Linux Einführung

Letzte Aktualisierung: 26. April 1999

Inhaltsverzeichnis

Kapitel 1

Entwicklung

Setzt man sich heute vor einen Rechner, startet ein Programm, wartet einige Sekunden, bis dieses endlich die erhofften Resultate präsentiert, dann blättert man alsbald durch Prospekte der Computer–Discounter und liebäugt mit neuer, leistungsfähigerer Technik.
Mit keinem Gedanken erinnert man sich an die monströsen Ungetümer aus Röhren und Transistoren im Turnhallenformat, die noch in den 70iger Jahren die Leistung kleiner Kraftwerke erforderten, um letztlich nur einen Bruchteil an heutiger Rechenleistung zu vollbringen.
Zu jener Zeit wurden die "Rechengenies" noch per Hand mit Daten gefüttert, vom Administrator, der den Lochkartenstapel einlegte und nach Stunden oder gar erst Tagen die Resultate ebenfalls in Form von Lochkarten entnahm. Ein *Betriebssystem* kannte und brauchte man damals noch nicht.
Die Rechner wurden kleiner, paßten schon in einen einzigen Raum, ihre Verarbeitungsgeschwindigkeit stieg und übertraf bald die Fähigkeit des Administrators, die Magnetbänder rechtzeitig zu wechseln. Es wurde Zeit, den Ballast der reinen Stapelverarbeitung (ein Job nach dem anderen) aufzugeben und zur Timesharing–Verarbeitung überzugehen. Man erweiterte den damals sündhaft teuren Arbeitsspeicher, so daß mehrere Jobs gleichzeitig rechenbereit auf die CPU warten konnte und teilte jedem Job eine genau festgelegte Zeitscheibe an CPU–Zeit zu. Von einer Interaktion mit dem Nutzer war man noch meilenweit entfernt.
MULTICS (Multiplexed Information and Computing System) wurde vom MIT, den Bell Labs und General Electrics ins Leben berufen, in der Vorstellung, ein Betriebssystem für eine Maschine zu entwickeln, die über soviel Rechenkapazität verfügen sollte, um alle Einwohner von Boston bedienen zu können. Auch wenn das Projekt wegen seiner Komplexität zum Scheitern verurteilt war, brachte es für das spätere Verständnis des Aussehens von Betriebssystemen wertvolle Erkenntnisse.
1969 schließlich griffen Ken Thompson und Denis Ritchie, zwei Mitentwickler von MULTICS, die Idee wieder auf und verwirklichten Teile davon in UNICS.
Grundlegende Überarbeitung erfuhr das Betriebssystem durch Forschungen an amerikanischen Universitäten, nachdem AT&T die Source–Lizenzen preisgünstig an diese abgab (der kommerzielle Vertrieb war AT&T zunächst per Gerichtsbeschluß untersagt worden).
Erstes kommerzielles Produkt war UNIX Version 7, das unter den Namen Sy-

stem V vertrieben wurde. Es folgten zahllose weitere UNIX–Derivate mit mehr oder minder großem Erfolg (AIX vom IBM, HP–UX von Hewlett Packard, Xenix von Microsoft, Sinix von Siemens, Solaris von SUN, BSD der Berkely University, Parix von Parsytec, ...).
Professor Andrew S. Tanenbaum implementierte 1987 ein zu UNIX Version 7 kompatibles System mit dem Namen **Minix**. Es diente ihm als Lehrsystem für seine Studenten und wurde auch im Quelltext für ein gewisses Entgeld verfügbar. Größtes Manko von Minix waren die durch den Autor beschränkten Erweiterungsmöglichkeiten im Kernel, so daß z.B. das X–System niemals unter Minix laufen kann.
Um seinen 386er genauer zu inspizieren, schrieb ein finnischer Student namens Linus Torvalds im Jahr 1991 einige Assemblerroutinen, die schließlich ein minimales Betriebssystem formten. **Linux** erblickte das Licht der Welt und da Torvalds den Sourcecode an interessierte Minixer verschickte, fanden sich bald weltweit Interessenten zusammen, die fortan über das Internet kommunizierten und ihre neuesten Erweiterungen zu Linux verbreiteten.
Von Anfang an stellte Torvalds seine Sourcen unter die Verantwortung der GPL[1], so daß diese frei kopiert werden konnten und jedem Interessenten zur Verfügung standen. Gleichzeitig war man bei der Entwicklung auf Konformität zum POSIX–Standard bedacht, wodurch Linux ohne großen Aufwand auf andere Hardware–Plattformen portierbar wurde.

[1] GNU General Puclic Licence

Kapitel 2

Erste Schritte

2.1 Login

Linux (und UNIX im Allgemeinen) ist ein Mehrbenutzersystem, d.h. gleichzeitig können theoretisch unbegrenzt viele Nutzer mit nur einem System arbeiten. Um Dateien eindeutig einem Nutzer zuordnen zu können, benötigt Linux Informationen über diesen, d.h. ein Nutzer muß sich beim System anmelden:

```
login:
```

Ein Nutzer könnte sich leicht als ein anderer ausgeben, z.B. als *root*, einer Nutzerkennung, die auf jedem Unix–System existiert. Um vor Mißbrauch zu schützen, muß dieser sich dem System gegenüber auch ausweisen, ein geheimes Paßwort sorgt für den Schutz:

```
Password:
```

Stimmen Nutzerkennung und Paßwort überein, gelingt das Anmelden am System und der Nutzer landet in seinem Heimatverzeichnis. Linux startet eine *Shell*, einen Kommandointerpreter, der die Interaktion zwischen Nutzer und System steuert: die Shell nimmt Eingaben vom Nutzer entgegen, interpretiert diese und veranlaßt das Betriebssystem zu entsprechenden Aktionen...

2.2 Logout

Unter DOS war es noch üblich, den Ausschalter zur Beendigung einer Sitzung am Rechner zu nutzen. Auch unter Linux dient der Ausschalter dem endgültigen Knock–out des Rechners; zuvor jedoch ist ein ordnungsgemäßes Herunterfahren des Systems dringend zu empfehlen. Der Grund hierfür ist in erster Linie die Verfahrungsweise beim Zugriff auf dauerhafte Speichermedien, wie z.B. die Festplatte. Aus Performancegründen werden Schreibzugriffe auf diese nicht sofort durchgeführt, sondern erst bei Bedarf bzw. nach Ablauf einer voreingestellten Zeitspanne (meist 30 sec). Somit werden Systembus und DMA–Controller entlastet. Würde der Ausschalter Linux in die Jagdgründe befördern, wären die Daten für immer verloren, aber zum Glück existieren mehrere Möglichkeiten, dem System einen anständigen Abtritt zu ermöglichen:

```
root@sonne > reboot
root@sonne > shutdown -r now
```

booten den Rechner neu.

```
root@sonne > halt
root@sonne > shutdown -h now
```

hält das System dagegen an.

Obige Befehle bedürfen aber der Berechtigung des Administrators (also root), dem gewöhnlichen Nutzer steht bestenfalls (falls nicht vom Administrator unterbunden) die Tastenkombination *CTRL–ALT–DEL* zur Verfügung, die ein *reboot*[1] bewirkt.

2.3 Virtuelle Konsolen

Linux ist ein Multitasking–System und bietet damit mehrere Möglichkeiten, Kommandos gleichzeitig auszuführen. Mit den Tastenkombinationen (ALT)(Fn) ($n = 1..6$) läßt sich zwischen den sogenannten virtuellen Konsolen wechseln. Auf diesen Konsolen erscheint ein Login–Prompt und man kann auf jeder Konsole eine Sitzung eröffnen. So kann man auf der ersten Konsole ein Programm editieren und dieses auf der nächsten Konsole compilieren, ohne den Editor beenden zu müssen. Zu jeder Zeit kann natürlich nur eine physische Konsole (Bildschirm) aktiv sein; um die Ausgaben der einzelnen Programme regulieren zu können, werden alle "normalen" Ausgaben auf der virtuellen und nur Fehlermeldungen auf dem Bildschirm ausgegeben.
Arbeitet man unter dem X–Windows–System, erreicht man eine der Textkonsolen über die Tastenkombinationen (CTRL)(ALT)(Fn). Der X–Server selbst ist auf die 7.Konsole aktiv, zu der man über (ALT)(F7) gelangt.

2.4 Verzeichnisse und Dateien

Anmerkung: In allen weiteren Beispielen sind Kommandofolgen, die die Rechte des Systemadministrators **root** erfordern, symbolisch durch das Prompt:

```
root@sonne >
```

dargestellt; Kommandofolgen, die dem "gewöhnlichen" Nutzer zur Verfügung stehen, werden durch:

```
user@sonne >
```

eingeleitet.

Dateien dienen zum Speichern von Daten. Verzeichnisse gruppieren die Dateien in einer baumartigen Struktur. Ein Baum besitzt (falls er nicht gefällt wurde :-) eine Wurzel, also ist der Ursprung aller Verzeichnisse das *root*–Verzeichnis "/". Mit

```
user@sonne > cd /
```

[1] "reboot" ist die Default–Einstellung, der Administrator kann zum Beispiel mittels "YaST" ein anderes Verhalten einstellen.

wechselt man in das Wurzelverzeichnis (*cd – change directory*) [2], wo ein Listing (*ls – list*) auf den meisten Systemen die folgende Ausgabe liefern sollte:

```
System.map  cdrom  lib         proc      tmp       vmlinuz.new
System.old  dev    lost+found  projects  usr       vmlinuz.old
bin         etc    mnt         root      var       windoof
boot        home   opt         sbin      vmlinuz
```

Wollen wir wieder nach Hause (`/home/user`), geben wir einfach

```
user@sonne > cd
```

ein und landen in unserem Home–Verzeichnis. Das Kommando `cd` wertet im Falle fehlender Argumente (oft wird auch von Optionen gesprochen) die Shell–Variable `HOME` aus, die den Pfad zum entsprechenden Verzeichnis enthält. Den Inhalt der Shell–Variablen kann man sich mittels des Aufrufes

```
user@sonne > echo $HOME
```

betrachten. Weitere wichtige Shell–Variablen seien an dieser Stelle kurz erwähnt:

`DISPLAY`	Name des Displays der Standardausgabe
`HOSTNAME`	Name des Rechners
`MANPATH`	Suchpfade für Manual pages
`PATH`	Suchpfade für Programme
`SHELL`	enthält den Namen der Shell (z.B. bash)
`USER`	Nutzerkennung

Gibt man

```
user@sonne > echo $(TAB)(TAB)
```

ein, erhält man eine Auflistung aller Shell–Variablen. `echo` schreibt sein Argument auf die Standardausgabe, vergißt man das "$"–Zeichen, würde `echo` den Namen der Shellvariablen wiedergeben.

```
user@sonne > mkdir new_directory
```

erstellt ein neues Verzeichnis *new_directory*[3]. Betrachtet man dessen Inhalt:

```
user@sonne > ls -a new_directory
```

findet man – sofern man sich nicht im Wurzelverzeichnis befindet – "." für das Verzeichnis selbst und "..", was auf das übergeordnete Verzeichnis verweist. Möchte man schließlich das Verzeichnis wieder löschen, trommelt man

```
user@sonne > rmdir new_directory
```

auf die Tastatur und vermißt beim nächsten Listing das Verzeichnis, sofern dieses *leer* war! Das zum Löschen von Dateien gedachte Kommando `rm` – *remove* kann auch auf Verzeichnisse angewendet werden:

[2] Unter Unix werden Befehle von Optionen grundsätzlich getrennt; im Unterschied zu DOS muß beim Wechsel ins übergeordnete Verzeichnis ein Leerzeichen zwischen cd und .. stehen.

[3] Die Berechtigung dazu hat der Nutzer nur in seinem Home–Verzeichnis und in /tmp!

```
user@sonne > rm -r new_directory
```

Mit der Option "-r" (für rekursiv) entfernt `rm` auch alle Unterverzeichnisse und letztlich das Verzeichnis selbst.
Erinnert man sich des Namens des aktuellen Verzeichnisses nicht, hilft

```
user@sonne > pwd
```

(print working directory), das den Namen (mit Pfad) wiedergibt.

Wenden wir uns nun den Dateien zu. Wie das Löschen funktioniert, haben wir schon kennengelernt: mit dem Kommando `rm`. Zum Erstellen neuer Dateien gibt es mehrere Möglichkeiten. Die Einfachste ist

```
user@sonne > touch Dateiname
```

das eine leere Datei anlegt, sofern der Dateiname noch nicht im aktuellen Verzeichnis existierte. Gab es sie bereits, ändert `touch` ihr letztes Änderungs– und Zugriffsdatum; der Grund hierfür wird uns im Zusammenhang mit `make` klar werden.
Ausgaben von Kommandos lassen sich in eine Datei umlenken, zum Beispiel legt

```
user@sonne > ls / > root_inhalt
```

eine Datei *root_inhalt* mit folgendem Inhalt an:

```
System.map  cdrom  lib         proc      tmp       vmlinuz.new
System.old  dev    lost+found  projects  usr       vmlinuz.old
bin         etc    mnt         root      var       windoof
boot        home   opt         sbin      vmlinuz
```

Den Inhalt einer solchen Datei schaut man sich am besten mit einem sogenannten Pager an:

```
user@sonne > less root_inhalt
```

oder mit

```
user@sonne > more root_inhalt
```

Pager haben gegenüber Editoren (diese behandeln wir später) den Vorteil, daß man nicht aus Versehen den Inhalt der Datei ändern kann. Ob man `more` oder `less` bevorzugt, ist Geschmackssache; beide Pager unterscheiden sich nur geringfügig in der Bedienung[4] (beide beendet man durch Q).
Für sehr lange Dateien interessiert man sich meist nur für die ersten/letzten Zeilen (z.B. werden in `/var/log/messages` alle Systemmeldungen protokolliert, wobei man den Grund für ein ungewöhnliches Systemverhalten sicher am Ende der Datei finden wird).

```
user@sonne > head Dateiname
```

zeigt die ersten (Voreinstellung 10) Zeilen der Datei *Dateiname* an,

[4] Tatsächlich beziehen sich unter Linux beide meist auf einunddasselbe Programm "less".

```
user@sonne > tail Dateiname
```

dagegen die letzten (10) Zeilen.
Für das Betrachten kurzer Dateien (deren gesamter Inhalt auf einer Bildschirmseite darstellbar ist) eignet sich auch folgendes Kommando:

```
user@sonne > cat /etc/passwd
```

`cat` schreibt in diesem Beispiel den Inhalt der Paßwortdatei auf die Standardausgabe.
Es soll nützlich sein, Dateien unter Umständen kopieren zu können. Auch an ein solches Kommando haben die UNIX–Götter gedacht:

```
user@sonne > cp quelle ziel
```

quelle und *ziel* sind nun zwei Dateien mit identischem Inhalt.
Eine andere Möglichkeit zum Kopieren bietet die Umleitung von Ausgaben:

```
user@sonne > cat quelle > ziel
```

Aber warum sollte man Dateien mit gleichem Inhalt mehrfach speichern? Einziger Grund hierfür kann doch nur der Zugriff auf diesselbe Datei aus verschiedenen Verzeichnissen heraus sein. Jedoch wird hiermit Speicherplatz verschwendet. Abhilfe schaffen hier sogenannte *Links*, also Verweise auf Dateien/Verzeichnisse.

```
user@sonne > ln quelle ziel
user@sonne > ln -s quelle ziel2
```

In der ersten Zeile wird ein sogenannter statischer (fester) Link angelegt. *ziel* ist jetzt nur ein anderer Name für *datei*; eine Änderung von *ziel* ändert auch den Inhalt von *datei*. Wird einer der Dateinamen gelöscht, kann über den anderen Namen weiter auf den Inhalt zugegriffen werden.
Anders verhält sich der Link der zweiten Zeile, ein sogenannter symbolischer Link. Im Unterschied zum statischen Link, wo der Verwaltungseintrag (I–Node, siehe Seite 24) der Ursprungsdatei kopiert wird, speichert der symbolische Link den Speicherplatz des I–Nodes der Datei. Löscht man nun *quelle*, existiert *ziel2* weiter, zeigt aber ins "Nichts" (auf einen nicht mehr existierenden I–Node), ist sozusagen unbrauchbar geworden. Der Vorteil symbolischer Links besteht in der Möglichkeit, ihn auch auf Verzeichnisse anwenden zu können (Der Superuser darf dies auch für feste Links!). Ein Beispiel erläutert den Sachverhalt in Abschnitt 3.6.
Das Umbenennen bzw. Verschieben von Dateien erfolgt mit dem Kommando `mv` – *move*:

```
user@sonne > mv quelle ziel
```

Die Datei *quelle* erhält den neuen Namen *ziel*.

2.5 Hände weg von fremdem Eigentum

Vielleicht hat man bei den obigen Beispielen schon Bekannschaft mit Ausgaben wie `permission denied` geschlossen. Wenn nicht, wechselt man z.B. ins root–Verzeichnis und versucht einmal eine Datei in diesem anzulegen.

Der Grund hierfür ist, daß alles (Prozesse, Dateien, Verzeichnisse...) in Linux einen Besitzer hat und dieser bestimmt, wer was damit tun und wer gefälligst was zu unterlassen hat. Ausnahme ist hier wieder einmal der Oberboß – also *root* –, der über alles sein Zepter schwingt und die üblichen Gesetze zu seinem Gunsten außer Kraft setzen kann. Betrachten wir drei typische Einträge aus einem Verzeichnis, so wie sie vom Kommando `ls -l` dargestellt werden:

```
user@sonne > ls -l
lrwxrwxrwx   1 root  root       8 Aug 11 18:40 asm -> asm-i386
-rw-------   1 user  users   8139 Oct 13 07:53 av2.txt
drwxr-xr-x  16 user  users   1024 Oct 19 21:39 tmp
```

Interessant ist hier die erste Kolonne, die die jeweiligen Zugriffsrechte angibt. Das erste Zeichen gibt dabei den Typ der Datei an:

–	alles, außer das Nachfolgende
b	blockorientiertes Gerät (Device, siehe Seite 19)
c	zeichenorientiertes Gerät
d	Verzeichnis
l	symbolischer Link (Verweis)
p	FIFO–Datei (pipe)

Es folgen jeweils drei Spalten, die die Rechte des Eigentümers, der Gruppe und der anderen (in der angegebenen Reihenfolge) bezeichnen:

r	Datei darf gelesen werden
w	Datei darf geschrieben werden
x	Datei darf ausgeführt werden (Binary, Shell–Skript)

Theoretisch sind alle Kombinationen der Zugriffsrechte denkbar, aber der Eigentümer einer Datei darf diese immer lesen, selbst wenn das Lesebit nicht gesetzt ist. Genauso macht es wenig Sinn, eine gewöhnliche Textdatei als ausführbar zu setzen; die Shell wird damit nichts anfangen können. Ein `x` für ein Verzeichnis gibt an, daß in dieses gewechselt werden kann.
Interessant ist auch ein Konstrukt folgender Art:

```
-rwx---rwx
```

, das angibt, daß mit der Datei alles angestellt werden kann, außer für Nutzer derselben Gruppe. Versucht irgendjemand, der nicht der Gruppe des Eigentümers angehört, die Datei zu modifizieren, wird ihm dies gelingen, einem Gruppenmitglied bleibt dies versagt, obwohl er ja gleichzeitg ein "anderer" ist. D.h. die Rechte der Gruppe sind verbindlicher, als die Rechte der anderen!

2.6 Hilfe

Unixe verfügen von Haus aus über ein ausgereiftes Hilfesystem, den Manual Pages. Zu nahezu jedem Kommando findet man dort eine ausführliche Beschreibung und jeder Programmierer ist dazu aufgefordert, zu seinen Programmen ebenfalls Manuals zu erstellen.

```
user@sonne > man ls
```

zeigt eine kompakte Beschreibung des Kommandos `ls` mit all seinen Optionen, Argumenten, Informationen zu bekannten Problemen, Verweise zu verwanden Kommandos...
Selbstverständlich gibt es zur Bedienung von `man` selbst ein Manual.

Andere Informationsquellen sind die *info*–Seiten, die man durch Eingabe von

```
user@sonne > info
```

erreicht. Zuerst sollte man sich die Hilfe zu `info` betrachten, da die Bedienung doch eher seltsam ist.

In den *HOWTO's* findet man Informationen zur Installation, Konfiguration, zu von Linux unterstützter Hardware usw. Am besten schaut man sich die HTML–Version der HOWTO's an (unter X). Die komprimierten Quellen finden sich im Verzeichnis `/usr/doc/howto/` und lassen sich z.B. mit dem Pager `less` betrachten.
Viele Programmpakete bringen darüber hinaus eine eigene umfangreiche Dokumentation mit, die vor allem Neuerungen, Installationshinweise und Fehlerbehandlungen umfassen. Die entsprechenden Dateien finden sich meist in den jeweiligen Unterverzeichnissen in `/usr/doc/packages`.

2.7 Editoren

Unix verfügt über eine Fülle von Editoren. Der Vater aller Editoren ist der `ed`. Aber nur für masochistisch veranlagte Menschen wird der zeilenweise arbeitende Editor heute noch interessant sein.
Viele mögen über die Benutzer des `vi` ähnlich denken, jedoch ist der `vi` unbestritten einer der schnellsten Editoren überhaupt und zumindest in geklonter Form auf jedem Unix–System verfügbar. Das macht ihn zu dem Standardeditor für Systemadministratoren. Im Anhang B findet man eine ausführliche Darstellung.

Komfortabler (und leider auch langsamer) ist der `emacs` bzw. sein grafischer Verwander der `xemacs`. Der "bunteren" der beiden Varianten werden wir uns in einem eigenen Kapitel 11 zuwenden.

Kapitel 3

Das Dateisystem

Ein Exkurs durch das *ext2fs*–Dateisystem soll uns vor allem die Verzeichnisstruktur näher bringen, aber auch Informationen über die Unterstützung anderer Filesysteme (msdos, vfat, iso9660) liefern. Nicht zu vergessen sind Kenntnisse über die Besitzverhältnisse und Zugriffsrechte, die zu den wesentlichsten Merkmalen von Unix–Dateisystemen zählen.

3.1 Dateien und Verzeichnisse

Wichtigste Merkmale des Linux–Filesystems *ext2fs* sind:

- Dateinamen können bis zu 255 Zeichen lang sein.
- Groß– und Kleinschreibung werden unterschieden.
- Alle Zeichen sind erlaubt (aber auf Sonderzeichen sollte man besser verzichten). Es ist sogar möglich, einen aus Leerzeichen bestehenden Dateinamen zu bilden:

```
user@sonne > touch "   "
user@sonne > ls -l
-rw-r--r--   1 user   users         0 Nov  6 07:33
user@sonne > rm "   "
```

Eine Dateinamenserweiterung wie unter DOS existiert in Linux nicht; der Punkt ist ein ganz normales Zeichen. Dennoch verwenden Archivierungs– und Komprimierungsprogramme wie *tar* und *gzip* derartige Suffixe `.tar, .gz`[1].

3.2 Jokerzeichen

Um sich nur Dateien mit einem bestimmten Namensmuster anzuschauen, verwendet man sogenannte Jokerzeichen. Bevor wir uns Beispielen zuwenden, listen wir die Möglichkeiten zur Suchmusterangabe auf:

[1] Auch ohne die entsprechenden Dateiendungen erkennen die meisten Programme das jeweilige Format, Ausnahme ist z.B. der C–Compiler *gcc*, der als Eingabe eine **.c* Datei erwartet.

?	genau ein beliebiges Zeichen
*	beliebig viele (auch 0) beliebige Zeichen
[def]	eines der Zeichen
[^czx]	keines der angegebenen Zeichen
[!czx]	wie oben
[a-f]	alle Zeichen aus dem Bereich
~	steht für das Heimatverzeichnis

```
user@sonne > ls /*.map
/System.map

user@sonne > ls -d /?[abc]*
/sbin  /var

user@sonne > ls /?[abc]*
/sbin:
SuSEconfig  fsck.minix  ldconfig  rmmod
YaST        genksyms    lilo      route ...

/var:
X11R6  cron    lock  named    spool   texfonts
adm    games   log   openwin  squid   tmp
```

Betrachten wir folgendes Beispiel:

```
user@sonne > rm -r *~
```

Die Option `-r` weist das Kommando `rm` an, rekursiv in allen Unterverzeichnissen Backup–Dateien (Endung ~) zu löschen. Wir erwarten jetzt, das tatsächlich auch in allen Unterverzeichnissen die Dateien entfernt wurden, stellen aber fest, daß dies nur im aktuellen Verzeichnis geschehen ist: Der Grund dafür ist die Shell, die für die Auflösung der Jokerzeichen verantwortlich ist. Die Shell betrachtet also nur *~ und findet entsprechende Muster nur im aktuellen Verzeichnis. `rm` bekommt also nur eine Liste dieser Dateien!
Genau aus diesem Grund funktioniert in Linux (im Unterschied zu DOS) ein Kommando wie

```
user@sonne > cp *.x *.y
```

niemals. Die Shell expandiert zwar alle *.x, kann aber *.y nicht auflösen und übergibt dieses unverändert an das Kommando `cp`, welches damit aber nichts anzufangen vermag.

3.3 Verzeichnisstruktur

Bei der Unmenge von Dateien in Unix–Systemen (auf meinem System sind es 106839!) ist eine hierarchische Struktur unabdingbar. Lange Zeit hat jedes Unix–Derivat seine eigenen Vorstellungen vom Aufbau seiner Dateiverwaltung mitgebracht, aber unterdessen ist man sich zumindest in der Linux–Gemeinde mehr oder weniger einig geworden und erarbeitete einen Standard, der wichtige Strukturen definiert. Festgelegt wurden Richtlinien bzgl. der gemeinsamen

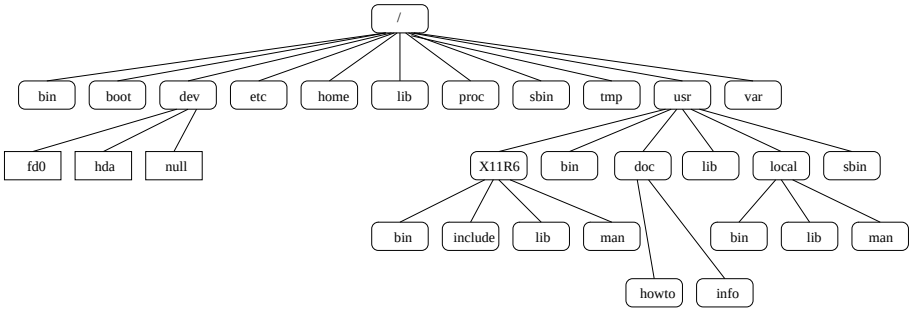

Abbildung 3.1: Die Linux–Verzeichnisstruktur (Ausschnitt)

Nutzung über das Network File System (NFS), der Unterteilung in die üblichen Kategorien (Programme, Dokumentation, Konfigurationsdateien...), sowie der Unterteilung der Konfigurationsdateien in lokale und netzweit verteilte Daten. Sobald der Kernel aktiv ist, lädt er als erstes das Root–Filesystem, in dem alle für die Aufgaben des Kernels notwendigen Programme und Konfigurationsdateien angesiedelt sein müssen.
Zu den Programmen gehören:

- Dienstprogramme zum Prüfen und Reparieren des Dateisystems
- Programme zum Sichern der Systemdaten und zur Installation neuer Systemteile
- eventuell wichtige Netzwerkprogramme

`/bin`

Die wichtigsten Kommandos zur Systemverwaltung findet man hier, sie dürfen von allen Nutzern ausgeführt werden.

`/boot`

Hier findet man die Dateien des Bootmanagers `lilo` und die Kernel. In früheren Linux–Versionen wurden der "Haupt"–Kernel im Root–Verzeichnis installiert und nur die optionalen Kernel in diesem Verzeichnis.

`/dev`

In diesem Verzeichnis stehen die Gerätedateien (Devices), die die gesamte Hardware beschreiben (Festplatte, Floppy, RAM...), sowie einige Devices mit speziellen Aufgaben.

Drei Informationen sind für jedes Device relevant:

1. *Typ des Zugriffs*
   - blockorientiert (b), gepufferter Zugriff, z.B. Festplatten
   - zeichenorientiert (c), ungepufferter Zugriff, z.B. Drucker über die parallele Schnittstelle

2. *Major Device Number* gibt die Nummer des zu verwendenden Treibers an (siehe `/usr/include/linux/major.h`).

3. *Minor Device Number* zur weiteren Unterscheidung der Treiber. Z.B. haben alle Diskettenlaufwerke dieselbe Major Device Number; durch die Minor Device Number kann der Typ (Zollgröße, Format) konkretisiert werden.

Ein etwas ungewöhnliches Device ist `/dev/null`, der Mülleimer von Linux. Möchte man z.B. bei der Suche nach Dateien Fehlermeldungen (permission denied) vermeiden, hilft Folgendes:

```
user@sonne > find / -name "*" 2> /dev/null
```

Zur Erklärung des Mechanismus schaue man auf Seite 59.

Wichtige Gerätedateien sind:

cdrom	Link auf eine entsprechende Datei (z.B. cdu535)
cua*	Serielle Schnittstellen (ausgehende Verbindungen)
fd*	Diskettenlaufwerke
hd*	IDE–Festplatten
kmem	Speicherauszug (core)
lp*	Parallele Schnittstellen
mouse	Link auf die entsprechende Datei
port	IO–Ports
sd*	SCSI–Festplatten
tty*	Terminalkonsolen
ttyS*	Serielle Schnittstellen (eingehende Verbindungen)

`/etc`

enhält alle globalen Konfigurationsdateien (Tastatur, X, Netzwerk...)[2].

`/home`

Alle Heimatverzeichnisse der Nutzer findet man hier. Nach dem Login landet jeder Benutzer in seinem Home. Heimatverzeichnisse können vom Systemverwalter auch an anderer Stelle angesiedelt werden.

`/lib`

Einige gemeinsam genutzte Bibliotheken (bzw. Links darauf) stehen hier. Diese Bibliotheken werden von den Programmen während des Systemstarts benötigt.

`/lost+found`

Wurde Linux unsachgemäß beendet, ist oft eine Reparatur des Dateisystems notwendig. Daten, die dabei nicht eindeutig zugeordnet werden konnten, werden in diesem Verzeichnis gesammelt.

[2]Leider sind sich im Aufbau dieses Verzeichnisses die verschiedenen Distributionen nicht einig, so stehen die X–spezifischen Dateien bei *SuSE* direkt in /etc, während *Redhat* diese in einem eigenen Unterverzeichnis /X11 hält.

`/mnt`

Dient als mount–Point für zusätzliche Dateisysteme (ist normalerweise leer).

`/opt`

Programme, die nicht zum üblichen Installationsumfang von Linux gehören, werden oft unter diesem Zweig installiert. So werden nahezu alle kommerziellen Softwarepakete hier eingerichtet; auch die Programme zur KDE befinden sich hier.

`/proc`

Ist sozusagen das Laufzeitsystem von Linux. Hier werden Informationen zu allen laufenden Prozessen, zur Hardware (CPU, Interrupts...) gehalten. Die Dateien dieses Verzeichnisse existieren einzig im Hauptspeicher. Der Aufbau dieses speziellen Dateisystems wird auf Seite 35 detailliert behandelt.

`/sbin`

Kommandos zur Systemverwaltung, die nur von *root* ausgeführt werden dürfen.

`/tmp`

Temporäre Dateien können hier abgelegt werden, jeder Nutzer ist dazu berechtigt.

`/usr`

Ein Blick in dieses Verzeichnis offenbart eine starke Ähnlichkeit mit dem Root–Filesystem, nur befindet man sich eine Privilegierungsebene tiefer. Hier sind alle Anwendungsprogramme angesiedelt, sowie das X–System, LaTeX, ..., also alles, was dem gewöhnlichen Nutzer zur Arbeit mit dem System genügen muß.

