überprüft werden. Dies geschieht über (im Kernelquellcode-Basisverzeichnis):

```
root@ServerI # make dep
```

Anschließend können die alten Objekt-Dateien gelöscht werden (nicht immer notwendig), der Kernel kompiliert, die Module kompiliert und installiert werden. Dies wird erreicht über:

```
root@ServerI # make clean             # alte Objekt-
Dateien löschen
root@ServerI # make make bzImage      # Kernel kompilieren
root@ServerI # make modules           # Module kompilieren
root@ServerI # make modules_install   # Module installieren
```

Sie können auch alle Schritte in eine Befehlszeile schreiben:

```
root@ServerI # make dep clean bzImage modules modules_install
```

Vor allem bei der ersten Kompilierung, nach einem `make clean` und je nach Umfang der Konfiguration ist Ihr Rechner einige Zeit beschäftigt. Sie können in der Zwischenzeit, da Linux ja ein Multitasking-System ist, auf einer anderen Konsole weiterarbeiten.

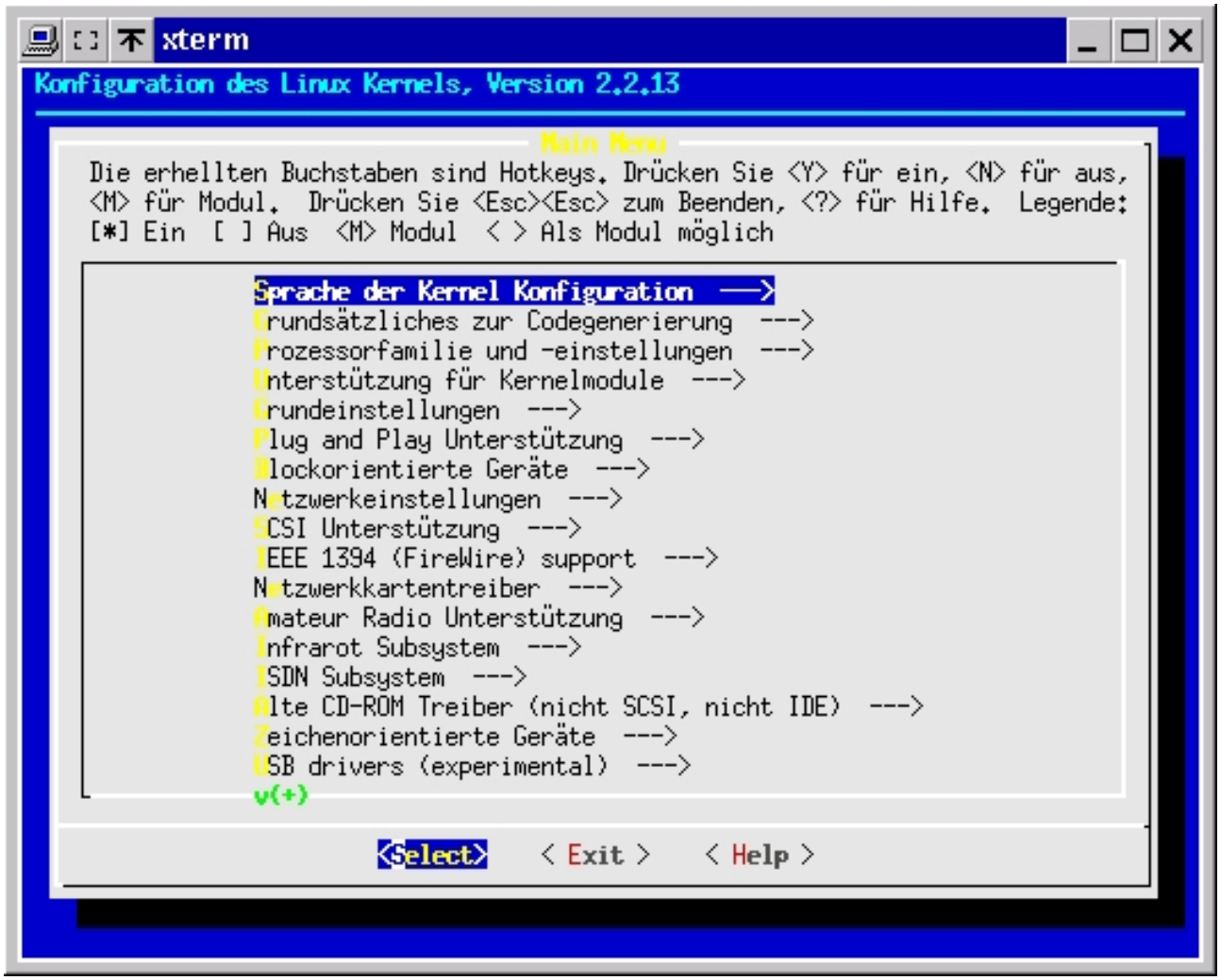


Abbildung 9.6: Kernel-Konfiguration über `make xconfig`

Den neuen Kernel `bzImage` finden Sie im Verzeichnis `/usr/src/linux/arch/i386/boot/`, der nun noch installiert werden muß. Zur Sicherheit sollte zuvor der alte Kernel und die Datei `System.map` gesichert werden. Z.B. mit

```
root@ServerI # mv /boot/vmlinuz /boot/vmlinuz.bak
root@ServerI # mv /boot/System.map /boot/System.map.bak
```

Anschließend kann der Kernel und die Datei `System.map` ins Verzeichnis `/boot` kopiert werden:

```
root@ServerI # cp ./System.map /boot
root@ServerI # cp ./arch/i386/boot/bzImage /boot/vmlinuz
```

Nun sollten Sie noch `LILO` so konfigurieren, daß beide Kernel gestartet werden können (falls der neu kompilierte Kernel nicht lauffähig sein sollte), `LILO` neu installieren (siehe Kapitel 6.4.2 auf Seite 33) und das System neu starten. Sollten Sie `loadlin` verwenden, muß der Kernel in ein von DOS lesbare Partition oder Diskette kopiert werden und die `loadlin`-Konfigurationsdateien entsprechend modifiziert werden.

Teil III

Shells

Kapitel 10

Allgemeines zu den Shells

Die UNIX-Shell ist der Kommandointerpreter auf einem Unix- und Linux-System, die Möglichkeiten gehen aber weit darüber hinaus, was Sie vielleicht vom Kommandointerpreter eines MS-DOS Systems (COMMAND.COM) her kennen. Er liest die vom Benutzer eingegebenen Kommandos und bringt diese nach einigen Aufbereitungen zur Ausführung. Desweiteren kennen Shells Konstrukte, die von höheren Programmiersprachen bekannt sind wie Schleifen, Verzweigungen und Funktionen. Mit diesen Konstrukten ist es möglich, Aufgaben effizient zu lösen, ohne direkt in einer Programmiersprache wie Perl oder C zu programmieren. Das macht die Shell zu einem mächtigen und unverzichtbarem Werkzeug für System-Administratoren, Programmierer und allen, die routinemäßige Abläufe automatisch steuern möchten.

Die Shell hat ihren Namen daher, weil sie sich wie eine Muschel (engl. Shell) um den Betriebssystemkern legt. Direkte Kommunikation mit dem Betriebssystemkern wäre auch viel zu komplex und benutzerunfreundlich. Dabei nimmt die Shell die Kommandos vom Benutzer entgegen, interpretiert diese und setzt sie in Systemaufrufe um. Die Rückmeldungen des Systems werden wieder von der Shell entgegengenommen, aufbereitet und an den Benutzer ausgegeben. UNIX-Shells sind kein Bestandteil des Betriebssystems, sondern eigenständige Programme. Diese Tatsache hat einige Vorteile - Shells können wie andere Anwendungsprogramme aufgerufen und auch ausgetauscht werden. Daraus folgt, daß es nicht *die* UNIX-Shell gibt, sondern viele verschiedene mit unterschiedlicher Leistungsfähigkeit und Syntax. Von einigen UNIX-Linien wurden eigene Shell-Varianten entwickelt.

10.1 Shell-Varianten

Nachfolgend werden die einzelnen Shells kurz beschrieben, wobei diese Liste nicht vollständig ist.

Bourne-Shell (`sh`)

- Die Bourne-Shell ist die erste UNIX-Shell überhaupt. Ihr Erfinder ist Steve Bourne, daher auch der Name und bildet die Grundmenge aller Shells.

Korn-Shell (`ksh`)

- Eine Weiterentwicklung der Bourne-Shell, Erfinder ist David Korn. Sie bietet Verbesserungen gegenüber der Bourne-Shell und eine History-Funktion.

Bourne-Again-Shell (`bash`)

- Die BASH ist der Nachfolger der Bourne-Shell und die Standardshell unter Linux-Systemen. Eine History-Funktion ist implementiert. Derzeit gibt es zwei Versionen der BASH (Version 1.x und 2.x), wobei die Version 2.x sehr viel restriktiver in der Syntax ist.

C-Shell (`csh`)

- Sie wurde auf der BSD-UNIX[1] Linie entwickelt und basiert auf der Bourne-Shell. Ihre Syntax ist stark an der Programmiersprache C angelehnt (daher auch der Name) und ist deswegen bei Programmierern sehr beliebt. Zusätzlich zu den Funktionen der Bourne-Shell bietet sie Job-Kontrolle und eine History-Funktion

TC-Shell (`tcsh`)

- Die TC-Shell ist der Nachfolger von der C-Shell mit Erweiterungen und Neuigkeiten.

PDK-Shell (`pdksh`)

- Sie ist ein Korn-Shell Clone

Almquist-Shell (`ash`)

- Eine Bourne-kompatible Shell aus der NetBSD-Linie. Da sie nur 62 kByte groß ist, eignet sie sich gerade für Rechner mit wenig Arbeitspeicher sowie Installation- und Rettungs-Disks, dafür bietet sie keine Job-Kontrolle[2].

Z-Shell (`zsh`)

- Diese Shell ist ein Zusammenschluß von mehreren Shells, sie besitzt die meisten Funktionen der `ksh`, Features der `csh` und einige Optionen der `tcsh`.

[1] BSD = Berkley System Distribution
[2] Z.B. benutzt die Slackware-Distribution diese Shell für ihr Setup.

Kapitel 11

Aufruf der Shells und Shell-Skripten

Die Bourne-Again-Shell, kurz `bash`, ist die Standard-Shell auf Linux-System und wird deswegen in diesem Kapitel näher besprochen. Viele dieser Funktionen sind auch übertragbar auf andere Shells, manchmal mit leicht abgeänderter Syntax. Bei der Einrichtung eines neuen Benutzers wird die Login-Shell mit angegeben und in der Datei `/etc/passwd` abgespeichert. Möchten Sie die Login-Shell eines Benutzers ändern, können Sie dies entweder direkt in dieser Datei eintragen oder über das Distributions-Tool (z.B. YaST bei SuSE-Systemen) modifizieren. Im laufenden Betrieb einer Shell kann durch Starten einer anderen Shell die Shell gewechselt werden. Wie schon erwähnt, sind die Shells eigenständige Programme und können wie jedes andere Programm gestartet werden, mit Options- und Parameterübergabe. Z.B. wechselt der Aufruf von `zsh` in die Z-Shell und der Aufruf von `exit` beendet diese Shell wieder.

Shell-Skripte sind Text-Dateien, die eine Folge von Shell-Befehlen enthalten (ähnlich den Batch-Dateien unter MS-DOS). Hierfür gibt es zwei Aufrufvarianten. Entweder über

```
<Shell-Name> [<Optionen>] <Skript-Name und Pfad> [<Parameter>]
```

wobei das Skript mit der angegebenen Shell gestartet wird. Hier können auch spezielle Shell-Optionen mit angegeben werden und die Parameter werden dem Skript übergeben. Die etwas gebräuchlichere Aufrufsyntax lautet aber:

```
<Skript-Name und Pfad> [<Parameter>]
```

Hierbei muß aber die Skript-Datei als ausführbar markiert werden. Dem Skript werden die angegebenen Parameter übergeben und in einer Kind-Shell der aktuellen Shell gestartet. Enthält das Shell-Skript Befehle und Syntax einer speziellen Shell, kann über Angabe in der ersten Zeile des Skripts die zu benutzende Shell festgelegt werden. Die Form sieht folgendermaßen aus:

```
#!/bin/ksh
```

um z.B. das Skript in einer Korn-Shell ablaufen zu lassen. Somit stehen Ihnen die Möglichkeiten offen, für jedes Skript die geeignete Shell auszuwählen und den effizientesten Code zu programmieren. Die grundlegende Bedienung der Kommandozeile finden Sie in Kapitel 7, Seite 35 und die Konfigurations-Dateien in Kapitel 15, Seite 78.

Kapitel 12

Begriffe der Shell

Bevor wir mit dem Arbeiten in einer Shell beginnen, müssen zuerst einige Begriffe geklärt werden (entnommen aus [Herold, Seite 9 - 10])

Kommandozeile (*command line*)

ist eine Eingabezeile, die von der Shell zu interpretieren ist.

Metazeichen (*metacharacter*)

sind Zeichen (keine Ziffern oder Buchstaben), denen von der entsprechenden Shell eine Sonderbedeutung zugeordnet ist.

Trennzeichen (*blank*)

sind Leer- oder Tabulatorzeichen.

Bezeichner (*identifier*)

ist eine Folge von Buchstaben (keine Umlaute oder ß), Ziffern oder Unterstrichen, wobei diese Folge mit einem Buchstaben oder einem Unterstrich beginnen muß. Bezeichner werden z.B. für Namen von Shell-Variablen oder Shell-Funktionen verwendet. Bezeichner wie

```
vier_*_hotel
2A
7_und_40_elf
kündigung
```

sind also nicht erlaubt.

Wort (*word*)

ist eine Folge von Zeichen, die durch ein oder mehrere der folgenden Metazeichen

```
; & ( ) | < > Neuezeile-Zeichen Leerzeichen Tabulatorzeichen
```

abgegrenzt ist; mit \ ausgeschaltete Metazeichen (wie z.B. `\;`) sind allerdings Teil eines Wortes.