`/var`

Hier stehen die veränderlichen Daten, z.B. werden Systemmeldungen protokolliert, Druckaufträge werden hier zwischengespeichert usw.

Etwas genauer betrachten wir den unter `/usr` befindlichen Ast des Dateibaumes:

`/usr/bin`

Die meisten Nutzerprogramme findet man hier.

`/usr/sbin`

“Systemnahe” Nutzerprogramme (z.B. Netzwerktools).

`/usr/lib`

Bibliotheken der Nutzerprogramme.

`/usr/X11R6`

Unter diesem Ast verbirgt sich das gesamte X–Windows–System.

`/usr/man`

Hierunter befinden sich die Manual Pages, geordnet nach den jeweiligen Sektionen.

`/usr/var`

Noch ein Verzeichnis für variable Daten, wie z.B. die PID laufender Prozesse (in run), Sperren von Ressourcen (unter lock), Zwischenlager auszudruckender Dateien und Mails (spool)...

3.4 Partitionen

Festplatten müssen vor ihrer Benutzung als Speichermedium in Partitionen unterteilt werden. Notwendig sind solche Einteilungen, um eine Art Inhaltsverzeichnis auf der Festplatte einzurichten, anhand dessen die gespeicherten Daten effizient verwaltet werden können.
Eine Festplatte kann auch nur eine einzige Partition (max. 2 Gigabyte bei Linux) beherbergen; unabdingbar wird eine Unterteilung erst, wenn man mehrere Betriebssysteme gleichzeitig auf einer Platte installiert haben will. Unter Linux benötigen wir noch eine Swap–Partition, die im Falle mangelnden Hauptspeichers zur temporären Auslagerung von Speicherseiten dient. Günstig erweist sich auch die weitere Unterteilung in eigene Partitionen für die home–Verzeichnisse `/home`, für das Verzeichnis variabler Daten `/var` und für den `/usr`–Ast.
Für Festplatten mit mehr als 1024 Zylindern ist zu beachten, daß zumindest der Linux–Kernel innerhalb der ersten 1024 Zylinder liegen muß. Diese Restriktion ist keine Einschränkung des Betriebssystems, sondern den mangelnden Fähigkeiten heutiger BIOS–Implementierungen geschuldet. Somit kann es günstig sein, eine kleine Partition (10 MB) für das Verzeichnis `/boot` innerhalb dieser 1024 Zylinder anzulegen.

Auf den einzelnen Partitionen wird nun das Dateisystem von Linux *ext2fs* eingerichtet. Mindestens einer Partition muß nun ein sogenannter Mount–Point zugeordnet werden, der `/` für das Root–Filesystem. Mindestens das Root–Filesystem wird dabei bereits vom Kernel "gemountet", alle anderen Dateisysteme können auch per Hand nachträglich eingehangen werden:

```
root@sonne > mount /dev/hda1 /
```

`/dev/hda1` bezeichnet hier die erste Partition auf der ersten IDE–Festplatte, also das, was unter Dos/Windows als Laufwerk C: angesprochen wird. Unterschiedliche Laufwerke, ob Festplatten, Disketten– oder CDROM– Laufwerke werden im Linux–Dateisystem verborgen, indem sie in ein (leeres[3]) Verzeichnis eingehangen werden:

```
root@sonne > mount -t iso9960 /dev/cdrom /cdrom
root@sonne > mount /dev/hdb2  /home/ter/lwd
root@sonne > mount -t msdos /dev/fd0 /mnt
```

[3]Ist der Mount–Point kein leeres Verzeichnis, werden alle darin gespeicherten Dateien durch die Dateien des zu mountenden Dateisystems verdeckt.

Der Mount–Point (das Verzeichnis) muß existieren, genauso muß dem `mount`–Kommando der Typ des Dateisystems mitgeteilt werden, falls dieses nicht *ext2fs* ist.
Ohne Parameter aufgerufen, listet `mount` alle momentan gemounteten Partitionen auf. Um ein Dateisystem wieder abzuhängen, nutzt man

```
root@sonne > umount /cdrom
root@sonne > umount /dev/fd0
```

Beide Kommandos können (Wer hätte etwas anderes erwartet?) nur von `root` ausgeführt werden – es sei denn, er entscheidet sich für eine Liberalisierung. Dazu kann er entsprechende Einträge in der Datei `/etc/fstab` vornehmen (Seite 85).

3.5 Interna

Die Repräsentation der Dateien spiegelt sich in den I–Nodes *(information nodes)* wieder. Dazu wird die Partition in Blöcke fester Länge (1024, 2048 oder 4096 Byte) unterteilt. Abbildung 3.2 veranschaulicht das Prinzip.

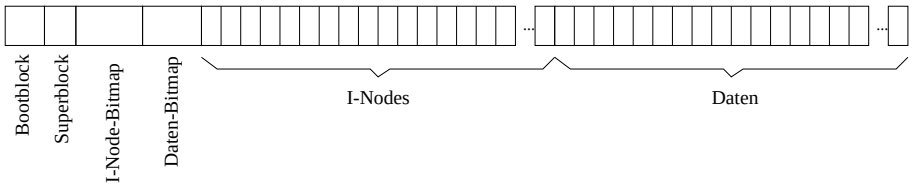

Abbildung 3.2: Vereinfachte Darstellung des Dateisystems auf eine Festplatte

- **Bootblock** Der erste Block jeder Partition kann einen Bootloader (zum Start eines Betriebssystems) enthalten, er wird beim Einschalten des Rechners vom BIOS gelesen.
- **Superblock** Hier stehen Informationen zum Typ und internen Aufbau des Dateisystems.
- **Bitmaps** Jedem I–Node und jedem Datenblock ist ein Bit in einer Bitmap zugeordnet. Ein gesetztes Bit zeigt an, daß der zugehörige Block benutzt wird. Die Bitmaps dienen dem schnellen Auffinden von freien Speicherkapazitäten.
- **I–Node** Jede Datei wird durch mindestens einen I–Node dargestellt. Im I–Node werden dazu alle Informationen bzgl. dieser Datei in einer eigenen Datenstruktur gehalten (Abbildung 3.3).

  Ist die Datei sehr klein, werden ihre Daten direkt im I–Node gespeichert, ansonsten verweist ein Eintrag im I–Node auf einen (oder mehrere) Datenblock, in dem nun der Inhalt der Datei gespeichert wird.

  Reichen die in einem I–Node referenzierten Blöcke für eine Datei nicht aus, zeigt ein Eintrag im I–Node auf einen (oder mehrere) weiteren I–Node,

welcher nun die eigentlichen Verweise zu den Datenblöcken beinhaltet. Man spricht von einem einfach indirekten Block.

Bis zu dreifach indirekte Blöcke sind möglich, so daß als maximale Dateigröße 16 GByte erzielt werden.

- **Datenblock** Der Inhalt von Dateien (außer von sehr kleinen) wird hier gespeichert.

  Da die Blockgröße festliegt, wird im Falle, daß die Dateigröße diese nicht erreicht, Plattenplatz verschwendet. In Anbetracht der durchschnittlichen Dateigrößen in einem Unix–System hat sich 4096 Byte als optimale Blockgröße herausgestellt (Wählt man einen kleineren Wert, erhöht sich zwar die mittlere Auslastung der einzelnen Blöcke, allerdings benötigt man nun mehr I–Nodes sowie größere Bitmaps).

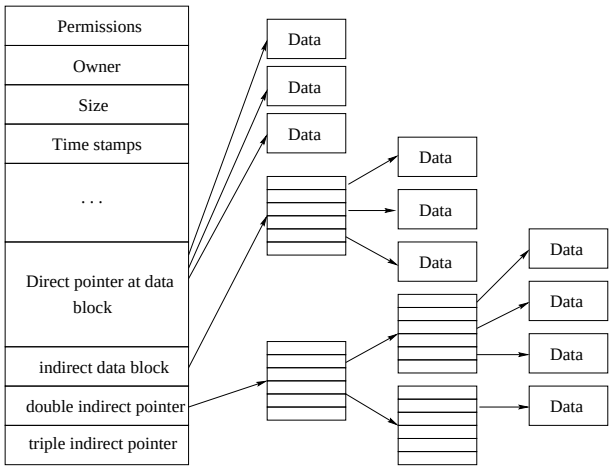

Abbildung 3.3: Speicherung der Dateien mit Hilfe von I–Nodes.

3.6 Links

Mit den Kenntnissen über den internen Aufbaus eines Linux–Dateisystems vollziehen wir in Abbildung 3.4 die Repräsentation eines festen und eines symbolischen Links. Von den Daten des I–Nodes sind nur die relevanten Informationen dargestellt.
Während bei einem festen Link der originale I–Node kopiert wird und gleichzeitig in einem Zähler die Anzahl der auf die Datenblöcke zeigenden festen Verweise notiert werden, speichert der symbolische Link in einem neuen I–Node einzig die Blocknummer des originalen I–Nodes.
Wird nun die originale Datei entfernt, wird der I–Node selbst gelöscht. Nicht aber die Datenblöcke, da anhand des Verweiszählers eine Benutzung dieser feststellbar ist. Diese würden nur im Falle eines auf “1” stehenden Zählers tatsächlich als frei markiert werden.
Für den symbolischen Verweis sind nun die Informationen verloren, um auf die Daten zugreifen zu können, nicht aber für den festen Link.

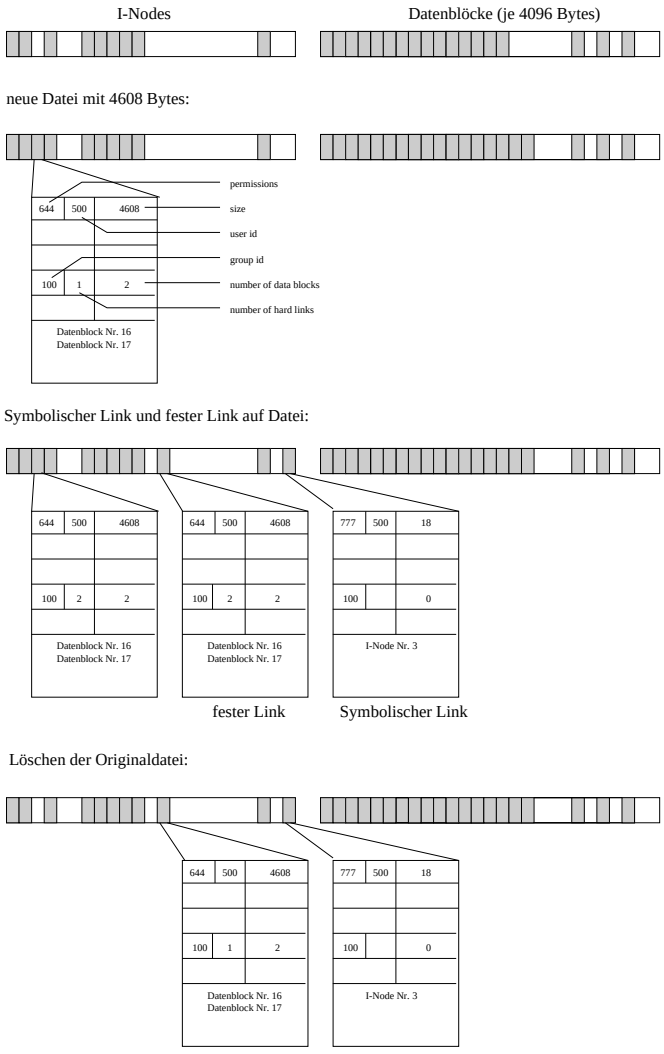

Abbildung 3.4: Der Unterschied zwischen statischen und symbolischen Links.

3.7 Zugriffsrechte

In der Einleitung auf Seite 13 wurde schon die Bedeutung der Zugriffsrechte erklärt. Zunächst werden wir zwei spezielle Zugriffrechte kennenlernen, bevor wir uns den Möglichkeiten zur Änderung der Rechte zuwenden.

3.7.1 Spezielle Zugriffsrechte

Typische Listings im Langformat führen zu Ausgaben folgender Gestalt:

```
user@sonne > ls -l
drwxr-xr-x  23 root   root   1024 Feb 15 15:54 /usr
-rwxr-xr-x   1 root   root 110890 Dec 12 18:01 /usr/bin/zoo
-rw-r--r--   1 user  users   9655 Apr  6 09:38 nis.tex
```

Was die einzelnen Bits zu bedeuten haben, sollte bereits bekannt sein.
Betrachten wir nun aber folgende Ausgaben:

```
user@sonne > ls -ld /t*
drwxrwxrwt   7 root   root   1024 Apr  7 10:07 /tmp

user@sonne > ls -l /usr/bin/passwd
-rwsr-xr-x   1 root shadow  32916 Dec 11 20:47 /usr/bin/passwd
```

Die erste Besonderheit betrifft das Bit "t" des Verzeichnisses `/tmp`. Der Buchstabe steht für "save program text on swap device" und bewirkt, daß in dieses Verzeichnis zu schreibende Daten solange wie möglich nur in der Swap–Partition (bzw. Hauptspeicher) existieren und eventuell erst beim Shutdown des Systems tatsächlich zurückgeschrieben werden. Für kurzlebige Daten (temporäre Daten) kann ein solches Verhalten zur Performancesteigerung beitragen, da zeitaufwendige Schreiboperationen vermieden werden.

Zur Erläuterung des s–Bits "set user or group ID on execution" betrachten wir ein kurzes C–Programmfragment, so wie das Programm `passwd` (dient zum Ändern des Paßwortes) realisiert sein könnte:

```
// Programmfragment passwd.c

int main() {
    FILE *new;
    //...

    new = fopen ("/etc/passwd","a");

    if ( new == NULL ) {
       perror("fopen");
       // ...
    }

    /...
}
```

Der Systemadministrator übersetzt das Programm und installiert es mit folgenden Zugriffsrechten im Verzeichnis `/usr/bin`:

```
-rwxr-xr-x  1 root  shadow  32916 Dec 11 20:47 /usr/bin/passwd
```

Ein normaler Nutzer möchte nun sein Paßwort ändern und startet das Programm `passwd`:

```
user@sonne > passwd
fopen: permission denied
```

Was ist die Ursache?
Das Programm versucht die Datei `/etc/passwd` zum Schreiben zu öffnen und scheitert natürlich, da nur Root die Schreibberechtigung besitzt (Dies darf auch nicht anders sein, sonst könnte jeder die Paßwörter manipulieren). Grund hierfür

ist, daß der Prozeß, der das Programm ausführt die Nutzerkennung von `user` erhält.
Mit Hilfe des s–Bits kann die Nutzerkennung des Prozesses vom rufenden Nutzer/Gruppe in die des Besitzers / der besitzenden Gruppe umgewandelt werden, so daß `passwd` nun "im Auftrag" von Root die Datei `/etc/passwd` öffnet. Ein s–Bit darf nur bei Binardateien gesetzt werden (keine Shellskripte).

3.7.2 Ändern der Zugriffsrechte

Zum Ändern der Zugriffsrechte steht das Kommando `chmod` zur Verfügung:

```
chmod [options] <mode> <dateiname>
```

Für die möglichen Optionen sei auf die Manuals verwiesen.

Symbolischer Modus

Symbolisch lassen sich die Rechte setzen durch eine Kombination aus der betreffenden Rechtegruppe:

Symbol	Rechtegruppe
u	Eigentümer (user)
g	Gruppe (group)
o	Andere (others)
a	Alle (all)

und den entsprechenden Rechten:

Symbol	Recht
r	Leserecht
w	Schreibrecht
x	Ausführungsrecht
s	s–Bit setzen
t	t–Bit setzen
+	Recht(e) hinzufügen
-	Rechte entfernen

Beispiele

```
user@sonne > ls -l hello
-rw-r--r--   1 user  users   108 Apr  7 12:10 hello

user@sonne > chmod a+w hello
-rw-rw-rw-   1 user  users   108 Apr  7 12:10 hello

user@sonne > chmod g-rw,u+xs hello
-rws---rw-   1 user  users   108 Apr  7 12:10 hello
```

3.7.3 Numerischer Modus

Rechte werden bitweise dargestellt, für Eigentümer, Gruppe und Andere gelten:

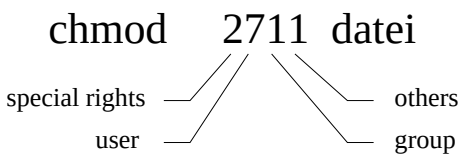

Abbildung 3.5: Syntax von chmod, numerisch

Recht	Bitdarstellung	Wert
Execute	0000 0001	1
Write	0000 0010	2
Read	0000 0100	4
rwx	0000 0111	7

Jede Rechtegruppe wird durch einen numerischen Wert repräsentiert, die Sonderrechte (s,t) werden durch einen eigenen Wert dargestellt. Somit werden dem Kommando `chmod` maximal vier Werte angegeben (Abbildung 3.5). Fehlen Werte, werden diese von links her (!) als Null (keine Rechte) angenommen, d.h. `chmod 1 <datei>` ist gleichbedeutend mit `chmod 0001 <datei>`!
Für die Sonderrechte gelten folgende Bitdarstellungen:

Recht	Bitdarstellung	Wert
t–Bit	0000 0001	1
s–Bit der Gruppe	0000 0010	2
s–Bit des Eigentümers	0000 0100	4

Beispiele

```
user@sonne > ls -l hello
-rw-r--r--   1 user  users   108 Apr  7 12:10 hello

user@sonne > chmod 666 hello
-rw-rw-rw-   1 user  users   108 Apr  7 12:10 hello

user@sonne > chmod 4706 hello
-rws---rw-   1 user  users   108 Apr  7 12:10 hello
```

3.7.4 Voreinstellung

Neue Dateien/Verzeichnisse werden offensichtlich stets mit einunddenselben Zugriffsrechten erzeugt:

```
user@sonne > mkdir testdir
user@sonne > touch testdat
user@sonne > ls -ld test*
-rw-r--r--  1 user  users     0 Apr  7 13:11 testdat
drwxr-xr-x  2 user  users  1024 Apr  7 13:11 testdir
```

Zuständig für dieses Verhalten ist die sogenannte `umask`, die oft in der Datei `/etc/profile` auf den Wert `022` voreingestellt wird.

Zusätzlich wird noch eine Maximalmaske benötigt, die sich für Verzeichnisse und andere Dateien unterscheidet. Die bei der Erzeugung gesetzten Rechte entstehen nun, indem von der Maximalmaske der durch `umask` vorgegebene Wert subtrahiert wird:

Verzeichisse	
Maximalmaske	777
umask	022
Ergebnis	755

Dateien	
Maximalmaske	666
umask	022
Ergebnis	644

Mit dem Kommando `umask` läßt sich die Voreinstellung ändern.

3.8 Gruppenzwang

Unter Unix muß jeder Benutzer mindestens einer Benutzergruppe angehören. Durch eine solche Gruppenzugehörigkeit lassen sich Zugriffsrechte auf Dateien für bestimmte Anwender ableiten. So kann ein einzelnes Verzeichnis nur für Mitglieder einer Projektgruppe zugänglich gemacht werden und auch für bestimmte Programme läßt sich der Zugriff auf bestimmte Ressourcen durch eine entsprechende Gruppeneinteilung kontrollieren.
Gruppen können durch Paßwörter geschützt werden und nur bei Kenntnis dieses kann ein Benutzer sich dieser Gruppe anschließen. Mitglieder einer solchen Gruppe (in der Datei `/etc/group` festgelegt) können ohne Eingabe des Paßwortes die Zugehörigkeit zu einer Gruppe ändern. Mit dem Kommando:

```
user@sonne > newgrp <group>
```

wechselt der Nutzer die effektive Nutzergruppe. Die Default–Gruppe (der man automatisch nach em Login angehört) ist in der Datei `/etc/passwd` festgelegt. Nach einem Gruppenwechsel gehören alle neu erstellten Dateien zur Gruppe `<group>`.
Eigene Dateien dürfen anderen Gruppen zugeordnet werden, falls der Besitzer auch Mitglied der neuen Gruppe ist:

```
user@sonne > chgrp <newgroup> <datei>.
```

3.9 Besitzerwechsel

Zum Ändern des Eigentümers einer Datei dient das Kommando `chown` *(change owner)*. Aus Sicherheitsgründen darf man allerdings nicht einfach seine Dateien einem anderen Nutzer unterschieben, deshalb erfordert dieses Kommando (meist) Root–Rechte. `chown` kann ebenso zum Wechsel der Gruppe verwendet werden:

```
root@sonne > chown user datei; ls -l datei
-rw-r--r--  1 user users     0 Apr  7 12:10 datei

user@sonne > chown root datei
chown: datei: Operation not permitted

user@sonne > chown :modem datei; ls -l datei
```

```
-rw-r--r--  1 user modem     0 Apr  7 12:10 datei

root@sonne > chown root.root datei; ls -l datei
-rw-r--r--  1 root  root     0 Apr  7 12:10 datei
```

Der Gruppenwechsel zur Gruppe "modem" ist nur gestattet, falls "user" Mitglied dieser Gruppe ist.

3.10 Andere Dateisysteme

Linux ist ein Multitalent und versteht die Sprachen vieler Dateisysteme. Voraussetzung für den Zugriff auf diese ist natürlich ein entsprechend konfigurierter Kernel (wird später behandelt), sowie eine aktuelle Version desselben.
Von Linux unterstützte Dateisysteme sind (Auswahl):

ext	Vorgänger des ext2
ext2	Linux
hpfs	OS–2 (nur lesend)
iso9660	CD–ROM
minix	Minix, heute oft für Disketten unter Unix verwendet
msdos	DOS–Partitionen oder –Disketten
nfs	Network File System
SMB	NT, Windows for Workgroups
swap	Swap–Partitionen oder –Dateien
System V	verschiedene Unixe
proc	Linux–internes System zur Prozeßverwaltung
umsdos	direkte Verwendung vom MS–DOS-Dateisystem unter Linux
vfat	Windows95

Um alle unterstützten Dateisysteme unter einem Hut zu vereinigen, ist eine einheitliche Schnittstelle erforderlich. Diese wird vom Virtuellen Dateisystem bereitgestellt.

3.11 Das virtuelle Dateisystem

Für einen Prozeß im System spielt es offenbar keine Rolle, auf welchem Medium sich die zu bearbeitende Datei befindet, ob im Linux–Dateisystem (ext2), ob auf einer MSDOS–formatierten Diskette oder auf einem anderen Rechner irgendwo im Netzwerk... Der Prozeß weiß nichts über Details der Implementierung; er kennt nur die Schnittstelle des Virtuellen File Systems (VFS) und greift über dessen Funktionen auf die Dateien zu.
Das Virtuelle Dateisystem vereint alle unterstützten Dateisysteme und setzt deren interne Strukturen in eine klar definierte Struktur um, die letztlich den Prozessen zur Arbeit mit den Daten dient. Abbildung 3.6 veranschaulicht die Interna des VFS.

3.12 Ausblick

Zu Erwarten sind demnächst folgende Erweiterungen zum *ext2fs*:

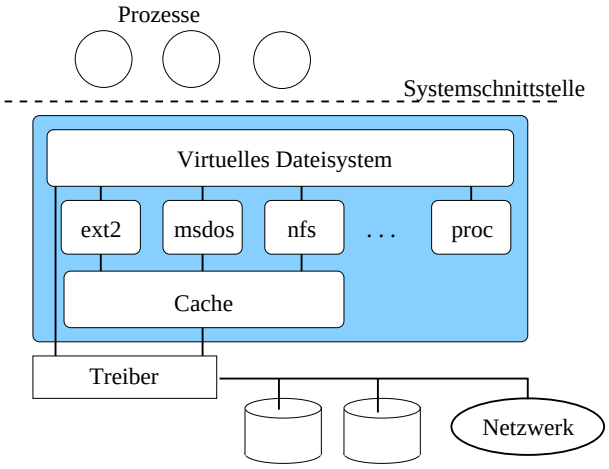

Abbildung 3.6: Interne Struktur des Virtuellen Dateisystems

- Rekonstruktion gelöschter Dateien
- Kennzeichnung brisanter Dateien (die beim Löschen mit Zufallszahlen überschrieben werden)
- komprimierte Speicherung von Dateien
- verschlüsselte Speicherung der Daten
- Partitionen bis zu 4 Terabyte
- CD-RW als "normales" Filesystem mounten (die Beta–Version ist verfügbar: IFS)

3.13 Prozesse

Unter Linux können mehrere Prozesse gleichzeitig ausgeführt werden. "Gleichzeitig" bedeutet hier quasi–parallel, d.h. dem Nutzer wird durch die sequentielle Abarbeitung mehrerer Prozesse in kleinen Zeitscheiben die Existenz mehrerer Prozessoren vorgegaukelt[4].

Welche Prozesse im Augenblick laufen, verrät uns z.B. das Kommando `ps`:

```
user@sonne > ps ax
  PID TTY STAT TIME COMMAND
    1  ?  S    0:03 init [2]
    2  ?  SW   0:00 (kflushd)
    3  ?  SW<  0:00 (kswapd)
    8  ?  S    0:00 update (bdflush)
   57  ?  S    0:00 /sbin/kerneld
```

[4]Linux unterstützt auch SMP (Symmetrisches Multiprocessing); jedoch laufen unter Linux schon so viele interne Prozesse quasi–parallel, daß für die wirklich gleichzeitige Abarbeitung mehrere Dutzend Prozessoren notwendig wären.

```
   68  ?  S    0:00 /usr/sbin/klogd -c 1
   70  ?  S    0:00 /usr/sbin/syslogd
   89  ?  S    0:00 /usr/sbin/rpc.mountd
   91  ?  S    0:00 /usr/sbin/rpc.nfsd
   93  ?  S    0:00 /usr/sbin/rpc.ugidd
  105  ?  S    0:00 /usr/sbin/cron
  110  ?  S    0:00 /usr/sbin/inetd
  112  a0 S    0:00 /usr/bin/gpm -t ms -m /dev/mouse
  115  ?  S    0:00 /usr/sbin/lpd
  127  ?  S    0:01 /usr/sbin/sshd
  129  ?  S    0:00 sendmail: accepting connections on port 25
  144   3 S    0:00 /sbin/mingetty tty3
  145   4 S    0:00 /sbin/mingetty tty4
  146   5 S    0:00 /sbin/mingetty tty5
  147   6 S    0:00 /sbin/mingetty tty6
  160  ?  S    1:54 /usr/X11R6/bin/Xwrapper :0 -bpp 8
 1561   2 S    0:00 bash
 1568   2 R    0:00 ps x
```

Aus obigem Listing sind folgende Informationen abzulesen:

PID — Prozeßnummer, jeder Prozeß erhält bei seiner Erstellung eine eindeutige PID. Der erste Prozeß beim Systemstart erhält die PID 1, der init–Prozeß. Den Vererbungsbaum veranschaulicht das Kommando `pstree`.

TTY — Welcher Prozeß läuft auf welchem virtuellen Terminal (Konsole)? *tty3–6* warten auf ein Login (mingetty). Ein Fragezeichen kennzeichnet Prozesse, die kein Terminal kontrollieren.

STATE — Zustand des Prozesses:

- R Der Prozeß wartet auf die Zuteilung der CPU (runable).
- S Der Prozeß wartet auf das Eintreten eines Ereignisses, z.B. das Beenden eines IO–Transfers... (spleeping).
- D Der Prozeß wartet auf einen bestimmten Hardwarezustand (uninterruptible sleep).
- T Der Prozeß befindet sich im Einzelschrittlauf, z.B. für Debuggingzwecke (traced).
- Z Der Prozeß existiert nicht mehr, aber der Vaterprozeß hat den Rückgabestatus noch nicht geprüft (zombie).
- W Die dem Prozeß zugehörige Speicherseite befindet sich nicht im Hauptspeicher (ist also ausgelagert).

TIME — Verbrauchte CPU–Zeit des Prozesses.

COMMAND Name des Programmes, welches vom Prozeß ausgeführt wird.

Abbildung 3.7 stellt zwei typische Situationen zur Entstehung und Beendigung Sunixvon Prozessen gegenüber.

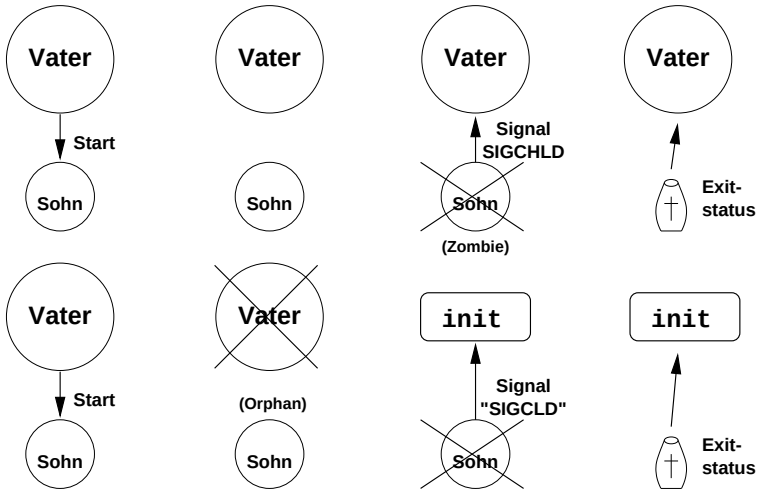

Abbildung 3.7: Geburt und Tod eines Prozesses

Ein Prozeß entsteht, indem ein Elternprozeß mittels des Systemrufes `fork()` einen neuen Prozeß erzeugt (Ausnahme ist `init`, der einzige "von Hand generierte" Prozeß). Dieser Kindprozeß teilt sich alle Ressourcen mit dem Elternprozeß, wesentliche Unterschiede sind nur ein eigener Stack und eine eigene PID.
Anhand der PID ist es den beiden Prozessen nun möglich, zu unterschieden, ob es sich um den Vorfahren oder einen Nachkommen handelt. In Abhängigkeit hiervon wird nun ein Kindprozeß mittels des Systemrufes `exec()` ein neues Programm laden.
Irgendwann wird ein Kindprozeß seine Arbeit beenden und signalisiert diesen Zustand seinem Elternprozeß durch ein entsprechendes Signal. Obwohl der Kindprozeß bereits jetzt aus Speicher und Prozeßtabelle entfernt ist, muß dieses Signal noch verarbeitet werden. Für gewöhnlich zeichnet der Elternprozeß dafür verantwortlich.
Zwei Situationen könnten den "normalen Werdegang" durcheinander bringen:

1. Der Elternprozeß ist andersweitig beschäftigt (befindet sich im Zustand D o.a.). In einem solchen Fall symbolisiert der Zustand Z das noch zu behandelnde Signal. (Situation 1 der Abbildung 3.7)

2. Der Elternprozeß existiert nicht mehr (Programmfehler o.a.). Jetzt ist der `init`–Prozeß für der Signalbehandlung verantwortlich (Situation 2 der Abbildung 3.7).

3.14 Interprozeßkommunikation

Signale sind **eine** Möglichkeit zur Interprozeßkommunikation. Vom Kernel verwendet, dienen sie dazu, Prozesse über bestimmte Ereignisse zu informieren;

der Nutzer bedient sich ihrer, um hängengebliebene Prozesse abzubrechen oder diese anzuweisen, ihre Konfigurationsdateien neu einzulesen.
Die Nutzersignale unterliegen bestimmten Restriktionen, sie dürfen nur an dem Nutzer zugeordnete Prozesse versandt werden. Nicht einmal *root* hat dem *init*–Prozeß etwas zu sagen.

Signal	Nummer	Aktion
SIGHUP	1	Terminal–Hangup, bei Dämonen verwendet, um ein erneutes Einlesen der Konfigurationsdateien zu erzwingen
SIGINT	2	Tastatur–Interrupt
SIGQUIT	3	Ende von der Tastatur
SIGILL	3	Illegaler Befehl
SIGABRT	4	Abbruch–Signal von *abort(3)*
SIGFPE	8	Fließkommafehler (z.B. Division durch Null)
SIGKILL	9	Unbedingte Beendigung eines Prozesses
SIGSEGV	11	Speicherzugriffehler
SIGPIPE	13	Schreiben in eine Pipe, ohne daß ein Prozeßs daraus liest
SIGTERM	15	Prozeß soll sich beenden (default von kill)
SIGCHLD	17	Ende eines Kindprozesses
SIGCONT	18	Gestoppter Prozeß wird fortgesetzt
SIGSTP	20	Ausgabe wurde angehalten
SIGUSR1	30	Nutzerdefiniertes Signal

Die genaue Zuordnung zwischen symbolischem Signal und Nummer ist in Linux in der Datei /usr/include/linux/asm/signal.h zu finden.

3.15 Das Prozeßdateisystem

Das Prozeßdateisystem stellt zur Laufzeit die Daten des Kernels in Form eines normalen Dateisystems dar. Als Mount–Point dient normalerweise /proc. Dieses Dateisystem existiert allein im Hauptspeicher und nicht auf der Festplatte!
Ein Blick in das Verzeichnis offenbart den Inhalt:

```
user@sonne > ls /proc
1         129       146       177       93          modules
102       1299      147       182       cmdline     mounts
105       1300      1532      183       cpuinfo     net
110       1320      1533      184       devices     pci
112       1337      155       185       dma         scsi
115       1338      156       186       filesystems self
1199      1364      1561      2         interrupts  stat
1202      141       159       3         ioports     sys
1204      142       160       57        kcore       uptime
1205      143       1624      68        kmsg        version
1211      1439      163       70        ksyms
1212      144       170       8         loadavg
1225      1442      173       85        locks
1226      1444      174       89        meminfo
```

```
127     145     176     91      memstat
```

Sinn dieses Abbildes der Kerneldaten ist es, Programmen das Lesen dieser Daten zu ermöglichen, ohne auf den Kernelbereich zugreifen zu müssen (Sicherheit!!!).

Im Einzelnen bedeuten die Einträge:

- **1..** Diese Zahlen entsprechen den PIDs der laufenden Prozesse und sind Unterverzeichnisse, die wiederum die relevanten Daten der Prozesse beinhalten.
- **cmdline** Enthält die Bootzeile, mit der der Kernel gestartet wurde. In den Unterverzeichnissen zu den einzelnen Prozessen stehen hier die Optionen beim Start des jeweiligen Prozesses.
- **cpuinfo** Typ und Leistung der CPU findet man hier.
- **devices** Major und Minor Number der im Kernel geladenen Treiber stehen hier.
- **filesystems** Die vom Kernel unterstützten Dateisysteme.
- **interrupts** Liste der belegten Hardwareinterrupts.
- **kcore** Zugang zum Arbeitsspeicher (nur für root).
- **locks** Liste der aktiven Dateisperren.
- **meminfo** Speicher– und Swap–Auslastung.
- **stat** Statusinformation des Kernels.
- **sys** Verschiedene Informationen zum Laufzeitsystem.
- **version** Kernelversion und Übersetzungsdatum dieses.

Kapitel 4

Wichtige Kommandos

An dieser Stelle soll eine (unvollständige) Auflistung der unter Linux verfügbaren Kommandos folgen. Die Gliederung erfolgt nach thematischen Gesichtspunkten, wobei einige Kommandos wiederholt erscheinen werden. Die möglichen Optionen werden nur teilweise erwähnt; hier informiere man sich in den entsprechenden Manuals. Es wird nicht zwischen bashspezifischen und allgemeinen Kommandos unterschieden!

4.1 Kommandos zur Dateiverwaltung

cat — verkettet seine Argumente

```
user@sonne > cat file.1 file.2 file.3 > file.all
```

cd — wechselt aktuelles Verzeichnis

```
user@sonne > cd /usr/src          absolute Pfadangabe
user@sonne > cd ../../usr         relative Pfadangabe
user@sonne > cd                   wechselt nach $HOME
```

cp — kopiert Dateien

```
user@sonne > cp datei.txt file.txt
user@sonne > cp -R dir dest       rekursiv mit
                                  Unterverzeichnissen
```

find — sucht Dateien nach Namen `-name`, Datum `-[a,c]time`, Größe `-size`, Typ `-type` usw.

```
user@sonne > find /usr/include -name "*.h"
```

ln — richtet einen Link ein

```
ln file.1 link2.1                 fester Link (Hardlink)
ln -s file1. symlink2.1           symbolischer Link
```

ls zeigt den Inhalt eines Verzeichnisses an

```
user@sonne > ls /boot
System.map  boot.b      map         vmlinuz
boot.0803   chain.b     os2_d.b
user@sonne > ls -l /boot/m*                 langes Format
-rw-r--r-- 1  root  root  9728  Feb 15 16:01 /boot/map
```

mkdir legt ein neues Verzeichnis an

```
user@sonne > mkdir ~/testdir
```

mv verschiebt Dateien und/oder ändert ihren Namen

```
user@sonne > mv old new                          Umbenennen
user@sonne > mv new /tmp                         Verschieben
user@sonne > mv /tmp/new ~/old Verschieben und Umbenennen
```

rm löscht Dateien

```
user@sonne > rm datei
user@sonne > rm -r dir       rekursiv mit Unterverzeichnissen
```

rmdir löscht leere Verzeichnisse

```
user@sonne > rmdir dir
```

split zerlegt eine Datei in Teile vorgegebener Größe

```
user@sonne > ls -l archiv.tgz
-rw-r--r-- 1 user users  3998311 Apr 7 08:41 archiv.tgz
user@sonne > split -b 1400000 archiv.tgz
user@sonne > ls -l
-rw-r--r-- 1 user users  3998311 Apr 7 08:41 archiv.tgz
-rw-r--r-- 1 user users  1400000 Apr 7 08:42 xaa
-rw-r--r-- 1 user users  1400000 Apr 7 08:42 xab
-rw-r--r-- 1 user users  1198311 Apr 7 08:42 xac
```

tee vervielfältigt die Standardeingabe (Siehe auch Seite 59)

```
user@sonne > ls -l | tee inhalt
```

4.2 Zugriff auf DOS–Disketten

fdformat Formatiert eine Diskette. Das Format ist am Kernel festgelegt und wird durch das gewählte Device bestimmt, die Option `-n` unterdrückt eine anschließende Überprüfung der Diskette. Es wird kein Dateisystem angelegt.

```
user@sonne > fdformat -n /dev/fd0
```

mattrib ändert die Attribute einer Datei

```
user@sonne > mattrib a:\hash.h
  A          A:/hash.h
user@sonne > mattrib -a a:\hash.h
             A:/hash.h
```

mcd wechselt Verzeichnis

```
user@sonne > mmd tmp                          Siehe mmd
user@sonne > mcd tmp
user@sonne > mdir                             Siehe mdir
 Volume in drive A has no label
Directory for A:/tmp

.            <DIR>     04-14-1999   9:51
..           <DIR>     04-14-1999   9:51
        2 files                   0 bytes
                            484 864 bytes free
```

mcopy kopiert Dateien von/nach Linux

```
mcopy file.1 a:                               Laufwerk A
mcopy file.1 b:                               Laufwerk B
```

mdel löscht Datei auf Diskette

```
mdel file.1                          löscht auf Laufwerk A
```

mdir zeigt das Inhaltsverzeichnis der Diskette, siehe Beispiel zu `mcd`

mformat Legt ein DOS–Dateisystem an. Optional kann das Format (Spuren, Köpfe, Zylinder) angegeben werden. Fehlt die Angabe, wird versucht, das Format aus der Datei `/etc/mtools.conf` zu entnehmen.

```
user@sonne > mformat a:
```

mlabel ändert Namen der Diskette

```
user@sonne > mlabel a:
 Volume has no label
Enter the new volume label : test
user@sonne > mlabel a:
 Volume label is TEST
```

mmd — erzeugt ein Verzeichnis; siehe Beispiel zu `mcd`

mrd — löscht ein Verzeichnis

```
user@sonne > mrd a:/tmp
```

mread — Kopiert Dateien von DOS nach Linux. Mit der Option `-t` wird das DOS–Zeilenendezeichen CR/LF in das Unix–Zeilenendezeichen LF umgewandelt.

```
user@sonne > mread -t a:hash.h .
```

mren — ändert einen DOS–Dateinamen

```
user@sonne > mren a:hash.h a:hash.old
```

mtype — zeigt den Inhalt von Textdateien

```
user@sonne > mtype a:hash.old
```

mwrite — kopiert von Linux nach DOS; Gegenstück zu `mread`

4.3 Texte bearbeiten

csplit — Zerlegt den Text an vorgegebenen Stellen (Muster) in einzelne Dateien. Das Muster wird zwischen zwei Slashes eingeschlossen.

```
user@sonne > csplit -k --prefix mail \
> /var/spool/mail/user /^From/
```

cut — extrahiert Spalten aus jeder Zeile eines Textes

```
user@sonne > ls -l /boot| cut -b 1-11,56-
total 718
-rw-r--r-- System.map
-rw-r--r-- boot.b
-rw-r--r-- chain.b
-rw------- map
-rw-r--r-- vmlinuz
```

expand — ersetzt Tabulatoren durch Leerzeichen

```
user@sonne > expand file1 > file2
```

fold — bricht lange Textzeilen an vorgegebenen Positionen um

```
user@sonne > cat > testdat
Das ist eine lange Zeile(ENTER),(CTRL)(DEL)
user@sonne > fold -12 testdat
Das ist eine
 lange Zeile
```

fromdos — konvertiert DOS–Zeilenenden ins Linux–Format

grep — sucht Textmuster innerhalb der Eingabe

```
user@sonne > ps eax | grep bash
 167   2 S    0:00 -bash
 166   1 S    0:00 -bash TERM=linux HZ=100 HOME=/...
1630  p0 S    0:00 grep bash PWD=/home/user...
```

head — zeigt die ersten (10) Zeilen einer Datei an

```
user@sonne > head -20 /var/log/messages
```

less — zeigt seitenweise Dateien an

more — wie less

paste — vereint mehrere Texte zeilenweise

```
user@sonne > less test1.txt
  Zeile1 aus test1.txt
  Zeile2 aus test1.txt(Q)
user@sonne > more test2.txt
  Zeile1 aus test2.txt
  Zeile2 aus test2.txt
user@sonne > paste test1.txt text2.txt
  Zeile1 aus test1.txt   Zeile1 aus test2.txt
  Zeile2 aus test1.txt   Zeile2 aus test2.txt
```

recode — konvertiert zwischen verschiedenen Zeichensätzen

sed — Stream–Editor (programmierbar, siehe Manual Page)

sort — sortiert seine Eingabe

```
user@sonne ls -l /boot | sort +4
-rw-r--r-- 1 root root    300 Dec 11 22:08 chain.b
-rw-r--r-- 1 root root   4536 Dec 11 22:08 boot.b
-rw------- 1 root root   9728 Feb 15 16:02 map
-rw-r--r-- 1 root root 131719 Feb 15 16:01 System.map
-rw-r--r-- 1 root root 579254 Feb 15 16:01 vmlinuz
```

tac — “verkehrtes” cat

```
user@sonne > tac test1.txt
Zeile2 aus test1.txt
Zeile1 aus test1.txt
```

tail — zeigt die letzten (10) Zeilen einer Datei an

```
user@sonne > tail /var/log/messages
```

todos — Gegenstück zu fromdos

tr — ersetzt in einer Datei vorgegebene Zeichen durch andere Zeichen

```
user@sonne > cat testdat
  Weiteres zu TR siehe: man TR
user@sonne > tr TR tr < testdat
  Weiteres zu tr siehe: man tr
```

uniq — entfernt mehrfach auftretende Zeilen in einer sortierten Datei

```
user@sonne > less testdat
  eine erste Zeile
  eine zweite Zeile
  eine erste Zeile
user@sonne > sort testdat | uniq
  eine erste Zeile
  eine zweite Zeile
```

4.4 Packen/Entpacken/Archivieren

Bemerkung: Um die Güte der einzelnen Komprimierungsverfahren zu demonstrieren, wird jeweils einunddiesselbe Datei komprimiert:

```
user@sonne > ls -l mtar
-rw-rw-rw-  1 user  users   1295872 Apr 14 11:51 mtar
```

bzip2 — Neueres Komprimierungstool, verwendet einen effizienteren Algorithmus als `gzip`. Leider ist dieses Tool noch neu und wenig verbreitet.

```
user@sonne > bzip2 mtar; ls -l
-rw-rw-rw- 1 user users  552899 Apr 14 11:51 mtar.bz2
user@sonne > bunzip2 mtar.bz2
```

cpio — Überträgt Archive zwischen unterschiedlichen Dateisystemen.

Beispiel: Alle Dateien, die zum Paket `bash` gehören (im `rpm`–Format und im System installiert) sollen in ein `tar`–Archiv gepackt werden:

```
user@sonne > rpm -ql bash | cpio -o -H ustar -O mtar
```

compress — Ein älteres Werkzeug zum Komprimieren von Dateien.

```
user@sonne > compress mtar; ls -l
-rw-rw-rw- 1 user users  806217 Apr 14 11:51 mtar.Z
user@sonne > uncompress mtar.Z
```

gzip — Ein weit verbreitetes Komprimierungsprogramm, daß nach dem Lempel–Ziv–Algorithmus arbeitet.

```
user@sonne > gzip mtar; ls -l
-rw-rw-rw- 1 user users  586426 Apr 14 11:51 mtar.gz
user@sonne > gunzip mtar.gz
```

mt — Dient der Steuerung von Streamern.

rpm — Der Redhat–Package–Manager; wird von vielen Distributoren als Standardformat für Pakete genommen. Ein `rpm`–Paket enthält neben den Dateien auch Informationen, Beschreibungen und Versionsnummern (im Unterschied z.B. zu einem `tar`–Archiv).

```
root@sonne > rpm -i bash.rpm                    Installieren
user@sonne > rmp -q bash                          Abfrage
bash-2.02.1-13
user@sonne > rpm -ql less                Abfrage nach Inhalt
/etc/lesskey
/etc/lesskey.bin
/usr/bin/less
/usr/bin/lessecho
/usr/bin/lesskey
/usr/bin/lesspipe.sh
/usr/man/man1/less.1.gz
/usr/man/man1/lesskey.1.gz
root@sonne > rmp -e less                      Deinstallieren
```

Mit `rpm` läßt sich noch wesentlich mehr anstellen; auch lassen sich eigene Pakete generieren. Näheres findet man in den Manuals.

tar — Archivierungsprogramm, das Verzeichnisstrukturen erhält.

```
user@sonne > tar czf archiv.tgz file* dir   Einpacken,
Komprimieren
user@sonne > tar tzf archiv.tgz          Inhalt auflisten
dir/contents
dir/file
file
file_01
user@sonne tar xzf archiv.tgz                  Entpacken
```

4.5 Prozeßverwaltung

bg — setzt Prozeß im Hintergrund fort; Beispiel auf Seite 59

fg — setzt einen Hintergrundprozeß im Vordergrund fort

```
user@sonne > sleep 100&
[1] 2085
user@sonne > sleep 110&
[1] 2086
user@sonne > fg %1
sleep 100
```

halt — beendet Linux und hält den Rechner an

```
root@sonne > halt
```

kill — sendet Prozessen (PID) Signale

```
user@sonne > kill -SIGTERM 235
user@sonne > kill -15 236
```

killall — sendet Prozessen (Name) Signale

```
user@sonne > kill -SIGTERM xinit
```

nice — Startet Prozeß mit veränderter Priorität. Die Priorität bestimmt, wieviel CPU–Zeit einem Prozeß zugeteilt wird. Eine Verringerung ist z.B sinnvoll, um mit rechenintensiven Prozessen nicht das System zu lähmen; eine Erhöhung könnte bei Damönenprozessen erforderlich sein und darf nur von Root vorgenommen werden.

```
user@sonne > nice -n 19 gcc bigprogram.c
user@sonne > nice -n -10 inetd
```

nohup — Führt ein Kindprozeß unabhängig vom Vater aus. Stirbt der Vater, werden normalerweise alle seine Nachfahren beendet. Dieses Verfahren wird mit `nohup` unterbunden.

```
user@sonne > bash
user@sonne > ./sleepproc&
[1] 776
user@sonne > exit
user@sonne > ps eax | grep  sleepproc
user@sonne >
user@sonne > bash
user@sonne > nohup ./sleepproc&
[1] 786
user@sonne > exit
user@sonne > ps eax | grep  sleepproc
user@sonne > 786  ?  S N  0:00 sh ./sleepproc ...
```

ps — Listet laufende Prozesse auf.

```
user@sonne > ps
  PID TTY STAT TIME COMMAND
  166   2 S    0:00 -bash
  169   4 S    0:00 (mingetty)
  170   5 S    0:00 (mingetty)
  171   6 S    0:00 (mingetty)
 1109   2 R    0:00 ps
```

reboot — beendet Linux und startet den Rechner erneut

shutdown — beendet Linux

```
root@sonne > shutdown -r +5          Reboot in 5 Minuten
root@sonne > shutdown -h now         sofortiges Halt
```

time Mißt die Zeit zur Ausführung eines Programmes. Im Beispiel wird die Zeit zum Übersetzen des Kernels (ohne Module) gemessen.

```
root@sonne > time (make dep; make clean; make zImage)
real    4m47.858s       Gesamtzeit (inkl. Warten auf CPU)
user    3m10.590s                  Zeit im Nutzermodus
sys     0m16.560s                  Zeit im Kernelmodus
```

top listet alle Prozesse auf und aktualisiert per Voreinstellung diese Liste aller 5 Sekunden

4.6 Nutzerverwaltung

useradd Richtet einen neuen Benutzeraccount ein.

```
root@sonne > useradd testuser -m
```

userdel Entfernt einen Benutzeraccount.

```
root@sonne > userdel testuser
```

groups Zeigt die Gruppen des Benutzers an.

```
user@sonne > groups
users
root@sonne > groups
root bin uucp shadow dialout nogroup
```

passwd Ändert das Paßwort eines Benutzers:

```
user@sonne > passwd
Changing password for user
Old password:
Enter the new password (min of 5, max of 8 characters)
Please use a combination of upper and lower case
letters and numbers.
New password:
Re-enter new password:
```

4.7 Dateisystem

dd — Dient dem Kopieren von Datenblöcken, z.B zum Sichern des Bootsektors einer Festplatte:

```
root@sonne > dd if=/dev/hda of=/boot/bootsektor \
> bs=512 count=1
```

e2fsck — Zur Reparatur des Linux–Dateisystems ext2fs.

```
root@sonne > e2fsck /dev/hda3
```

fdformat — Diskette formatieren, siehe Seite 39.

fdisk — Dient der Partitionierung von Festplatten.

```
root@sonne > fdisk /dev/hda
```

```
Command (m for help): p
/dev/hda1 *    237    375  1116486    7  OS/2 HPFS
/dev/hda2      376    527  1220940    5  Extended
/dev/hda3        1    220  1767118+  83  Linux native
/dev/hda4      221    236   128520   82  Linux swap

Command (m for help): q
```

fsck — Reparatur von Dateisystemen (xiaf, minix, ext2). `fsck` ist ein Frontend und ruft entsprechend des Dateisystemtyps andere Programme auf. Per default versucht `fsck` den Typ anhand der `/etc/fstab` zu erkennen.

```
root@sonne > fsck /dev/hda3
```

mkfifo — Erzeugt eine named pipe (FIFO–Datei), ein Beispiel ist auf Seite 59 zu finden.

mkfs — Linux–Dateisysteme einrichten, korrespondiert zu `fsck`.

mknod — Erstellt eine neue Device–Datei (FIFO, Block– oder Chardevice).

mkswap — Existierende Datei oder Partition als Swap–Bereich einrichten.

```
root@sonne > dd if=/dev/zero of=/tmp/swpfile bs=1024\
> count=4096
4096+0 records in
4096+0 records out
root@sonne > mkswap /tmp/swpfile 4096
root@sonne > swapon /tmp/swpfile
```

mount — Einhängen eines Devices in ein Dateisystem, siehe Seite 22.

swapoff — Deaktivieren eines Swap–Bereiches.

```
root@sonne > swapoff /tmp/swpfile
```

swapon Aktivieren eines Swap–Bereiches, siehe `mkswap`.

4.8 Terminals

reset Restauriert den Zeichensatz:

```
stupiduser@sonne > cat /usr/bin/grep
¡ ¿— ¿                                   "reset" eingeben!!!
Erase is delete.
Kill is control-U (^U).
Interrupt is control-C (^C).
stupiduser@sonne >
```

restorefont Speichert/restauriert den VGA–Zeichensatz.

restorepalette Restauriert die Farbpalette.

setfont Verändert den momentan verwendeten Zeichensatz (nur auf der Konsole). Mit `showfont` erhält man den Zeichensatz.

```
user@sonne > setfont lat4u-19.psf.gz
```

setterm Verändert verschiedene Terminaleinstellungen, z.B. löscht die nachfolgende Eingabe den Bildschirminhalt

```
user@sonne > setterm -clear
```

4.9 Netzwerk–Administration

arp Anzeige und Manipulation des lokalen ARP–Caches (Address Resolution Protocol)

```
root@sonne > arp
Address         HWtype  HWaddress         Flags Mask
  Iface
www.saxedu.de  ether   00:80:C8:47:17:A1    C
  eth0
root@sonne > arp -d www.saxedu.de       Eintrag löschen
root@sonne > arp
root@sonne >
```

ifconfig Konfiguration eines Netzwerk–Interfaces; wird hauptsächlich beim Systemstart zur Initialisierung benötigt.

```
root@sonne > ifconfig eth0 192.168.10.101 broadcast
 194.180.239.255 netmask 255.255.255.0 up
root@sonne > ifconfig eth0 down           deaktivieren
```

netstat — Abfrage der Netzwerk–Schnittstellen, Anzeige von Statistiken und der Kernel–Routingtabelle.

Wichtige Optionen:

ohne	Anzeige der geöffneten Ports
-a	Anzeige aller geöffneter und überwachter Ports
-i	Statistik der Netzwerkschnittstellen
-r	Anzeige der Routingtabelle des Kernels
-t/-u	Anzeige der TCP/UDP–Sockets

Beispiele:

```
user@sonne >  netstat -r
Kernel IP routing table
Destination     Gateway       Genmask         Flags
  MSS Window  irtt Iface
sonne.galaxis.d *             255.255.255.255 UH
 1500 0          0 dummy0
loopback        *             255.0.0.0       U
 3584 0          0 lo
default         gw.galaxis.de 0.0.0.0         UG
 1500 0          0 eth0
```

```
user@sonne > netstat -i
Kernel Interface table
Iface   MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK
  TX-ERR TX-DRP TX-OVR Flags
lo     3584   0     50      0      0      0     50
       0      0      0 BLRU
dummy  1500   0      0      0      0      0      0
       0      0      0 BORU
eth0   1500   0   5413      4      0      0   1346
       0      0      0 BRU
```

Bedeutung der Felder:

MTU	Maximal Transfer Unit
Met	Metrik (in Linux ohne Bedeutung)
RX	empfangene Pakete
TX	gesendete Pakete
OK	Paket i.O.
ERR	fehlerhaftes Paket
DRP	verworfenes Paket
OVR	Überlauf des Paketes

ping — Sendet ICMP–Pakete mit einem ECHO_REQUEST an einen Netzwerkrechner.

```
user@sonne > ping -c 2 -s 1024 198.41.0.4
PING 198.41.0.4 (198.41.0.4): 1024 data bytes
1032 bytes from 198.41.0.4: icmp_seq=0 ttl=247
     time=175.3 ms
1032 bytes from 198.41.0.4: icmp_seq=1 ttl=247
     time=176.2 ms

--- 198.41.0.4 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet
loss
round-trip min/avg/max = 175.3/175.7/176.2 ms
```

route — Anzeige und Manipulation der Kernel–Routingtabelle.

```
root@sonne > route
Kernel IP routing table
Destination     Gateway         Genmask         Flags
   Metric Ref    Use Iface
sonne.galaxis.d *               255.255.255.255 UH
   0      0        0 dummy0
loopback        *               255.0.0.0       U
   0      0        1 lo
default         gw.galaxis.de   0.0.0.0         UG
   0      0        6 eth0
```

Hinzufügen einer Route ins Netz `192.168.85.0` über das Gateway `gw.galaxis.de`:

```
root@sonne > route add -net 192.168.85.0 \
> gw gw.galaxis.de dev eth0
root@sonne > route
...
92.68.85.0 gw.galaxis.de 255.255.255.0 UG !
   0      0        0 eth0
...
root@sonne > route del 192.168.85.0
```

traceroute Zeichnet die Route auf, die ein Paket im Netzwerk durchläuft. `traceroute` sendet dazu je 3 IP–Pakete mit gesetztem Time to life (TTL). Durch Erhöhung des Wertes im TTL–Feld wird ein TIME_EXCEEDED beim jeweils nächsten Host erzwungen. Ein Stern (*) zeigt einen Fehlversuch an.

```
root@sonne > traceroute 198.41.0.4
traceroute to 198.41.0.4 (198.41.0.4), 30 hops max,
 40 byte packets
 1 base.saxedu.de (194.180.239.1)
                          1.247 ms  0.853 ms  0.829 ms
 2 195.211.141.1 (195.211.141.1)
                          4.153 ms  3.912 ms  2.489 ms
 3 frankfurt1.ipf.NET (195.211.224.1)
                        18.129 ms  18.121 ms  16.695 ms
 4 hssi4-0.frankfurt2.gigabell.NET (195.211.199.30)
                        18.220 ms  26.968 ms  33.149 ms
 5 atm3-0.washington2.gigabell.NET (195.211.199.105)
                      142.528 ms  148.544 ms  141.750 ms
 6 mae-east3.noc.conxion.NET (192.41.177.168)
                      136.715 ms  130.753 ms  132.033 ms
 7 hevacis-01-pos900.conxion.NET (206.204.250.41)
                      128.383 ms  145.647 ms  143.854 ms
 8 206.204.250.74 (206.204.250.74)
                      143.390 ms  138.117 ms  133.604 ms
 9 206.204.221.1 (206.204.221.1)
                      135.003 ms  132.150 ms  150.069 ms
10 * * *
11 A.ROOT-SERVERS.NET (198.41.0.4)
                                  161.153 ms  250.809 ms *
```

4.10 Weiteres

alias Definiert eine Abkürzung, `unalias` entfernt eine Abkürzung. Beispiele findet man auf Seite 57.

cksum CRC–Prüfsumme[a] einer Datei berechnen.

```
user@sonne > cksum archiv.tar.gz
2645260944 586426 archiv.tar.gz
```

[a] cyclic redundancy check

diff Vergleicht zwei Dateien und gibt die Unterschiede aus.

```
user@sonne > cat source1.txt
Das ist das Original.
Es wird erweitert werden.
user@sonne > cat source2.txt
Das ist das Original.
Es wurde erweitert.
Hier die Erweiterung
user@sonne > diff source1.txt source2.txt
2c2,3
< Es wird erweitert werden.
---
> Es wurde erweitert.
> Hier die Erweiterung
```

`diff` hat vor allem im Kombination mit `patch` Bedeutung erlangt, wir werden darauf zurück kommen (Siehe `patch`).

df, du Informationen zur Auslastung der Festplatte.

```
user@sonne > df
Filesystem 1024-blocks Used Available Capacity Mounted
/dev/hda3    1709661 1059753   561553     65%   /
/dev/hda7     200568   36008   164560     18%   /msdos
```

expr Integer–Berechnung.

```
user@sonne > expr 3 + 45
48
user@sonne > expr 3 \* 500
1500
user@sonne > expr 3 \< 5
1
```

file Versucht den Typ einer Datei zu erkennen (Siehe auch 6.2).

```
user@sonne > file linux.ps
linux.ps: PostScript document text conforming at
level 2.0
```

free Informationen über RAM und Swap.

```
user@sonne > free
             total   used   free shared buffers cached
Mem:         62768  60996   1772  37344     976  15524
-/+ buffers/cache: 44496  18272
Swap:      128516  15248 113268
```

hash Zeigt die Hashtabelle an. Ein einmal aufgerufenes Kommando wird mit seinem Pfad in einer Tabelle gehalten, um bei einem erneuten Aufruf den Zugriff zu beschleunigen (jetzt muß nicht erst PATH durchsucht werden).

```
user@sonne > hash
hits    command
   1    /bin/more
   1    /usr/bin/patch
  10    /bin/tar    ...
```

lpr Zum Ausdruck von Dateien (siehe 7.4).

patch Fügt Änderungen in eine Datei ein. Vor allem in der Programmierung werden sich die Quelldateien von Version zu Version unterscheiden. Um uptodate zu sein, kann man sich jedes Mal den gesamten Sourcebaum neu besorgen. Das ist sicher nur für kleinere Quelltexte praktikabel. Viel günstiger wäre es ja, wenn man nur die tatsächlichen Änderungen holen und diese in die alten Dateien einpassen würde. Genau zu diesem Zweck wurde `patch` entwickelt. Das folgende Beispiel baut auf die Einführung zu `diff` auf:

```
user@sonne > diff source1.txt source2.txt > s1-2.diff
user@sonne > patch < s1-2.diff
Hmm...  Looks like a normal diff to me...
File to patch: source1.txt
Patching file source1.txt using Plan A...
Hunk #1 succeeded at 2.
done
```

`source1.txt` und `source2.txt` besitzen nun denselben Inhalt. Das oben demonstrierte Vorgehen wird zum Beispiel bei der Verteilung neuer Kernelversionen genutzt. Anstatt ca. 30 MByte Daten zu übertragen, besorgt man sich das wenige kByte große `diff`–File.

printenv Zeigt Umgebungsvariablen an (Environment).

```
user@sonne > printenv
PWD=/home/user
WINDOWID=37748750
PAGER=less  ...
```

rdev Dient zur Veränderung des Kernels, z.B. kann der default–Textmodus eingestellt oder (im Beispiel) der VGA–Modus verändert werden (wird hier auf 80x50 gesetzt):

```
root@sonne > rdev -v /boot/vmlinuz -2
```

set Zeigt alle Shellvariablen an.

sum — Prüfsummenberechnung einer Datei.

```
user@sonne > sum archiv.tar.gz
55618   573
```

tty — Zeigt den Device–Namen des aktiven Terminals an.

type — Gibt an, wie eine Kommandos von der Shell interpretiert werden würde:

```
user@sonne > type passwd
passwd is /usr/bin/passwd
user@sonne > type test
test is a shell builtin
```

uname — Liefert Informationen zum Betriebssystem.

```
user@sonne > uname
Linux
user@sonne > uname -n
sonne
user@sonne > uname -r
2.0.36
```

Kapitel 5

Der Kommandozeileninterpreter *bash*

Die bash (*bourne again shell*)[1] ist standardmäßig das erste Programm, das von Linux nach dem erfolgreichen Einloggen gestartet wird.
Die Shell allgemein ermöglicht eine bequeme Eingabe und Ausführung von Kommandos; daher auch die Bezeichnung eines Kommandozeileninterpreters. Zusätzlich stellen die Shells unter Unix eine Programmiersprache zur Verfügung, um sogenannte Shell–Skripte erstellen zu können.