Kapitel 13

Die bash

13.1 Jokerzeichen

Einige Jokerzeichen werden Sie schon von MS-DOS oder anderen Betriebssystemen her kennen, andere werden neu für Sie sein. Die wohl bekanntesten Jokerzeichen sind `*` und `?`, welche unter Linux null oder beliebig viele Zeichen bzw. genau ein Zeichen abdecken. Dabei wertet die Shell zuerst die Jokerzeichen in einer Kommandozeile aus und expandiert diese, bevor sie an das angegebene Kommando übergeben werden. Z.B. wird mit dem Aufruf von

```
ls -l a*
```

zuerst `a*` expandiert und anschließend die komplette Liste an den Befehl `ls` übergeben. Der eigentliche Aufruf von `ls` könnte dann so aussehen:

```
ls -l amplitude.gif anhang1.tex axinom.tex
```

In den Shells gibt es zur verfeinerten Auswahl noch weitere Jokerzeichen (Übersicht Tabelle 13.1). Die Angabe von `[abc]` sucht nach genau eines der angegebenen Zeichen, Bereichsangaben können über `[a-f]` angegeben werden. Soll keines der Zeichen vorhanden sein, wird ein Ausrufezeichen vorangestellt `[!abc]`. Die Tilde ist ein Alias-Name für das Heimatverzeichnis des Benutzers, wenn sie als Pfadname angegeben wird, `~logname` bezeichnet das Heimatverzeichnis des Benutzers *logname*. Ansonsten kennzeichnet eine Tilde häufig Backup-Dateien. Ein Wort kann dabei auch mehrere Jokerzeichen enthalten, z.B.

```
ls -l [a-f]*.[tl][ex]x
```

Hierbei werden alle Dateien angezeigt, die mit `a-f` beginnen, dann beliebig viele Zeichen (auch null) enthalten, gefolgt von einem Punkt und dem Suffix `tex` oder `lyx` enden. Wenn Sie aber versuchen sollten, mehrere Dateien gleichzeitig umzubenennen wird Ihnen das nicht gelingen. Der Aufruf von

```
mv *.x *.y
```

um alle Dateien mit Suffix `.x` zum Suffix `.y` umzubenennen wird mit einer Fehlermeldung abbrechen. Sie können sich vielleicht schon denken, warum das so ist. Zuerst ersetzt die Shell `*.x` durch alle Dateien mit diesem Suffix, für `*.y` wird jedoch keine Datei gefunden und der

Parameter so an mv übergeben. mv kann aber nicht mehrere Dateien gleichzeitig umbenennen (nur verschieben) und bricht mit einer Fehlermeldung ab. Hierfür muß also eine andere Lösung gefunden werden (z.B. eine Schleife).

Ein anderes Problem entsteht z.B. beim Befehl ls, wenn durch das Pattern ein Verzeichnis mit abgedeckt wird. Hier wird nicht nur der Verzeichnisname ausgegeben, sondern auch der Inhalt des Verzeichnisses (was vielleicht gar nicht erwünscht ist). Gibt man ls die Option -d mit an, werden nur die Verzeichnisnamen und nicht deren Inhalt angezeigt.

Jokerzeichen	Bedeutung
?	genau ein Zeichen
*	null oder beliebig viele Zeichen
[abc]	genau eines der angegebenen Zeichen
[a-f]	genau ein Zeichen aus dem angegebenen Bereich
[!abc]	keines der angegebenen Zeichen
~	Alias für das Heimatverzeichnis

Tabelle 13.1: Übersicht der Jokerzeichen

13.2 Substitutionsmechanismen (Quoting)

In manchen Anwendungsfällen kann es nützlich sein, die Ausgabe eines Kommandos als Teil der Kommandozeile interpretieren zu lassen. Dazu bietet die Shell die sogenannten Substitutionsmechanismen an. Kommandos werden in Metazeichen ' ', " ", ` ` oder $() geklammert, die verschiedene Arten des Substitutionsmechanismus erlauben. Dabei muß zwischen Variablensubstitution und Kommandosubstitution unterschieden werden. Z.B. möchten Sie den String "Heute ist der " gefolgt von dem aktuellen Datum ausgeben. Die Kommandozeile

```
echo Heute ist der date '+%d.%m.%y (%a)'
```

führt nicht zu der gewünschten Ausgabe, da nicht date durch sein Ergebnis substituiert wird. Wird jedoch

```
echo Heute ist der `date '+%d.%m.%y (%a)'`
```

eingegeben, wird das in Gegen-Apostrophen geklammerte Kommando zuerst ausgeführt und das Ergebnis im echo-Kommando eingesetzt. Der daraus resultierende Befehl lautet:

```
echo Heute ist der 23.07.00 (Sun)
```

Wird vor dem Substitutionszeichen, ebenso bei allen anderen Metazeichen, ein Backslash angegeben, so hebt dieser die Sonderbedeutung auf. In Tabelle 13.2 sehen Sie die vier Substitutionsmetazeichen und welche Substitutionsmechanismen darin möglich sind.

13.3 Umleitungen und Pipes

Manchmal ist es notwendig, die Standardeingabe und -ausgabe umzulenken. Die üblichen Voreinstellungen sind dabei:

Zeichen	Variablensubstitution durchführen	Kommandosubstitution durchführen
’ ’	nein	nein
“ “	ja	nein
` `	ja	ja
$()	ja	ja

Tabelle 13.2: Übersicht der Substitutionsmetazeichen

Standardeingabe: Dialogstation (Tastatur)

Standardausgabe: Dialogstation (Bildschirm)

Standardfehlerausgabe: Dialogstation (Bildschirm)

Dabei wird zwischen Standardausgabe und -fehlerausgabe unterschieden, um echte Daten von Fehler- oder Diagnosemeldungen zu trennen. Um nun diese zu ändern, werden u.a. die Zeichen < und > benötigt (Übersicht über alle Sonderzeichen in Tabelle 13.3). Der Eingabestrom kann über `<datei` und der Ausgabestrom über `>datei` jeweils von / in eine Datei umgelenkt werden. Dabei können Umlenkungsanweisungen an beliebiger Stelle in einem einfachen Kommando angegeben werden. Die nachfolgenden Kommandozeilen sind identisch, aus Gründen der Lesbarkeit sollte aber die Form (1) oder (2) bevorzugt werden.

```
cat <ein >aus (1)
cat >aus <ein (2)
<ein cat >aus (3)
<ein >aus cat (4)
>aus cat <ein (5)
>aus <ein cat (6)
```

Werden zwei Kommandos über eine Pipe (|-Zeichen) verbunden, so leitet das Kommando 1 seine Ausgabe an Kommando 2 weiter. Die wohl bekannteste Form der Pipe ist die Ausgabe von Text, der mittels des Befehls `more` seitenweise angezeigt wird.

```
cat text | more
```

Hierbei wird zuerst die Datei text von `cat` ausgegeben und diese Standardausgabe an den Befehl `more` als Standardeingabe weitergereicht, der dann die seitenweise Ausgabe der Datei `text` vornimmt. Manchmal macht der Unterschied zwischen einer Pipe und der Kommandosubstitution anfänglich Schwierigkeiten. Das nachfolgende Beispiel (aus [Herold, Seite 28f]) soll diesen Unterschied helfen zu verdeutlichen. Eine Datei `zaehle.txt` soll Namen von Dateien enthalten, zu denen die darin enthaltene Wortzahl zu bestimmen ist:

```
root@ServerI # cat zaehle.txt
/etc/magic
/etc/inittab
/usr/include/stdio.h
root@ServerI #
```

Der Aufruf von

```
root@ServerI # cat zaehle.txt | wc -w
3
```

liefert dann die falsche Ausgabe, da in diesem Fall nicht der Inhalt der in `zaehle.txt` genannten Dateien, sondern der Inahlt von `zaehle.txt` selbst ausgewertet wird. Mit dem Aufruf

```
root@ServerI # wc -w `cat zaehle.txt`
```

dagegen wird - wie gefordert - der Inhalt der in `zaehle.txt` angegebenen Dateien und nicht der Inhalt von `zaehle.txt` selbst ausgewertet. Nach der Durchführung der Kommandosubstitution `` `cat zaehle.txt` `` würde folgende Kommandozeile aus der obigen resultieren:

```
wc -w /etc/magic /etc/inittab /usr/include/stdio.h
```

und z.B. folgende Ausgabe liefern:

```
 572 /etc/magic
 112 /etc/inittab
 365 /usr/include/stdio.h
1049 total
```

Gelegentlich möchten Sie vielleicht auch die Ausgabe eines Programmes am Bildschirm betrachten und gleichzeitig in eine Datei schreiben. Für dieses Problem steht Ihnen das Programm `tee` zur Verfügung, welches den Ausgabestrom verdoppelt und die parallele Ausgabe auf zwei Ausgabeströme ermöglicht. Dabei wird dem Programm `tee` die Ausgabe eines Kommandos über eine Pipe als Eingabe übergeben.

```
<Kommando> | tee <Dateiname>
```

Die nachfolgende Kommandozeile soll den Gebrauch von `tee` demonstrieren. In diesem Beispiel wird mittels `ls -l` der Inhalt des aktuellen Verzeichnisses einmal in die Datei inhalt1 gespeichert, der zweite Ausgabestrom wird über eine weitere Pipe an das Kommando `sort` weitergegeben, dort nach der Dateigröße (fünfte Spalte, also Option +4) sortiert und in `inhalt2` gespeichert.

```
ls -l | tee inhalt1 | sort +4 > inhalt2
```

13.4 Kommandoausführung

Normalerweise läuft eine Kommandoausführung so ab, wobei Sie das Kommando eingeben, dann von der Shell interpretiert und ausgeführt wird und anschließend die Ausgabe auf dem Bildschirm erfolgt und die Shell für ein neues Kommando bereitsteht. In den vorherigen Abschnitten haben Sie die Möglichkeiten kennengelernt, mehrere Kommandos gleichzeitig einzugeben und ausführen zu lassen. Für die weitere Steuerung der Kommandoausführung gibt es Sonderzeichen, womit Sie, ohne Umleitung der Ausgaben, mehrere Kommandos einer Kommandozeile eingeben können und diese sogar nur bedingt ausgeführt werden.

Syntax	Bedeutung
kom > datei	Standardausgabeumleitung in Datei
kom < datei	Standardeingabeumleitung von Datei
kom 2> datei	Standardfehlerumleitung in Datei
kom >& datei	Standardausgabe und -fehler in Datei umleiten
kom >> datei	hängt Standardausgabe an vorhandene Datei an
kom1 | kom2	leitet Ausgabe von kom1 an kom2 weiter
kom <> datei	Datei wird zum Lesen und Schreiben geöffnet
kom <&-	Standardeingabe schließen
kom >&-	Standardausgabe schließen
kom | tee datei	zeigt die Ausgaben an und speichert gleichzeitig eine Kopie in der Datei

Tabelle 13.3: Übersicht der Umleitungssonderzeichen

Werden zwei Kommandos mit einem Semikolon getrennt, wird zuerst das Kommando 1 ausgeführt und bei Beendigung das Kommando 2. Soll das Kommando 2 nur dann ausgeführt werden, wenn Kommando 1 erfolgreich ablief, können Sie die Kommandos mit && verknüpfen. Für das Gegenteil gibt es die Verknüpfung || (doppeltes Pipe-Symbol), also nur wenn Kommando 1 fehlerhaft ablief, soll Kommando 2 gestartet werden.

Da Linux ein Multitasking-System ist, können Kommandos auch im Hintergrund gestartet werden, so daß die Shell nicht auf die Beendigung des Kommandos wartet sondern gleich wieder zur Eingabe bereit steht. Sie sollten aber darauf achten, daß Ein- und Ausgaben trotzdem auf dem Bildschirm erscheinen und somit möglicherweise zu einer Vermischung der Ein- und Ausgaben des Vorder- und Hintergrundprozesses kommen kann. Hier sollten Sie die Möglichkeiten der Umleitung benutzen. Ein Programm kann man über Angabe des Kaufmanns-UND am Ende des Kommandos im Hintergrund starten:

```
kommando &
```

Hierbei wird dann noch die PID-Nummer ausgegeben, die für verschiedene Programme, z.B. `kill`, benötigt wird. Sollten Sie vergessen haben, das Kommando als Hintergrundprozeß zu starten, so können Sie jederzeit das Hintergrundprogramm über die Tastenkombination `STRG+Z` stoppen und durch Eingabe des Befehls `bg` im Hintergrund fortführen. Also brauchen Sie nicht das Kommando mittels STRG-C brutal zu stoppen und neu zu starten. Werden Kommandos in runden Klammern ( ) gefaßt, führt die Shell die Kommandos in derselben Shell aus und liefen ein gemeinsammes Ergebnis (ansonsten wird für jedes Kommando eine neue Sub-Shell gestartet). Z.B. durch

```
root@ServerI # (ls ; date) > inhalt
```

wird in die Datei `inhalt` eine Dateiliste sowie das aktuelle Datum geschrieben. In Tabelle 13.4 finden Sie eine Übersicht zu den Sonderzeichen der Kommandoausführung.

13.5 Zeichenkettenbildung und Berechnungen

13.5.1 Zeichenkettenbildung

Sie standen vielleicht schon mal vor dem Problem, viele ähnliche Verzeichnisse zu erstellen, z.B. `teil1`, `teil2`, `teil3` ... Die `bash` kann Ihnen hierfür den Tippaufwand abnehmen, da

Syntax	Bedeutung
kom1 ; kom2	führt die Kommandos nacheinander aus
kom1 && kom2	führt kom2 nur aus, wenn kom1 erfolgreich war
kom1 || kom2	führt kom2 nur aus, wenn kom1 Fehler liefert
kommando &	startet das Kommando im Hintergrund
kom1 & kom2	startet kom1 im Hintergrund, kom2 im Vordergrund
(kom1 ; kom2)	führt beide Kommandos in der gleichen Shell aus

Tabelle 13.4: Übersicht der Kommandoausführungssonderzeichen

sie aus Zeichenketten, die in geschweiften Klammern angegeben sind, alle denkbaren Zeichenkettenkombinationen zusammenstellt. Die einzelnen Zeichenketten werden durch Kommatas getrennt. Der Vorteil gegenüber Jokerzeichen besteht darin, daß auch nicht existierende Dateinamen gebildet werden können, z.B. für mkdir.

```
root@ServerI # echo {a,b}{1,2,3}
a1 a2 a3 b1 b2 b3
```

13.5.2 Berechnung arithmetischer Ausdrücke

Die bash kann auch ganzahlige arithmetische Ausdrücke mit 32-Bit-Integerzahlen (Zahlenbereich zwischen +/- 2147483648) berechnen. Hierfür müssen Sie den Ausdruck in eckige Klammern stellen und ein $-Zeichen voranstellen.

```
root@ServerI # echo $[100+45]
145
```

Als Operatoren sind die meisten aus der Programmiersprache C bekannten erlaubt: + - * / für die vier Grundrechenarten, % für Modulo-Berechnungen, == != < <= > >= für Vergleiche, << und >> für Bitverschiebungen, ! && || für logisches NICHT, UND und ODER etc. Wenn einzelne Werte aus Variablen entnommen werden sollen, muß ein $-Zeichen vorangestellt werden.

Alternativ und bei Shells, die keine Berechnungen durchführen können, kann das Komando expr verwendet werden.

```
root@ServerI # expr 100 + 45
145
```

13.6 Shell-Variablen

Viele Einstellungen der bash und anderer Anwendungsprogramme werden über Shell-Variablen gesteuert. Bei den Variablen muß unterschieden werden zwischen globalen, lokalen und dynamisch-veränderlichen. Beim Systemstart werden schon viele Variablen gesetzt, die Sie entweder über Konfigurationsdateien oder im laufenden Betrieb verändern können (z.B. über die Datei /etc-/profile, Siehe Kapitel 15, Seite 78). Andere verändern sich im laufenden Betrieb dynamisch, so z.B. die Variable PWD, die immer den aktuellen Pfad enthält. Die Zuweisung einer Variablen erfolgt durch den Zuweisungsoperator =:

```
var = <wert>
```

Soll der Inhalt von der Variablen Leerzeichen oder andere Sonderzeichen enthalten, muß die Zeichenkette in einfache oder doppelte Hochkommata gestellt werden. Bei der Zuweisung können mehrere Zeichenketten unmittelbar aneinandergereiht werden und die Kommandosubstitution vorgenommen werden. Beachten Sie, daß alle durch einfache Zuweisung entstandenen Variablen als lokale angelegt werden. Beim Beenden der Shell (oder der Sub-Shell) gehen diese verloren. Soll eine lokale Variable als globale makiert werden, muß `export <var>` oder `declare -x <var>` aufgerufen werden. Die Tabelle 13.5 gibt die verschiedenen Kommandos zur Variablenverwaltung wieder, dabei gibt es mehrere funktionelle Überlappungen.

Kommando	Bedeutung
`var = <wert>`	Kurzschreibweise für `let`, `var` ist lokal
`declare var = <wert>`	weist der lokalen Variablen `var` einen Wert zu (wie `let`)
`declare -x var = <wert>`	weist der globalen Variablen `var` einen Wert zu (wie `export`)
`export`	zeigt alle globalen Variablen an
`export var`	macht `var` zu einer globalen Variablen
`export var = <wert>`	weist der globalen Variablen `var` einen Wert zu
`let var = <wert>`	weist der lokalen Variablen `var` einen Wert zu
`local var = <wert>`	definiert `var` als lokal (nur in Shell-Funktionen)
`printenv`	zeigt wie `export` alle globalen Variablen an
`set`	zeigt alle (lokale und globale) Variablen an
`unset var`	löscht die Variable var

Tabelle 13.5: Kommandos zur Variablenverwaltung

Zu beachten ist noch, daß Variablenzuweisungen, auch wenn sie global sind, nur für eine Shell gelten. Arbeiten Sie auf mehreren Terminals, so laufen dort voneinander unabhänginge Shells. Oft benötigte Variablen können Sie in den Konfigurationsdateien (siehe Kapitel 15) eintragen und somit bei jedem Systemstart verfügbar halten.

Sie sollten darauf achten, daß Sie nicht vorhandene System-Variablen überschreiben, die für den Betrieb der Shell und anderer Anwendungsprogramme notwendig sind. Die Tabelle 13.6 beschreibt die wichtigsten System-Variablen.

Name	Bedeutung
`HOME`	Pfadname des Heimatverzeichnis des aktuellen Benutzers
`PATH`	Suchpfad für ausführbare Programme
`BASH`	enthält den Dateinamen der bash
`LOGNAME`	enthält den Login-Namen (User-Namen)
`HOSTNAME`	enthält den Rechner-Namen
`MANPATH`	enthält den Pfad zu allen Verzeichnissen, die man-Seiten enthalten
`INFOPATH`	enthält den Pfad zu allen Verzeichnissen, die info-Seiten enthalten
`OLDPWD`	enthält den Pfad des zuletzt aktiven Verzeichnisses
`PWD`	enthält den Pfad des aktuellen Verzeichnisses
`PS1`	enthält die Zeichenkette des Prompts
`PS2`	enthält die Zeichenkette des Sekundär-Prompts (bei mehrzeiligen Eingaben)

Tabelle 13.6: Die wichtigsten System-Variablen

Desweiteren gibt es noch einige Variblen, die durch die Shell vordefiniert sind und sich dynamisch ändern. Diese werden vor allem in Shell-Skripten verwendet. Die Tabelle 13.7 zeigt die

Bedeutung dieser Variablen.

Name	Bedeutung
$?	Rückgabewert des letzten Kommandos
$!	PID des zuletzt gestarteten Hintergrund-Prozesses
$$	PID der aktuellen Shell
$0	Dateiname des gerade ausgeführten Shell-Skripts (oder des symbolischen Links)
$#	Anzahl der dem Shell-Programm übergebenen Parameter
$1 bis $9	Parameter 1 bis 9, die dem Shell-Skript übergeben wurden
$*	Alle Positionsparameter des Skripts als ein String
$@	Alle Positionsparameter des Skripts als einzelne Strings

Tabelle 13.7: vordefinierte dynamisch-änderliche Shell-Variablen

Die Shell unterscheidet bei Variablen zwischen undefinierten Variablen und Variablen, denen explizit der "Nullwert" (leere Zeichenkette) zugewiesen wurde. Dieses Vorgehen ist gerade bei der Fehlersuche in Skripten wichtig und kann über verschiedene Methoden festgestellt werden (z.B. über die Parametersubstitution, s.u.).

Mit dem bash-Kommando read können Sie während des Ablaufs eines Shell-Skripts Eingaben vornehmen und in Variablen speichern. Wird die Option -n angegeben, erfolgt kein Wechsel in eine neue Zeile, bevor die Werte eingegeben werden können. Die Aufruf-Syntax lautet:

```
read [Optionen] <var>
```

13.7 Parametersubstitution

Unter diesem Namen stellt die bash einige Kommandos zur Verfügung, die die Bearbeitung von Zeichenketten in Variablen ermöglichen. Die Syntax hierfür lautet ${var__muster}, wobei __ für ein bis zwei Sonderzeichen steht, die die Bearbeitung näher spezifizieren. Zu beachten ist, daß die Variablen ohne vorangestelltes Dollarzeichen ($) angesprochen werden, außer wenn Vergleichsmuster aus einer Variablen gelesen werden sollen. In Tabelle 13.8 sind die Mechanismen einzelnd aufgeführt.

Syntax	Bedeutung
`${var:-default}`	Wenn die Variable leer ist, liefert die Konstruktion die Defaulteinstellung als Ergebnis, anderfalls den Inhalt der Variablen. Die Variable wird nicht verändert.
`${var:=default}`	Wie oben, es wird aber gleichzeitig der Inhalt der Variablen geändert, wenn diese bisher leer war.
`${var:+neu}`	Wenn die Variable leer ist, wird sie nicht verändert. Wenn die Variable dagegen bereitsbelegt ist, wird der bisherige Inhalt durch eine neue Einstellung ersetzt. Die Konstruktion liefert denneuen Inhalt der Variablen.
`${var:?fehlermeldung}`	Wenn die Variable leer ist, wird der Variablenname und die Fehlermeldung ausgegeben und dasShell-Programm anschließend beendet. Andernfalls liefert die Konstruktion den Inhalt der Variablen.
`${#var}`	Liefert die Anzahl der in der Variablen gespeicherten Zeichen als Ergebnis, die Variable wird nichtverändert.
`${var#muster}`	Vergleicht den Anfang der Variablen mit dem angegebenen Muster. Wenn das Muster erkannt wird, liefertdie Konstruktion den Inhalt der Variablen abzüglich des kürzestmöglichen Textes, der dem Suchmusterentspricht. Wird das Muster dagegen nicht gefunden, wird der ganze Inhalt der Variablen zurückgegeben.Im Suchmuster können die zur Bildung von Dateinamen bekannten Jokerzeichen verwendet werden. DieVariable wird in keinem Fall verändert.
`${var##muster}`	Wie oben, allerdings wird jetzt die größtmögliche Zeichenkette, die dem Muster entspricht, eliminiert.
`${var%muster}`	Wie `${var#muster}`, allerdings erfolgt der Mustervergleich jetzt am Ende des Variableninhalts. Es wirddie kürzestmögliche Zeichenkette vom Ende der Variablen eliminiert. Die Variable selbst bleibt unverändert
`${var%%muster}`	Wie oben, allerdings wird die größtmögliche Zeichenkette eliminiert.
`${!var1}`	Liefert den Inhalt der Variablen, dessen Name in `var1` als Zeichenkette enthalten ist.

Tabelle 13.8: Übersicht der Parametersubstitutions-Syntax

Kapitel 14

Programmiersprachenkonstrukte der bash

14.1 Auswertung von Ausdrücken

Bevor wir auf die einzelnen Programmiersprachenkonstrukte eingehen, muß noch die Auswertung von Ausdrücken angesprochen werden, da sie als Bedingungen für die Konstrukte notwendig sind. Hierfür gibt es zwei Kommandos, `test` um Bedingungen zu prüfen und `expr` um einfache arithmetische Berechnungen durchzuführen. Zu beachten ist, daß die Shells bzw. die Anwendungsprogramme bei einer erfolgreichen Ausführung, anders als in den Programmiersprachen, einen `exit`-Wert von 0 und bei einem Fehler einen `exit`-Wert unterschiedlich von 0 liefern. Das gleiche gilt für die Wahrheitswerte `true` (= 0) und `false` (<> 0).

14.1.1 Formulieren von Bedingungen mittels `test`

Das builtin-Kommando `test` formuliert eine Vielzahl von Bedinungen über Optionen, die in den Tabellen 14.1, 14.2, 14.3 und 14.4) aufgelistet werden[1]. Ist die Bedingung wahr, so liefert `test` einen `exit`-Wert von 0, ansonsten einen von 0 unterschiedlichen Wert. Sollten keine Argumente `test` übergeben werden, so liefert `test` immer einen Wert unterschiedlich von 0 (falsch). Für den Aufruf gibt es zwei Möglichkeiten:

```
test <ausdruck>
```

oder

```
[ ausdruck ]
```
[2]

Sollen Zeichenketten mit Leerzeichen verglichen werden, müssen diese in Hochkommata gestellt werden, damit der komplette String verglichen wird. Gerade bei Variablen kann dies eine Fehlerquelle sein.

[1] Die Online-Hilfe ist mit den Kommandos `man test` und `info test` verfügbar.
[2] Nach [und vor] muß mindestens ein Leer- oder Tabulatorzeichen angegeben werden.

Ausdruck	liefert wahr, wenn
`zk`	die Zeichenkette nicht leer ist
`-z zk`	die Zeichenkette leer ist [0 Zeichen] (***z**ero*)
`-n zk`	die Zeichenkette nicht leer ist (***n**ot **z**ero*)
`zk1 = zk2`	die Zeichenketten identisch sind
`zk1 != zk2`	die Zeichenketten voneinander abweichen

Tabelle 14.1: `test`-Bedingungen auf Zeichenketten

Ausdruck	liefert wahr, wenn
`z1 -eq z2`	die Zahlen gleich sind (***e**qual*)
`z1 -ne z2`	die Zahlen ungleich sind (***n**ot **e**qual*)
`z1 -gt z2`	`z1` größer `z2` ist (***g**reater **t**han*)
`z1 -ge z2`	`z1` größer gleich `z2` ist (***g**reater **e**qual*)
`z1 -lt z2`	`z1` kleiner `z2` ist (***l**ess **t**han*)
`z1 -le z2`	`z1` kleiner gleich `z2` ist (***l**ess **e**qual*)

Tabelle 14.2: `test`-Bedingungen auf Zahlen

Ausdruck	liefert wahr, wenn
`-b dat`	`dat` existiert und eine blockspezif. Gerätedatei ist (***b**lock*)
`-c dat`	`dat` existiert und eine zeichenspezif. Gerätedatei ist (***c**haracter*)
`-d dat`	`dat` existiert und es sich um ein Verzeichnis handelt (***d**irectory*)
`-e dat`	die Datei existiert (***e**xist*)
`-f dat`	`dat` existiert und es sich um eine einfache Datei handelt (***f**ile*)
`-L dat`	`dat` existiert und es sich um einen symbolische Link handelt (***L**ink*)
`-r dat`	`dat` existiert und die Datei gelesen werden darf (***r**ead*)
`-s dat`	`dat` existiert und die Datei mindestens 1 Byte lang ist (***s**ize*)
`-w dat`	`dat` existiert und die Datei verändert werden darf (***w**rite*)
`-x dat`	`dat` existiert und die Datei ausgeführt werden darf (***e**xecute*)
`-O dat`	`dat` existiert und der `dat`-Eigentümer der effektiven UID entspricht (***O**wner*)
`-G dat`	`dat` existiert und die `dat`-Gruppe der effektiven GID entspricht (***G**roup*)
`-S dat`	`dat` existiert und eine spezielle Datei vom *socket*-Typ ist (***S**ocket*)
`-p dat`	`dat` existiert und eine named pipe ist (***p**ipe*)
`dat1 -ef dat2`	beide Dateien denselben I-Node haben (***e**qual **f**ile*)
`dat1 -nt dat2`	`dat1` neuer als `dat2` ist (***n**ewer **t**han*)
`dat1 -ot dat2`	`dat1` älter als `dat2` ist (***o**lder **t**han*)

Tabelle 14.3: `test`-Bedingungen auf Dateien (Auszug)

Ausdruck	Bedeutung
`!ausdruck`	negiert den Wahrheitswert des nachfolgenden Ausdrucks
`ausdr1 -a ausdr2`	verknüpft die Wahrheitswerte der beiden Ausdrücke mit dem UND-Operator (***and***)
`ausdr1 -o ausdr2`	verknüpft die Wahrheitswerte der beiden Ausdrücke mit dem ODER-Operator (***or***)
`( ausdr )`	Klammerung eines Ausdruckes, um andere als die voreingestellten Prioritätenfür die Operatoren bei der Auswertung eines Ausdrucks festzulegen

Tabelle 14.4: verknüpfte `test`-Bedinungen

14.1.2 Das Kommando `expr`

Dieses Kommando kann zur Berechnung von arithmetischen Ausdrücken (siehe auch Kapitel 13.5.2), zum Vergleich zweier Zeichenketten etc. benutzt werden. Die Aufrufsyntax lautet:

```
expr argument(e)
```

Als Argumente können Zeichenketten, ganze Zahlen oder Konstruktionen, die eine ganze Zahl oder Zeichenketten liefern, wie Variablenwerte oder Kommandosubstitution, angegeben werden. Folgendes ist dabei zu beachten:

- einzelne Argumente müssen durch Leer- und/oder Tabulatorzeichen getrennt werden;
- sollten sich Shell-Metazeichen im Argument befinden, muß die Sonderbedeutung durch Quoting (voranstellen eines "\") ausgeschaltet werden;
- um negative ganzzahlige Argumente darzustellen, wird ein Minuszeichen vorangestellt;
- bei den Vergleichen ist der Rückgabewert 1 für wahr und 0 für falsch (C-Konvention);
- liefert die Berechnung des angegebenen Ausdrucks Null, so wird der arithmetische Wert 0 zurückgegeben;

Nachfolgend in Tabelle 14.5 werden die einzelnen Operatoren beschrieben (in der Reihenfolge ihrer Prioritäten).