5.1 Andere Shells

Viele kluge Köpfe haben sich die Frage gestellt, was eine Shell leisten muß. Entsprechend vielfältig ist die Auswahl an unter Linux verfügbaren Kommandozeileninterpretern.
Sie reicht von Shells mit minimalem Umfang *ash*, über historische Exemplare *csh*, *ksh* bis hin zu den modernen und weit verbreiteten Shells *bash*, *tcsh*[2].
Sofern installiert, läßt sich auch unter Linux die entsprechende Shell nutzen. Die Loginshell ist durch einen Eintrag in der Paßwortdatei festgelegt `/etc/passwd`.

```
user1:x:500:100:example user:/home/user1:/bin/bash
user2:x:501:100:example user:/home/user2:/bin/tcsh
```

Durch Eingabe des entsprechenden Shellnamens (z.B. tcsh) kann man die aktive Shell wechseln und mittels `exit` kehrt man in die alte Shell zurück.
Möchte man seine Shell permanent verändern, steht einem das Kommando `chsh` zur Verfügung:

[1] Ein Übersetzungsversuch liefert ein Konstrukt in der Art "Wiedergeburtsmuschel":-), "Bourne" ist der Name des Entwicklers dieser Shell.
[2] Letztere findet man oft als Standardshell unter Sun/Solaris.

```
user@sonne > cat /etc/passwd
...
user:x:500:100:example user:/home/user:/bin/bash
...
user@sonne > chsh user -s /bin/tcsh
Password:
user@sonne > cat /etc/passwd
...
user:x:500:100:example user:/home/user:/bin/tcsh
...
```

5.2 Kommandoeingabe

Der grundlegende Aufbau (nach weit verbreiteter Syntax) eines Kommandos sollte wie folgt aussehen:

```
kommandoname [-Optionen] [Argumente...]
```

Optionen werden hierbei mittels "-" eingeleitet und bestehen zumeist aus nur einem Buchstaben[3]. Mehrere Optionen können gruppiert werden `ls -la`. Dem Kommando können beliebig viele Argumente übergeben werden. Meist können Optionen und Argumente in ihrer Reihenfolge vertauscht werden.
Das führende Minus zur Einleitung von Optionen wird vermutlich nach und nach entfallen (Unix98–Standard). Einige Kommandos folgen schon den neuen Richtlinen (`ps`, `tar`,...), akzeptieren aber weiterhin die alte Syntax (`ps` warnt bei Gebrauch des Minus).

Expansion von Datei– und Kommandonamen

Beispiel: Durch Eingabe von

```
user@sonne > z(TAB)
```

wird die *bash* mit einem Signalton reagieren. Durch Eingabe von (TAB) versucht die *bash* die bisher eingegebenen Zeichen eindeutig einem Datei– oder Kommandonamen zuzuordnen. Dazu durchsucht sie das aktuelle Verzeichnis und alle in der Shell–Variablen $PATH angegebenen Verzeichnisse nach entsprechenden Einträgen.

```
user@sonne > z(TAB)(TAB)
```

veranlaßt die *bash*, alle möglichen Ergänzungen am Bildschirm anzugeben:

```
user@sonne > z(TAB)(TAB)

zcat  zdiff   zgrep  zipcloak zipnote  zless  znew
zcmp  zforce  zip    zipinfo  zipsplit zmore  zsoelim
```

[3]Kommandos des X–Windows–System benutzen oft auch längere Optionen *-geometry*, GNU–Kommandos bieten oft auch die Möglichkeit der Eingabe vollständiger Namen –– *verbose*.

Ist der Name eindeutig aufzulösen, ergänzt die *bash* selbständig:

```
user@sonne > zd(TAB)
user@sonne > zdiff
```

Dieser Mechanismus funktioniert auch bei Shell–Variablen und Heimat–Verzeichnissen:

```
user@sonne > $PA(TAB)
user@sonne > $PATH
user@sonne > ls ~ro(TAB)
user@sonne > ls ~root/
```

Wichtige Tastenkürzel

Die folgende Tabelle faßt die wichtigsten in der Shell verfügbaren Tastenkombinationen zusammen.

(↑) , (↓)	durch die zuletzt eingegebenen Kommandos scrollen
(←) , (→)	Cursor bewegen
(POS 1) , (ENDE)	Cursor an Beginn/Ende der Zeile
(CTRL)+(A), (CTRL)+(E)	wie oben
(ALT)+(B) , (ALT)+(F)	Cursor um ein Wort vor/zurück
(ALT)+(D)	Wort löschen
(CTRL)+(K)	alles bis Zeilenende löschen
(CTRL)+(T)	die beiden vorangegangenen Zeichen vertauschen
(ALT)+(T)	die beiden vorangegangenen Wörter vertauschen
(CTRL)+(L)	Bildschirm löschen
(CTRL)+(R)	bereits eingebenes Kommando suchen

5.3 Alias

Zur Abkürzung immer wiederkehrender Kommandofolgen lassen sich für diese sogenannte Aliasse definieren:

```
user@sonne > pwd
/home/user

user@sonne > alias cdlin='cd /usr/src/linux'
user@sonne > cdlin
/usr/src/linux-2.0.36.SuSE

user@sonne > cd
user@sonne > pwd
/home/user

user@sonne > unalias cdlin
user@sonne > cdlin
bash: cdlin: command not found
```

Mit `alias` definiert man einen Alias und mit `unalias` läßt sich dieser wieder entfernen. Aliasse existieren bis zum Löschen oder bis zum Beenden der aktiven Shell. Möchte man einen Alias permanent einrichten, trägt man die entsprechende Befehlszeile in die Datei `.profile` in seinem Home–Verzeichnis ein. Der Aufruf von `alias` ohne Argumente bewirkt eine Auflistung aller definierten Abkürzungen.

Ein– und Ausgabeumleitung

Die *bash* kennt drei sogenannte Standarddateien:

- **Standardeingabe (0)** laufende Programme erwarten von hier ihre Eingaben (normalerweise handelt es sich um die Tastatur).
- **Standardausgabe (1)** Programme schreiben auf diese ihre Ausgaben (Bildschirm).
- **Standardfehlerausgabe (2)** Fehlerausgaben landen hier (Bildschirm, aber nur die aktive Konsole).

Ein– und Ausgaben können umgeleitet werden.
Beispiele:

```
user@sonne > ls -l > inhalt
```

schreibt die Ausgaben von `ls -l` in die Datei *inhalt*.

```
user@sonne > touch /etc/passwd 2> fehler
```

leitet die Standardfehlerausgabe (Wert 2) in die Datei *fehler*.

```
user@sonne > find /var -name "*.txt" >& datei
```

leitet Standard– und Standardfehlerausgabe in die Datei *datei* um.

5.4 Pipes

Pipes (Röhren) dienen der Verknüpfung eines Ausgabe– mit einem Eingabestrom:

```
user@sonne > cd /dev
user@sonne > ls -l | less
```

speist die Ausgabe von `ls -l` in die Eingabe von `less` (ohne diese Pipe hätte man keine Chance, die Ausgaben – ca. 1500 Zeilen – am Bildschirm zu verfolgen).
Anstelle von Pipes können auch FIFO–Dateien[4] verwendet werden (kaum gebraucht):

```
user@sonne > mkfifo fifo
user@sonne > ls -l > fifo&
user@sonne > less y fifo
user@sonne > rm fifo
```

[4] First In First Out

`ls -l` schreibt nun in die Datei *fifo*, aus dieser `less` seine Daten bezieht. Um die Shell für die nächste Eingabe frei zu bekommen, wurde `ls -l` im Hintergrund gestartet (& bewirkt diesen Effekt).
Alternativ kann man auch wie folgt vorgehen:

```
user@sonne > ls -l > fifo
^z
[1]+  Stopped                 ls --color=tty -l >fifo
user@sonne > bg
[1]+ ls --color=tty -l >fifo &
user@sonne > less < fifo
[1]+  Done                    ls --color=tty -l >fifo
```

(CTRL)(Z) stoppt die Ausführung des aktiven Prozesses (es erscheint eine Meldung: `..Stopped...`), `bg` läßt den Prozeß nun im Hintergrund laufen. Mit Beenden von `less` terminiert auch `ls -l`. Mit `fg` kann man den zuletzt in den Hintergrund beförderten Prozeß wieder in den Vordergrund holen (`fg jobnr` holt den Job mit der angegebenen Nummer in den Vordergrund).

Ausgabevervielfachung

Manchmal möchte man die Ausgaben eines Programmes in einer Datei speichern und sie gleichzeitig am Bildschirm betrachten. `tee` bewirkt eine Vervielfachung einer Ausgabe:

```
user@sonne > ls | tee inhalt
```

Die Ausgaben von *ls* werden auf dem Bildschirm angezeigt und gleichzeitig in *inhalt* gespeichert.

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

speichert die Ausgaben von `ls -l` in der Datei *inhalt1* und weist die Standardausgabe dem Kommando `sort` zu, welches den Inhalt nach dem vierten Eintrag (Dateigröße) sortiert und das Ergebnis in die Datei *inhalt2* umleitet.

Die Syntax zur Umleitung von Datenströmen sei nochmals zusammengefaßt:

`kommando > datei`	leitet die Standardausgaben in datei
`kommando < datei`	liest Eingaben aus datei
`kommando 2> datei`	leitet die Standardfehlerausgaben in datei
`kommando >& datei`	leitet Ausgaben und Fehler um
`kommando >> datei`	hängt Standardausgaben an datei an
`kommando1 | kommando2`	leitet Ausgabe von kommando1 an Eingabe von kommando2
`kommando | tee datei`	zeigt Ausgaben an und leitet diese gleichzeitig in datei

5.5 Mehrere Kommandos

Die *bash* unterstützt mehrere Möglichkeiten, mehrere Kommandos nacheinander und in Abhängigkeit voneinander zu starten:

```
user@sonne > ls; date
```

Die Kommandofolge führt zunächst das Kommando *ls* aus und zeigt dann das aktuelle Datum an.

```
user@sonne > ls; date > datei
```

In *datei* steht das aktuelle Datum.

```
user@sonne > (ls; date) > datei
```

In *datei* stehen nun der Verzeichnisinhalt und das aktuelle Datum. Die Klammerung bewirkt die Ausführung der eingeschlossenen Kommandos in derselben Shell, so daß diese ein Ergebnis zurückliefern.

Nachfolgende Tabelle faßt die weiteren Möglichkeiten zusammen:

`komm1; komm2`	führt die Kommandos nacheinander aus
`komm1 && komm2`	führt komm2 nur aus, wenn komm1 erfolgreich war
`komm1 || komm2`	führt komm2 nur aus, wenn komm1 einen Fehler liefert
`komm1 &`	führt Kommando als Hintergrundprozeß aus
`komm1 & komm2`	startet komm1 im Hintergrund, komm2 im Vordergrund
`(komm1; komm2)`	startet beide Kommandos in einer Shell

5.6 Substitution

Die einfachsten Substitutionsmechanismen der *bash* sind die Jokerzeichen ? (genau ein beliebiges Zeichen) und * (beliebig viele (auch 0) beliebige Zeichen):

```
user@sonne > ls *.tex
Buch.tex buch.tex BUCH.tex
user@sonne > ls ?uch.tex
Buch.tex buch.tex
user@sonne > ls -d /*
/System.map  /dev         /mnt       /sbin      /vmlinuz.new
/System.old  /etc         /opt       /tmp       /vmlinuz.old
/bin         /home        /proc      /usr       /winboot
/boot        /lib         /projects  /var       /windows
/cdrom       /lost+found  /root      /vmlinuz
```

Durch eckige Klammern lassen sich Jokerzeichen aus einem Bereich angeben

```
user@sonne > ls -d /*[a-c]*
/System.map  /boot   /etc  /proc      /sbin     /winboot
/bin         /cdrom  /lib  /projects  /var
```

Wird als erstes Zeichen im Bereich "^" angegeben, bedeutet dies "alle Zeichen, außer den Angegebenen":

```
user@sonne > ls -d /*[^d-z]*
/System.map  /boot   /lib        /projects /vmlinuz.new
/System.old  /cdrom  /lost+found /sbin     /vmlinuz.old
/bin         /etc    /proc       /var      /winboot
```

?	genau ein beliebiges Zeichen
*	beliebig viele (auch 0) beliebige Zeichen
[def]	eines der Zeichen
[^def]	keines der angegebenen Zeichen
[!def]	wie oben
[a-d]	alle Zeichen aus dem Bereich
~	steht für das Heimatverzeichnis
.	aktuelles Verzeichnis
..	übergeordnetes Verzeichnis

Die *bash* setzt aus in Klammern eingeschlossenen Zeichenketten alle möglichen Kombinationen zusammen:

```
user@sonne > echo {a,b}{1,2,3}
a1 a2 a3 b1 b2 b3
user@sonne > echo {a,b}.{1,2,3}
a.1 a.2 a.3 b.1 b.2 b.3
```

Kommandosubstitution

Auch Kommandos lassen sich substituieren:

```
user@sonne > find . -name "inhalt*" | xargs ls -l
```

listet alle Dateien auf, die mit *inhalt* beginnen. `xargs` trennt hier die Ausgaben von `find` nach jedem Leerzeichen und verbindet diese mit dem nachfolgenden Kommando zu Kommandozeilen.
Mit Kommandosubstitution läßt sich dies auch wie folgt ausdrücken:

```
user@sonne > ls -l ‘find . -name "inhalt*"‘
user@sonne > ls -l $(find . -name "inhalt*")
```

Die beiden Zeilen sind semantisch äquivalent.

5.7 Shell–Variablen

Shell–Variablen dienen in erster Linie zur Steuerung von Programmen, z.B. benötigt ein X–Server Informationen, welches Display er für die Anzeige verwenden soll. Normalerweise wird beim Systemstart die Shell–Variable *DISPLAY* auf *:0.0* gesetzt, was die Standardausgabe des lokalen Rechners bezeichnet. Arbeit man z.B. über *telnet* auf einem entfernten Rechner, muß dem dortigen X–Server mitgeteilt werden, wohin er seine Ausgaben umleiten soll:

```
user@sonne > export DISPLAY=sonne.galaxis.de:0.0
```

5.7.1 Globale Shell–Variablen

Wie wir im einführenden Beispiel gesehen haben, dienen Shellvariablen dazu, das Verhalten bestimmter Programme zu beeinflussen. Eine globale Variable ist ab der Shell ihrer Einführung sichtbar, also auch in allen daraus abgeleiteten Shells und in allen darin gestarteten Programmen.

Bereits beim Systemstart werden einige Variablen global belegt. So zum Beispiel `PATH`, die die Suchpfade der Programme enthält oder `WINDOWMANAGER`, die Variable, die den Default–Windowsmanager setzt.

Manchmal entsprechen die vom Administrator getroffenen Einstellungen nicht den Vorstellungen des Nutzers. So wird dieser die Variablen überschreiben.
Um die Variablen auch außerhalb der aktiven Shell sichtbar zu machen, muß diese möglichst in der ersten aktiven Shell gesetzt werden (da alle Programme/Shells Nachfahren der Login–Shell sind). Um den Windowsmanager zu ändern und das HOME–Verzeichnis in den Suchpfad aufzunehmen, sind solche Variablen zu exportieren:

```
user@sonne > export WINDOWMANAGER=/usr/X11R6/bin/fvwm2
user@sonne > export PATH=$PATH:$HOME
```

Möchte man solche Einstellungen für alle zukünftigen Sitzungen geltend machen, trägt man obige Zeilen in eine Datei `.profile` in seinem Home–Verzeichnis ein.

5.7.2 Lokale Shell–Variablen

Für die jeweils aktive Shell lassen sich lokale Variablen vereinbaren. Solche Variablen sind einzig in dieser Shell und in den darin ausgeführten Programmen sichtbar:

```
user@sonne > var=10
user@sonne > echo $var
10

user@sonne > string="abc efg"
user@sonne > echo $string
abc efg

user@sonne > bash
user@sonne > echo $var

user@sonne > exit
user@sonne > echo $var
10
```

Lokale wie auch globale Shell–Variablen werden mittels

```
unset <variable>
```

gelöscht.

5.7.3 Spezielle Shell–Variablen

Die Bash kennt einige besondere Shellvariablen (Auswahl):

Variable	Bedeutung
?	Rückgabewert des letzten Kommandos
$	PID der akuellen Shell
!	PID des letzten Hintergrundprozesses
0	Name des ausgeführten Shellskripts
#	Anzahl der Elemente einer Liste
*	Inhalt einer Liste
1	1.Element einer Liste
⋮	⋮
9	9.Element einer Liste

Einige – zugegeben unnütze – Beispiele sollen den Sachverhalt verdeutlichen:

```
user@sonne > echo $0
-bash

user@sonne > set {a,b,c}{1,2}
user@sonne > echo $*
a1 a2 b1 b2 c1 c2

user@sonne > echo $#
6

user@sonne > echo $4
b2
```

5.8 Programmierung der Bash

Niemand wird auf die Idee kommen, mit den Mitteln der Bash ein Office–System zu programmieren. Für solche Zwecke fehlen der Programmiersprache einfach die Ausdruckmittel und selbst wenn jemand sich die Arbeit machen würde, wäre ein solches Programm furchtbar langsam...
Sinn der Programmiersprache der Bash ist z.B die Unterstützung der Automatisierung von wiederkehrenden Befehlsfolgen.

Kommen wir auf ein reales Problem zurück. Wir haben im System mit dem Programm `tar` 80 Archive erzeugt und diese anschließend mit dem Programm `gzip` komprimiert.
Die Archive bezeichneten wir mit `archiv01` bis `archiv80`. `tar` ergänzte das Suffix `.tar` und `gzip` hing nochmals `.gz` hintenan, so daß die Dateinamen nun `archivXX.tar.gz` lauten.
Um die Archive auf ein anderes (nicht vernetztes) System zu kopieren, bedienen wir uns einer MSDOS–formatierten Diskette und speichern die Archive mittels `mcopy archiv*.tar.gz a:` auf diese.
Auf dem anderen System speichern wir die Dateien zurück. Zu unserer Überraschung nennen sich diese aber nun `archi~XX.gz`, da in MSDOS zwei Punkte in einem Dateinnamen unzulässig sind und diese maximal 8 Zeichen enthalten dürfen. Unsere Namen wurden automatisch manipuliert...

Wir müssen nun also die ursprünglichen Namen restaurieren. Eine Lösung wäre, achtzig Mal eine Zeile der Art

```
user@sonne > mv archiv~01.gz archiv01.tar.gz
```

einzugeben... eine mühevolle Arbeit :-(

Alternativ läßt sich die Arbeit von einigen wenigen Shellskript–Zeilen bewerkstelligen:

```
user@sonne > for i in $(ls archiv*.gz); do
> tmp=${i#*~}
> mv $i archiv${tmp%.*}.tar.gz
> done
```

Die enthaltene Kommandosubstitution (`$(ls archiv*.gz)`) haben wir schon kennengelernt; den weiteren Mechanismen weiteren wir uns in diesem Abschnitt zuwenden.

5.8.1 Parametersubstitutionen

Die Frage: "Was ist Programmieren?" läßt sich in zwei Worten beantworten: "Werte manipulieren". Die *bash* stellt wieder einmal komplexe Mechanismen bereit, um den Inhalt von Zeichenketten zu bearbeiten.

Substitution	Wirkung (Beispiel)
`${var:-default}`	Ist `var` leer, liefert das Konstrukt die Zeichenkette *default* zurück, ansonsten den Inhalt von `var`. `var` bleibt unverändert. **Beispiel:** Einem Shellskript kann optional ein Verzeichnis als erstes Argument übergeben werden. In Abhängigkeit davon soll das Shellskript dorthin wechseln oder im aktuellen Verzeichnis bleiben. Eine mögliche Lösung wäre: `cd ${1:-./}` 1 ist eine spezielle Shell–Variable (Siehe 5.7.3).
`${var:=default}`	wie oben, der Inhalt von `var` wird auf *default* gesetzt, wenn diese leer war.
`${var:+neut}`	Ist `var` leer, bleibt die Variable unverändert, ansonsten wird die Variable mit *neu* belegt und der Wert zurückgeliefert.

`${var:?error}`	Ist `var` leer, wird der Variablennamen und error ausgegeben und das Shell–Programm beendet. Ansonsten wird der Inhalt der Variablen zurückgegeben. **Beispiel:** Ein Shellprogramm soll herausfinden, an welchem Terminal ein bestimmter Nutzer angemeldet ist. Dazu muß dem Shellskript die Nutzerkennung als Argument übergeben werden. Das Shellprogram terminiert mit einer Fehlermeldung, wenn das Argument fehlt: `user=${1:?"Usage: $0 <username>"}`

`${#var}`	Liefert die Anzahl der Zeichen in `var`.
	
	Beispiel:
	
	`user@sonne > var="Zeichenkette"`
	`user@sonne > echo ${#var}`
	`12`

`${var#muster}`	Mustervergleich, beginnend am Anfang der Variablen. Wird das Muster gefunden, wird der Inhalt der Variablen ab dem ersten Zeichen **nach** dem **ersten** gefundenen Muster zurückgegeben. Ist das Muster nicht enthalten, wird der Inhalt der Variablen geliefert.
	
	Beispiel:
	
	`user@sonne > i="archi~01.gz"`
	`user@sonne > tmp=${i#*~}`
	`user@sonne > echo $tmp`
	`01.gz`

`${var%muster}`	wie oben, der Vergleich beginnt am Ende der Variablen. Geliefert wird, falls das Muster enthalten ist, alles vom Beginn des Variableninhaltes bis zum ersten Zeichen **vor** dem Muster.
	
	Beispiel:
	
	`user@sonne > echo $tmp`
	`01.gz`
	`user@sonne > echo ${tmp%.*}`
	`01`

	Die beiden zuletzt genannten Mechanismen substituieren jeweils das erste auftretende Muster. Es läßt sich ebenfalls das längstmögliche Muster eliminieren:
	
	`user@sonne > dat=/home/user/text.ps.tgz`
	`user@sonne > echo ${dat##*/}`
	`text.ps.gz`
	
	`user@sonne > echo ${dat%%.*}`
	`/home/user/text`

5.8.2 Testen von Bedingungen

Die Formulierung von Bedingungen ist in der *bash* nicht durch einfache Ausdrücke zu realisieren, da zum einen viele Sonderzeichen (<, >) bereits reserviert sind und zum anderen alle Aktionen über ein einheitliches Kommandokonzept durchgeführt werden sollen. Deswegen exisitiert für solche Zwecke das *build—in* Kommando `test`[5].

`test` liefert 0, falls die Bedingung erfüllt ist und eine 1 sonst. Die Schreibweise in eckigen Klammern ist eine Ersatzdarstellung.

```
user@sonne > test "$i"
user@sonne > [ "$i" ]
```

gibt 0 zurück, wenn die Variable `i` belegt ist, andernfalls eine 1.

```
user@sonne > test -e $i
```

testet, ob eine Datei mit dem Namen in `i` existiert.
Für Dateien existieren folgende Tests:

Option	Bedeutung
-b/-c	Test auf Gerätedatei (Block/Character)
-d	Test auf Verzeichnis
-e	Existenz der Datei
-f	Test auf normale Datei
-p	Test auf Pipe
-r/-w/-x	Test auf Lese–/Schreib–/Ausführungsrecht
-s	Test, ob Datei nicht leer ist

Mehrere Tests können kombiniert werden:

Option	Bedeutung
!	Negation
-a	logisches UND zweier Tests
-o	logisches ODER zweier Tests

[5]Wer eigene Programme *test* benennt, sollte sich nicht wundern, wenn die Ausgabe vom Erwarteten abweicht. Das Programm muß unbedingt mit dem konkreten Suchpfad aufgerufen werden, sonst führt die Shell ihr eigenes Kommando aus!

Um sich mit dem Kommando vertraut zu machen, kann man dieses gleich auf der Shell ausprobieren:

```
user@sonne > var=5
user@sonne > test $var -gt 5; echo $?
1

user@sonne > test -f /dev/console; echo $?
1

user@sonne > test -c /dev/console; echo $?
0

user@sonne > test -e /bin/passwd -a -x /bin/passwd; echo $?
0

user@sonne > test -b /dev/null -o -c /dev/null
user@sonne > test $? -eq 0 && echo "Geraetedatei"
Geraetedatei
```

5.8.3 Werte einlesen mit read

Mit `read` werden in Shellprogrammen Variablen mit Eingabewerten belegt:

```
#!/bin/sh
# numerischen Wert einlesen
a=            # a loeschen
while [ -z "$a" ]; do
  echo -n "Geben Sie eine Zahl ein: "
  read a
  a=${a##*[^0-9,' ',-]*}
  if [ -z "$a" ]; then
     echo "Ungueltige Eingabe!"
  fi
done
echo $a
```

5.8.4 Berechnungen

Die Bash ist kein Taschenrechner. So sollte es nicht verwundern, daß sie nur elementarste Grundrechenoperationen auf ganzen Zahlen beherrscht.
Der einfachste Weg zur Durchführung einer Berechnung ist die Verbindung einer Variable mit einem bestimmten Typ:

```
user@sonne > declare -i a=3
```

Die Variable `a` ist eine Ganzzahl und wird mit 3 initialisiert. Mit einer typisierten Variablen läßt sich "ganz normal" rechnen:

```
user@sonne > a=a+3; echo $a
6
user@sonne > a=100*a; echo $a
600
```

Selten verwendet man die Typisierung einer Variablen. Meist nutzt man die implizite Konvertierung der Bash, die versucht, alle typenlose Variablen in ganze Zahlen umzuwandeln.

```
user@sonne > b=5; b=$[$b+1]; echo $b
6

user@sonne > b="Ist b keine Zahl, wird b zu 0 konvertiert"
user@sonne > echo $b
Ist b keine Zahl, wird b zu 0 konvertiert

user@sonne > b=$[$b+1]; echo $b
1

user@sonne > a="netter Versuch"; echo $a
0
```

Dollarzeichen in Verbindung mit einer Klammer besitzt in der Bash eine Sonderbedeutung, die hier nochmals zusammengefaßt sei:

`$(...)` – Kommandosubstitution
`${...}` – Parametersubstitution
`$[...]` – Berechnung

5.8.5 Verzweigungen

if und case

Mit `if`–Anweisungen steuert man in Abhängigkeit eines Ausdruckes den Programmablauf:

```
if <Bedingung>; then
   <Kommando(s)>
[elif <Bedingung>; then]
   <Kommando(s)>
[else]
   <Kommando(s)>
fi
```

Beispiel:

```
#!/bin/sh

if test $# -ne 2; then
   echo "Das Kommando benoetigt zwei Argumente!"
   exit 1
else
   echo "Parameter 1: $1, Parameter 2: $2"
fi
```

if–Verzweigungen können verschachtelt werden:

```
if [ $# -gt 2 ]; then     # Zeile 1
   # tue irgendwas
else
  if test $# -ne 2; then  # Zeile 4
     # tue etwas anderes
  fi
fi
```

Die Tests in Zeile 1 und 4 sind semantisch äquivalent, sieht man einmal davon ab, das im ersten Test überprüft wird, ob die Anzahl der Parameter größer 2 ist und im Zweiten auf "ungleich 2" getestet wird.
Um große Verschachtelungstiefen bei `if`–Konstrukten zu vermeiden, kann die `case`–Anweisung verwendet werden:

```
case "Variable" in
    <Muster1> ) <Kommando(s)> ;;
    <Muster2> ) <Kommando(s)> ;;
    <Muster3> ) <Kommando(s)> ;;
esac
```

Beispiel:

```
#!/bin/sh
option=
daten=
for i do               # Erklaerung im naechsten Abschnitt
  case "$i" in
     -* ) option="$option $i";;
     *  ) daten="$daten $i";;
  esac
done
echo "Optionen: $option"
echo "Daten: $daten"
```

`case` arbeitet mit Mustervergleich von Zeichenketten, deshalb wird der Vergleichsausdruck in Doppelanführungszeichen gesetzt (`"$i"`). Mit welchem Muster der Ausdruck zu vergleichen ist, steht vor einer schließenden runden Klammer; `-*` bedeutet also: Die Muster stimmen überein, sobald der Vergleichsausdruck mit einem Minus beginnt. Die Anweisungen ab dem ersten zutreffenden Vergleich werden abgearbeitet bis zu einem expliziten Abbruch (zwei Semikola in Folge `;;`) oder bis zum Ende der `case`–Anweisung.

5.8.6 Schleifen

for

Die `for`–Schleife arbeitet auf einer Liste, wobei eine Schleifenvariable der Reihe nach mit jedem Element der Liste belegt wird.

```
for Variable [in Liste]; do
     <Kommando(s)>
done
```

Fehlt die Angabe der Liste, wird versucht, die Schleifenvariable mit einer Liste aus der Umgebung zu verbinden. In einem Shellprogramm ist diese Liste z.B. die Liste der übergebenen Kommandozeilenparameter. Mit dem Kommando `set` kann auch explizit eine Liste erzeugt werden.

Beispiele:

```
user@sonne > for i in a b c; do echo $i; done
a
b
c

user@sonne > for i in *.tex; do cp $i $i~; done

user@sonne > for i in $(find / -name "*.tex");
> do echo $i; cp $i $i~; done

user@sonne > set 1 2 3 4
user@sonne > for i do echo $i; done
1
2
3
4
```

while

while führt die Schleife solange aus, bis die Bedingung nicht mehr erfüllt ist.

```
while <Bedingung>; do
     <Kommando(s)>
done
```

Beispiel:

```
user@sonne > i=2;z=1; while [ $z -le 5 ]; do
> echo $z $i; i=$[$i*$i]; z=$[$z+1]; done
1 2
2 4
3 16
4 256
5 65536
```

until

Im Unterschied zu *while*–Schleifen wird bei *until* die Bedingung negiert formuliert.

```
until <Bedingung>; do
     <Kommando(s)>
done
```

Beispiele:

```
user@sonne > i=1; until [ $i -gt 5 ]; do echo $i; i=$[$i+1];
> done
1
2
3
4
5

user@sonne > i=2;z=1; until [ $z -gt 5 ]; do
> echo $z $i; i=$[$i*$i]; z=$[$z+1]; done
1 2
2 4
3 16
4 256
5 65536
```

5.8.7 Ein komplexeres Beispiel

Aufgabenstellung

Im Laufe der Zeit sammelt sich in einem Dateisystem so allerhand Schrott an, seien es Sicherungsdateien der Editoren (Endung ~) oder core–Files (Name core) abgestürzter Programme oder Objekt–Dateien übersetzter Programmodule (Endung .o). Es wäre sicherlich sinnvoll, ein Shellskript zu haben, was das Dateisystem von solchem Ballast befreit.
Also schreiben wir ein Programm mit folgenden Merkmalen:

- Das Skript soll alle Verzeichnisse nach zu löschenden Kandidaten durchsuchen.
- Durch Optionen kann gesteuert werden, welche Datei–Art zu löschen ist; fehlen die Optionen, sind alle entsprechenden Arten zu löschen.
- Durch eine zusätzliche Option soll ermöglicht werden, vor jedem Löschen dieses zu bestätigen.

Eine mögliche Lösung

Das nachfolgende Shellprogramm löscht im aktuellen Verzeichnis einschließlich aller Unterverzeichnisse core–, Backup– und Objektdateien. Welche Dateien zu entfernen sind, kann in der Kommandozeile angegeben werden.

```
1    #!/bin/sh

2    object=
3    tilde=
4    core=
5    inter=

6    for i do
7        case "$i" in
8             -c ) core=core;;
9             -o ) object=.o;;
10            -t ) tilde=\~;;
11            -i ) inter=-i;;
12       esac
13   done

14   for i in $core $object $tilde; do
15      for x in $(find $(pwd) -name "*$i"); do
16          rm $inter $x
17      done
18   done
```

Erläuterungen

Zeile	Bemerkung
1	`#` leitet normalerweise einen Kommentar ein, es sei denn, es folgt ein Ausrufezeichen. Hieran erkennt die Shell, daß die nachfolgenden Zeilen mit der angegebenen Shell auszuführen sind. So wird ein Shellprogramm portabel und kann auch unter einer anderen Shell aufgerufen werden.
2–5	Löschen der Variablen, um Konflikte mit gleichnamigen globalen Variablen zu vermeiden.
6, 13	Die Schleifenvariable `i` wird mit den Kommandozeilenparametern verbunden.
7–12	Es wird getestet, welche Optionen per Kommandozeile übergeben wurden. Die entsprechenden Variablen werden ggf. belegt.
14, 18	Die Schleifenvariable wird mit der Liste der gesetzten Optionen verbunden. Jede gesetzte Variable (Option) ist somit ein Element der Liste. Ist keine Variable (Option) gesetzt, terminiert die Schleife sofort (leere Liste).
15,17	Nach den entsprechenden Dateien (`*core`,`*.o`,`*~`) wird ausgehend vom aktuellen Pfad gesucht. Die Schleifenvariable `x` wird jeweils mit dem Ergebnis von `find` belegt.
16	Die Dateien werden gelöscht. Wurde die Option `-i` in der Kommandozeile übergeben, expandiert die Zeile zu `rm -i <dateiname>`.