14.2 Die `if`-Anweisung

Die `if`-Anweisung ist eine Verzweigungsanweisung, wobei zuerst der `exit`-Status des letzten Kommandos in der Bedingung überprüft wird. Liefert dieser einen `exit`-Status von 0, wird die `kdoliste` ausgeführt, ansonsten diese einfach übersprungen. Hierbei muß zwischen der einseitigen Auswahl (`if` ohne `else`-Teil), der zweiseitigen Auswahl (`if` mit `else`-Teil) und der Mehrfach-Auswahl (`if` mit `elif`-Teil) unterschieden werden.

Operator	**Bedeutung**
`op1 : op2`	überprüft, ob der reguläre Ausdruck des Operanden `op2` den Operanden `op1` abdeckt;liefert die Anzahl der abgedeckten Zeichen, sonst 0; soll der abgedeckte Teil selbst ausgegeben werden,so ist der entsprechende reguläre Ausdruck in `op2` mit `\(...\)` zu klammern;
`op1 \* op2`	Multipliziert `op1` mit `op2`
`op1 / op2`	Dividiert `op1` durch `op2` und liefert ein ganzzahliges Ergebnis
`op1 % op2`	Liefert den Rest aus der Ganzzahldivision von `op1` durch `op2` (Modulooperator)
`op1 + op2`	Addiert `op1` mit `op2`
`op1 - op2`	Subtrahiert `op2` von `op1`
`op1 = op2`	prüft, ob `op1` gleich `op2` ist
`op1 \> op2`	prüft, ob `op1` größer als `op2` ist
`op1 \>= op2`	prüft, ob `op1` größer oder gleich `op2` ist
`op1 \< op2`	prüft, ob `op1` kleiner als `op2` ist
`op2 \<= op2`	prüft, ob `op1` kleiner oder gleich `op2` ist
`op1 != op2`	prüft, ob `op1` ungleich `op2` ist
`op1 \& op2`	liefert als Ergebnis `op1`, wenn keiner der beiden Operatoren `op1` oder `op2` der Wert 0 oder die leere Zeichenkette ist (nicht zu verwechseln mit einer undefinierten Variablen), ansonsten wird als Ergebnis 0 geliefert;
`op1 | op2`	liefert als Ergebnis `op1`, wenn der Operand `op1` nicht der Wert 0 oder die leere Zeichenkette ist, ansonsten wird als Ergebnis `op2` geliefert

Tabelle 14.5: Übersicht der `expr`-Operatoren

Syntax

```
if Bedingung1
 then
   then_kdoliste1
 [elif Bedingung2
     then
        then_kdoliste2]
 :
 :
 [else
     else_kdoliste1]
fi
```

Beispiel für eine einseitige Auswahl:

```
if test $# -eq 0
then
   echo "Ein Argument muss mindestens angegeben werden"
   exit 1
fi
```

In diesem Beispiel wird überprüft, ob beim Starten des Shell-Skripts mindestens ein Parameter angegeben wurde. Ist dies nicht der Fall, so ist die Bedingung wahr, der `then`-Teil wird ausgeführt und eine Meldung ausgegeben sowie das Skript abgebrochen. Ansonsten wird nach dem Schlüsselwort `fi` fortgefahren.

Beispiel für eine zweiseitige Auswahl:

```
if test $# -eq 0
then
   echo "Es wurde kein Argument angegeben"
else
   echo "Es wurden die Argumente " $* " angegeben"
fi
```

Dieses Skript funktioniert ähnlich wie das vorherige. Zuerst wird überprüft, ob Parameter dem Skript übergeben wurde. Ist dies nicht der Fall, so ist die Bedingung wahr und es wird der `then`-Teil ausgeführt. Ansonsten wird der `else`-Teil ausgeführt und alle Parameter ausgegeben. Anschließen wird nach dem Schlüsselwort `fi` fortgefahren.

Beispiel für die Mehrfach-Auswahl:

```
if [ "$1" = "gruen" ]
then
   echo "Fahren"
elif [ "$1" = "gelb" ]
then
   echo "Wenn rot -> gelb: Vorsichtig fahren"
```

```
   echo "Wenn gruen -> gelb: Stoppen"
elif [ "$1" = "rot" ]
then
   echo "Stoppen"
else
   echo "Ampel ausser Betrieb"
fi
```

Dieses Shell-Skript gibt für eine Ampelfarbe die erforderliche Reaktion eines Autofahrers aus (aus [Herold, Seite 119]). Dabei wird die Ampelfarbe als erstes Argument übergeben und über die `if`- und `elif`-Anweisungen ausgewertet. Sollte kein Argument oder eine falsche Zeichenfolge angegeben werden, so wird der `else`-Teil ausgeführt.

14.3 Die case-Anweisung

Die `case`-Anweisung eignet sich besser für die Mehrfach-Auswahl, z.B. um Menüs zu realisieren, als die `if`-Anweisungen. Dabei wird die Zeichenkette `wort` in der angegebenen Reihenfolge zunächst mit `pattern1`, dann mit `pattern2 ...` verglichen, bis eine Übereinstimmung gefunden wird. Bei einer Übereinstimmung wird die zugehörige `kdoliste` ausgeführt und anschließend nach dem Schlüsselwort `esac` fortgefahren. Das Zeichen `)` trennt die `pattern`-Angabe von der zugehörigen `kdoliste` und achten Sie auf die beiden Semikla am Ende der `kdoliste`, sie müssen angegeben werden. Bei den Pattern können Sie die Substitutionsmechanismen benutzen, `/` oder der führende Punkt oder ein Punkt, welcher unmittelbar `/` folgt, müssen nicht explizit wie bei der üblichen Dateinamenexpandierung in `wort` angegeben sein.

Den Substitutionsmechanismus wird häufig auch für den Default-Zweig gewählt in der Form `*)` bzw. `?)`. Achten Sie aber darauf, diesen als letzten anzugeben, da der Reihenfolge nach verglichen wird und sonst, wenn Sie diesen womöglich als ersten Pattern angeben, immer nur dieser ausgeführt wird.

Syntax

```
case wort in
   pattern1) kdoliste1;;
   pattern2) kdoliste2;;
   :
   :
   patternn) kdolisten;;
esac
```

Beispiel für einecase-Auswahl

```
case "$Menuitem" in
   1)      echo "Menüpunkt 1 wird ausgeführt";;
   2)      echo "Menüpunkt 2 wird ausgeführt";;
   [EeXx]) echo "Menü wird beendet";;
   *)      echo "Falsche Eingabe";;
esac
```

Befindet sich in der Variablen Menuitem der Wert 1 bzw. 2, werden die entsprechenden Kommandolisten ausgegeben. Sollte der Wert ein “E”, “e”, “X” oder “x” sein, so wird die Meldung ausgegeben, daß das Menü beendet wird. Befindet sich ein anderer Wert in der Variablen, so wird die Meldung “Falsche Eingabe” ausgegeben.

14.4 Die while-Schleife

Mit der while -Schleife lassen sich Kommandos abhängig von der Bedingung wiederholt ausführen. Wenn der exit-Status des letzten Kommandos in der kdoliste1 gleich 0 (erfolgreich) ist, wird die kdoliste2 ausgeführt. Dieser Ablauf wiederholt sich so lange, bis die kdoliste1 einen exit-Status unterschiedlich von 0 liefert. Dann wird hinter dem done-Schlüsselwort die Bearbeitung weitergeführt. Natürlich kann es dabei passieren, daß die kdoliste2 gar nicht ausgeführt wird, wenn schon die erste Ausführung der kdoliste1 einen exit-Wert unterschiedlich von 0 liefert.

Syntax

```
while kdoliste1
 do
   kdoliste2
 done
```

Beispiel für eine while-Schleife

```
while who | grep $1 >> /dev/null
do
   echo "Benutzer $1 ist immer noch angemeldet"
   sleep 300
done
echo "Benutzer $1 hat sich abgemeldet"
```

Dieses Skript prüft alle 300 Sekunden, ob der Benutzer, der als Parameter übergeben wurde, noch auf dem System angemeldet ist und gibt dann eine Meldung aus. Die Ausgabe der kdoliste1 wird auf /dev/null (einem großen, schwarzen Loch) umgeleitet, da nicht die Ausgabe, sondern nur der exit-Wert relevant ist.

14.5 Die until-Schleife

Die until-Schleife funktioniert ähnlich wie die while-Schleife, sie ist die Umkehrung dazu. Der Schleifenrumpf (kdoliste2) wird so lange ausgeführt wie kdoliste1 einen exit-Wert verschieden von 0 liefert. Wie auch schon bei der while-Schleife kann es auch hier passieren, daß der Schleifenrumpf nicht durchlaufen wird, und zwar dann, wenn die kdoliste1 beim ersten Ausführen sofort einen exit-Status von 0 liefert.

Syntax

```
until kdoliste1
 do
   kdoliste2
 done
```

Beispiel für eine until-Schleife

```
until who | grep $1 >> /dev/null
do
   echo "Benutzer $1 ist noch nicht angemeldet"
   sleep 300
done
echo "Benutzer $1 hat sich angemeldet"
```

Dieses Beispiel ist die Umkehrung des Beispiels zur while-Schleife, wobei alle alle 300 Sekunden geprüft wird, ob sich der als Parameter übergebene Benutzer noch nicht angemeldet hat. Die Schleife wird so lange durchgeführt, bis sich der Benutzer angemeldet hat.

14.6 Die for-Schleife

Anders als in den beiden vorherigen Schleifen, wo die Durchführung des Schleifenrumpfes an eine Bedinung geknüpft ist, wird bei der for-Schleife die Anzahl der Schleifendurchläufe über eine wort-Liste festgelegt. Hierfür gibt es zwei Möglichkeiten, entweder implizit alle Positionsparameter (keine in-Angabe) oder alle danach angegebenen worte (bei einer in-Angabe). Der Laufvariablen wird dabei nacheinander jedes einzelne wort zugewiesen.

Syntax

```
for laufvariable [in wort1  wortn]
 do
   kdoliste
 done
```

Beispiel für eine for-Schleife ohne in-Angabe

```
nummer=1
for i
do
   echo "$nummer. Argument: $j"
   nummer=`expr $nummer + 1`
done
```

Bei diesem Skript werden alle übergebenen Parameter nacheinander ausgegeben, vorangestellt wird die Zeichenfolge "<Argumentnummer>. Argument: "

Beispiel für eine for-Schleife mit in-Angabe

```
for i in *
do
   cat $i
   chmod 755 $i
done
```

Dieses Skript gibt nacheinander alle Dateien im aktuellen Verzeichnis mit `cat` aus (außer die, deren Name mit . beginnt) und ändert die Zugriffsmaske auf "`rwxr-xr-x`".

14.7 Die Kommandos continue, break und exit

`break` bewirkt das unmittelbare Verlassen der direkt umgebenen Schleife (`while`, `until` und `for`). Dabei kann `break` mit einem optionalen Zahlenwert `n` aufgerufen werden. Dieser Zahlenwert legt dann die Schachtelungstiefe fest, welche verlassen werden soll. Ohne Angabe dieses Zahlenwertes wird die aktuelle Schleife abgebrochen und die Ausführung nach dem Schleifenrumpf fortgesetzt. Häufig ist die Ausführung von `break` an eine Bedinung mittels `if`- oder `case`-Konstrukt verknüpft.

Syntax: `break [n]`

Der Aufruf des Kommandos `continue` in einem Schleifen-Körper (`while`, `until` und `for`) bewirkt die unmittelbare Ausführung des nächsten Schleifendurchlaufs. Es wird nicht wie bei `break` die Schleife verlassen, sondern sofort zur Schleifenbedingung gesprungen und dort fortgefahren. `continue` kann ein optionaler Zahlenwert `n` angegeben werden, der die Anzahl der Schachtelungstiefe festlegt, die übersprungen werden soll, ansonsten wird für die direkt umschließende Schleife neu geprüft und durchlaufen. Üblich ist der Gebrauch von `continue`-Anweisungen in Verbindung mit `if`- oder `case`-Konstruktionen in Schleifen.

Syntax: `continue [n]`

Das Kommando `exit` beendet unmittelbar das Shell-Skript bzw. die aktive Shell. Wird `exit` ohne optionales Argument (Zahlenwert) aufgerufen, liefert es den `exit`-Status 0 zurück, ansonsten den Zahlenwert. Dieser kann vor allem zur Fehlersuche benutzt und ausgewertet werden, da der Rückgabewert des letzten Kommandos (hier dann also `exit`) über die Variable `$?` ausgelesen werden kann.

Syntax: `exit [n]`

14.8 Funktionen

Wie in höheren Programmiersprachen ist es in der `bash` erlaubt, Funktionen zu definieren. Dabei muß zwischen `{` und dem ersten Kommando aus `kdoliste` mindestens ein Leerzeichen, Tabulatorzeichen oder Neuezeile-Zeichen angegeben sein. Zwischen dem letzten Kommando aus `kdoliste` und dem abschließenden `}` muß immer ein Semikolon angegeben sein, wenn sich diese beiden in derselben Zeile befinden. Anschließend kann die Funktion über ihren Namen aufgerufen (ohne Klammern) und Parameter mit übergeben werden, welche dann der entsprechenden Funktion in Form von Positionsparametern zur Verfügung gestellt werden. Der Parameter `$0` enthält dann allerdings nicht den Funktionsnamen, sondern den Namen der ausführenden Shell. Der Bezeichner `function` bei der Funktionsdefinition ist optional, verhilft aber zur besseren Lesbarkeit des Skripts. Über das Variablendefinitionskommando `local` können Variablen definiert werden, die nur innerhalb der Funktion gültig sind. Das Schlüsselwort `return` bewirkt das Verlassen der Funktion.

Syntax

```
[function] funktionsname() {
 kdoliste;}
```

Beispiel

```
funktion ll() {
ls -CF "$@"
echo "------\n`ls "$@" | wc -l` Dateien";}
```

Hier wird eine neue Funktion `ll` definiert, welches alle Dateien bzw. die über Parameter angegebenen Dateien eines Verzeichnisses mit der Opton `-CF` listet und anschließend eine Trennlinie sowie die Anzahl der Dateien ausgibt. Die Funktion kann dann über

```
root@ServerI # ll
```

aufgerufen werden, um alle Dateien aufzulisten oder beispielsweise mittels

```
root@ServerI # ll a*
```

aufgerufen werden, um alle Dateien, die mit “a” beginnen, aufzulisten.