Eine Auflistung der wichtigsten Shellkommandos findet man im Anhang A.1.

5.8.8 Dialogboxen

Wer zum ersten Mal Linux installiert, wird mit dem Installationstool *yast*[6] konfrontiert. *yast* war in den Anfängen nichts anderes als ein Shell–Skript (und ist heute ein Binary), das die Möglichkeiten von Dialogboxen[7] ausnutzte .

Das Kommando *dialog* wird durch Optionen gesteuert:

Option	Wirkung
`--checklist`	dient der Auswahl von Optionen
`--clear`	löscht bei Beendigung des Dialogs den Bildschirm (der allerdings blau bleibt, ein Aufruf von *setterm -clear* hilft)
`--infobox`	eine Nachricht wird eingeblendet
`--inputbox`	dient der Eingabe von Daten
`--menu`	dient zur Auswahl von einem aus mehreren Menüpunkten
`--msgbox`	wie `--infobox`, allerdings muß der Nutzer die Nachricht mit "OK" bestätigen
`--textbox`	eine als Parameter übergebene Datei wird angezeigt. Kann diese nicht vollständig im Fenster dargestellt werden, werden Scrollbars eingefügt
`--title`	in der ersten Zeile des Dialogs wird ein Titel gezeigt
`--yesno`	ein YES und ein NO Button erscheinen

Aufgabenstellung

Als Beispiel wollen wir das vorherige Shellskript zum Löschen von Dateien durch die Verwendung von Dialogboxen verbessern.

Eine mögliche Lösung

Das Löschen der Dateien aus Abschnitt 5.8.7 erfolgt nun menügesteuert.

```
    #!/bin/sh

    object=
    tilde=
    core=
    inter=

1   dialog --clear --checklist \
2   'Make clean your file system' 10 70 4 \
3   '1' 'Core-Files' off \
4   '2' 'Object-Files' off \
5   '3' 'Backups' off \
```

[6] Nur bei SuSE.
[7] Der Einsatz dieser ist nur auf der Konsole sinnvoll, für X nimmt man besser Tk.

```
6    '4' 'Prompt whether to remove each file?' on 2> ~/tmp.$$

7    options=$(cat ~/tmp.$$)
8    rm ~/tmp.$$

9    for i in $options; do
10       case "$i" in
11           \"1\" ) core=core;;
12           \"2\" ) object=.o;;
13           \"3\" ) tilde=\~;;
14           \"4\" ) inter=1;;
15       esac
16   done

     for i in $core $object $tilde; do
         for x in $(find $(pwd) -name "*$i"); do
17           if [ $inter ]; then
18               dialog --yesno "Removing $x?" 5 70
19               if [ $? = 0 ]; then
20                  rm $x
21               fi
22             else
23               dialog --infobox "Removing $x" 5 70
24               rm $x
25           fi
         done
     done

26   setterm -clear
```

Erläuterungen

Zeile	Bemerkung
1–6	Ein checklist–Dialog wird begonnen. Alle Zeilen müßten eigentlich auf einer einzigen Zeile stehen, durch das Fortsetzungszeichen \ lassen sich derartige langen Zeilen aber splitten. Der Dialog erhält einen Titel (Zeile 2), Höhe (10 Zeilen), Breite (70 Spalten) und Anzahl der Einträge (4) werden vereinbart. Die vier Schalter werden beschriftet (Zeilen 3-6), die letzte Box wird vorbelegt (on). Abbildung 5.1 zeigt diesen Dialog. Bei Dialogende muß die Auswahl explizit gespeichert werden. Eine Umlenkung in eine temporäre Datei ist sinnvoll. Als (hoffentlich eindeutiger) Name dient die PID des Shellskripts ($$). Waren alle Optionen gesetzt, steht in der Datei tmp.PID Folgendes: `"1" "2" "3" "4"`
7,8	Die Optionen werden aus der temporären Datei ausgelesen, in der Variable “options” gespeichert und die temporäre Datei gelöscht.

9–16	Analog zum Abschnitt 5.8.7. Um ein Muster wie `"1"` zu vergleichen, müssen die Doppelanführungszeichen als solche gekennzeichnet werden.
17	Im interaktiven Fall soll die Abfrage vor dem Löschen als Ja/Nein–Dialog erfolgen.
18	Abfrage, ob Datei gelöscht werden soll.
19–20	Der yesno–Dialog kehrt mit Status “0” zurück, falls der yes–Button gewählt wurde ⇒ Löschen!
23	Das Löschen einer jeden Datei wird kurz in einer Infobox angezeigt.
26	Der Bildschirm wird gelöscht (bleibt sonst blau).

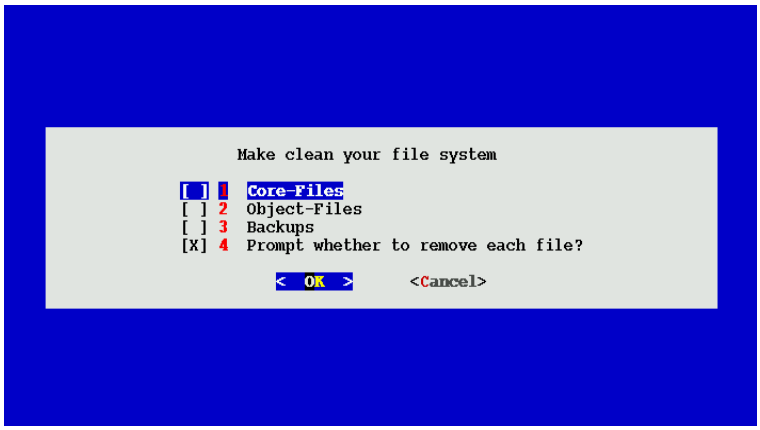

Abbildung 5.1: Eine Checkbox.

Kapitel 6

Linux–Systemadministration

6.1 Das Bootkonzept

Nachfolgende Aussagen treffen uneingeschränkt auf die Distributionen von SuSE zu; die Vorgänge beim Booten können bei anderen Distributionen geringfügig davon abweichen.

Unabhängig vom Betriebssystem kontrolliert nach dem Einschalten des Rechners zunächst das BIOS (Basic Input Output System) die Hardware und erledigt elementare Intialisierungen (Bildschirm, Tastatur). Je nach Version des BIOS werden aus dem CMOS Informationen über Peripherie (SCSI, Floppy, Harddisks), Systemzeit... entnommen. Meist ist nun die Geometrie der ersten Festplatte bekannt, so daß die ersten 512 Bytes von dieser geladen werden (der sogenannte Master Boot Record MBR). Der Aufbau des MBR ist standardisiert und besteht aus einem Bootloader (gewöhnlich reichen die vorgesehenen 446 Bytes nicht für den gesamten Code des Bootloaders aus, so daß nur ein Intialisierungsteil desselben im MBR zu finden ist) und der Partitionstabelle. Diese 64 Bytes sind der Grund für die Beschränkung auf maximal 4 primäre Partitionen. Die beiden übrigen Bytes enthalten die Zahl `AA55`, die den MBR für das BIOS als gültig deklariert.

An dieser Stelle wollen wir nicht weiter auf den Bootloader eingehen. Wir nehmen an, der Linux Loader LILO (Abschnitt 9.3) befindet sich im MBR und bestimmt das weitere Vorgehen. Er kennt den Linux–Kernel und lädt diesen in den Hauptspeicher.
Mit der Meldung `Uncompressing Linux...` übernimmt der Kernel die Kontrolle über die Hardware des Systems. Er liest Einstellungen aus dem BIOS und initialisiert die Schnittstellen. Anschließend testen und konfigurieren die im Kernel eingebundenen Treiber die vorgegebene Hardware.
Jetzt wird das Root–Filesystem einer Prüfung unterzogen und gemountet. Hat bisher alles funktioniert, wird der Vater aller Prozesse **/sbin/init** gestartet (Abbildung 6.1).

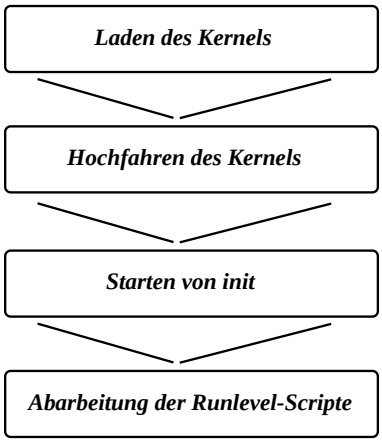

Abbildung 6.1: Ablauf des Boot–Vorganges

6.1.1 init

init ist der wichtigste Prozeß überhaupt im System. Stirbt er, ist auch Linux hinüber (was im Vergleich zu anderen Betriebssystemen eher selten der Fall ist). *init* kann auch nicht gekillt werden, nicht einmal durch das sonst tödliche Signal 9!
Was nun *init* zu tun hat, bestimmt die Datei `/etc/inittab` sowie die unter `/etc/init.d` befindlichen Skripte. Der Systemadministrator bestimmt, in welchem *Runlevel* Linux starten soll. Dazu bedient er sich entweder des Tools "Yast" *(Administration des Systems ⇒ Login–Konfiguration)*, oder aber er modifiziert den entsprechenden Eintrag in der `/etc/inittab` per Hand:

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

6.1.2 Runlevel

Durch Runlevel soll ermöglicht werden, das System in verschiedenen Zuständen zu starten[1]:

Level	Bedeutung
0	halt
S	Single User Mode
1	Multi User without Network
2	Multi User with Network (default)
3	Multi User with Network and X
4,5	free
6	reboot

Der Systemverwalter hat nun im laufenden Betrieb die Möglichkeit, durch

[1] Das Prinzip der Runlevel existiert bei allen Distributionen; die Belegung differiert allerdings (Runlevel 1 bei RedHat entspricht Runlevel S bei SuSE ...)

```
root@sonne > init S
```

das Runlevel zu wechseln. Der Wechsel der Modi dient u.a. der Pflege des Systems (Single User Mode) oder der voreingestellten Benutzung der grafischen Oberfläche (Multi User with Network and X). Durch

```
root@sonne > init 0
```

kann das System angehalten (wie `halt`) und durch

```
root@sonne > init 6
```

neu gestartet (wie `reboot`) werden.

Ein Wechsel des Runlevels im aktuellen Betrieb läuft intern wie folgt ab:

1. `init` ruft das Shellskript `/sbin/init.d/rc` mit dem neuen Runlevel auf
2. `rc` ruft daraufhin alle Stop–Skripte des aktuellen Runlevels auf (Verzeichnis `/sbin/init.d/rc`X`.d`, X entspricht dem alten Runlevel). Diese Skripte sind mit Nummern zwischen 0 und 99 versehen, die besagen, daß eine Skript mit niedriger Nummer zwingend vor dem Skript mit einer höheren Nummer abzuarbeiten ist. Bei gleicher Nummer (Priorität) ist die Reihenfolge der Ausführung nicht vorgeschrieben, erfolgt jedoch meist in alphabetischer Reihenfolge. Abbildung 6.2 zeigt die Namenskonvention der Skripte.
3. analog zum letzten Punkt werden nun von `rc` die Start–Skripte des neuen Runlevels abgearbeitet
4. bei erneutem Start desselben Runlevels überprüft `init` nur die Datei `/etc/inittab` auf Veränderungen und führt entsprechende Aktionen aus

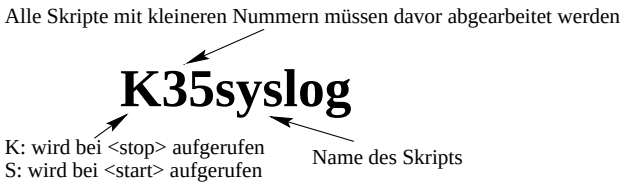

Abbildung 6.2: Namenskonvention der Skripte

6.1.3 Skripte in /sbin/init.d

Alle Skripte dieses Verzeichnissen dienen dem Wechsel eines Runlevels.

Skript	Aufgabe
`boot`	Wird direkt von `init` beim Systemstart (und nur dann) ausgeführt. Der für das Laden von Modulen zuständige Kernel–Dämon wird gestartet, die Dateisysteme auf ihre Konsistenz überprüft, P&P–Hardware wird initialisiert... Schließlich werden alle Skripte in `/sbin/init.d/boot.d` abgearbeitet.

`boot.local`	Hier können weitere Dinge vorgenommen werden, wie das Laden der deutschen Tastaturtabellen, die Einstellung von Baud–Raten der Schnittstellen (Modem)...
`boot.setup`	Einstellungen für den Übergang vom Single– in den Multiuser–Mode findet man hier.
`halt`	Beim Aufruf der Runlevel 0 oder 6 reagiert dieses Skript mit einem halt oder reboot.
`rc`	Skript, das beim Übergang von einem Runlevel in ein anderes aufgerufen wird. Zunächst werden alle Stop–Skripte des aktuellen Levels ausgeführt, dann die Start–Skripte des neuen Runlevels.

6.1.4 /etc/rc.config

`/etc/rc.config` ist die zentrale Konfigurationsdatei bei SuSE. Hier werden alle wichtigen Umgebungsvariablen gesetzt. Eine Änderung der Datei kann mittels des Programmes `yast` erledigt oder per Hand vorgenommen werden (danach sollte man `/sbin/SuSEConfig` aufrufen, damit die Änderungen wirksam werden).
Verändert man nur einige bestimmte Konfigurationen, z.B Netzwerkeinstellungen, lassen sich die Änderungen auch durch

```
root@sonne > /sbin/init.d/network stop
root@sonne > /sbin/init.d/network start
```

per Hand verbreiten.
Prinzipiell sollten Eingriffe in `/etc/rc.config` nur in den Runleveln 0 bzw. 1 erfolgen (es sei denn, man arbeitet an einem Stand–Alone–Rechner).

6.2 Verwaltung der Nutzer

Die relevanten Daten der Nutzerverwaltung befinden sich in den beiden Dateien `/etc/passwd` und `/etc/group`.

Neue Nutzer könnten durch manuelle Eingriffe in die Datei `/etc/passwd` (Siehe 6.2.1) angelegt werden, einfacher ist jedoch die Zuhilfename von *Yast (Administration des Systems ⇒ Benutzerverwaltung)* bzw. des Kommandos `useradd`

```
root@sonne > useradd  -g users -s /bin/bash new_user
root@sonne > cat /etc/passwd
...
new_user:x:1000:100::/home/new_user:/bin/bash
root@sonne > userdel new_user
```

Beim Anlegen des Nutzers sollte der Systemverwalter dessen Home erzeugen und den Inhalt von von `/etc/skel` in dessen Home–Verzeichnis kopieren, womit ihm voreingestellte Konfigurationen zur Verfügung stehen. Wie im Beispiel gezeigt, entfernt `userdel` den Nutzer; sein HOME samt aller darin befindlichen Daten ist explizit zu löschen.

Gruppen werden entweder über *Yast (Administration des Systems ⇒ Gruppenverwaltung)* erzeugt oder per Hand in `/etc/group` (Siehe 6.2.2)eingetragen.

6.2.1 Die Datei /etc/passwd

Die Daten zur Benutzerverwaltung findet man hier. Der Begriff "Benutzer" wird hier etwas weiter gefaßt, indem auch Pseudobenutzer angelegt werden, um Rechte für bestimmte Dateien/Verzeichnisse/Prozesse nur bestimmten Programmen einzuräumen. So findet man einen Benutzer "lp", hinter dem sich der Druckerdämon verbirgt. Und sucht man ihm gehörende Dateien, wird man unterhalb von `/var/spool` fündig.

Der Aufbau eines Eintrags ist immer gleich (sofern es sich nicht um einen NIS–Eintrag handelt, siehe Seite 167):

```
Username: Password: UID: GID: Info: Home: Shell
```

Username	druckbare Zeichen, meist Kleinbuchstaben
Password	*leer* Login ohne Paßwortabfrage
	x Paßwort steht in `/etc/shadow`
	sonst verschlüsseltes Paßwort
UID	nichtnegative Zahl < 64000, für normale Benutzer > 100
GID	nichtnegative Zahl < 64000
Info	meist Name des Nutzers, wird oft für Mails benutzt
Home	Startverzeichnis nach Login
Shell	Default–Shell

Daß das verschlüsselte Paßwort in der `/etc/passwd` steht, wird man nur noch auf älteren Systemen vorfinden. Das Problem ist in der allgemeinen Lesbarkeit der Datei begründet, die manche Programme benötigen. Leider stürzten sich auch die Entschlüsselungstools auf diese Erscheinung, indem sie vorab alle nur erdenklichen Paßwörter mit dem bekannten `crypt`–Algorithmus verschlüsselten und in einer Datenbank sortiert ablegten. Jetzt mußten derartige Programme "nur" noch die in den Paßwortdateien verschlüsselten Sequenzen in der Datenbank suchen und erhielten in 80% der Fälle den Klartext. Um solche Datenbankenabfragen zu vermeiden, wurde das verschlüsselte Paßwort in die lesegeschützte Datei `/etc/shadow` ausgelagert (und andere Maßnahmen wurden getroffen...).

Der Bedeutung der Felder der `/etc/shadow` unterscheidet sich von denen in der Datei `/etc/passwd`:

```
Username: Password : DOC : MinD : MaxD: Warn : Exp: Dis : Res
```

Username	identischer Eintrag wie in der `/etc/passwd`
password	verschlüsseltes Paßwort
DOC	Day of last change, Tag ab dem 1.1.1970, an dem das Paßwort zuletzt geändert wurde
MinD	minimale Anzahl Tage, die das Paßwort gültig ist
MaxD	maximale Anzahl Tage, die das Paßwort gültig ist
Warn	Anzahl der Tage vor Ablauf der Lebensdauer des Paßwortes, ab der vor dem Verfall zu warnen ist
Exp	Expire, wieviele Tage gilt Paßwort trotz Ablauf der MaxD?
Dis	bis zu diesem Tag (gezählt ab 1.1.1970) ist dieser Account gesperrt
Res	Reserve, Feld wie derzeit nicht ausgewertet

Manchmal ist es für den Systemadministrator erforderlich, den Nutzern den Zugang zum Rechner temporär zu versagen. Die Existenz einer (auch leeren) Datei `/etc/nologin` – ermöglicht das Einloggen nur noch für Root. Versucht sich ein anderer Nutzer beim System anzumelden, wird der Inhalt dieser Datei ausgegeben.

6.2.2 Die Datei /etc/group

In dieser Datei befinden sich die verschiedenen Benutzergruppen und ihre Mitglieder. Ein Eintrag besitzt folgenden Aufbau:

```
Gruppenname:Passwort:Gruppennummer:Mitgliederliste
```

Paßwort	Nutzer können bei Kenntnis die Gruppe wechseln, auch wenn sie keine Mitglied der Gruppe sind; es gelten die gleichen Aussagen wie für die Paßworte der `/etc/passwd`
Gruppennummer	nichtnegative Zahl < 64000
Mitgliederliste	durch Komma getrennte Liste der Nutzerkennzeichen

Der zweite und vierte Eintrag kann entfallen, d.h. die Gruppe ist nicht durch ein Paßwort geschützt bzw. in die Gruppe kann kein Nutzer wechseln, der diese nicht als default–Gruppe hat. Die Paßwortabfrage entfällt für Gruppenmitglieder; Paßwörter und Gruppenverwalter werden in der Datei `/etc/gshadow` gespeichert.

6.3 Die Datei /etc/login.defs

Mit dieser Datei wird das Verhalten beim Login–Vorgang gesteuert. An dieser Stelle sollen nur die wichtigsten Parameter Erwähnung finden:

```
#
# /etc/login.defs - Configuration control definitions for the
#                   login package (incomplete).
#
# Drei Eintraege muessen definiert sein:
#      MAIL_DIR   -- wo wird Mail zwischengespeichert
```

```
#      ENV_SUPATH -- minimale PATH--Vorbelegung fuer Root
#      ENV_PATH   -- minimale PATH--Vorbelegung fuer alle
# Die weiteren Eintraege sind optional.
#
# Wartezeit in Sekunden , bevor bei fehlgeschlagenem Login das
# Prompt erneut erscheint
#
FAIL_DELAY              1

# Protokollierung und Anzeige fehlgeschlagener Logins
#
FAILLOG_ENAB            yes

# Protokollierung erfolgreicher Logins
#
LOG_OK_LOGINS           no

# Zeit des letzen Logins merken und anzeigen
#
LASTLOG_ENAB            yes

# Test auf neue Mail
#
MAIL_CHECK_ENAB         yes

# Root darf sich nur auf den angegebenen Konsolen anmelden
#
CONSOLE         /etc/securetty
CONSOLE tty1:tty2:tty3:tty4:tty5:tty6:tty7:tty8

#
MAIL_DIR        /var/spool/mail

#
ENV_SUPATH      PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/X11R6/bin
ENV_PATH        PATH=/bin:/usr/bin:/usr/X11R6/bin:/usr/local/bin

#       UMASK -- Vorbelegung
#
UMASK           022

# Passwort--Kontrolle
#
#       PASS_MAX_DAYS   soviele Tage gilt Passwort maximal
#       PASS_MIN_DAYS   soviele Tage gilt Passwort mindestens
#       PASS_MIN_LEN    Mindestlaenge
#       PASS_WARN_AGE   ab wann ist vor Verfall zu warnen
#
PASS_MAX_DAYS   99999
```

```
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

#
# Min/max Werte fuer UID-Wahl von useradd
#
UID_MIN                  1000
UID_MAX                 60000

#
# Min/max Werte fuer GUID-Wahl von groupadd
#
GID_MIN                   100
GID_MAX                 60000

# Anzahl Login-Versuche (dann wird Bildschirm geloescht)
#
LOGIN_RETRIES           4

# Max. Zeitspanne fuer Eingabe von Nutzerkennung und Passwort
#
LOGIN_TIMEOUT           120

# Anzahl Versuche, ein Passwort zu aendern
#
PASS_CHANGE_TRIES       5

# Warnung vor "schwachem" Root-Passwort
#
PASS_ALWAYS_WARN        yes

# Signifikante Passwortlaenge (durch crypt beschraenkt)
#
PASS_MAX_LEN            8

# Passwortabfrage bei Programm chfn/chsh
#
CHFN_AUTH               yes

# Soll Login moeglich sein, falls nicht ins Home gewechselt
# werden kann? (z.B /home ueber NFS, NFS nicht verfuegbar)
#
DEFAULT_HOME    yes

# Globale Einstellungen koennen hier vorgenommen werden
#
#ENVIRON_FILE   /etc/environment
```

6.4 Die Datei /etc/fstab

Die Datei `/etc/fstab` enthält Parameter über das (permanente) Dateisystem. Jedes Dateisystem wird durch eine eigene Zeile beschrieben; anhand einer typischen `/etc/fstab` betrachten wir die wichtigsten Einträge:

```
/dev/hda1     /           ext2         defaults        1   1
/dev/hdb2     /usr        ext2         defaults        1   2
/dev/hda3     swap        swap         defaults        0   0

/dev/hdc      /cdrom      iso9660      ro,noauto,user  0   0
/dev/fd0      /floppy     auto         noauto,user     0   0

proc          /proc       proc         defaults        0   0
```

Drei der Einträge dieser Beispieldatei sind für alle Systeme zu empfehlen:

- Zeile 1: Betrifft das Root–Dateisystem (notwendig)
- Zeile 3: Einbinden einer Swap–Partition (optional)
- Zeile 8: Einbinden des Prozeßdateisystems (notwendig)

Alle weiteren Zeilen sind optional. Jede Zeile besteht aus 6 Spalten, die im Einzelnen folgende Bedeutung besitzen:

```
Device  Mountpoint  Type  Options  Dump  Check
```

Eintrag	Bedeutung
Device	Blockdevice oder entferntes Dateisystem, welches zu mounten ist
Mountpoint	Verzeichniseintrag, in dem das Dateisystem erscheinen soll
Type	Typ des Dateisystems (auto steht für automatische Erkennung bei Diskettenlaufwerken)
Options	Optionen (siehe nachfolgende Tabelle)
Dump	Gibt an, ob das Dateisystem vom Kommando `dump` zu sichern ist
Check	Gibt an, ob das Dateisystem vor dem mounten zu überprüfen ist. Beim Root–Dateisystem sollte hier eine "1" stehen und bei allen anderen entweder eine "0" (keine Prüfung) oder eine "2". Dateisysteme mit gleicher Nummer werden parallel überprüft, das Root–Dateisystem sollte immer allein und als erstes getestet werden.

Mit den Optionen kann der Mount–Vorgang gesteuert werden (Auswahl):

defaults	Voreinstellungen (rw, suid, auto, nouser...)
noauto	kein automatisches Mounten beim Booten
user	Device darf von normalen Nutzern gemountet werden
ro, rw	read only, read write
exec	Ausführung von Binaries gestattet
sync	ungepuffertes Schreiben

Prinzipiell lassen sich alle Dateisysteme (sofern vom Kernel unterstützt) auch per Hand mit dem Kommando `mount` einbinden, jedoch erfordert dieses Vorgehen immer die vollständige Syntax. Als Beispiel importieren wir das Home–Verzeichnis via NFS (Network File System, siehe Seite 165) von einem Rechner mit den Namen `erde.galaxis.de`. Das Dateisystem soll in einem Verzeichnis `/nfs` sichtbar sein. Die notwendige Kommadozeile lautet:

```
root@sonne > mount -t nfs erde.galaxis.de:/home /nfs
```

Möchte der Systemadministrator nun zulassen, daß auch ein normaler Nutzer dieses Verzeichnis mounten kann, trägt er eine entsprechende Zeile in die Datei `/etc/fstab` ein:

```
erde.galaxis.de:/home   /nfs   nfs   noauto,user  0  0
```

Ein solches Dateisystem darf natürlich nicht überprüft werden, da diese Aufgabe dem entfernten Rechner obliegt.
Für den Nutzer genügt jetzt folgene Kommandozeile:

```
user@sonne > mount /nfs
```

`mount` sucht die fehlenden Parameter in der Datei `/etc/fstab` und wird fündig... Alle momentan eingebundenen Dateisysteme werden in einer Datei `/etc/mtab` verwaltet.

6.5 Protokollieren von Systemmeldungen

Programmen ist es möglich, über spezielle Systemrufe dem Protokollanten von Linux `syslogd` Nachrichten zu senden, die dieser je nach Konfiguration entweder in Dateien speichert, an syslog–Dämonen anderen Rechner weiterleitet oder direkt auf die Konsole ausgibt. Vor allem bei der Fehlersuche oder bei der Aufdeckung vermuteter Sicherheitslücken können die protokollierten Daten nützlich sein. Nachrichten werden unterschieden nach ihrer Art:

Herkunft	Erläuterung
auth	Authentifizierung (z.B. login)
authpriv	vertrauliche Nachrichten der “inneren” Sicherheitsdienste
cron	Meldungen des `cron`–Dämonen
daemon	Meldungen von anderen Dämonen
kern	Kernelmitteilungen
lpr	Meldungen des Druckerdämonen
mail	Meldungen vom Mailsystem
news	Meldungen des Nachrichtensystems
security	wie *auth*
syslog	Meldungen vom `syslogd` selbst
user	Meldungen von Anwenderprogrammen
local0–local7	zur freien Verfügung

und ihrer Dringlichkeit (Priorität, aufsteigend geordnet):

Priorität	Bedeutung
none	Nachrichten dieser "Pseudopriorität" nicht beachten
debug	Debugmeldungen, innere Programmzustände
info	Informationen über das alltägliche Geschehen ...
notice	etwas Auffälliges im Normalbetrieb ...
warn	alle möglichen Warnungen ...
warning	wie *warn*
err	Fehlermeldungen aller Art
error	wie *err*
crit	kritische Fehlermeldungen, gerade noch gutgegangen ...
alert	dringend, sofortiger Eingriff notwendig
emerg	unmittelbar vor einem Systemcrash
panic	wie *emerg*

Der ganze Vorgang des Protokollierens kann durch die Datei `/etc/syslog.conf` gesteuert werden. Ein Eintrag umfaßt eine Zeile:

```
Herkunft.Prioritaet[;Herkunft.Prioritaet]   Wohin damit
```

Eine Beispieldatei soll die verschiedenen Möglichkeiten demonstrieren:

```
#
# /etc/syslog.conf - Konfigurationsdatei fuer syslogd(8)
#
# Weitere Informationen erhalt man mit "man syslog.conf".
#
# Meldungen vom Kernel der Prioritaeten warn  und hoeher, alle
# Fehlermeldungen und Meldungen  hoeherer Prioritaet jeglicher
# Herkunft (*) nach /dev/console. Meldungen der inneren
# Sicherheitdienste nicht anzeigen:

kern.warn;*.err;authpriv.none                    /dev/console

# Kurz vor einem Systemcrash diesen allen angemeldeten Nutzern
# (*) sofort anzeigen und an anderen Rechner  weiterleiten (um
# spaeter die Ursache finden zu koennen):

*.emerg                                                    *
*.emerg                                       @atairgalaxis.de

#
# alle Nachrichten des email-Systems in eine Datei
#

mail.*                                            /var/log/mail

#
# all news-System-Nachrichten in eine Datei
#
```

```
news.*                                                  /var/log/news

#
# alle Warnungen in eine Datei
#

*.warn                                                  /var/log/warn

#
# den Rest (ausser mail und news) in eine gemeinsame Datei
#

*.*;mail.none;news.none                         /var/log/messages


# ueber  fehlgeschlagene Loginversuche  wird der Adminstrator,
# sofern angemeldet, informiert

auth.*                                                          root
```

6.6 Alle Jahre wieder...

Es gibt Programme, die sollten in regelmäßigen Abständen laufen, z.B. um das Postfach auf neu ankommende Mail zu überprüfen. Hierfür existiert ein eigener Dämon `crond`, der alle in bestimmten Tabellen aufgeführten Maßnahmen zu gegebener Zeit veranlaßt.
Für den Systemverwalter besteht die Möglichkeit, bestimmte wiederkehrende Verwaltungsaufgaben in die Datei `/etc/crontab` einzutragen:

```
# Kommandos werden unter dieser Shell ausgefuehrt
#
SHELL=/bin/sh

# Kommandos werden in diesen Pfaden gesucht
#
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin

# die Ausgaben der Kommandos werden per Mail an root geleitet
#
MAILTO=root

53  6 * * * $HOME/bin/cron.daily
 0  * * * * test -x /usr/sbin/faxqclean && /usr/sbin/faxqclean
25 23 * * * test -e /usr/sbin/faxcron && sh /usr/sbin/faxcron |\
mail FaxMaster
```

Zu Beginn einer solchen `crontab` können bestimmte Einstellungen vorgenommen werden, die die Ausführung der in der Tabelle enthaltenen Kommandos steuern. Im Beispiel werden die Shell, die zur Interpretation der Kommandos

genutzt wird, die Suchpfade nach ausführbaren Programmen und die Ausgabeumleitung festgelegt.
Es folgen die Einträge der Art "Wann ist was zu tun", wobei 5 Felder das "Wann" beschreiben und der Rest das "Was". Schauen wir uns die Zeitfelder genauer an; von links nach rechts besitzen sie folgende Bedeutung:

Feld	Zeit	zulässige Werte
1	Minute	0–59
2	Stunde	0–23
3	Tag	0–31
4	Monat	0–12, jan, feb, ..., dec
5	Wochentag	0–7, sun (entspricht 0 oder 7), mon, ..., sat

Jedes Feld erlaubt auch die Eingabe mehrerer Werte. Die nachfolgende Tabelle faßt die verschiedenen Syntaxvarianten zusammen, die Beispiele beziehen sich – sofern nicht anders angegeben – auf Minuten:

	Syntax	Beispiel/Bemerkung
voller Bereich	`*`	0, 1, 2, . . . , 59
ausgewählte Bereiche	`1-5`	1, 2, 3, 4, 5
Liste	2,3,11,12	nur an den angegebenen Werten
	2,3,30-40	Kombination aus Liste und Bereich
Schrittweite	*/2	aller zwei Minuten (0, 2, . . . , 58)

Tag, Wochentag und Monat können auch als englische Namen (die ersten drei Buchstaben) angegeben werden. Klein– und Großschreibung werden dabei nicht unterschieden. Bereiche sind bei der Verwendung von Namen nicht erlaubt.
Das Editieren der `/etc/crontab` ist selbstverständlich nur dem Systemadministrator gestattet, dem gewöhnlich Nutzer steht das Kommando `crontab` zur Verfügung, das ihm eine eigene Tabelle im Verzeichnis `/var/crond/tabs` unter seinem Nutzernamen generiert. Ein Nutzer kann mit

```
user@sonne > crontab -e
```

seine eigene crontab–Datei editieren und mit

```
user@sonne > crontab -l
```

die Einträge betrachten. Ein direktes Editieren ist nicht möglich, da die Datei `root` gehört!
Um zum Beispiel den Feierabend am Rechner nicht zu verschlafen hilft:

```
user@sonne > crontab -e
no crontab for user - using an empty one
...
0 20 * * 1-5 echo -e "E.T. will nach Hause \a"
...
crontab: installing new crontab
```

das einem 20.00 Uhr an jedem Wochentag an den rechtzeitigen Abschied erinnert[2], indem ein Piepton erklingt und wir vom `crond` eine Mail erhalten.
Wir überzeugen uns noch schnell vom richtigen Eintrag:

```
user@sonne > crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.1491 installed on Fri Nov 13 13:30:46 1998)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37
vixie Exp $)
0 20 * * 1-5 echo -e "E.T. will nach Hause \a"
```

6.6.1 Neuigkeiten für den Nutzer

Linux befindet sich in steter Entwicklung und von Zeit zu Zeit wird der Systemadministrator Änderungen am System vornehmen, die jeder Nutzer kennen sollte. Z.B sollten die Nutzer auf neu integrierte Technik aufmerksam gemacht werden oder auf die Installation einer aktuellen Version eines Programmpaketes. Das per Mail abzuhandeln, ist die eine Möglichkeit. Eine Andere ist die Anzeige einer Kurzinformation nach dem erfolgten Login. Dazu tragen wir die anzuzeigende Nachricht in der Datei `/etc/motd` ein:

```
root@sonne > vi /etc/motd
Neues von Systemadministrator...
```

Systeminformationen (Version der Distribution, des Kernels...) werden oft vor dem Login–Prompt eingeblendet. Welcher Art die Anzeige sein soll, steht in der Datei `/etc/issue`:

```
user@sonne > less /etc/issue
Welcome to S.u.S.E. Linux 6.1 (i386) - Kernel \r (\l).
```

6.6.2 Was der Nutzer braucht

Während des Starts der Login–Shell besteht die Möglichkeit, globale Einstellungen jedem Nutzer verfügbar zu machen. Dem Systemadministrator steht hierfür die Datei `/etc/profile` zur Verfügung; jeder Nutzer kann diese durch eigene Einstellungen in `$HOME/.profile` überschreiben. Beide Dateien sind im Aufbau identisch, deswegen schauen wir uns Ausschnitte aus einer profile–Datei genauer an:

```
PROFILEREAD=true

umask 022

# adjust some limits (see bash(1))
ulimit -c 0             # don't create core files
ulimit -d unlimited     # max data size of a program
ulimit -s unlimited     # max stack size of a program

#
```

[2] Es wird uns erfreuen, daß zum Bearbeiten der *vi* gestartet wird.

```
# make path more comfortable
#
MACHINE=`test -x /bin/uname && /bin/uname --machine`
PATH=/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin
for DIR in ~/bin/$MACHINE ~/bin ; do
    test -d $DIR && PATH=$DIR:$PATH
done
test "$UID" = 0 && PATH=/sbin:/usr/sbin:$PATH
for DIR in /usr/openwin/bin \
           /usr/lib/java/bin \
           /var/lib/dosemu \
           /usr/games/bin \
           /usr/games \
           /opt/gnome/bin \
           /opt/kde/bin ; do
    test -d $DIR && PATH=$PATH:$DIR
done
test "$UID" = 0 || PATH="$PATH:."
export PATH

# for all programs that use the GNU readline library (bash, gdb)
if ! test -f ~/.inputrc ; then
  INPUTRC=/etc/inputrc
  export INPUTRC
fi

#
# set some environment variables
#
POVRAYOPT=-l/usr/lib/povray/include
export POVRAYOPT
test -z "$WINDOWMANAGER" && WINDOWMANAGER=/usr/X11R6/bin/startkde
export WINDOWSMANAGER
TEXINPUTS="$TEXINPUTS:~/.TeX:/usr/doc/.TeX"
export TEXINPUTS
PRINTER='lp'
export PRINTER

# Further options for the 'ls' command are in /etc/DIR_COLORS.
alias ls='ls --color=tty'
alias dir='ls -l'
alias ll='ls -l'
alias la='ls -la'
alias l='ls -alF'
alias ls-l='ls -l'
alias o='less'
alias ..='cd ..'
alias ...='cd ../..'
alias +='pushd .'
```

```
if [ -z "$KSH_VERSION" ]; then
    alias -- -='popd'
fi
alias rd=rmdir
alias md='mkdir -p'
alias unix2dos='recode lat1:ibmpc'
alias dos2unix='recode ibmpc:lat1'
alias unzip='unzip -L'
alias which='type -p'
```

Es handelt sich hierbei um einen Auszug einer globalen profile–Datei. Wünscht man eigene Einstellungen, kopiert man sich diese Datei als .profile in sein Home und editiert die entsprechenden Einstellungen. Wie man sieht, verfolgt einen die Shell–Programmierung :-)

6.6.3 Weitere Konfigurationsdateien

Nichts gesagt wurde bislang über die Einrichtung eines Netzwerkes, aber darüber und über die entsprechenden Konfigurationsdateien informieren wir uns im Kapitel 12.
Mit dem Einbinden eines Druckers und der damit verbundenen Konfiguration der Datei `/etc/printcap` befassen wir uns im folgenden Kapitel, und auch der grafischen Oberfläche – dem X–Windows–System – widmen wir uns zu einem späteren Zeitpunkt (Kapitel 8).

An dieser Stelle sollen in einem Kurzüberblick einige weitere Konfigurationsdateien im Verzeichnis `/etc` vorgestellt werden.

Datei	Inhalt
adjtime	Diese Datei enthält Daten zur Korrektur der CMOS–Uhr. Bekanntlich ist die Ganggenauigkeit fernöstlicher Quarze nicht immer das Gelbe vom Ei... aber wenigstens ist die Abweichung dieser relativ konstant. Genau diese Abweichung nutzt man, um in bestimmten Abständen die Zeit zu justieren. Diese Methode ist eher für Stand–Alone–Rechner von Bedeutung, da im Internet auf einen Zeitserver zurückgegriffen werden könnte.
aliases	Eine Konfigurationsdatei für `sendmail`. Der Aufbau einer Zeile ist `name: alias[,alias]` und dient z.B. zur Umleitung von Mail. `man aliases` gibt eine detaillierte Auskunft. Nach einer Änderung in dieser Datei ist unbedingt `newaliases` aufzurufen.
fdprm	Dient der Programmierung des Floppycontrollers mittels des Programmes `setfdprm`. Die gängigsten Formate erkennt der Kernel selbsttätig, die Datei spielt erst bei "exotischen" Parametern eine Rolle. (Siehe auch `man setfdprm`)
ld.so.conf	Suchpfade nach (shared) Bibliotheken für das Programm `ldconfig`.

magic	Merkmale von Dateien, anhand derer das Kommando `file` versucht, den Typ zu spezifizieren.
man_db.config	Suchpfade für die Manual Pages. Durch die Reihenfolge der Einträge wird auch die Suchreihenfolge und damit das Ergebnis beeinflußt. (Z.B. ist es für einen C–Programmierer sicher sinnvoll, die Seiten der Sektion 3 (Bibliotheksroutinen) zunächst auszuwerten.)
securetty	Dient der Zugangsbeschränkung von Root. Hier kann angegeben werden, unter welchem Terminal sich Root anmelden darf. Z.B. ist es üblich, ein Root–Login über das Netz zu verhindern.
shells	Hier stehen verfügbare bzw. zugelassene Shells.

Kapitel 7

Drucken unter Linux

Das Einrichten eines Druckers unter Linux erscheint auf den ersten Blick schwierig. Die hinter dem kompliziert anmutenden System stehende Idee aber ist simpel und gibt dem Administrator alle Freiheiten bezüglich der Konfiguration. Um diese Konfigurationsmöglichkeiten soll es in diesem Abschnitt gehen.
Der bzw. die Drucker eines Linuxsystems werden über sogenannte Druckerwarteschlangen angesprochen, die der Dämon `lpd` verwaltet, wobei einem Drucker mehrere Warteschlangen zugeordnet sein können. Das ist z.B. sinnvoll, wenn der Drucker mehrere Sprachen versteht (z.B. PCL[1] und PS[2]) und über diese beiden auch angesprochen werden soll. Ein weiterer Vorteil des Warteschlangensystems ist, daß Druckjobs unabhängig vom Drucker abgeschickt werden können. Ist ein Drucker z.B. zur Zeit nicht eingeschaltet, so speichert das System die Jobs solange, bis der Drucker wieder verfügbar ist.
Das Drucksystem realisiert gleichzeitig – für den Nutzer transparent – die Anbindung von Netzwerk–Druckern.

7.1 Der Druckerdämon lpd

Der Druckerdämon `lpd` steuert alle Ausgaben an den Drucker. Er wird mittels der Datei `/etc/printcap` konfiguriert. Viele Distributionen bieten Programme zur Einstellung dieser Datei an. Da diese Beschreibung distributionsübergreifend einsetzbar sein soll, wird hier der Aufbau und die Funktion einzelner Einträge am Beispiel erläutert. Folgende wichtige Einträge sind notwendig:

erster Eintrag	Name der Druckerwarteschlange
lp	lokales Ausgabegerät (z.B. /dev/lp1 = LPT1)
rm	Netzwerkadresse des Remote–Druckers (wird dieser Eintrag benutzt, muß "*lp*=" gesetzt werden !!!)
rp	Name des Netzwerk–Druckers (z.B. auf einem anderen Linux–Rechner der Name der Druckerwarteschlange)
sd	Verzeichnis, in dem Jobs zwischengespeichert werden
lf	Logdatei

[1] PCL – Printer Control Language
[2] PS – Postscript

if	Inputfilter (siehe 7.2)
of	Outputfilter (siehe 7.2)
mx	Maximale Dateigröße des Jobs
sh	Unterdrückung der Statusseite

Eine weitergehende Beschreibung findet man in der Manual Page zur Datei `/etc/printcap`.

Das folgende Beipiel zeigt eine Konfiguration mit drei Warteschlagen. Die erste Warteschlange ist für einen lokalen, die zweite für einen entfernten Drucker konfiguriert. Bei Verwendung von `apsfilter` werden für verschiedene Dateiarten (ASCII, DVI, PS, PCL, GIF, ...) automatisch Einträge in der Datei `/etc/printcap` erzeugt, die die Konvertierung in das Druckerformat beschreiben. Der dritte Eintrag zeigt eine derartige Warteschlange.

Beispiel:

```
# /etc/printcap on sonne.galaxis.de
#
# lokaler Drucker mit Namen lp an /dev/lp1 (LPT1)
lp:\
  :lp=/dev/lp1:\
  :sd=/var/spool/lp:\
  :lf=/var/spool/lp/log:\
  :af=/var/spool/lp/acct:\
  :if=/var/spool/lp/mein_filter:\
  :mx#0:\
  :sh:
#
# Remote-Drucker am Rechner 191.168.1.200, Warteschlange lp
lp-remote:\
  :lp=:\
  :rm=191.168.1.200:\
  :rp=lp:\
  :lf=/var/spool/lp-remote/log:\
  :af=/var/spool/lp-remote/acct:\
  :if=:\
  :mx#0:\
  :sh:
#
# Ein von apsfilter erzeugter Eintrag zur automatischen
# Konvertierung von ASCII-Dateien ins Druckerformat Laserjet 4
#
ascii|lp1|ljet4-a4-ascii-mono-600|ljet4 a4 ascii mono 600:\
        :lp=/dev/lp1:\
        :sd=/var/spool/lpd/ljet4-a4-ascii-mono-600:\
        :lf=/var/spool/lpd/ljet4-a4-ascii-mono-600/log:\
        :af=/var/spool/lpd/ljet4-a4-ascii-mono-600/acct:\
        :if=/var/lib/apsfilter/bin/ljet4-a4-ascii-mono-600:\
```

```
        :la:mx#0:\
        :sh:sf:
#
```

7.2 Filter

Wie schon erwähnt, werden alle Dateien über sogenannte Input– und Outputfilter geleitet. Diese Filter können vom Nutzer frei definiert bzw. sogar selbst geschrieben werden. Ein Inputfilter könnte z.B. das Format der zu druckenden Datei feststellen und diese dann in ein Standardformat (z.B. ps[3]) umwandeln. Ein Outputfilter könnte dann dieses Standardformat in eine für den speziellen Drucker nötige Sprache umwandeln. Zu jeder Distribution gehört normalerweise ein Filterpaket, das eine sinnvolle Konfiguration erlaubt.
Das Paket *apsfilter* ist sehr gut geeignet und läßt sich auf allen Plattformen und Distributionen sehr gut integrieren. Das Paket enthält außerdem ein Setuptool, mit dem alle Einstellungen menugesteuert realisiert werden können. Konfiguriert wird der `apsfilter` entweder mittels `yast`[4] oder dem Shellskript `/var/lib/apsfilter/SETUP`.

7.3 Administration mittels lpc

Das Programm `lpc` dient zum Managen des `lpd`–Dämons auch während der Laufzeit. Mit ihm können z.B. einzelne Druckerwarteschlangen gestoppt oder gelöscht, die Reihenfolge der Abarbeitung verändert ... werden.

```
root@sonne > lpc
lpc> ?
Commands may be abbreviated.  Commands are:

abort   enable  disable help    restart status  topq    ?
clean   exit    down    quit    start   stop    up
lpc> status
ascii:
        queuing is enabled
        printing is enabled
        no entries
        printer idle
lp:
        queuing is enabled
        printing is enabled
        no entries
        printer idle
raw:
        queuing is enabled
        printing is enabled
        no entries
```

[3] PS – Postscript
[4] Nur bei SuSE.

```
        printer idle
lpc> exit
```

7.4 Drucken mit `lpr,lpq,lprm`

Das Kommando `lpr` fügt einer Druckerwarteschlange einen neuen Auftrag hinzu. Der Druckjob wird in die mit "-P" angegebene Druckerwarteschlange oder, wenn die Option nicht mitgegeben wurde, in die Standard–Druckerschlange, die mit Hilfe der Umgebungvariablen `PRINTER` gesetzt wird, eingereiht.

Beispiel:

```
user@sonne > lpr -Plp1 mein_lebenslauf.ps
```

Unter Umgehung der Durckerwarteschlangen kann man auch direkt auf die Schnittstelle schreiben (im Beispiel wird angenommen, daß der Drucker an der ersten parallelen Schnitstelle angeschlossen ist):

```
user@sonne > cat Datei > /dev/lp0
```

Mit dem Kommando `lpq` kann jeder Nutzer den Status und den Inhalt einer Warteschlange abfragen. Die Angabe der Warteschlange erfolgt wieder mit der Option "-P".

Beispiel:

```
user@sonne > lpq -Plp1
no entries
```

Mit dem Kommando `lprm` entfernt man einen noch nicht bearbeiteten Job aus der Warteschlange, die wieder mit "-P" angegeben werden kann. Zum Löschen benötigt man die Jobnummer.

Beispiel:

```
user@sonne > lpq
lp is ready and printing
Rank   Owner   Job       Files                    Total Size
active user    10        standard input           53795 bytes
user@sonne > lprm 10
dfA10sonne dequeued
cfA10sonne dequeued
```

7.5 Unterstützte Drucker

Wer sich ernsthaft mit der Erstellung von Dokumenten unter Linux beschäftigt, der wird über kurz oder lang auf das Textsatzsystem TEX bzw. seiner Makro–Erweiterung LATEX zurückgreifen. `latex` erzeugt aus einem TEX–Dokument ein DVI–File (device independent), aus welchem wiederum mittels `dvips` das Postscript–Format generiert werden kann.

Verfügt man nun über einen Postscript–Drucker, läßt sich das Dokument direkt ausgeben... leider sind Postscript–Drucker relativ teuer...
Schlaue Köpfe entwickelten *Ghostscript*, ein Programm, das aus Postscript–Dateien ein für viele Drucker verständliches Ausgabeformat erzeugen kann. "Viele" bedeutet hier wirklich "nahezu alle" Druckertypen, und selbst ein nicht direkt unterstützter Drucker kann durch ein verwandtes Format zur Arbeit erzogen werden. Definitiv nicht unterstützt werden sogenannte GDI–Drucker, wie sie neuerdings für Windows angeboten werden[5]. Vor dem Kauf eines Druckers – und von peripherer Hardware allgemein – sollte man sich der gesicherten Einbindung in Linux vergewissern. Im Falle von Ghostscript findet man im Paketverzeichnis eine Datei *drivers.txt*, die alle von der jeweiligen Version direkt unterstützten Druckertypen beschreibt [6].

[5]Treiber sind angeblich in Entwicklung.

[6]Aus Lizenzgründen wurde in Linux–Distributionen nur die veraltete Version 4.03 von Ghostscript ausgeliefert. Aktuelle Pakete (>5.10, unterstützen mehr Druckermodelle) finden sich aber auf allen FTP–Servern und auch auf SuSE 6.1.

Kapitel 8

Das X–Windows–System

Das X–Windows–System wurde von DEC und dem MIT entwickelt und stellt heute einen defacto–Standard für grafische Unix–Oberflächen dar. Die erste Version (X11R1) wurde 1987 vorgestellt (Wann sprach man erstmals von Windows? :-); ab der Release 6 übernahm das X–Consortium die Verantwortung für die weitere Entwicklung.
Die grafische Oberfläche eines Linux-Systems ist *X11* [1]. Dieses, auf vielen Unixen beheimatete System, ist schichtenartig aufgebaut. Ein Hauptbestandteil ist dabei der X–Server, der der darauf aufsetzenden Software (meist ein Windowmanager) die nötigen Mechanismen bereitstellt und die Hardware abstrahiert. Ein besonderes Merkmal ist die Netzwerkfähigkeit, d.h. Anwendungen auf dem Rechner X können auf dem (sonstwo auf der Welt beheimateten) Rechner Y dargestellt werden.
In diesem Abschnitt soll kurz auf die Konfiguration des X–Servers eingegangen werden. Ebenfalls finden einige wenige Windowmanager Erwähnung, die die in ihnen laufenden Anwendungen z.B. mit Rahmen versehen und dem Nutzer einige Interaktions– und Konfigurationsmöglichkeiten bieten.

8.1 Auswahl und Konfiguration des X–Servers

Zur Auswahl des X–Server benötigt der Administrator des Systems einige Informationen zur Grafikkarte und zum Monitor. Die technischen Daten lassen sich meist aus der diesen Geräten beigelegten Dokumentation entnehmen. Eine weitere Hilfe ist das Kommando `SuperProbe`, das anhand von Tests versucht, den Grafikkartentyp sowie den zugehörigen Chipsatz herauszufinden.
Hat der Administrator diese Informationen eingeholt, kann er den entsprechenden X–Server installieren. Die meisten Kartentreiber sind im SVGA-Server enthalten. Für viele Karten existieren aber spezielle X–Server, die meist nach dem Chipsatz bzw. dem Hersteller der Karte benannt sind (z.B. XF86Com_Matrox für Matrox–Karten).
Ist der entsprechende Server installiert, muß das System noch konfiguriert werden. Dazu wird das Programm `xf86config`[2] aufgerufen.

[1] Unter Linux steht genau genommen nur ein Klon von X11R6 zur Verfügung: XFree86, der dasselbe tut, wie sein Vorbild, aber nichts kostet...

[2] Dieses Programm sollte auf allen Linux–Distributionen verfügbar sein; während z.B. SaX

Die Ausführung von *xf86config* erfordert die Rechte des Administrators!

8.2 Mauseinstellung

Zunächst fordert das Programm zur Spezifikation der Maus auf:

```
First specify a mouse protocol type. Choose one from the following list:

 1.  Microsoft compatible (2-button protocol)
 2.  Mouse Systems (3-button protocol)
 3.  Bus Mouse
 4.  PS/2 Mouse
 5.  Logitech Mouse (serial, old type, Logitech protocol)
 6.  Logitech MouseMan (Microsoft compatible)
 7.  MM Series
 8.  MM HitTablet
 9.  Microsoft IntelliMouse
```

Den entsprechenden Typ entnimmt man am besten aus der Beschreibung. Man beachte, daß z.B viele Mäuse von Logitech als PS/2 zu konfigurieren sind (i.A. ist für die meisten Maustypen entweder die Option 1 oder 4 zu wählen).
Als Nächstes wird (bei Option 1) gefragt, ob ein dritter Button simuliert werden soll (indem man beide Buttons gleichzeitig drückt). Diese Option ist bei 2–Button–Mäusen zu bejahen. Bei Option 4 wird nur nach Aktivierung der dritten Taste gefragt.
Das Device sollte in der Voreinstellung übernommen werden.

8.3 Tastatur

Für deutsche Tastaturen ist die Aktivierung der ALT– und ALTGR–Tasten zu empfehlen, um unter X auf deutsche Sonderzeichen zugreifen zu können. Außerdem ist das Protokoll 5 zu wählen.
Später läßt sich die Tastaturbelegung mit dem Kommando `xmodmap` anpassen (so kann man auch die ansonsten unnützen Tasten einer Windows–Tastatur mit eigenen Funktionen belegen).

8.4 Monitor

Die Frequenzen für den Monitor erfährt man aus dem Handbuch; für verschiedene Modelle findet man eventuell unter /usr/X11R6/lib/X1/doc/Monitors die korrekten Werte.

```
   hsync in kHz; monitor type with characteristic modes
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87Hz interlaced (no 800x600)
 4  31.5,35.15,35.5; SuperVGA, 1024x768@ 87Hz interlaced, 800x600 @ 56Hz
 5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
```

nur auf SuSE existiert.

```
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
 8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
 9  31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10  31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11  Enter your own horizontal sync range
```

Analog verfährt man mit den vertikalen Frequenzen. Die Angaben eines Monitornamens u.a. kann man ignorieren.

8.5 Grafikkarte/Server

```
Do you want to look at the card database?
```

Jetzt sollte man hoffen, daß man seine Karte in der Liste der unterstützten Grafikkarten auch findet. Die gewählte Option sollte der Karte zu einhundert Prozent entsprechen, sonst ist für das Überleben dieser nicht zu garantieren!
Steht die Karte nicht zur Verfügung, ist als Option "q" zu wählen.
Ein der Karte zugeordneter X–Server muß nachfolgend angegeben werden. Wurde die Karte nicht in der Liste erwähnt, kann man hier entweder den Monochrome– oder den 16–Farb–Server auswählen, ansonsten nimmt man den am besten geeigneten Server (wird oft bei Wahl der Karte vorgeschlagen).
Hat man sich für einen Server entschieden, läßt man vom Programm den Link von `/usr/x11R6/bin/X` auf den Server setzen. Der Link ist ebenso in `/var/X11R6/bin` zu ziehen.

Für beschleunigte Server erfolgt an dieser Stelle eine Abfrage der konkreten Auswahl dieser.

Weitere Informationen betreffen wieder die Grafikkarte:

- Speichergröße des Video–RAMs
- Name, Hersteller, Typ – kann ignoriert werden
- RAMDAC und Clock–Chip bei beschleunigten Servern (sollte nur angegeben werden, wenn in der Grafikkartenbeschreibung diese auch exakt spezifiziert sind)

Zuletzt lassen sich die verschiedenen Grafikauflösungen für jede Farbtiefe angeben.
Die Konfiguration muß noch abgespeichert werden und anschließend sollte mit etwas Glück der X–Server zu starten sein. Wenn nicht, liegt vermutlich eine falsche Konfiguration vor.
In der Datei `/etc/XF86Config` lassen sich die Einträge per Hand optimieren, worauf hier aber nicht eingegangen werden soll.

8.6 Windowmanager

Der Windowmanager setzt auf dem X–Server auf und liefert den in ihm laufenden Programmen z.B. eine Umrandung. Außerdem gibt er deml Nutzer verschiedene Einstellungsmöglichkeiten bezüglich Menüs, Maustastenbelegungen usw.
Die folgenden Windowmanager liegen den meisten Distributionen bei:

- fvwm
- fvwm2
- fvwm95 (Windows95 nachempfunden)
- kdm (KDE)
- olvwm (Openlook)

Diese Windowmanager können über verschiedene globale und nutzereigene Konfigurationsdateien eingestellt werden.

Für die beiden gebräuchlichen Windowmanager `fvwm2` und `kwm` seien einige Konfigurationsmöglichkeiten kurz aufgeführt.

fvwm2

Die globale Einstellung wird durch die Datei `/usr/X11R6/lib/X11/fvwm2/.fvwm2rc` gesteuert. Um eigene Einstellungen vorzunehmen, speichert man sich diese Datei im Heimatverzeichnis unter dem Namen `.fvwm2rc` ab.

Wer sich die soeben kopierte Datei zu Gemüte führt, wird vielleicht vor der Fülle der Einstellungen zurückschrecken (ca. 2500 Zeilen). An dieser Stelle möchte ich auch nicht auf alle Details eingehen, sondern nur einige wenige Konfigurationen modifizieren. Wer sich mit der Materie umfassend auseinanderzusetzen wünscht, sollte einen Blick in das Manual zum `fvwm2` werfen.

Zunächst nehmen wir einige globale Einstellungen (Auswahl) vor; diese sollten zu Beginn der Datei stehen:

```
# Anzahl virtueller Desktops auf 12 (Format 4 x 3)
#
DeskTopSize 4x3

# Wie verhaelt sich die Farbdarstellung bei Mausbewegung?
#
ColormapFocus FollowsMouse

# Zeit fuer Mausklick
#
ClickTime 750

# Module befinden sich in diesem Pfad
#
ModulePath /usr/X11R6/lib/X11/fvwm2

# ... und Bitmaps hier
#
PixmapPath /usr/X11R6/include/X11/3dpixmaps/small:#...weitere

# Pfade fuer Icons
#
```

```
IconPath /usr/X11R6/include/X11/bitmaps

# Farbe Fenstertitel und Fensterrahmen
#
HilightColor   Black CadetBlue

# Schriften fuer Fenster und Icons
#
WindowFont -misc-fixed-bold-r-normal-*-13-*-75-75-c-80-iso8859-1
IconFont -misc-fixed-medium-r-normal-*-10-*-75-75-c-60-iso8859-1

# Fokusiertes Fenster in den Vordergrund (Autoraise)
#
Module FvwmAuto 200

#
```

Damit ist die Fülle globaler Einstellungen noch lange nicht erschöpft, so lassen sich z.B die Icons beliebig positionieren, mit der Maus über den Bildschirmrand hinaus in andere Desktops scrollen... Wir wenden uns aber den Einträgen in den Menüs zu.

Startet man den `fvwm2` erscheint ein oberer Button, dessen Optik durch ein Modul mit dem Namen `FvwmButtons` gesteuert wird. Schauen wir uns die einzelnen Möglichkeiten zu dessen Konfiguration genauer an:

```
# generelles Aussehen:
#
*FvwmButtonsFont     6x13       # Schriftgroesse
*FvwmButtonsFore     Black      # Schriftfarbe
*FvwmButtonsBack     grey67     # Hintergrundfarbe
*FvwmButtonsGeometry +0+0       # relative Bildschirmposition

# Anzahl Spalten fuer den Button
#
*FvwmButtonsRows     1

# Groesse des Feldes fuer den virtuellen Desktop und
# absolute Position innerhalb des Buttons (0 0)
#
*FvwmButtons (2x1) - - Swallow "FvwmPager" Module FvwmPager 0 0

# manche Buttons werden direkt mit Programmen verbunden
# Position ist relativ angegeben (+0+0) "linksbuendig anordnen"
#
*FvwmButtons  - - Swallow "FvwmXeyes" Exec xeyes -name
"FvwmXeyes" -geometry +0+0 -bg grey67 &

# manche Buttons werden mit einem Menue verbunden
# *FvwmButtons <Menuetext> <Bitmap> Function <Functionname>
#
```

```
*FvwmButtons Fvwm2... window3d.xpm Function barthiswmpopupfunc

#
```

Abbildung 8.1: FvwmButtons

Damit stehen uns alle Kenntnisse zur Verfügung, um einen eigenes Menü hinzuzufügen. In diesem sollen die Editoren `vi` und `xemacs` erscheinen. Außerdem ändern wir die Wirkung des Mailbuttons dahingehend, daß nun nicht das Programm `pine`, sondern der Mailclient von Netscape (dieser ist allerdings noch zu konfigurieren) gestartet wird (Abbildung 8.1):

```
# Modifizierung des FvwmButtons fuer den Mailaufruf
#
*FvwmButtons  - - Swallow "coolmail" Exec coolmail
-geometry +0+0 -vol 100 -int 12 -e "netscape -messanger" &

# Erzeugen eines Eintrages in der Buttonleiste:
#
*FvwmButtons Editors... pencil_3d.xpm Function bareditpopupfunc

# Was soll die Funktion realisieren? Sie soll ein Untermenue
# oeffnen.Das "I" bewirkt, dass bei Fokusierung des Buttons jede
# Aktion (beliebiger Mausklick, Enter) die Funktion ausloest:
#
AddToFunc bareditpopupfunc
 + "I"   Popup bareditpopup

# das Untermenue "bareditpopup" muss noch erzeugt werden
#
AddToMenu bareditpopup "Editors" Title
 + "Vi*wordprocess_3d.xpm*"   Exec xterm -e vi
 + "Xemacs*emacs_3d.xpm*"  Exec xemacs

#
```

Nach dem Neustart des `fvwm2` steht uns nun ein Button “Editors...” zur Verfügung. Klicken wir auf diesen, öffnet sich ein Untermenü mit der Überschrift “Editor” und den beiden Einträgen “Vi” (als Bitmap dient wordprocess_3d.xpm) und “Xemacs” (emacs_3d.xpm). Die Auswahl einer der Einträge bewirkt den Start von `xemacs` bzw. eines `xterm`, in dem wiederum der `vi` gestartet wird.
Betrachten wir nun noch, wie ein Eintrag in ein mit der Maus erreichbares Menü gelangt. Dazu verfolgen wir die Realisierung des Menüs, das mittels der rechten Maustaste eröffnet wird:

```
# Die rechte Maustaste wird an das Menue gebunden
#
Mouse 3  R  A  Menu  allpopup Nop

# ueber die Tasten ALT-F3 ist ebenfalls das Menue erreichbar
#
Key   F3 A  M  Menu  allpopup

# das Menue selbst...
#
AddToMenu allpopup "Alles" Title

 + "&System-Werkzeuge%small.sysinfo_3d.xpm%" Popup systempopup
 + "&Programmieren%small.program_3d.xpm%" Popup develpopup
 + "Ed&itoren%small.TextEditor.xpm%"   Popup editorpopup
 + "Prod&uctivity%small.edit_doc_3d.xpm%" Popup prodpopup
 + "&Multimedia%small.Multimedia3.xpm%"   Popup multimediapopup
 + "&Dokumentation%small.helpme_3d.xpm%"  Popup helppopup
 + "Bildschi&rm%small.window3d.xpm%"   Popup wmpopup
 + "Spi&ele%small.Game.xpm%"  Popup gamespopup

# als Beispiel betrachten wir das Untermenue <helppopup>
#
AddToMenu helppopup "Dokumentation" Title

 + "&SuSE Hilfesystem%small.logo_suse_3d.xpm%"  Exec hilfe
 +   "" Nop   # Linie darstellen
 + "&XMan%small.xman3d.xpm%"  Exec xman
 + "&TkMan%small.library_3d.xpm%"   Exec tkman
 +   "" Nop
 + "&Howto%small.question2_3d.xpm%" Popup h

#
```

Wie man erkennt, wird mit

```
AddToMenu <menuname> "<Text>" Title
```

ein neues Menü eingeleitet und jeder Eintrag besitzt die Form

```
+ "<&Text>%<Bitmap>%" <was ist zu tun>.
```

Ein vorgestelltes `&` erklärt den nachfolgenden Buchstaben zum Shortcut für den Eintrag; die zugehörige Bitmap wird zwischen Prozentzeichen `%` angegeben. Schließlich wird die auszuführende Aktion angegeben. Hierbei kann es sich um die Ausführung eines Programmes handeln `Exec <programm>`, um ein weiteres Untermenü `Popup <subpopupname>`, oder aber um keine Operation `Nop`.

kwm

Die Konfiguration des Windowmanagers des KDE unterscheidet sich stark von der der meisten anderen Manager. Startet man zum ersten Mal den `kwm`, wird

im Heimatverzeichnis ein Unterverzeichnis `.kde` und eine Datei `.kderc` angelegt. Letztere enthält nur globale Einstellungen zu Farben der verschiedenen Elemente. Die interessanten Konfigurationsdateien befinden sich im Unterverzeichnis `~/.kde/share/config`.

```
user@sonne > ls ~/.kde/share/config
kbanner.kssrc     kdisplayrc      klines.kssrc     kslip.kssrc
kbgndwmrc         kflame.kssrc    klissie.kssrc    kssrc
kblankscrn.kssrc  kfmrc           kpanelrc         kswarm.kssrc
kblob.kssrc       kforest.kssrc   kpolygon.kssrc   kvtrc
kbouboule.kssrc   kghostviewrc    kpyro.kssrc      kvtrc.1
kcmbellrc         khop.kssrc      krock.kssrc      kwmrc
kcminputrc        kioslaverc      krootwmrc        kwmsoundrc
kcontrolrc        klaser.kssrc    kscience.kssrc
```

Aber auch diese bearbeitet man besser nicht von Hand, sondern nutzt die KDE–eigenen Konfigurationstools.