Teil IV

Konfigurationsdateien

Kapitel 15

Die Profildateien

Profildateien sind Konfigurationsdateien, die Einstellungen für die Shells vornehmen und Variablen, die für den Ablauf der Programme notwendig sind, setzen. Die globle Profil-Datei ist `/etc/profile`, die jedem Benutzer einen Satz an Konfigurationseinstellungen bereitstellt. Für eigene, erweiterte Konfigurationen ist die Datei `.profile` im Heimatverzeichnis zuständig sowie spezielle Konfigurationsdateien im Heimatverzeichnis. Der Name hängt von der benutzten Shell ab, z.B. für die `bash` ist es die Datei `.bashrc`. Hierbei werden globale Einstellungen von lokalen überschrieben.

In der globalen Profildatei werden Sie häufig die Form

```
<variable> = <wert>
export <variable>
```

entdecken. Hiermit wird zuerst einer Variablen ein Wert zugewiesen (möglich sind auch Substitutionsmechanismen) und anschließend die Variable als globale Variable exportiert. Häufiger finden Sie auch Programmierkonstrukte, um die Variablen speziell an Ihr System anpassen zu können. Z.B. wird der Variablen `MANPATH` zuerst ein Wert zugewiesen (eine Reihe von `man`-Verzeichnissen, einzelne Verzeichnisangaben werden durch einen Doppelpunkt getrennt) und anschließend eine `for`-Schleife durchlaufen, die weitere Verzeichnisse mit möglichen `man`-Seiten enthält und bei Existenz an die `MANPATH`-Variablen angehängt. Hiermit wird erreicht, daß spezielle Sprachversionen von `man`-Seiten verfügbar gemacht werden[1].

Die Variable `PS1` ist für den normalen Prompt zuständig. Hier finden Sie ein ganzes Programmiersprachenkonstrukt (bei SuSE 6.3), um den Prompt in Abhängigkeit der benutzten Shell sowie des Benutzers zu setzten. Für den Wert dieser Variablen ist eine Zeichenkette vorgesehen, die neben normalem Text die in Tabelle 15.1 wiedergegebenen Zeichenkombinationen enthalten kann.

Die beiden nachfolgenden Beispiele zeigen eine `.profile`-Datei, die beim Login interpretiert wird sowie die Datei `.bash_logout`, die von der `bash` beim `logout` ausgeführt wird.

```
# .profile
# Prompt setzen
export PS1="\h:\w\$ "
# Kurzbefehle definieren
```

[1] Der Befehl `man <Befehl>` zeigt normalerweise immer nur die zuerst gefundene `man`-Seite an. Möchten Sie, daß zuerst deutsche Seiten angezeigt werden, sollten Sie den Pfad zu diesen Seiten direkt an erster Stelle von `MANPATH` eintragen.

Sonderzeichen	Bedeutung
\t	aktuelle Zeit im Format HH:MM:SS
\d	aktuelles Datum im Format “Wochentag Monat Tag”
\w	aktuelles Verzeichnis
\W	letzter Teil des aktuellen Verzeichnisses
\u	User-Name
\h	Host-Name
\$	Promptzeichen ($ bzw. # für root)
\n	Neuezeile-Zeichen
\s	Name der Shell (Basisname von $0)
\#	Nummer des aktuellen Kommandos
\!	History-Nummer des aktuellen Kommandos
\nnn	Zeichen mit oktalem Ascii-Code nnn
\\	Backslash
\[	Beginn für eine Reihe von nichtdruckbaren Zeichen
\]	Ende einer Reihe von nichtdruckbaren Zeichen

Tabelle 15.1: Zeichenkombinationen für den Prompt

```
alias win=startx
alias move=mv
alias copy=cp
# Loginmeldung ausgeben
echo "Hallo $USER, es ist jetzt `date +"%A %d %B, %k Uhr $M"` "
# Größe des Heimatverzeichnis ausgeben
echo "Dein Verzeichnis $HOME belegt `du -sh` "
```

Diese `.profile`-Datei setzt zuerst einen Benutzerspezifischen Prompt, danach werden Kurzbefehle (Alias-Namen) definiert, z.B. daß das Verschieben von Dateien auch über das Kommando `move` erfolgen kann. Zum Schluß wird eine Loginmeldung mit Datum und Uhrzeit ausgegeben sowie die Größe des Heimatverzeichnisses.

```
# .bash_logout
# Bildschirm löschen
tput clear
# Sichern aller neuen oder geänderten Texte
# vom Verzeichnis $HOME/text nach $HOME/text.bak
cp -u $HOME/text/*.* $HOME/text.bak
# Logoutmeldung ausgeben
echo "Hallo $USER, es ist jetzt `da-
te +"%A %d %B, %k Uhr $M"`. Auf Wiedersehen!"
```

Durch diese `.bash_logout` wird beim `logout` nun zuerst der Bildschirm gelöscht, anschließend die veränderten Dateien des Verzeichnisses text unterhalb des Heimatverzeichnisses ins Verzeichnis text.bak gesichert. Anschließend wird eine Meldung ausgegeben.

Kapitel 16

Globale SuSE-Konfigurationsdatei

SuSE benutzt für die Konfiguration eine eigene Text-Datei `rc.config` im Verzeichnis `/etc`. Hier werden alle relevanten Einstellungen vorgenommen und mit Hilfe eines Skripts in die eigentlichen Konfigurationsdateien übertragen. Sie können diese Datei entweder mit YaST bearbeiten oder sie direkt mit einem Texteditor verändern. Sollten Sie die Konfigurationsdatei mit einem Texteditor verändern, müssen Sie anschließend das Skript `SuSEconfig` aufrufen, damit die Einstellungen übertragen werden - bei Veränderung über YaST startet dieses Skript automatisch. Damit die neuen Einstellungen aktiv werden, müssen die veränderten Dienste neu gestartet werden. Entweder starten Sie die Skripte manuel neu (im Verzeichnis `/sbin/init.d`) oder veranlassen, daß der Runlevel neu durchlaufen wird. Generell ist es zu empfehlen, Konfigurationsänderungen im Single-User-Mode auszuführen und danach in den eigentlichen Runlevel zurückzukehren (zur Änderung des Runlevels siehe 2.1, Seite 10).

Der Vorteil einer solchen globalen Konfigurationsdatei ist sicherlich die leichtere Handhabung der Konfiguration. Sie müssen nicht mehr wissen, welche Konfigurationsdatei für welchen Dienst zuständig ist und alle wichtigen Einstellungen können über eine Datei gelöst werden. Dieser Vorteil hat jedoch, wie so häufig, auch Nachteile. Sollten Sie mal ein anderes Linux-System konfigurieren, welches nicht auf SuSE basiert, werden Sie nicht wissen, wo Sie welche Einstellungen vornehmen müssen. Komplexere Konfigurationen sind mittels der Konfigurationsdatei nicht möglich. Probleme könnte es auch geben, wenn Sie eine Installationsroutine einer Software starten, die nichts von SuSE und `rc.config` weiß bzw. damit nicht umgehen kann. Diese Installationsroutine modifiziert die Konfigurationsdatei direkt und beim nächsten Durchlauf von `SuSEconfig`, weil Sie z.B. ein Softwarepaket von der Distributions-CD nachinstalliert haben, werden diese Änderungen überschrieben, sie stehen ja nicht in `rc.config`, und die Software funktioniert womöglich nicht mehr so, wie Sie es erwarten.

Es sprechen also auch Gründe dafür, diese Konfigurationsdatei kritisch zu sehen. Möchten Sie die größtmögliche Kontrolle über das System haben, können Sie auch `SuSEconfig` abschalten. Hierfür muß in der `rc.config` der Parameter `ENABLE_SUSECONFIG` auf “no” gesetzt werden und, das letzte Mal, `SuSEconfig` gestartet werden.

Mit Hilfe dieser Konfigurationsdatei können Sie Grundeinstellungen verändern, lokale Hardware initialisieren, Netz- und lokale Dienste starten und konfigurieren sowie lokale Wartungsdienste modizifieren. Dabei haben die Einträge die Form:

```
<Parameter>=<Wert>
```

Die meisten Parameter werden im SuSE-Handbuch beschrieben, z.B. [Bauer et al, Seite 399ff] und sind auch in der Konfigurationsdatei meist ausführlich kommentiert.

Kapitel 17

Die Mount-Datei /etc/fstab

Diese Konfigurationsdatei gibt an, welche Datenträger in das Dateisystem aufgenommen werden sollen (siehe hierzu auch Kaptiel 2, Seite 39). Entweder werden sie automatisch beim Systemstart eingebunden oder können bei Angabe der Option noauto bequem über mount name eingebunden werden. Die Syntax sieht folgendermaßen aus:

```
<Gerätedatei> <Mount-Point> <Dateisystem> <Optionen> <Dump-
Nr> <Prüf-Nr>
```

ext2	Linux-Standard
minix	gebräuchlich für Unix-Disketten
msdos	DOS-Partition
vfat	DOS-Partition mit langen Dateinamen
iso9660	CD-ROM-Dateisystem
ntfs	NT-Dateisystem (Schreibzugriff sehr gefährlich)
nfs	Zugriff auf Verzeichnisse anderer Rechner via Netz
smbfs	Zugriff auf Windows-Freigaben
proc	Prozeßverwaltung
swap	Swap-Partition oder -Datei

Tabelle 17.1: einige mögliche Dateisysteme für mount und /etc/fstab

block=*n*	Blockgröße (für CD-ROM *n*=2048)
conv=auto	automatische Konversion von DOS-Textdateien (gefährlich)
default	Voreinstellung
gid=n	Gruppenzugehörigkeit der Dateien
noauto	Datenträger wird nicht automatisch eingebunden
noexec	keine Programmausführung erlaubt
ro	Dateisystem Read-Only einbinden
sync	Schreibzugriffe nicht puffern
uid=n	Benutzerzugehörigkeit der Dateien
umask	Zugriffsbits der Dateien
user	jeder Benutzer darf (u)mount ausführen

Tabelle 17.2: Die wichtigsten Optionen für /etc/fstab und mount

Für die Gerätedatei muß der komplette Pfad angegeben werden (z.B. /dev/hda1) und der angegebene Mount-Point muß im Dateibaum als Verzeichnis vorhanden sein. Sollte dieses Verzeichnis nicht leer sein, werden die Daten im Verzeichnis überdeckt. Es besteht nur noch der Zugriff auf die eigentlichen Daten im eingebundenen Dateisystem. Wenn Verzeichnisse fremder Rechner via NFS eingebunden werden, lautet der Device-Name rechnername:/verzeichnis. Die möglichen Dateisysteme sind abhängig von der Kernelkonfiguration, es besteht nur die Unterstützung für die Dateisysteme, die im Kernel eingebunden oder als Modul verfügbar sind. Einige Dateisysteme sind in Tabelle 17.1 aufgeführt. Optionen sind vor allem für CD-ROM und Dateisysteme wie MS-DOS und OS/2 notwendig, die wichtigsten Optionen sind in Tabelle 17.2 wiedergegeben[1].

Die <Dump-Nr> enthält Informationen für das Programm dumpfs und wird zur Zeit ignoriert. <Prüf-Nr> gibt an, ob und wie die Dateisysteme auf Richtigkeit überprüft werden. Dabei sollte für das Wurzelverzeichnis 1 und für alle anderen veränderlichen Dateisystem 2 angegeben werden. Dateisysteme, die nicht überprüft werden sollen (z.B. CD-ROM, proc und swap), bekommen 0 [Kofler, Seite 191].

Das nachfolgende Listing zeigt eine etwas komplexere fstab-Datei. Neben den Linux-Partitionen, Swap-Partition und proc-Dateisystem wird beim Systemstart die Partition /dev/hda8 mit einem vfat-Dateisystem eingebunden. Die Gruppenzugehörigkeit wird auf Gruppe 100 gesetzt und der Besitzer aller Dateien und Verzeichnisse ist root (uid=0 ist auf jedem System der Benutzer root). Für den liberalen Zugriff werden die Dateirechte über umask=0002 auf Lesen, Schreiben und Ausführen für Besitzer und Gruppe gesetzt, alle anderen erhalten Lese- und Ausführrechte. Für das CD-ROM Laufwerk werden die Einträge vorbereitet, so daß jeder Benutzer mit (u)mount /cdrom das Laufwerk ein- und ausbinden kann. Beim Diskettenlaufwerk gibt es beim Dateisystemtyp eine Besonderheit, hier wird der Type auto eingestellt. Hier versucht Linux, den Dateisystemtyp selbstständig zu ermitteln, gerade bei DOS-Disketten mit langen Dateinamen schlägt dies jedoch fehl.

```
/dev/hda2  swap    swap    defaults                  0 0
/dev/hda3  /       ext2    defaults                  1 1
/dev/hda5  /usr    ext2    defaults                  1 2
/dev/hda1  /boot   ext2    defaults                  1 2
/dev/hda7  /home   ext2    defaults                  1 2
/dev/hda8  /daten  vfat    gid=100,umask=0002,uid=0  0 0
/dev/hdb   /cdrom  iso9660 ro,noauto,user,block=2048 0 0
/dev/fd0   /floppy auto    noauto,user               0 0
none       /proc   proc    defaults                  0 0
```

Linux speichert in der Datei /etc/mtab eine Liste aller Datenträger, die *momentan* eingebunden sind und ändert diese dynamisch mit jedem mount- und umount-Befehl.

[1] Die komplete Optionsliste erhalten Sie mit man mount, weitere Informationen erhalten Sie auch mit man fstab

Teil V

Tools

Kapitel 18

Das SuSE-Konfigurationstool yast

YaST ist das zentrale Konfigurationstool der SuSE-Distribution und bildet die Schnittstelle zu vielen Systemeinstellungen. Der Umfang beschränkt sich auf die wichtigsten Einstellungen, für spezielle Konfigurationen wie Samba, Apache, DNS ... muß jedoch direkt in den Konfigurationsdateien gearbeitet werden. Bei der Installation eines SuSE-Systems werden Sie zum ersten Mal auf YaST treffen, da es auch für die Installation zuständig ist. Bei einem laufenden System können Sie hiermit Pakete nach- und deinstallieren, das Netzwerk konfigurieren, Benutzer und Gruppen bearbeiten, diverse Konfigurationen ändern und die Datei `/etc/rc.config`, die zentrale Konfigurationsdatei von SuSE, konfigurieren. Aufgerufen wird YaST entweder mit `YaST` oder `yast`.

Abbildung 18.1: Das Programm YaST

Kapitel 19

Das Konfigurationstool `Webmin`

Die Konfigurationsmöglichkeiten sind mit dem Tool Webmin sehr viel weitreichender als mit YaST. Webmin ist ein Browser-basiertes Tool und damit hervorragend zur Fernadministration geeignet. Es kann über www.webmin.de als `rpm`- oder `tar`-Achiv kostenlos heruntergeladen werden. Bei der Installation werden verschiedene Einstellungen zum System abgefragt, wie Betriebssystem, Distribution und Version, Port, Administrator-Benutzer und Paßwort ... Aufgerufen wird Webmin im Browser über `http://<Rechner>:<Port>`, wobei der Standardport 10000 ist (dieser sollte auch belassen werden, da zum einen der Port einfach zu merken ist und von keinem anderen Dienst benutzt wird). Nach der Anwahl von Webmim erscheint ein Anmeldedialog und nach erfolgreicher Autentifizierung die eigentliche Oberfläche. Die Oberfläche gliedert sich in 5 Bereiche: Webmin-Konfiguration, System, Servers, Hardware und Others, womit man (fast) alles konfigurieren kann. Für Webmin können mehrere Benutzer eingerichtet werden, die unterschiedliche Aufgaben erledigen dürfen. Webmin befindet sich in der ständigen Entwicklung, auf vielen Internet-Seiten finden Sie zusätzliche Module, die Sie einbinden können Sollten Sie das passende Modul nicht finden, programmieren sie es einfach selber!

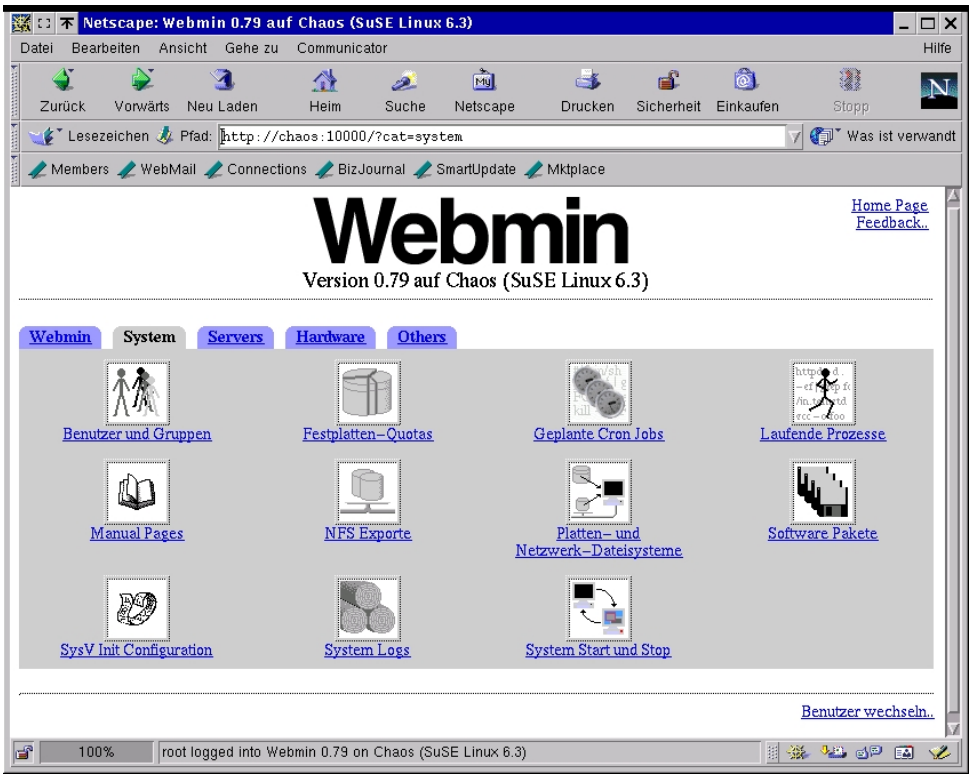


Abbildung 19.1: Das Konfigurationstool Webmin

Kapitel 20

Das Samba-Konfigurationstool swat

Für Samba gibt es auch ein browser-basiertes Konfigurationstool mit Namen `swat`. Hierfür sind zuerst Änderungen an den Konfigurationsdateien `/etc/inetd.conf` und `/etc/services` nötig. In `/etc/services` muß die Zeile `swat 901/tcp` nachgetragen werden[1] .
In `/etc/inetd.conf` muß die Zeile

```
swat    stream    tcp    nowait.400    root    /usr/sbin/swat    swat
```

hinzugefügt werden[2][3] und danach den `inetd`-Dämon neu starten, z.B. bei SuSE über `/sbin/init.d/inetd restart`. Anschließend kann die Samba-Konfiguration über einen Browser erfolgen, die Adresse lautet: `http://<rechner>:901`. In `swat` können die globalen Einstellungen eingestellt werden, genauso können Freigaben eingerichtet, gelöscht und modifiziert werden. Denken Sie daran, daß nach Beendigung der Konfigurationsarbeiten die Samba-Dämonen neu gestartet werden müssen. Swat kann dieses auch für Sie erledigen, in der Rubrik “Status” finden Sie entsprechende Buttons. In dieser Rubrik sehen Sie auch alle aktiven Zugriffe, Shares und offene Dateien, hier können Sie auch einzelne Zugriffe gezielt “killen”.

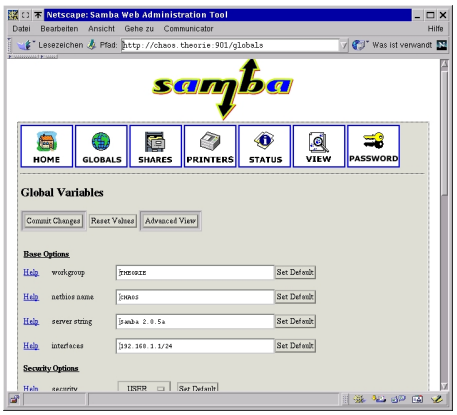


Abbildung 20.1: Das Samba-Konfigurationstool `swat`

[1] Bei SuSE 6.3 ist sie schon enthalten und muß gegebenenfalls noch das #-Zeichen entfernt werden.
[2] Natürlich müssen Sie darauf achten, daß der Pfad in der sechsten Spalte dieser Zeile Ihrem Installationspfad von Samba entspricht.
[3] Bei SuSe 6.3 ist diese Zeile schon vorhanden, es muß nur noch das Kommentarzeichen (#) entfernt werden.

Teil VI

Anhang

Anhang A

Der Unix-Editor vi

Der vi (gespr. wie-ei) ist der Standard-Editor unter allen erhältlichen Unix- sowie Linux-Distributionen. Auch wenn noch andere Editoren unter diesen Systemen zur Verfügung gestellt werden, die dazu auch noch komfartabler in der Bedieung sind, ist man immer wieder gezwungen, auf den vi zurück zu greifen, z.B. bei Linux-Rettungssystemen oder einer Unix-Distribution, die nur den vi zur Verfügung stellt [Borrotzu].

Vom vi gibt es mehrere Erweiterungen, wie den vim. Da vi keine Menüsteuerung enthält, gibt es zwei Betriebsmodi, den Befehlsmodus und den Eingabemodus, die über Tastatursequenzen gesteuert werden. Um in den Eingabemodus zu gelangen, gibt es die folgenden Tastatursequenzen:

I	Um Text vor der aktuellen Cursorposition einzufügen
A	Um Text hinter der aktuellen Cursorposition einzufügen
O	Um eine freie Textzeile unterhalb der aktuellen Zeile einzufügen
<Shift>+I	Um Text am Anfang der aktuellen Zeile einzufügen
<Shift>+A	Um Text am Ende der aktuellen Zeile einzufügen
<Shift>+O	Um eine freie Textzeile oberhalb der aktuellen Zeile einzufügen

Der Befehlsmodus bietet die Möglichkeiten, die Sie von Editoren gewöhnt sind (und geht sicherlich darüber hinaus). In den Befehlsmodus gelangt man über die Escape-Taste <ESC>. Alle Tastatursequenzen müssen grundsätzlich mit <ENTER> abgeschlossen werden. In den nachfolgenden Tabellen sind die, nach Meinung des Autors, wichtigsten Tastatursequenzen wiedergegeben.

Cursorbewegung

Pfeiltasten und Positionstasten	wie gewohnt
+	Cursor an den Anfang der nächsten Zeile
-	Cursor an den Anfang der vorherigen Zeile
W	Cursor ein Wort nach rechts
B	Cursor an den Anfag des aktuellen Wortes
C	Cursor an das Ende des aktuellen Wortes

Text löschen

X	Löscht das Zeichen an der aktuellen Cursorposition
D$	Löscht alle Zeichen von der aktuellen Cursorposition bis zum Zeilenende
DW	Löscht alle Zeichen von der aktuellen Cursorposition bis zum Anfang des nächsten Wortes
DD	Löscht die gesammte aktuelle Zeile

Kopieren und Einfügen

YW	Kopiert von der Cursorposition im aktuellen Wort bis zum Anfang des nächsten Wortes
Y$	Kopiert von der Cursorposition bis zum Zeilenende
YY	Kopiert die aktuelle Zeile
P	Fügt das Objekt rechts von der Cursorposition ein
<Shift>+P	Fügt das Objekt links von der Cursorposition ein

Suchen

/[Zeichenkette]	Sucht [Zeichenkette] vorwärts
?[Zeichenkette]	Sucht [Zeichenkette] rückwärts
N	Sucht in aktueller Richtung weiter
<SHIFT>+P	Sucht in entgegengesetzter Richtung weiter

vi-Steuerungssequenzen

U	Letzten Befehl rückgängig machen
.	Letzten Befehl wiederholen
:q	vi beenden, wenn keine Änderungen am Text vorgenommen wurden
:q!	beenden, ohne Speichern
:w ?[Dateiname]?	Text speichern; der Dateiname ist optional
:wq ?[Dateiname]?	Text speichern und vi beenden
:x	wie :wq, erzwingt aber das Überschreiben einer bestehenden Datei

Sollte Ihnen der Umgang mit solchen kryptischen Tastatursequenzen für die alltägliche Arbeit zu aufwendig sein, bietet Linux viele verschiedene Text-Editoren, von einfach bis zu komplex, wie z.B. den `Emacs`. Manche können Syntaxhervorhebung und sogar als Mail- und News-Client fungieren. Für einige bestehen auch spezielle Varianten für das X-Windows-System. Der Autor arbeitet z.B. am liebsten mit dem `fte`-Editor, einem freien Editor mit Syntaxhervorhebung und Menüführung, der auch für andere Betriebssysteme wie Windows, DOS und OS/2 verfügbar ist. Trotz allem sollte man die grundlegenden Tastatursequenzen für `vi` kennen (oder griffbereit liegen haben), da es der Editor ist, der auf jedem Linux- und Unix-System jederzeit verfügbar ist.

Anhang B

Kommandoreferenz

B.1 Hilfe

Manual-Seite zu Titel (meist Kommando-Name) ausgeben (manual)

```
man [Optionen] [Bereich] Titel
```

Optionen:
`-k Schlüsselwort(e)`
Infos zu Schlüsselwort(e) ausgeben;
`-f Datei`
Infos zu einer bestimmten Datei ausgeben;

alle Überschriften der man-Seiten anzeigen, die Schlüsselwort enthalten

```
apropos [Optionen] Schlüsselwort
```

Optionen:
`-e oder --exact`
Sucht nach einem genau passenden Wort;
`-w oder --wildcard`
das Schlüsselwort enthält die aus der Shell bekannten Jokerzeichen

Kurzbeschreibung zu Kommando/Datei ausgeben

```
whatis [Optionen] Kommando/Datei
```

Optionen:
`-w oder --wildcard`
das Schlüsselwort enthält die aus der Shell bekannten Jokerzeichen

startet info-Online-Hilfesystem

```
info [Optionen] [Thema]
```

erzeugen oder aktualisieren der Indexdatenbank für `man`

```
mandb [Optionen] [Pfad]
```

Optionen:
`-q oder --quiet`
Unterdrückt die Warnungen;
`-c oder --create`
erzwingt die Erzeugung einer neuen Indexdatenbank;

erzeugen oder aktualisieren der Indexdatenbank für whatis

```
makewhatis [Optionen]
```

Optionen:
`-u`
Aktualisieren der Indexdatenbank mit neuen Einträgen;
`-v`
Prozeßinformationen ausgeben;
`-m manpath`
Angabe des Pfades für man-Seiten (überschreiben der Systemeinstellungen);

B.2 Dateiverwaltung

wechseln zum Verzeichnis (change directory)

```
cd [Verzeichnis]
```

Ist kein Verzeichnis angegeben, dann wird zum Home-Verzeichnis gewechselt.

Dateien kopieren (copy)

```
cp [-optionen] Quelle Ziel
cp [-optionen] Dateien Zielverzeichnis
```

Optionen:
`-a bzw. --archive`
behält möglichst alle Attribute der Dateien bei;
`-b bzw. --backup`
bereits vorhandene gleichnamige Dateien werden nicht überschrieben, sondern in Backup-Dateien umbenannt (Dateiname plus ~);
`-i bzw. --interactive`
fragt, bevor vorhandene Dateien überschrieben werden;
`-p bzw. --preserve`
beläßt die Informationen über Besitzer, Gruppenzugehörigkeit, Zugriffsrechte und den Zeitpunkt der letzten Änderung unverändert. Ohne diese Option gehört die Kopie demjenigen, der `cp` ausführt (Benutzer und Gruppe), die Zeitangabe wird auf die aktuelle Zeit gesetzt;

`-R bzw. --recursive`
kopiert auch Unterverzeichnisse und die darin enthaltenen Dateien;
`-u bzw. --update`
kopiert Dateien nur dann, wenn dabei nicht eine gleichnamige Datei mit neuerem Datum überschrieben wird;

auflisten von Dateinamen (list)

```
ls [Optionen] [Datei(en)]
```

Optionen:
`-a bzw. -all`
zeigt auch Dateien an, die mit "." beginnen;
`-d bzw. --directory`
zeigt nur den Namen des Verzeichnisses, nicht aber seinen Inhalt an;
`-l bzw. --format=long oder --format=verbose`
zeigt zusätzlich zum Dateinamen (in einer Liste) weitere Informationen an: die Dateigröße in Bytes, die Zugriffsrechte etc;
`-p bzw. -F`
hängt an die Dateinamen ein Sonderzeichen an, das den Typ der Datei kennzeichnet;
`-r bzw. --reverse`
dreht die Sortierreihenfolge um.;
`-R bzw. --recursive`
erfaßt auch Dateien in Unterverzeichnissen;
`-S bzw. --sort=size`
sortiert die Dateien nach ihrer Größe (größte zuerst);
`-t bzw. --sort=time`
sortiert die Dateien nach Datum (neueste zuerst);
`-X bzw. --sort=extension`
sortiert die Dateien nach ihrer Erweiterungskennung (nach dem letzten Punkt);