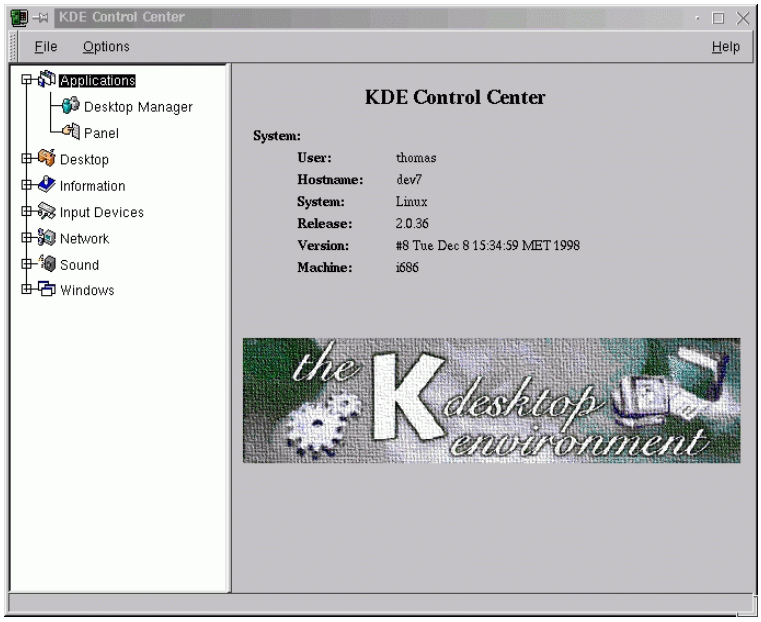

Abbildung 8.2: KDE Control Center

Erster Anlaufpunkt ist das “KDE Control Center” (Abbildung 8.2). Hier lassen sich Einstellungen zum Aussehen des Windowmanagers, Tastaturbelegung, Mausverhalten, Sound ... konfigurieren.

Um die Anzahl virtueller Desktops zu verändern, klickt man entweder mit der rechten Maustaste auf die Menüleiste des unteren Randes (Configure), oder man geht über das Symbol “K⇒Panel⇒Configure” oder man wechselt im Kontrollcenter zu Applications⇒Panel. Unter Desktops läßt sich nun die Anzahl zwischen 2 und 8 variieren.

Abbildung 8.3: KDE Taskleiste

Ins über das Symbol "K" der Taskleiste (Abbildung 8.3) erreichbare Menü lassen sich auch eigene Anwendungen integrieren. Dazu ist über "K⇒Panel ⇒Edit Menus" der Menu Editor zu öffnen. Handelt es sich um den ersten Eintrag, wird ein Feld "EMPTY" erscheinen. Mit der rechten Maustaste ist auf dieses zu klicken und "Change" zu wählen. Im selbsterklärenden Dialog ist der neue Eintrag entsprechend einzutragen. Um einen weiteren Eintrag hinzuzufügen muß "New" gewählt werden. Es entsteht ein neuer "EMPTY"–Eintrag, der wiederum anzupassen ist.

Einen neuen Eintrag in die untere Taskleiste erzeugt man über "K⇒Panel⇒Add application". Entweder wählt man hier eine Anwendung aus dem Menü aus, oder man wählt den Eintrag "EMPTY" an und modifiziert diesen anschließend auf der Taskleiste (rechte Maustaste⇒Properties).

Alle momentan geöffneten Anwendungen lassen sich in ein einziges Menü integrieren, indem "K⇒Panel⇒Add windowlist" gewählt wird. Das neue Menü erscheint auf der unteren Taskleiste.

Optionen beim Server–Start

Bei Start von X kann der gewünschte Manager angegeben werden (die Angabe ist notwendig, sofern nicht der default–Manager gewünscht wird):

```
user@sonne > startx kdm
```

Ebenso läßt sich die verwendete Farbtiefe einstellen (manche Applikationen arbeiten nicht mit allen Farbtiefen zusammen):

```
user@sonne > startx fvwm95 -- -bpp 16
```

`--` trennt hier die "normalen" Optionen von Optionen, die den X–Server betreffen. Und schließlich ist er für die Darstellung der Farben verantwortlich...

8.7 Login unter X11 – xdm

Der `xdm`–Dämon stellt dem Nutzer ein grafisches Login zur Verfügung. Der bei den meisten Distributionen dabeiliegende `xdm` ist sinnvoll voreingestellt und sollte bei Aufruf des entsprechenden Runlevels nach erfolgreicher Installation und Konfiguration der X–Servers (z.B. Runlevel 3, Einstellung in /etc/inittab) starten.
Die Konfigurationsdateien des `xdm` befinden sich im selben Verzeichnis wie die der Windowmanager (z.B. "/usr/X11R6/lib/X11/xdm" bei SuSE). Starteinstellungen werden vor allem in der Datei "Xsession" (global) und ".xsessionrc" (im Home–Verzeichnis) vorgegeben.

Kapitel 9

Der Kernel

Spricht man von Linux, so meint man den Kernel, die Shells, die Hilfsprogramme... Spricht man vom Betriebssystem Linux, denkt man ebenso an die Gesamtheit der mit einer Distribution ausgelieferten Pakete.
"Betriebssystem" und "Linux" werden meist synonym verwandt, obwohl ein Betriebssystem im Sinne seiner Definition einzig wenige Grundfunktionalitäten realisert (z.B. Prozeß–, Dateiverwaltung).

Das eigentliche Betriebssystem Linux ist einzig der Kernel. Dieser stellt alle nötigen Systemfunktionen für alle Programme bereit. Dieses Kapitel soll einen Einblick in den Kernel geben.

9.1 Nachladbare Funktionalität - Module

Damit die gesamte im System installierte Hardware unterstützt wurde, mußte der Kernel früher mit den entsprechenden Treibern konfiguriert und kompiliert werden. Seit Kernelversion 2.0 ist es möglich, fast alle Treiber auch als dynamisch ladbar bereitzustellen. Diese dynamisch ladbaren Treiber nennt man *Module*.

Die Module können von `root` mit Hilfe des Kommandos `insmod` geladen und mit dem Kommando `rmmod` wieder aus dem Kernel entfernt werden. Das erheblich komfortablere Kommando, das neben Anderem beide Funktionalitäten enthält, ist `modprobe`. Dieses Kommando lädt Module und erlaubt diesen das automatische Erkennen spezifischer Hardwareeinstellungen wie Interrupt oder Portadresse. Schlägt dieses "Autoprobing" fehl, kann der Administrator die Werte dem Modul fest in der Datei `/etc/conf.modules` zuordnen, die ebenfalls von `modprobe` durchsucht wird. Außerdem beachtet `modprobe` sogenannte "module-dependency[1]", wenn diese mit dem Kommando `depmod -a` erzeugt wurden. Dieses Kommando erstellt eine Datei, die die Zusammenhänge zwischen den Modulen enthält.

Das folgende Beispiel zeigt das Laden des Modules für eine DECchip–Tulip–(dc21x4x)–PCI Netzwerkkarte, die hier auf den Interrupt 11 und dem Port 300 konfiguriert wurde. Das Modul heißt "tulip.o".

[1] module-dependency bezeichnet den Fakt, daß ein Modul ein Anderes zum Laufen benötigt.

Beispiel:

```
root@sonne > modprobe tulip
```

lädt das Module `tulip.o` und alle vom ihm benötigten weiteren Module mit den Parametern aus der Datei `/etc/conf.modules`, wenn diese konfiguriert sind, bzw. mit der "Autoprobing"–Funktion

```
root@sonne > modprobe tulip io=0x300 irq=11
```

lädt das Module `tulip.o` und alle vom ihm benötigten weiteren Module mit den angegebenen Parametern: Portadresse 300 (**io**=0x300), Interrupt 11 (**irq**=11)

Die geladenen Module können mit dem Kommando `lsmod` angezeigt werden. Um das Laden von Modulen vom Nutzer fernzuhalten und damit die Arbeit mit dem System zu vereinfachen, wurde der `kerneld`–Dämon eingeführt. Dieser lädt die benötigten Module automatisch nach. Der `kerneld`–Dämon arbeitet wie `modprobe` auf der Datei `/etc/conf.modules` und der "Autoprobing"–Funktion der Module. Das folgende Beispiel zeigt die Einstellung für die schon erwähnte Netzwerkkarte.

Beispiel:

```
# /etc/conf.modules on master
#
# Alias, unter dem die Funktionalitaet des Modules verlangt wird.
alias eth0 tulip

# Optionen zum Modul
options tulip io=0x300 irq=11
```

9.2 Ein eigener Kernel

Bei jeder Distribution liegt ein Kernel bei, der allen Ansprüchen an Treiber usw. gerecht werden sollte. Durch den Modulsupport können fast alle Treiber dynamisch nachgeladen werden. Aus welchem Grund sollte man also einen eigenen Kernel bauen? Folgende Punkte sprechen für einen neuen Kernel:

- nur die benötigten Treiber und Grundfunktionen werden eingebunden; das verkleinert (und beschleunigt) den Kernel
- spezielle Eigenschaften können nur direkt im Kernel konfiguriert werden (z.B. Debug–Unterstützung, Automounter–Funktionalität usw.)
- der Kernel wird für den entsprechenden Prozessor optimiert
- neueste Kernel stehen immer als Sourcen, aber erst viel später fertig kompiliert zur Verfügung

Aus diesen Gründen soll hier kurz auf die Konfiguration und das Kompilieren eines eigenen Kernels eingegangen werden.

Alle folgenden Befehle können nur als `root` ausgeführt werden. Zuerst wird der neue Kernel `linux-x.x.x.tar.gz` (z.B. `linux--2.2.5.tar.gz`) im Verzeichnis `/usr/src` entpackt[2]. Um verschiedene Kernelsourcen zu unterscheiden und deren nebeneinander zu ermöglichen, wird "per Hand" ein Verzeichnis mit der Versionsnummer des Kernels angelegt (z.B. `mkdir linux-2.2.5`) und der Link `linux` auf dieses gerichtet (`ln -s linux-2.2.5 linux`). Danach wird der Kernel entpackt (`tar -xvzf linux-2.2.5`). im Verzeichnis `/usr/src/linux` (bzw. dort, wohin der Link `linux` zeigt) ist der folgende Befehl auszuführen, der eine graphische Oberfläche zur Konfiguration des Kernels zur Verfügung stellt:

```
root@sonne > make menuconfig
```

bzw. unter der grafischen Oberfläche X:

```
root@sonne > make xconfig
```

Jetzt wird der Kernel entsprechend konfiguriert. Zu jedem Menüpunkt ist in dieser Oberfläche eine Hilfe verfügbar. Kompiliert werden der Kernel und alle Module dann mit folgendem Befehl:

```
root@sonne > make dep clean zImage modules modules_install
```

Der fertige Kernel liegt nach dem Kompilieren als Datei `zImage` im Verzeichnis `/usr/src/linux/arch/i386/boot`. Dieser Kernel ist jetzt noch als Bootkernel einzurichten (z.B. im Bootloader `LILO`, siehe Kapitel 9.3). Nach der Installation muß das System gebootet werden.

9.3 Linux Boot Loader (LILO)

Der **L**inux **B**oot **Lo**ader ist ein mächtiges Tool zum Booten von Linux und anderen Systemen. Eine vollständige Beschreibung seiner Funktionalität ist hier an dieser Stelle nicht möglich. Eine sehr gute Beschreibung aller Möglichkeiten findet man in der beigelegten Dokumentation und einem entsprechenden Boot–HowTo. Das hier erwähnte Beispiel stellt nur eine Standardsituation dar.
Der **LILO** wird entweder in den **M**aster–**B**oot–**R**ecord (**MBR**) oder an den Anfang einer Partition innerhalb der ersten 1024 Zylinder der ersten oder zweiten Festplatte geschrieben. Diese Restriktionen werden nicht vom Linux, sondern vom BIOS gesetzt.
In diesem Beispiel liegt die **root**–Partition (hier `/dev/hda5`) des Linux–Systems in einer erweiterten Partition (hier `/dev/hda3`) der ersten Festplatte. Ein anderes System (z.B. Win95 oder DOS) liegt auf der ersten primären Partition der gleichen Platte. Das folgende Beispiel der Datei `/etc/lilo.conf`, mit der der **LILO** konfiguriert wird, zeigt die Einstellungen um die DOS-Partition und die Linuxpartition (mit 2 verschiedenen Kerneln) zu booten:

[2] Meist liefern die Distributionen die Kernel–Sourcen schon mit, so daß nur auf neue Versionen "gepachted" werden muß.

Beispiel:

```
# /etc/lilo.conf on sonne
#

# Geraet wohin der LILO installiert wird
# (hier MBR der 1. Festplatte)
# (/dev/hda3 waere die 3.Partition der 1. Platte)
#
boot=/dev/hda

# gibt an, dass der LILO beim Booten auf die Eingabe
# warten soll, welches System gebootet wird
#
prompt

# wie lange am Boot-Prompt gewartet wird (in Sekunden)
# (nach Ablauf der Zeit wird das 1.Image gebootet)
#
timeout=10

# Linux-Konsole bleibt normal
#
vga = normal

# mit append="" koennen Kernelparameter uebergeben werden
# Ende der allg. Konfiguration

# --- Erstes und damit standardmaessig gebootetes Image ---

# Kernel, der gebootet werden soll
#
image = /boot/vmlinuz

# Root-Partition, in der sich der Kernel befindet
#
        root = /dev/hda5

# Name, unter dem das Image am Bootprompt aufgerufen werden kann
#
        label = linux


# --- Zweites Image ---
#

image = /boot/vmlinuz_2.0.36
        root = /dev/hda5
        label = linuxold
```

```
# --- Anderes OS ---

# Partition, die gebootet werden soll
#
other = /dev/hda1

# auf welcher Platte die Partition liegt
#
        table = /dev/hda
# Name, unter dem das Image am Bootprompt aufgerufen werden kann
#
        label = Windoof

#
```

Die obige Konfiguration setzt voraus, daß das Booten aller Betriebssysteme vom Lilo aus steuerbar ist. Leider streikt bei derartiger Einstellung häufig der NT–Bootmanager, so daß man diesen zum Booten von Linux erziehen muß.

Für einen solchen Fall installiert man den Lilo nicht in den MBR, sondern in den Bootsektor der Rootpartition und konfiguriert die `lilo.conf` nur für die Linux–Images. Anschließend bootet man mit Hilfe einer Bootdiskette sein Linux und kopiert den Bootsektor in eine Datei (wir nehmen an, die Linux–Rootpartition ist `/dev/hdb2`):

```
root@sonne > dd if=/dev/hdb2 of=/tmp/bootsector bs=512 count=1
```

Die Datei `bootsector` speichert man sich auf eine von NT lesbare Diskette, z.B. eine DOS–formatierte Diskette:

```
root@sonne > mcopy /tmp/bootsector a:
```

Als Nächstes ist NT zu booten. Als Systemadministrator speichert man nun die Datei von der Diskette in ein NT–Verzeichnis und ergänzt die Datei `boot.ini` um einen entsprechenden Eintrag:

```
c:\bootsector="Linux"
```

Jetzt kann Lilo über den NT–Bootmanager aufgerufen werden.

9.4 Probleme beim Booten

Der Start des Bootmanagers erfolgt in mehreren Phasen. Erst wenn die Ausschrift `LILO boot:` erscheint, ist das Laden desselben rundum geglückt. Was aber tun, falls die Zeile nicht erscheint? LILO selbst sagt uns, wo der Fehler zu vermuten ist.
Häufigste Ursache wird eine falsche Geometrie der Festplatte sein, sei es die verrufene 1024–Zylinder–Grenze oder der gescheiterte Versuch, den Bootmanager von einer logischen Partition aus zu starten. Beide Fehler können in jeder Phase den LILO ins Nirvana schicken, aber es sind auch andere Ursachen möglich, auf die die verschiedenen Ausgaben hinweisen:

L Die Datei `/boot/boot.b` konnte nicht geladen werden. Der Datenträger könnte beschädigt sein.

LI Die Datei `/boot/boot.b` konnte geladen, aber nicht gestartet werden. Vermutlich wurde `/boot/boot.b` verschoben, der Bootmanager aber nicht neu installiert.

LIL Die Daten aus der Datei `/boot/map` lassen sich nicht lesen. Der Datenträger könnte beschädigt sein.

LIL? Die Daten des LILO wurden an eine falsche Speicheradresse geladen, vermutlich wurde wiederum LILO nach einer Kernelinstallation nicht gestartet.

LIL- Eine ungültige `/boot/map` ist die Ursache. Es wurde vergessen, den LILO neu zu installieren. Wird z.B. LILO über den NT–Bootmanager gestartet, ist dieser auf der NT–Partition nicht erneuert worden.

9.4.1 Plug and Play

Plug and Play ist eine feine Sache... Dem Slogan *Karte einbauen, Computer starten, fertig* glauben ohnehin nur noch die größten Optimisten. Auch wenn für Otto Normalverbraucher mit seiner einen Plug–and–Play–Karte tatsächlich der Administrationsaufwand schwindet, erweist sich die vom BIOS vorgenommene Initialisierung schnell als Falle, wenn sich in einem Rechner gleich mehrere solcher Karten um die Zuweisung der wenigen freien Interrupts und Adressen bewerben.

Während für Windows, DOS und Co. die Treiber für solche Karten noch mitgeliefert werden, müssen dieselben von Freaks in ihrer Freizeit erst noch für Linux entwickelt werden. Und während der Hersteller einer Karte noch weiß (wissen sollte), für welche Adressen und Interrupts er seine Treiber zurechtbastelt, beschränken sich die Linux–Treiber meist nur auf ausgewählte ("die normalen") Bereiche. Weist das BIOS einer Karte jetzt Ressourcen zu, die der Treiber nicht vorsieht, wird dieser die Hardware nicht entdecken...
Nun bleiben einem als Linux–Guru drei Offerten übrig:

- Plug and Play abschalten... funktioniert nur bei manchen Karten
- andere Karte kaufen... funktioniert nur bei manchen Geldbeuteln
- das Tool `isapnp` verwenden (das Paket liegt den meisten Distributionen bei)

Anhand der Installation einer Vibra 16 PnP Karte sollen die Schritte zur Einbindung demonstriert werden.

1. Der Treiber im Kernel ist als *Modul* zu realisieren. Am wenigsten Probleme bereitet es, wenn man die Einstellungen (Interrupt, IO–Adressen, DMA–Kanäle) entsprechend den Parametern unter DOS/Windows wählt. Ansonsten hilft nur experimentieren, wobei man vorab Interrupts bzw. IO–Adressen, die von anderer Hardware genutzt wird, ausschließen sollte.

   Die verwendeten Interrupts bereits benutzter (!) Geräte erfährt man mittels

```
user@sonne > cat /proc/interrupts
```

Vorsicht: Wurde z.B. der Drucker noch nicht angesprochen, wird der von ihm verwendete Interrupt nicht erscheinen.

Die Belegung der I/O–Ports vermittelt

```
user@sonne > cat /proc/ioports
```

2. Die Konfigurationsdatei `/etc/isapnp.conf` ist zu erzeugen:

   ```
   root@sonne > pnpdump >/etc/isapnp.conf
   ```

   `pnpdump` testet die Hardware und schreibt alle ermittelten Daten in die Datei. Diese muß nun bearbeitet werden und die Zeilen, die mit den Einstellungen im Modul korrespondieren, sind zu entkommentieren. Für eine Vibra16 PnP mit den Einstellungen IRQ=5, DMA 0 auf Kanal 1, DMA 1 auf Kanal 5, IO–Port 0x0220 sieht der entsprechende Ausschnitt wie folgt aus:

```
# ***** GEKUERZTE DATEI ****
#
# $Id: pnpdump.c,v 1.17 1998/11/10 22:45:04 fox Exp $
# This is free software, see the sources for details.
# This software has NO WARRANTY, use at your OWN RISK
#
# For details of this file format, see isapnp.conf(5)
#
# ...
#
# Board 1 has serial identifier e5 ff ff ff ff 70 00 8c 0e

# Card 1: (serial identifier e5 ff ff ff ff 70 00 8c 0e)
# Vendor Id CTL0070, No Serial Number (-1), checksum 0xE5.
# Version 1.0, Vendor version 1.0
# ANSI string -->Creative ViBRA16C PnP<--
#
# ...
#
# Don't forget to uncomment the activate (ACT Y) when happy

(CONFIGURE CTL0070/-1 (LD 0
#
# Multiple choice time, choose one only !

#     Start dependent functions: priority preferred
#       IRQ 5.
#           High true, edge sensitive interrupt (by default)
 (INT 0 (IRQ 5 (MODE +E)))
#       First DMA channel 1.
#             8 bit DMA only
```

```
#             Logical device is not a bus master
#             DMA may execute in count by byte mode
#             DMA may not execute in count by word mode
#             DMA channel speed in compatible mode
 (DMA 0 (CHANNEL 1))
#       Next DMA channel 5.
#             16 bit DMA only
#             Logical device is not a bus master
#             DMA may not execute in count by byte mode
#             DMA may execute in count by word mode
#             DMA channel speed in compatible mode
 (DMA 1 (CHANNEL 5))
#       Logical device decodes 16 bit IO address lines
#             Minimum IO base address 0x0220
#             Maximum IO base address 0x0220
#             IO base alignment 1 bytes
#             Number of IO addresses required: 16
 (IO 0 (SIZE 16) (BASE 0x0220))
#
# *** hier wurde stark gekuerzt ***
#
 (NAME "CTL0070/-1[0]{Audio              }")
 (ACT Y)
))
#
# *** es folgt weitere Hardware ***
#
```

3. Die Karte muß nun initialisiert und das Modul[3] geladen werden:

```
root@sonne > isapnp /etc/isapnp.conf
root@sonne > modprobe sound.o
```

4. Test: Ob die Karte korrekt erkannt wurde, verrät z.B. der belegte Interrupt (Alternativ zeigt `lsmod`, ob das Modul geladen wurde):

```
user@sonne > cat /proc/interrupts
 0:      90403   timer
 1:       1387   keyboard
 2:          0   cascade
 5:       2961   sound blaster
 8:          2 + rtc
10:      26394   aic7xxx
12:      35025   PS/2 Mouse
13:          1   math error
15:       1520   eth0
```

   Ist der Interrupt nicht aufgeführt, stimmt vermutlich die Ressourcenzuteilung im Modul und/oder im Konfigurationsfile nicht.

[3] Der Modulname "sound.o" ist entsprechend dem richtigen Namen zu ersetzen (Siehe Verzeichnis /lib/modules/<Kernel Version>).

5. Damit die Karte zukünftig gleich beim Hochfahren des Systems initialisiert wird, ist (bei SuSE) z.B. die Datei `boot.local` um die beiden Einträge aus Punkt 3 zu ergänzen.

Kapitel 10

Erstinstallation

Wir beschränken uns hier auf den Installationsablauf bei SuSE und die Verwendung des Tools `yast`. Viele Aussagen treffen aber unverändert auf Distributionen von *Redhat* und *Debian* zu.

Desweiteren setzen wir voraus, daß für Linux mindestens eine eigene Partition auf einer Festplatte existiert und diese unter der 1–GByte–Grenze angesiedelt ist.

Steht keine Partition für Linux zur Verfügung, kann man versuchen, die Partitionen von DOS und Windows mit dem Programm `fips` (auf erster CD) zu verkleinern und somit Platz zu schaffen. Günstig ist, zuvor die DOS/Windows–Partition zu defragmentieren.

Reicht der Platz für Linux noch immer nicht (mind. 250 MB), sollte man Windows löschen! :-)

10.1 Booten des Ur–Linux

Entweder verwendet man die mitgelieferte Bootdiskette[1], oder man bootet direkt von der ersten CD–ROM[2].

Nach Einstellungen zu Sprache, Bildschirm und Tastatur geht es um das Laden der Kernelmodule (Treiber). Hat man von CD–ROM gebootet und wünscht auch von dieser zu installieren, kann man den Punkt überspringen.

Bei Installation via NFS (Siehe 12.7) muß hier zumindest das Modul für die verwendete Netzwerkkarte geladen werden. Ebenso müssen für manche CD–ROM–Laufwerke die richtigen Treiber mit entsprechenden Parametern eingebunden werden.

Abschließend startet man die Installation `Installation starten` und wählt das Installationsmedium. In den meisten Fällen handelt es sich um die CD–ROM.

Wurde die Hardware korrekt erkannt, sollte jetzt `yast` starten.

[1] In älteren Distributionen mußten zunächst unter DOS ein oder zwei Bootdisketten erstellt werden.

[2] Ab SuSE 5.0 möglich; das BIOS muß dies unterstützen.

10.2 *Y*et *a*nother *S*etup *T*ool

`yast` begleitet uns nun durch die weitere Installation und fordert uns zur Partitionierung der Festplatte(n) auf, was bei einer Erstinstallation erforderlich ist.

10.2.1 Partitionen

In Abschnitt 3.4 wurde schon Grundsätzliches über die Partitionierung von Festplatten aufgeführt. Hier soll der Schwerpunkt auf die geeignete Anordnung in Bezug auf Linux liegen.

Eine Festplatte kann maximal *4 primäre* Partitionen (der Grund wurde in Abschnitt 6.1 beschrieben) beherbergen. Benötigt man weitere, so muß zumindest eine diese Partitionen eine *erweiterte* sein, die dann ihrerseits *mehrere logische* Partitionen enthalten kann.

Unter Linux benötigt man mindestens zwei (primäre oder logische) Partitionen. Eine dient zur Aufnahme der Root–Partition (siehe 3.3), die andere steht als Swap–Speicher zur Verfügung. Swap–Speicher dient der Auslagerung von Speicherseiten aus dem Hauptspeicher auf die Festplatte, falls im Hauptspeicher nicht genügend Platz zum Laden der Programme/Daten vorhanden ist. Als Summe von Swap–und Hauptspeicher sollte man mindestens 64MB veranschlagen, mit 128MB ist man bei den meisten Anwendungen auf der sicheren Seite. Der Typ der Partition ist bei Swap explizit zu setzen (Taste F3).

Stehen mehrere Partitionen zur Verfügung, ist es oft sinnvoll, für Teile des Dateisystems eine eigene Partition zu wählen. Z.B. reichen 100MB für das Root–Dateisystem, wenn der `/usr`–Ast und die optionalen Pakete des Verzeichnisses `/opt` auf eine eigene Partition gemountet werden. Auch für `/home` und `/var` sind eigene Partitionen oft sinnvoll. Installiert man Linux auf einer Festplatte mit mehr als 1024 Zylindern, muß der Kernel (ab SuSE 6.0 im Verzeichnis `/boot` angesiedelt) in einer Partition liegen, die sich innerhalb der ersten 1024 Zylinder befindet!

Anmerkungen: Der Zugriff auf Festplatten kostet Zeit. Deswegen empfiehlt es sich, bei Vorhandensein mehrerer Festplatten die Partitionen auf diese zu verteilen, da Zugriffe auf Daten verschiedener Partitionen dann nicht mehr konkurrierend sein müssen (und damit sind sie schneller).
Eine enorme Beschleunigung kann durch den Einsatz von Raid0 erreicht werden. Informationen hierzu findet man in den Kerneldokumentationen.

10.2.2 Mounten der Partitionen

Allen Partitionen *kann*, mindestens einer Partition *muß* ein Mountpoint zugeordnet werden. D.h. der Punkt, an dem die Partition ins Dateisystem eingebunden werden soll, muß spezifiziert werden. Die zwingend notwendige Einbindung betrifft das *root*–Filesystem (Mountpoint `/`), alle anderen Partitionen können (auch nachträglich) an beliebigen Mountpoints eingehangen werden. Alle hier getroffenen Mountpoints werden in die Datei `/etc/fstab` aufgenommen.

Nachfolgend werden die Dateisysteme auf den gemounteten Linux–Partitionen angelegt (Siehe 3.5).

10.2.3 Installationsumfang festlegen

Es stehen einige voreingestellte Konfigurationen zur Auswahl:
`Konfiguration laden`; alternativ läßt sich durch `Konfiguration erstellen/aendern` eine konkrete Paketauswahl treffen. `yast` warnt in letzterem Fall automatisch, falls ein Paket ein anderes bedingt. Zumindest die Pakete des Basissystems sollte man bei entsprechender Warnung auswählen!
Nicht installierte Pakete lassen sich nachträglich einfach integrieren.
Mit `Installation starten` erfolgt das Kopieren der Pakete auf die Festplatte(n).
Die weitere Vorgehensweise sollte aus den Meldungen von `yast` ersichtlich sein.