Verzeichnis neu anlegen (make directory)

```
mkdir [Optionen] Verzeichnis
```

Optionen:
`-m Modus bzw. --mode=Modus`
setzt die Zugriffsrechte des neuen Verzeichnisses wie durch Modus angegeben;
`-p bzw. --parents`
erstellt auch Zwischenverzeichnisse, wenn nötig;

umbenennen von Dateien bzw. verschieben (move)

```
mv [optionen] Datei1 [Datei2 ...] Ziel
```

Optionen:
`-b bzw. --backup`
bereits vorhandene gleichnamige Dateien werden nicht überschrieben, sondern in Backup-Dateien umbenannt (Dateiname plus ~);
`-i bzw. --interactive`
fragt, bevor vorhandene Dateien überschrieben werden;

Dateien / Verzeichnisbäume löschen (remove)

```
rm [optionen] Datei(en)
```

Optionen:
`-f`
löscht ohne Rückfragen;
`-i bzw. --interactive`
fragt, bevor vorhandene Dateien überschrieben werden;
`-r bzw. -R bzw. --recursive`
löscht auch Dateien in allen Unterverzeichnissen und das Unterverzeichnis;

leere Verzeichnisse löschen (remove directory)

```
rmdir [Optionen] Verzeichnis
```

Optionen:
`-p bzw. --parents`
löscht auch Unterverzeichnisse im angegebenen Verzeichnis;

B.3 Datei suchen

nach Dateien suchen, die Bedingung(en) erfüllen (find)

```
find Pfadangabe(n) Bedingung(en)
```

Bedingungen sind erfüllt, wenn:
`-name Datei`
Datei mit Name Datei;
`-perm oooo`
Datei mit Zugriffsrechten, die gleich Oktalzahl `oooo` sind;
`-user Name`
Datei, die Benutzer `Name` gehört ;
`-group Gruppenname bzw. -nogroup Gruppenname`
Datei, die Gruppe `Gruppenname` gehört;
`-atime n`
letzter Zugriff vor n Tagen war;

in eigens mit `updatedb` angelegter Datenbank nach Dateiname suchen, der Muster enthält (locate)

```
locate [Optionen] Muster
```

in den in PATH angegebenen Verzeichnissen nach Kommandoname(n) suchen und ausgeben (whereis)

```
whereis [Optionen] Kommandoname(n)
```

B.4 Bearbeiten von Texten

Datei(en) nacheinander ausgeben (concatenate)

```
cat [Optionen] [Datei(en)]
```

Optionen:
`-n`
sämtliche Zeilen werden nummeriert
-v
alle Kontrollzeichen außer tab und newline werden angezeigt

suchen nach Begriffen (regulärer-Ausdruck) in Datei(en)

```
grep [Optionen] regulärer-Ausdruck [Datei(en)]
```

Optionen:
`-c`
gibt nur die Anzahl der Zeilen an, in denen das Suchmuster gefunden wurde, nicht aber die Zeile selbst;
`-f Dateiname`
liest die hier aufgezählten Optionen der angegebenen Datei (für komplexe oder häufig benötigte Suchmuster);
`-i`
Groß-/Kleinbuchstaben nicht unterscheiden;
`-l`
zeigt nur die Dateinamen an, in denen das Suchmuster gefunden wurde;
`-n`
gibt bei der Ausgabe jeder Zeile auch deren Zeilennummer an;
`-w`
findet nur ganze Worte;

Datei(en) seitenweise ausgeben

```
less [Optionen] Datei(en)
```

Optionen:
`-f`
erzwingt die Anzeige, auch von Dateien mit nichtdruckbaren Zeichen

Datei(en) seitenweise ausgeben

```
more Datei(en)
```

GZip-Archiv-Datei(en) nacheinander ausgeben (concatenate)

```
zcat [Optionen] [Datei(en)]
```

GZip-Archiv-Datei(en) seitenweise ausgeben

```
zless [Optionen] Datei(en)
```

GZip-Archiv-Datei(en) seitenweise ausgeben

```
zmore Datei(en)
```

Datei(en) sortieren

```
sort [Optionen] Datei(en)
```

Optionen:
`-c`
überprüft, ob die Datei sortiert ist oder nicht;
`-f`
behandelt Klein- und Großbuchstaben als gleichwertig;
`-o Ergebnisdatei`
schreibt das Ergebnis in die Ergebnisdatei. Die Datei darf mit der zu sortierenden Datei übereinstimmen;
`-r`
sortiert in umgekehrter Reihenfolge;
`-t[z]`
Zeichen z als Trennzeichen zwischen zwei Spalten verwenden;
`-b`
führende Leerzeichen ignorieren;

B.5 Prozeßverwaltung

Prozeß beenden, Standard Stignal-Nr ist 15 (gewaltsame Beendigung)

```
kill [-Signal] ProzeßNr
kill [-Signal-Nr] ProzeßNr
```

alle Prozesse mit Namen “Name” beenden

```
killall Name
```

zeigt Liste der laufenden Prozesse an

```
ps [Optionen]
```

Optionen:
`-a`
zeigt auch die Prozesse anderer Benutzer;
`-l`
zeigt diverse Zusatzinformationen (Speicherbedarf, Priorität etc.) an;
`-m`

zeigt ausführliche Zusatzinformationen speziell zum Speicherbedarf an;
`-u`
zeigt zusätzlich zur Prozeßnummer eine Menge weiterer Informationen an;
`-x`
zeigt auch Prozesse, die keinem Terminal zugeordnet sind;

beendet alle laufenden Prozesse und startet Rechner neu (nur Root)

```
reboot [Optionen]
```

beendet alle laufenden Prozesse und bleibt stehen (reagiert nicht mehr) (nur Root)

```
halt [Optionen]
```

Linux beenden

```
shutdown [Optionen] Zeitpunkt [Nachricht]
```

Zeitpunkt: Als Zeitpunkt muß entweder eine Uhrzeit (`hh:mm`), die Anzahl der Minuten gerechnet von der aktuellen Zeit (`+m`) oder das Schlüsselwort `now` (also sofort) angegeben werden.

Optionen:
`-c`
bricht einen bereits eingeleiteten shutdown-Vorgang ab (wenn noch möglich);
`-f`
wie -r, aber schneller;
`-h`
nach dem Herunterfahren des Systems wird es angehalten; nach der Meldung "`system halted`" kann der Rechner ausgeschaltet werden;
`-n`
führt den Shutdown besonders schnell aus (unter Umgehung der Init-V-Prozesse);
`-r`
nach dem Herunterfahren des Systems wird ein Neustart veranlaßt ;
`-t sekunden`
bestimmt, wie lange zwischen der Warnnachricht und dem Kill-Signal für die Prozesse gewartet werden soll (Default: 20 Sekunden);

B.6 Archive

Dateien (ent)packen mit GZip

```
gzip [Optionen] Datei(en)
```

Optionen:
`-d bzw. --decompress oder --uncompress`
dekomprimiert die Datei ;
`-r bzw. --recursive`
(de)komprimiert auch Dateien in allen Unterverzeichnissen;

Dateien entpacken mit GZip

```
gunzip [Optionen] Datei(en)
```

sichern von Dateien und Verzeichnissen (zusammenpacken, keine Komprimierung)

```
tar Aktion [Optionen] Datei(en)
tar Aktion [Optionen] Verzeichnis(se)
```

Aktionen:
`-c bzw. --create`
erzeugt ein neues Archiv, ein altes, gleichnamiges wird überschrieben;
`-d bzw. --diff bzw. --compare`
vergleicht die Dateien des Archivs mit den Dateien des Verzeichnisses und stellt Unterschiede fest;
`-r bzw. --append`
erweitert das Archiv um zusätzliche Dateien;
`--delete`
löscht Dateien aus dem Archiv;
`-t bzw. --list`
zeigt das Inhaltsverzeichnis des Archivs an;
`-u bzw. --update`
erweitert das Archiv um neue oder geänderte Dateien;
`-x bzw. --extract`
extrahiert die angegebenen Dateien aus dem Archiv und kopiert sie in das aktuelle Verzeichnis. Dateien werden dabei nicht aus dem Archiv gelöscht;

Optionen:
`-C Verzeichnis`
extrahiert die Dateien in das angegebene Verzeichnis;
`-f Datei`
verwendet die angegebene Datei als Archiv (statt Streamer zu verwenden);
`-v bzw. --verbose`
zeigt während der Arbeit alle Dateinamen am Bildschirm an;
`-T Datei bzw. --files-from Datei`
archiviert bzw. extrahiert die in der Datei angegebenen Dateinamen;
`-W bzw. --verify`

überprüft nach dem Schreiben die Korrektheit der gerade archivierten Dateien;
`-z bzw. --gzip`
komprimiert bzw. dekomprimiert das gesamte Archiv durch `gzip` (`*.tgz`-oder `*.tar.gz`-Dateien);
`-Z`
komprimiert bzw. dekomprimiert das gesamte Archiv durch `compress` (`*.Z`-Dateien);
`-I`
komprimiert bzw. dekomprimiert das gesamte Archiv durch `bzip2` (`*.BZ2`-Dateien);

RedHat Paket-Manager

```
rpm [Aktion] [Option] <Datei- oder Paketname>
```

Aktionen:
`- i oder --install`
installiert das angegebene Paket;
`-U oder --upgrade`
vorhandenes Binärpaket aktualisieren
`-V oder --verify`
überprüft, ob sich irgendwelche Dateien eines Pakets gegenüber der Originalinstallation verändert haben, Paketname muß angegeben werden;
`-e oder --erase`
entfernt ein vorhandenes Paket, Paketname muß angegeben werden;
`-q oder --query`
Abfragen starten

Optionen:
`--test`
Dummyausführung ohne Veränderungen durchzuführen;
`-- root <verzeichnis>`
Installationsort verändern,
`--nodeps`
Installation, auch wenn Abhängigkeiten nicht erfüllt sind;
`--noscripts`
automatische Ausführung von Installationsskripten wird unterbunden;
`--oldpackage`
neueres Paket durch älteres ersetzten;
`-a`
unsortierte Liste aller Pakete;
`-f <Datei>`
feststellen, zu welchem Paket eine Datei gehört;
`-p <Paketdatei>`
Informationen über ein noch nicht installiertes Paket ermitteln;

B.7 Administration des Dateisystems

konvertierung und Kopieren von Dateien

```
dd [Optionen=Wert]
```

Optionen:
`conv=[Modus]`
konvertiert beim Kopieren im Modus Modus;
`bs=[n]`
bestimmt die Blockgröße n für Ein- und Ausgabedatei;
`count=[n]`
kopiert nur n Blöcke;
`if=[Quelldatei]`
gibt die Quelldatei (anstatt der Standardeingabe) an;
`of=[Zieldatei]`
gibt die Zieldatei (anstatt der Standardeingabe) an;

überprüft die Konsistenz des Dateisystems und kann Reparaturen durchführen

```
fsck [Optionen] Gerätedatei
```

Optionen:
`-A`
alle in `/etc/fstab` genannten Dateisysteme überprüfen;
`-t Typ`
gibt den Typ des Dateisystems an (etwa ext2);
`-n`
beantwortet alle Rückfragen mit `n` (`no`), führt keine Änderungen durch;
`-p`
führt Reparaturen im Dateisystem ohne Rückfrage durch;
`-y`
beantwortet alle Rückfragen mit y (yes), führt Änderungen durch;

partitionieren von Festplatten

```
fdisk [Optionen] Gerätedatei
```

richtet auf einer zuvor mit fdformat formatierten Diskette oder auf einer mit fdisk partitionierten Festplatte ein Dateisystem ein

```
mkfs [Optionen] Gerätedatei [Blocks]
```

Optionen:
`-t Dateisystem`
gibt den Typ des Dateisystems an, muß als erste Option angegeben werden;
`-b [n]`
bestimmt die Blockgröße, n muß eine Zweier-Potenz größer gleich 1024 sein;
`-c`
führt vor dem Einrichten des Datenträgers einen Test durch, ob defekte Blöcke existieren;
`-i [n]`
gibt an, nach wie vielen Bytes jeweils ein I-Node eingerichtet wird (Standard 4096);
`-m [n]`
gibt an, wieviel Prozent des Datenträgers für Daten von root reserviert werden sollen (Std.: 5%);

richtet eine Festplattenpartition oder eine Datei als Swap-Bereich ein.

```
mkswap Gerätedatei
mkswap Datei
```

bindet Datenträger in das Dateisystem ein

```
mount [Optionen] Gerätedatei Mount-Point
```

Optionen:
`-r`
Mountet read-only;
`-t Dateisystem`
gibt das Dateisystem an;

B.8 Sonstiges

ausgabe von Text, Variablen ...

```
echo [Optionen] [Zeichenketten]
```

Optionen:
`-n`
Ausgabe nicht mit einem Neuzeile-Zeichen abschließen;
`-e`
aus der Programmiersprache C bekannte Escape-Sequenzen einschalten;

definiert eine neue Abkürzung bzw. zeigt eine vorhandene Abkürzung an

```
alias [Abkürzung [=Kommando]]
```

zeigt an, wie der verfügbare Speicherplatz (RAM und Swap-Speicher) genutzt ist

```
free
```

ermittelt, ob es sich bei Kommando um ein Befehl, ein Shell-Kommando oder um eine Alias-Abkürzung handelt

```
type Kommando
```

freien Speicherplatz auf Dateisystem(e) ausgeben

```
df [Optionen] [Dateisystem(e)]
```

Optionen:
`-t`
total;
`-i bzw. --inodes`
statt freien Speicher in kBytes werden Informationen über die verfügbaren I-Nodes angegeben;

gibt Informationen über den Speicherbedarf von Dateien bzw. von Verzeichnissen

```
du [Optionen] [Verzeichnis/Datei]
```

Optionen:
`-b bzw. --bytes`
zeigt die Größenangaben in Bytes (statt in kBytes) an;
`-c bzw. --total`
zeigt als abschließenden Wert die Endsumme an;
`-s bzw. --summarize`
zeigt nur die Endsumme an;
`-S bzw. --dereference`
zeigt nur den Speicherbedarf unmittelbar im Verzeichnis an;

B.9 Drucken

Dateien drucken (line printer); nur Linux (Unix: lp)

```
lpr [Optionen] [Datei(en)]
```

Optionen:
`-#[n]`
Datei(en) n-mal drucken;
`-P id`
an Drucker id drucken;
`-m`
mail schicken, wenn Druck beendet;

Druckerwarteschlange anzeigen; nur Linux

```
lpq [Optionen] [Benutzerkennung(en)]
```

```
Optionen:
-l
```
im Langformat anzeigen;
`+[n]`
alle n Sekunden anzeigen;
`-P id`
Warteschlange von Drucker id;

Druckauftrag löschen

```
lprm [-Pprinter] [-] [job # ] [user ]
```

Optionen:
`-Pprinter`
gibt den Drucker an, ansonsten Standard-Drucker;
`-`

wird ein einzelnes "-"-Zeichen angegeben, werden alle Druckaufträge des Benutzers gelöscht (beim Super-User alle Druckaufträge);
`job`
die Job-Nummer des Druckauftrages, der gelöscht werden soll;
`user`
alle Druckaufträge des Benutzers `<user>` löschen, nur als Super-User;

B.10 Eigentümer/Gruppe und Zugriffsrechte

Eigentümer der Datei(en) ändern (change owner)

```
chown [Neuer_Eigentümer] Datei(en)
```

Gruppenzugehörigkeit der Datei(en) ändern (change group)

```
chgrp [Neue_Gruppe] Datei(en)
```

Zugriffsrechte der Datei(en)/Verzeichnis(se) ändern (change mode)

```
chmod [optionen] Modus Datei(en)/Verzeichnis(se)
```

Der Modus kann in Symbol-Modus oder in Oktal-Zahl angegeben werden.

Optionen:
`-R bzw. --recursive`
Änderungen auch in allen Unterverzeichnissen;

Modus:

u	Eigentümer (user)
g	Gruppe (group)
o	alle anderen (other)
a	alle drei Benutzergruppen (all)
+	hinzufügen
-	wegnehmen
=	gleich nach folgendem Muster setzen
r	Leserecht (read); Oktal-Wert = 4
w	Schreibrecht (write); Oktal-Wert = 2
x	Ausführrecht (execute); Oktal-Wert = 1

Anhang C

Partitionierungsvorschläge

Die nachfolgenden Partitionierungsvorschläge sind größtenteils aus [Bauer et al, Seite 55 - 60] entnommen und geben nur Beispiels-Konfigurationen wieder, die einen kleinen Überblick über die Partitionsgrößen wiederspiegeln. Die Partitionen hängen besonders vom Einsatzgebiet des Systems und der installierten Software ab. Hier hilft häufig nur "*Learning by doing*".

1. Beispiel: Druckserver / Router
   - 80 - 200 MB Plattenplatz
     - 16 MB Swap und der Rest als / (Root-Partition)
     - kein X-Windows, sollte nur als `root` zugänglich sein
2. Beispiel: Standard-Arbeitsplatzrechner (klein)
   - 200 - 500 MB Plattenplatz
     - 32 - 40 MB Swap und der Rest als / (Root-Partition)
3. Beispiel: Standard-Arbeitsplatzrechner (Durchschnitt)
   - 500 - 1200 MB Plattenplatz
     - 64 MB Swap
     - 5 - 10 MB für `/boot`, 180 MB für / (Root-Partition), 100 MB für `/home`, der Rest für `/usr`
     - sollten Programme benutzt werden, die in `/opt` installiert werden (z.B. `KDE`, `Gnome`, `Netscape` ...), muß entweder eine `/opt`-Partition eingerichtet werden oder die Root-Partition vergrößert werden
4. Beispiel: Arbeitsplatzrechner (groß)
   - ab 1,2 GB Plattenplatz
     - keine pauschale Partitionierung
     - z.B. für eine 3,4 GB Festplatte
       * 128 MB Swap, 5 - 10 MB für `/boot`, 300 MB für / (Root-Partition), 2,2 GB für `/usr`, 600 MB für `/opt` und 100 MB für `/home`

Werden viele Daten in `/var` gespeichert, z.B. bei einem News- oder Mail-Server, sollte hierfür eine eigene Partition eingerichtet werden. Möchten Sie Swap-Partitionen größer 128 MB benutzen, müssen Sie diese auf mehrere Partitionen aufteilen, da die max. benutzte Größe einer Swap-Partition diese Grenze hat.

Anhang D

Übungsaufgaben

1. Erstellen Sie in einem eigens dafür angelegtem Verzeichnis die Unterverzeichnisse “klasse05a” bis “klasse09d”.
2. Ändern Sie die Gruppenzugehörigkeit dieses Verzeichnisses und der eben angelegten Unterverzeichnisse auf “nobody”.
   (a) Zusätzlich sollen für den Besitzer Schreib/Lese/Ausführ-Rechte, für die Gruppe Schreib/Ausführ-Rechte und für alle anderen Lese/Ausführ-Rechte gesetzt werden.
3. Suchen Sie alle Dateien mit der Endung .conf im aktuellen und allen untergeordneten Verzeichnissen .
4. Mehrere Backup-Dateien (Endung ~) befinden sich im aktuellen Verzeichnis und den Unterverzeichnissen. Wie könnten diese gelöscht werden, ohne in alle Verzeichnisse einzelnd hineinzuwechseln?
5. Geben Sie alle Zeilen der Apache-Konfigurationsdatei (`/etc/httpd/httpd.conf`) aus, die den String “`Directory`” enthält.
6. Speichern Sie das Inhaltsverzeichnis eines Verzeichnisses in eine Datei und erstellen Sie eine weitere Datei, wobei diesmal das Inhaltsverzeichnis nach der Dateigröße sortiert ist.
7. Speichern des Inhaltsverzeichnisses eines Verzeichnisses sowie das aktuelle Datum in eine Datei.
8. Wiederholen Sie die vorherige Aufgabe , nur das Datums soll diesmal im Format “Tag.Monat.Jahr (lang) Stunde:Minute:Sek” und Tag in Klammern (Ausgeschrieben) dargestellt werden.
9. Legen Sie eine Datei mit den kompletten Pfad- und Dateinamen von 3 Dateien an. Anschließend soll mit Hilfe dieser Datei die Gesamtzahl der Wörter in den 3 Dateien bestimmt werden.
10. Kopieren Sie alle Dateien mit der Endung `*.tex` nach `*.tex~`.
11. Erstellen Sie ein Shell-Skript “zeichentyp”, welches den Typ eines dem Script übergebenes Zeichens bestimmt.
12. Sie möchten alle x min eine Meldung erhalten, das Benutzer y angemeldet ist. Wie könnte hierfür das Shell-Script aussehen?
13. Nun möchten Sie noch alle x min eine Meldung erhalten, das Benutzer y nicht angemeldet ist. Welches Shell-Script wäre hierfür notwendig?

Anhang E

Lösungen der Übungsaufgaben

1. Erstellen Sie in einem eigens dafür angelegtem Verzeichnis die Unterverzeichnisse “klasse05a” bis “klasse09d”.
   `mkdir klasse0{5,6,7,8,9}{a,b,c,d}`

   (a) Ändern Sie die Gruppenzugehörigkeit dieses Verzeichnisses und der eben angelegten Unterverzeichnisse auf “nobody”.
   `chgrp -R nobody Verzeichnis`

   (b) Zusätzlich sollen für den Besitzer Schreib/Lese/Ausführ-Rechte, für die Gruppe Schreib/Ausführ-Rechte und für alle anderen Lese/Ausführ-Rechte gesetzt werden.
   `chmod -R 735 Verzeichnis`

2. Suchen Sie alle Dateien mit der Endung .conf im aktuellen und allen untergeordneten Verzeichnissen .
   `find . -name '*.conf'`

3. Mehrere Backup-Dateien (Endung ~) befinden sich im aktuellen Verzeichnis und den Unterverzeichnissen. Wie könnten diese gelöscht werden, ohne in alle Verzeichnisse einzelnd hineinzuwechseln?
   `rm $(find -name '*~')`

4. Geben Sie alle Zeilen der Apache-Konfigurationsdatei (`/etc/httpd/httpd.conf`) aus, die den String “`Directory`” enthält.
   `cat /etc/httpd/httpd.conf | grep Directory`

5. Speichern Sie das Inhaltsverzeichnis eines Verzeichnisses in eine Datei und erstellen Sie eine weitere Datei, wobei diesmal das Inhaltsverzeichnis nach der Dateigröße sortiert ist.
   `ls -l | tee inhalt 1 | sort + 4 > inhalt2`

6. Speichern des Inhaltsverzeichnisses eines Verzeichnisses sowie das aktuelle Datum in eine Datei
   `(ls;date) > inhalt`

7. Wiederholen Sie die vorherige Aufgabe , nur das Datums soll diesmal im Format “Tag.Monat.Jahr (lang) Stunde:Minute:Sek” und Tag in Klammern (Ausgeschrieben) dargestellt werden.
   `(ls; date '+%d.%m.%Y %H:%M:%S (%A)') > inhalt`

8. Legen Sie eine Datei mit den kompletten Pfad- und Dateinamen von 3 Dateien an. Anschließend soll mit Hilfe dieser Datei die Gesamtzahl der Wörter in den 3 Dateien bestimmt werden.
   `wc -w 'cat zaehle.txt'`

9. Kopieren Sie alle Dateien mit der Endung `*.tex` nach `*.tex~`
   `for i in *.tex; do cp $i $i~; done`

10. Erstellen Sie ein Shell-Skript “zeichentyp”, welches den Typ eines dem Script übergebenes Zeichens bestimmt.
```
if [ $# -ne 1 ] then
echo "falsche Anzahl von Argumenten"; exit 1
fi
case "$1" in
[0-9]) echo "Ziffer";;
[a-z]) echo "Kleinbuchstabe";;
[A-Z]) echo "Großbuchstabe";;
?) echo "weder Ziffer noch Buchstabe";;
*) echo "nur ein Zeichen erlaubt";;
esac
```

11. Sie möchten alle x min eine Meldung erhalten, das Benutzer y angemeldet ist. Wie könnte hierfür das Shell-Script aussehen?
```
while who | grep $2 >> /dev/null; do
echo "$2 angemeldet"; sleep $1
done
```

12. Nun möchten Sie noch alle x min eine Meldung erhalten, das Benutzer y nicht angemeldet ist. Welches Shell-Script wäre hierfür notwendig?
```
until who | grep $2 >> /dev/null; do
echo "$2 nicht angemeldet"; sleep $1
done
echo "$2 angemeldet"
```

Literaturverzeichnis

[Schmidt] Einführung in die Linux Konzepte; Achim Schmidt;

[Arzberger et al] Tabellenbuch Informations- und Telekommunikationstechnik; Paul Arzberger et al; Verlag Dr. Max Gehlen, 2. Auflage 1998

[Kofler] Linux: Installation, Konfiguration, Anwendung; Michael Kofler; Addision Wesley Longman Verlag, 3. Auflage 1998

[Herold] Linux - Unix Shells; Helmut Herold; Addision Wesley Longman Verlag, 3. Auflage 1999

[Breuer] dtv-Atlas zur Informatik, Tafeln und Texte; Hans Breuer; Deutscher Taschenbuch Verlag (dtv), April 1995

[Bauer et al] SuSE Linux 6.0, Installation, Konfiguration, Erste Schritte (Distributions-Handbuch); Bodo Bauer et al; SuSE GmbH, 13. aktualisierte Auflage 1998

[Borrotzu] vi - DER Unix Editor; Oliver Borrotzu; Zusammenstellung während der Umschulung zum Fachinformatiker bei der RAG Dortmund GmbH; 1999

Tabellenverzeichnis

Abbildungsverzeichnis

Nachwort

Ich hoffe, Ihnen hat diese Zusammenstellung der Grundlagen eines Linux-Systems gefallen und wünsche Ihnen viel Freude mit dem Betriebssystem Linux. Sollten Sie Fehler entdecken oder Verbesserungsvorschläge haben, bitte ich Sie, mir diese per eMail an die Adresse t.brinkert@15bit.de mitzuteilen.

weiterführende Literatur, die ich empfehlen kann:

Linux: Installation, Konfiguration, Anwendung von Michael Kofler, erschienen im Addison-Wesley Verlag. Ein sehr gutes Buch über Linux, nicht nur für Anfänger, das detailliert Programme und Werkzeuge beschreibt, mit denen Sie täglich arbeiten. Positiv an diesem Buch ist noch, daß die Konfigurationen für verschiedene Distributionen erläutert werden.

Linux - UNIX Shells: Bourne-Shell, Korn-Shell, C-Shell, bash, tcsh von Helmut Herold, erschienen im Addison-Wesley Verlag. Ausführliche Beschreibung der am weitest verbreiteten Shells in der UNIX- und Linux-Welt mit vielen Beispielen. Dieses Buch ist das dritte aus der Reihe: Linux/UNIX und seine Werkzeuge. Weitere Titel dieser Reihe sind Linux - UNIX Grundlagen, Linux - UNIX Profitools, Linux - UNIX Systemprogrammierung und die Linux - UNIX Kurzreferenz.

TCP/IP Netzwerk-Administration von Craig Hunt, Deutsche Übersetzung von Peter Klicman, erschienen im O'Reilly Verlag. Beschreibt umfassend die Einrichtung und den Betrieb von TCP/IP-Netzwerken. Neben den Protokoll-Grundlagen werden fortschrittliche Routing-Protokolle, das Softwarepaket gated, Einführungen zur Konfiguration wichtiger Netzwerkdienste wie PPP, SLIP, sendmail, DNS (BIND), BOOTP, DHCP, NIS und NFS angesprochen. Zusätzliche Themen sind Fehlersuche und Sicherheit.

Apache: Das umfassende Referenzwerk von Ben Laurie & Peter Laurie, Deutsche Übersetzung von Peter Klicman, erschienen im O'Reilly Verlag. Die grundlegende Dokumentation für den beliebtesten Webserver der Welt. Behandelt werden sowohl die Unix- als auch die Win32-Distribution von Apache.

weiterführende Links:

www.15bit.de: Informatikseite des Autors, derzeit ist ein Script zur Konfiguration von DNS und BIND v4 verfügbar.

www.hamburger-bildungsserver.de: Hamburger Bildungsserver mit einer sehr guten Mailingliste zum Thema Linux

www.linux.de: Die Linux-Seite im Internet

www.linuxinfo.de: Informationen zu Linux mit Dokumentationen zu verschiedenen Programmen bzw. Anwendungsgebieten. Ein Newsletter kann abonniert werden

www.linuxbu.ch: Das Buch: **Linux im Windows-Netzwerk** aus dem Franzis-Verlag online. Zu diesem Buch gibt es auch eine Mailingliste.

www.pro-linux.de: Seite mit täglichen Neuigkeiten zum Thema Linux, die Überschriften der Neuigkeiten können als Newsletter abonniert werden.

www.64-bit.de: Sehr gute Sammlung von Linux-Dokumentationen.

www.suse.de: Internetauftritt der SuSE-Distribution

www.oreilly.de: Seiten des O'Reilly Verlages. Einige Bücher des Verlages sind auf diesen Seiten auch komplett als HTML-Datei verfügbar.

http://skyper.tmag.de/oreilly/books: 29 Bücher aus dem O'Reilly Verlag Online (in Englisch), Themen: Unix, Netzwerke, Perl, Java und Web.