Kapitel 11

xemacs–Kurzanleitung

Die folgende Kurzanleitung für den Gebrauch des *xemacs* beschreibt die wichtigsten Tastenkombinationen zur Bedienung des Editors, so wie sie zum Bearbeiten von Texten benötigt werden. Einige Spezialitäten zur Unterstützung des LaTeX– und TeX–Modus werden genauer erläutert.
"Der Emacs kann alles!", ist keine übertriebene Aussage. Bei entsprechender Konfiguration läßt er sich als Mail–und News–Reader, als grafischen Frontend für den Debugger *gdb* oder als Browser für die info–Seiten einsetzen. Außerdem kommt der Emacs mit einer kleinen Spielesammlung daher ...
Hinweis: Alle nachfolgenden Tastenkombinationen funktionieren nur bei korrekter Konfiguration des Emacs. Standardmäßig verwendet der Nutzer root eine andere Konfiguration, die die wenigsten der nachfolgenden Möglichkeiten unterstützt.

11.1 Laden und Speichern

Nachfolgende Tabelle enthält die verfügbaren Tastenkombinationen zum Speichern und Laden von Datein, sowwie zum Verlassen von Emacs. Anstelle mancher Tastenkombinationen können in der X–Variante auch die entsprechenden Menüs verwendet werden:

Tastenkombination	Wirkung
Ctrl+X,Ctrl+F datei Enter	Datei laden
Ctrl+X,4,F	Datei in neuem Fenster öffnen
Ctrl+X,I	Datei unter Cursor einfügen
Ctrl+X Ctrl+S	Datei speichern
Ctrl+X,Ctrl+W datei Enter	Datei als *datei* speichern
Ctrl+X,S	alle Dateien speichern
Ctrl+X,Ctrl+C	(x)emacs beenden
Ctrl+Z	(x)emacs unterbrechen, Fortsetzung mit %(x)emacs

11.2 Positionierung des Cursors

Neben der Positionierung mit der Maus und den Cursortasten stehen zum Scrollen durch den Text noch folgende Kommandos zur Verfügung:

Tastenkombination	Wirkung
Ctrl+F	Ein Zeichen nach rechts
Ctrl+B	Ein Zeichen nach links
Ctrl+P	Eine Zeile nach oben
Ctrl+N	Eine Zeile nach unten
Ctrl+V	Eine Bildschirmseite nach unten
Alt+V	Eine Bildschirmseite nach unten

11.3 Text markieren, löschen, einfügen

Löschen von Text

Tastenkombination	Wirkung
Alt+D	löscht Zeichen des Wortes rechts ab Cursor
Alt+←	löscht Zeichen des Wortes links ab Cursor
Ctrl+K	löscht bis Zeilenende ab Cursor
Ctrl+0, Ctrl+K	löscht vom Zeilenanfang bis Cursor
Alt+M	löscht nächsten Absatz
Alt+Z,xxx	löscht alle Zeichen bis zum ersten Auftreten von *xxx*
Ctrl+Y	fügt zuletzt gelöschten Text ab Cursor ein

Markieren von Text

Tastenkombination	Wirkung
Ctrl+Leertaste	setzt Markierung
Ctrl+X,R,S register Enter	speichert Text zwischen Markierung und Cursor im Register *register*
Ctrl+X,R,I register Enter	Fügt Inhalt des Registers *register* ab Cursor ein

Vertauschen von Text

Tastenkombination	Wirkung
Ctrl+T	vertauscht Zeichen unter den Cursor mit Zeichen links vom Cursor
Alt+T	vertauscht zwei Wörter: ∘ steht Cursor an Wortanfang, wird das Wort mit den vorhergehenden vertauscht ∘ steht Cursor mitten im Wort, wird dieses mit dem nachfolgenden Wort vertauscht
Ctrl+U	letzte Vertauschung rückgängig machen
Ctrl+X,Ctrl+T	aktuelle Zeile mit vorheriger vertauschen

11.4 Suchen und Ersetzen von Text

11.5 Suche

Tastenkombination	Wirkung
Ctrl+S text	Suche vorwärts, wiederholte Eingabe von Ctrl+S wechselt zum nächsten Muster
Ctrl+R	Suche rückwärts
Alt+P	wählt früher verwendeten Suchtext aus (vorherigen)
Alt+N	wählt früher verwendeten Suchtext aus (nächsten)
Ctrl+G	Abbruch der Suche
Ctrl+Alt+S	Suche nach Muster vorwärts (siehe nachfolgende Tabelle)
Ctrl+Alt+R	Suche nach Muster rückwärts (siehe nachfolgende Tabelle)

Ähnlich zum Mechanismus der Bash erlaubt der (x)emacs das Suchen nach Mustern mit bestimmten Jokerzeichen. Zum Beispiel findet Ctrl+Alt+S \<Taste ein Wort, was mit *Taste* beginnt. Einige Musterkombinationen sind in folgender Tabelle aufgeführt:

Ausdruck	sucht nach Muster...
\<	Wortanfang
\>	Wortende
^	Zeilenanfang
$	Zeilenende

.	ein beliebiges Zeichen
.*	beliebig viele (auch Null) beliebige Zeichen
.+	beliebig viele (nicht Null) beliebige Zeichen
.?	Null oder ein beliebiges Zeichen
`[abc]`	eines der Zeichen
`[^abc]`	keines der Zeichen
`\(`	Beginn einer Gruppe
`\)`	Ende einer Gruppe
`\Sonderzeichen`	Suche nach einem Sonderzeichen

Suchen und Ersetzen

Zum Suchen und Ersetzen von ext stehen zwei Kommandos zur Verfügung:

1. (Esc)+(%) sucht ein Muster und ersetzt es durch ein anderes, jede Ersetzung muß durch (Y) bestätigt bzw durch (N) verworfen werden
2. (Alt) query-replace-r (Enter) arbeitet wie das obige Verfahren, erlaubt aber die Eingabe von Jokerzeichen im Suchmuster

11.6 Spezialitäten

Um die Beschreibung nicht unendlich auszudehnen (allein die Beschreibung der Emacs–Konfiguration mittels LISP[1] umfaßt mehrere hundert Seiten), soll hier nur eine Beschreibung ausgewählter Modi erfolgen.

Bearbeitungsmodi von emacs

Der emacs unterstützt für verschiedene Texttypen spezielle Modi, die weitere spezifische Funktionalitäten (z.B das Syntax–Highlighting) definieren.
Einige Modi sind (anhand der Dateikennung kann der Emacs oft die entsprechenden Modi selbst erkennen):

(Alt)+(X) fundamental-mode (Enter)	Standardmodus
(Alt)+(X) indented-text-mode (Enter)	Fließtextmodus
(Alt)+(X) tex-mode (Enter)	TEX–Modus
(Alt)+(X) latex-mode (Enter)	LATEX–Modus
(Alt)+(X) c-mode (Enter)	C–Modus
(Alt)+(X) tcl-mode (Enter)	Tcl–Modus

11.6.1 TEX– und LATEX–Modus

Die wichtigsten zusätzlichen Funktionen sind:

[1] **L**ist **P**rocessor

Tastenkombination	Wirkung
Ctrl+C,Ctrl+E	Erzeugt eine Umgebung `\begin{name}` – `\end{name}`; verlangt die Umgebung Optionen und Parameter, wird zur Angabe dieser aufgefordert.
Ctrl+C,}	springt zur abschließenden Klammer der aktuellen Umgebung
Ctrl+J	Ende eines Absatzes, ein Syntaxtest für den Absatz wird durchgeführt

11.6.2 Abkürzungen

Für immer wiederkehrenden (lange) Textpassagen kann eine Abkürzung definiert werden, die bei entsprechdender Einstellung des Emacs dann automatisch zum vollen Text expandiert wird. Bevor wir ein beispiel betrachten, seinen die verschiedenen Möglichkeiten in Zusammenhang mit Abkürzungen aufgelistet:

Tastenkombination	Wirkung
Ctrl+X,A,I,G text Enter	Definiert zum zuvor eingegebenen Text eine global gültige Abkürzung
Ctrl+X,A,E	manuelle Expansion einer Abkürzung
Alt+X abbrev-mode Enter	aktiviert die automatische Expansion
Alt+X edit-abbrevs Enter	Abkürzungstabelle editieren
Alt+X write-abbrev-file Enter	Speichern der Abkürzungsdatei
Alt+X read-abbrev-file Enter	Laden der Abkürzungsdatei
Ctrl+X,Ctrl+S,Ctrl+X, B,Enter	zuvor editierte Abkürzungsdatei emacs–intern speichern

Beispiel: Um eine Abkürzung für den Text `\subsection{` zu definieren geben wir diesen auf einer neuen Zeile ein und betätigen die Tastenkombination Ctrl+X,A,I,G. Durch Eingabe vom `sbc` Enter definieren wir sbc als Abkürzung für `\subsection{`. Ist die automatische Expansion aktiv (Alt+X abbrev-mode Enter), wird bei einer nachfolgenden Eingabe von `sbc` Leertaste der Text automatisch vervollständigt. **Der Text der Abkürzung sollte kein Bestandteil eines Wortes sein!**

11.7 Emacs als Newsreader

In der X–Variante muß nur der verwendete Newsserver eingetragen werden: Options ⇒ Customize ⇒ Emacs ⇒ Applications ⇒ News ⇒ Gnus ⇒ Server. Alternativ läßt in der Datei `~/.emacs` eine Zeile in der Art

```
(setq nntp-address "news.irgendwo.de")
```

aufnehmen.

11.8 Sonstiges

Über die Tastenkombination "(Alt)+(X) command (Enter)" können alle Modi, Kommandos usw. erreicht werden. Z.B. startet (Alt)+(X) compile (Enter) den Kompilierungsvorgang.
Mit (Alt)+(X),(Tab) werden alle Möglichkeiten (einige hundert) aufgelistet.

Kapitel 12

Linux im Netzwerk

Gäbe es das Internet nicht, dann gäbe es heute wahrscheinlich auch kein Linux. Zumindest nicht in der jetzigen Qualität.
Ob wir Mails versenden oder News lesen wollen, oder von einer Web–Seite zur nächsten surfen wollen... Linux kann alles, falls es richtig konfiguriert wurde.

...aber andere Systeme können das auch... So richtig interessant wird der Einsatz von Linux erst im Serverbereich: ob als File– oder Nameserver, als Mail–Verteiler, ftp– oder Web–Server usw. – also überall dort, wo ein hoher Durchsatz gefragt ist.

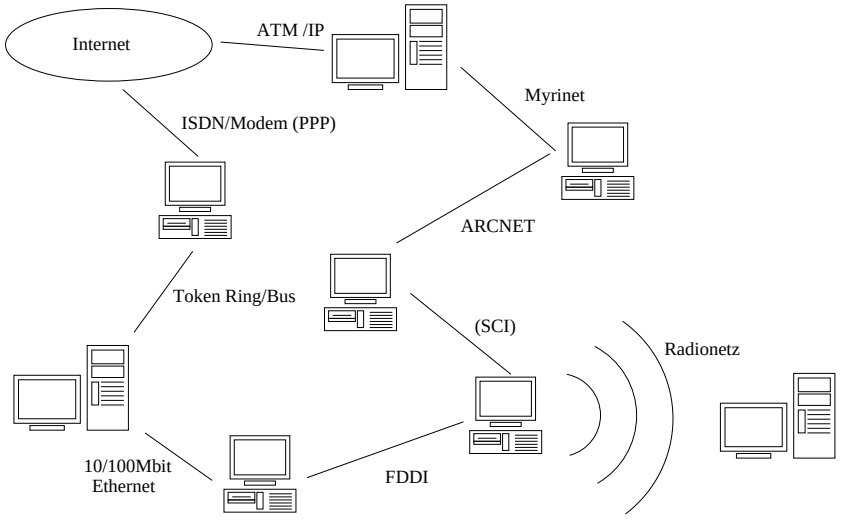

Abbildung 12.1: Einige von Linux unterstützte Protokolle und Technologien

12.1 Voraussetzungen

Bevor Linux im Netzwerk zum Einsatz gelangt, sollte man sich darüber informieren, ob die zugrunde liegende Hardware überhaupt vom Betriebssystem

unterstützt wird. Abbildung 12.1 zeigt einige derzeit von Linux unterstützte Netztechnologien und Protokolle.
Selbstverständlich muß auch der Kernel über die entsprechenden Treiber verfügen.

Weitere Punkte sollten im voraus geklärt sein:

- *Rechnername* Der Name eines Rechners (max. 8 Zeichen) muß innerhalb einer Domain eindeutig sein.
- *Domainname* Eine Domain beschreibt einen Abschnitt eines Netzwerkes.
- *IP–Adresse* Für Rechner mit Zugang zum Internet ist eine eindeutige IP–Adresse notwendig[1]. IP–Adressen sind sogenannten Klassen zugeordnet:

Klasse	Netzmaske	Netz–Adressen
A	255.0.0.0	0.0.0.0 – 127.255.255.255
B	255.255.0.0	128.0.0.0 – 191.255.255.255
C	255.255.255.0	192.0.0.0 – 223.255.255.255
Multicast	240.0.0.0	224.0.0.0 – 239.255.255.255

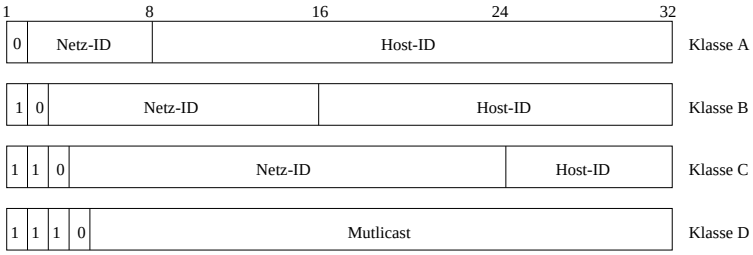

Abbildung 12.2: Interne Repräsentation der Klassen

Innerhalb jeder Klasse sind einige Adreßbereiche reserviert, wobei sichergestellt ist, daß diese Adressen im Internet nicht vermittelt werden:

Klasse	Netzmaske	reservierter Bereich
A	255.0.0.0	10.0.0.0 – 10.255.255.255
B	255.255.0.0	172.16.0.0 – 172.31.255.255
C	255.255.255.0	192.168.0.0 – 192.168.255.255

- *Gatewayadresse* Ein Rechner, der die Verbindung zwischen zwei Rechnernetzen herstellt wird als Gateway bezeichnet. Er ist für die netzwerkübergreifende Vermittlung von Datenpaketen verantwortlich. Im amerikanischen Sprachgebrauch wird Gateway oft mit einem Router gleichgesetzt. Beide verbinden verschiedene Netze miteinander, ein Gateway ermöglicht

[1] Diese 32 Bit lange Adresse wird von Providern verwaltet, so daß die einmalige Vergabe an zentraler Stelle überwacht wird. In absehbarer Zeit ist mit einer 128 Bit langen Adresse zu rechnen.

im Unterschied zu einem Router allerdings den Datentransfer zwischen verschiedenen Protokollwelten. In Abbildung 12.3 ist das Einsatzgebiet verschiedener Netzwerkelemente schematisch dargestellt.

- *Netzwerkmaske* Durch logische UND–Verknüpfung dieser mit einer IP–Adresse kann schnell entschieden werden, für welches Netz ein Paket bestimmt ist.

- *Nameserveradresse* Ein Nameserver ist für die Auflösung eines Rechnernamens in eine entsprechende IP–Adresse zuständig. Entweder kann dieser die Wandlung vollziehen, oder er kennt einen anderen Nameserver, der dies vielleicht vermag...

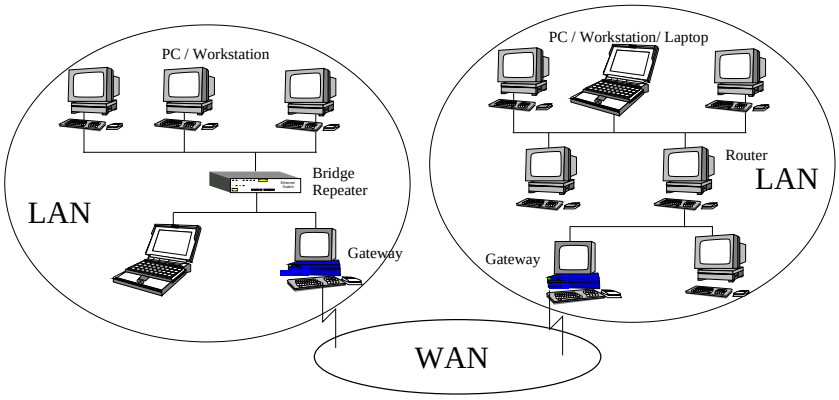

Abbildung 12.3: Local Area Network, Wide Area Network, Router, Gateway

Subnetze

Netze der Klassen A und B besitzen $2^{24} - 2$ bzw. $2^{16} - 2$ Hosts. Eine solche Vielzahl von Rechner in einem einzelnen Netzwerk zu administrieren, ist schlicht unmöglich. Man benötigt also eine Möglichkeit, ein solches großes Netz weiter zu strukturieren. Die dazu angewandte Technik ist die Bildung von Subnetzen[2].

Der Hostteil der IP–Adresse wird im jeweiligen Teilnetz nicht mehr als Rechneradresse interpretiert, sondern sie wird weiter untergliedert in eine Subnetzadresse und die Rechneradresse. In Abbildung 12.4 ist die Subnetzbildung für eine Klasse B Adresse dargestellt. Außerhalb unsers Teilnetzes sieht die Adresse wie jede andere IP–Adresse aus; ein Router wird sie anhand der Netzwerk–Adresse an ein Gateway unseres Teilnetzes senden. Innerhalb des Teilnetzes wird die IP–Adresse nun mit der Subnetzmaske bitweise UND verknüpft und man erhält die Adresse des Subnetzwerkes. Das Paket wird nun an ein Gateway/Router

[2]Im TCP/IP–Protokollstack spricht man im Zusammenhang mit Netzwerkschicht (Schichten 1 und 2 des OSI–Referenz–Modells) auch von Subnetzwerken. Diese haben aber nichts mit dem hier eingeführten Begriff zu tun, da letztere die OSI–Schicht 3 (Vermittlung) betrifft.

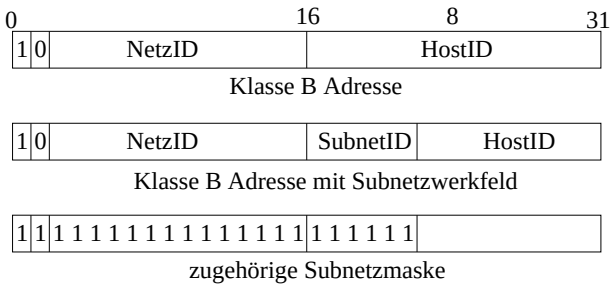

Abbildung 12.4: Subnetting

dieses Abschnittes weitergereicht. Die Subnetzbildung kann auch wiederholt auf den Hostteil einer schon aufgeteilten Adresse angewandt werden.

12.2 Konfigurationsdateien

Bei der SuSE–Distribution läßt sich ein Großteil der Netzwerkadministration im `yast` erledigen. Alternativ kann man die nachfolgend beschriebene Datei `/etc/rc.config` editieren und anschließend `SuSEconfig` aufrufen, wodurch die meisten Konfigurationsdateien automatisch generiert werden.
Weil aber die automatische Konfiguration nicht alle Bereiche abdeckt, ist das Verstehen des zugrunde liegenden Handwerkes Voraussetzung für eine befriedigende Netzwerkinstallation.
Wer unter SuSE–Linux arbeitet und sich zur "echten Handarbeit" berufen fühlt, der muß die Variable `ENABLE\_SUSECONFIG` in der Datei `/etc/rc.config` auf `no` setzen, da sonst das Tool `SuSEconfig` die Änderungen gemäß den Einstellungen in der Datei automatisch beim nächsten Aufruf vornimmt.

12.2.1 /etc/rc.config

Hier sollen nur die netzwerkspezifischen Ausschnitte kurz dargestellt werden.

```
# Mit dem loopback--Device laesst sich lokal die
# Netzwerkfunktionalitaet teilweise testen
#
START_LOOPBACK="yes"

# Anzahl von Netzwerkkarten "_0" fuer Eine
#                           "_0 _1 _2 _3" fuer Vier
#
NETCONFIG="_0"

# IP Adresse(n)
#
IPADDR_0="192.168.10.101"
IPADDR_1=""
IPADDR_2=""
```

```
IPADDR_3=""

# Netwerk Device Namen (z.B. "eth0")
#
NETDEV_0="eth0"
NETDEV_1=""
NETDEV_2=""
NETDEV_3=""

# Parameter fuer "ifconfig"
# Beispiel fuer Ethernet:
# IFCONFIG_0="192.168.81.38 broadcast 192.168.81.63
# netmask 255.255.255.224"
#
IFCONFIG_0="192.168.10.101 broadcast 192.168.10.255
netmask 255.255.255.0 up"
IFCONFIG_1=""
IFCONFIG_2=""
IFCONFIG_3=""

# Fuer nicht permanente Verbindungen (z.B. Modem)
#
SETUPDUMMYDEV=yes

# voller Hostname des System
#
FQHOSTNAME="sonne.galaxis.de"

# Verwendung von NIS zur Namensaufloesung
#
USE_NIS_FOR_RESOLVING=yes

# Liste bei der Aufloesung von Rechnernamen in IP-Adressen
#
SEARCHLIST="galaxis.de edu.saxedu.de de"

# Adresse(n) des (der) Nameservers
#
NAMESERVER="192.168.85.1"

# der Superdaemon <inetd> ist alle Netzwerkdienste notwendig
#
START_INETD="yes"

# Automatische Erstellung der /etc/resolv.conf
#
CREATE_RESOLVCONF=yes

# Portmapper starten?
```

```
# Wird fuer NFS und NIS benoetigt
#
START_PORTMAP="yes"

# Rechner als NFS-Server starten?
#
NFS_SERVER="yes"

# Rechner als NIS-Server starten?
#
START_YPSERV="no"

# Ist der Rechner NIS-Master-Server und sind weitere
# Slave-Server vorhanden, muss hier "yes" stehen
#
START_YPXFRD="no"

# Start yppasswdd (Passwortdaemon)?
#
START_YPPASSWDD="no"

#
```

12.2.2 /etc/HOSTNAME

Hier steht der Rechnername ohne die Domain.

```
user@sonne > cat /etc/HOSTNAME
sonne
```

12.2.3 /etc/hosts

In dieser Datei findet die Zuordnung von Rechnernamen in entsprechende IP–Adressen statt. Selbstverständlich kann hier nicht jeder Rechner des Internets aufgeführt werden, weshalb diese Datei nur in kleinen Netzwerken und für die unmittelbaren "wichtigen" Nachbarrechner (z.B. Nameserver, DNS–Server) genutzt wird.

```
# Typischer Aufbau einer /etc/hosts
# <IP-Adresse> <vollst. Rechnername> <Rechnername> [<Alias>]
#
127.0.0.0       localhost            # loopback
192.168.10.101  sonne.galaxis.de     sonne   mail
192.168.85.1    zeus.edu.saxedu.de   zeus
```

12.2.4 /etc/hosts.allow

Diese Datei dient der Zugangskontrolle von Nutzern/Diensten anderer Rechner. Für bestimmte Hosts/Netzwerke kann hier der Zugriff auf bestimmte lokale Dienste reguliert werden.

```
# Aufbau einer /etc/hosts.allow
# <service list> : <host list> [: command]
#
# Mail ist jedem gestattet
#
in.smtpd: ALL

# Telnet und FTP wird nur Hosts derselben Domain und dem
# Rechner Zeus erlaubt.
#
telnetd, ftpd: LOCAL, zeus.edu.saxedu.de

# Finger ist jedem erlaubt, aber root wird per Mail dar-
# ueber informiert
#
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)

#
```

12.2.5 /etc/hosts.deny

Hier kann der Zugang zu bestimmten Diensten des Rechners für bestimmte Hosts/Netzwerke explizit untersagt werden.

```
# Aufbau von /etc/hosts.deny analog zu /etc/hosts.allow
#
ALL: edu.saxedu.de

# Die remote--Shell ist eine bekannte Sicherheitsluecke
#
rsh: ALL

#
```

Beide Dateien werden u.a. vom TCP–Wrapper ausgewertet. Dabei wird zuerst die `hosts.allow` und dann die `hostsdeny` betrachtet. Es gilt: "Wurde etwas in der `hosts.allow` explizit gestattet, darf es nicht mehr verboten werden". Selbst, wenn ein entsprechender Eintrag in der `hosts.deny` steht.

12.2.6 /etc/hosts.equiv

Alle von diesen Rechnern zugreifenden Nutzer haben dieselben Rechte wie lokale Nutzer ("trusted hosts") und können ohne explizite Angabe eines Paßwortes über entfernte Dienste auf den Rechner zugreifen. Aus Sicherheitsgründen sollte auf die Möglichkeiten dieser Konfiguration verzichtet werden.

```
# Aufbau einer /etc/hosts.equiv
#
erde.galaxis.de erde

#
```

12.2.7 /etc/networks

Netzwerknamen werden hier in Netzwerkadressen umgesetzt:

```
# Typischer Aufbau einer /etc/networks
# Netzwerkname  Netzwerkadresse
#
loopback  127.0.0.0
localnet  192.168.10.0
edu-net   192.168.85.0

#
```

12.2.8 /etc/host.conf

Die Vorgehensweise beim Auflösen von Rechner– und Netzwerknamen in entsprechende Adressen wird hier bestimmt. Die Datei ist für den Resolver wichtig.

```
# Aufbau der /etc/host.conf
#
order hosts bind
multi on

#
```

In der Zeile `order...` wird die Reihenfolge bei der Auflösung von Rechnernamen in entsprechende Adressen festgelegt: Zuerst durch Nachschauen in der lokalen `/etc/hosts` (`hosts`), dann durch Befragen eines Nameservers (`bind`). Auch einen NIS–Server könnte man zu Rate ziehen (`nis`).
`multi on / multi off` erlaubt/verbietet die Vergabe von mehreren IP–Adressen an einen in `/etc/hosts` eingetragenen Rechner (Gateways haben mehrere IP–Adressen).

12.2.9 /etc/resolv.conf

Neben `/etc/host.conf` spielt auch diese Datei bei der Namensauflösung eine Rolle. Angegeben werden in dieser Datei:

- Domain des Rechners
- Liste von Domains zur Namensauflösung (unvollständige Hostnamen werden durch Anhängen der Domains vervollständigt)
- Adressen von Nameservern

```
# Aufbau einer /etc/resolv.conf
#
domain galaxis.de
search galaxis.de edu.saxedu.de de
nameserver 192.168.85.1

#
```

12.2.10 /etc/protocols

Die Protokollnummern der Transportprotokolle sind hier so eingetragen, wie sie im IP–Protokollkopf erscheinen.

```
# protocols This file describes the various protocols that are
#           available from the TCP/IP subsystem.  It should be
#           consulted instead of using the numbers in the ARPA
#           include files, or, worse, just guessing them.
#
ip     0     IP     # internet protocol,pseudo protocol number
icmp   1     ICMP   # internet control message protocol
igmp   2     IGMP   # internet group multicast protocol
ggp    3     GGP    # gateway-gateway protocol
tcp    6     TCP    # transmission control protocol
pup    12    PUP    # PARC universal packet protocol
udp    17    UDP    # user datagram protocol
idp    22    IDP    # WhatsThis?
raw    255   RAW    # RAW IP interface

#
```

12.2.11 /etc/services

Die Portnummern und Bezeichnungen der vorhandenen Netzwerkdienste stehen hier.

```
# Ausschnitte aus /etc/services
#
# Aufbau eines Eintrages:
# <Dienstname> <Port/Protokoll> [<Alias>]
#
tcpmux          1/tcp         # TCP port service multiplexer
echo            7/tcp
echo            7/udp
systat          11/tcp        users
netstat         15/tcp
ftp             21/tcp
ssh             22/tcp
ssh             22/udp
telnet          23/tcp
# 24 - private
smtp            25/tcp        mail
# 26 - unassigned
time            37/tcp        timserver
time            37/udp        timserver
rlp             39/udp        resource   # resource location
whois           43/tcp        nicname
domain          53/tcp        nameserver # name-domain server
domain          53/udp        nameserver
finger          79/tcp
```

```
kerberos        88/tcp          krb5        # Kerberos v5
kerberos        88/udp

#
```

12.3 inetd – Der Superdämon

Dämonen sind die guten Geister von Unix. Sie bleiben im Hintergrund und erwachen nur zum Leben, wenn es für sie etwas zu tun gibt. Dann erledigen sie die ihnen zugedachten Aufgaben und begeben sich wieder zur Ruhe, bis erneut ein Auftrag für sie hereinflattert.

Früher geisterten viele solcher Dämonen durch den Speicher des Rechners. Sobald das System gestartet wurde, rief man auch die Dämonen heran, selbst wenn der eine oder andere niemals wirklich etwas zu tun bekam. Jeder Server überwachte somit seine Ports, lag an einem etwas an, erzeugte er einen Kindprozeß und vererbte diesem die Verbindung. Der Server schloß unverzüglich die Verbindung und erwartete neue Anforderungen...

Heute überwacht der `inetd` alle Ports und erst bei Aktivierung an einem dieser Ports startet der inetd einen entsprechenden Server–Prozeß, der die bereits eröffnete Verbindung mittels eines Dateideskriptors übergeben bekommt und nun direkt mit dem Clienten kommuniziert. Hat der Server seine Arbeit erledigt, beendet er sich und erst eine erneute Anfrage beim inetd kann ihn wieder heraufbeschwören.

Als Fazit ist die meiste Zeit über nur noch ein Prozeß aktiv – der `inetd`.

Der `inetd` ist über die Datei `/etc/inetd.conf` konfigurierbar; ein Ausschnitt daraus sei kurz aufgeführt:

```
# See "man -S 8 inetd" for more information.
#
# If you make changes to this file, either reboot your machine
# or send the inetd a HUP signal:
# Do  a "ps x" as root and look up the pid of inetd. Then do a
# "kill -HUP <pid of inetd>".
# inetd will  re-read this file  whenever it gets that signal.
#
# <service_name> <sock_type> <proto> <flags> <user>
# <server_path> <args>
#
#echo    stream  tcp   nowait  root  internal
#echo    dgram   udp   wait    root  internal
discard stream  tcp   nowait  root  internal
discard dgram   udp   wait    root  internal
daytime stream  tcp   nowait  root  internal
daytime dgram   udp   wait    root  internal
#
telnet  stream  tcp   nowait  root  /usr/sbin/tcpd \
/usr/sbin/in.telnetd
```

```
ftp     stream  tcp   nowait  root  /usr/sbin/tcpd \
/usr/sbin/in.ftpd
#fsp    dgram   udp   wait    root  /usr/sbin/tcpd \
/usr/sbin/in.fspd
#
shell   stream  tcp   nowait  root  /usr/sbin/tcpd \
/usr/sbin/in.rshd
login   stream  tcp   nowait  root  /usr/sbin/tcpd \
/usr/sbin/in.rlogind
smtp    stream  tcp   nowait  root  /usr/sbin/tcpd \
/usr/sbin/in.smtpd

#
```

Im Einzelnen bedeuten:

- **service_name** Bezeichner des Dienstes
- **sock_type** Art des Kommunikationsmechanismus
- **prots** Protokollart
- **flags** für UDP: gibt an, ob der `inetd` sofort neue Anforderungen auf dem Port entgegennehmen, oder auf Terminierung des Servers warten soll
- **Programmname und Argumente** Programmname inklusive vollständigem Pfad und optionaler Parameterliste

Die Dienste nach `telnet` im obigen Ausschnitt aus der `/etc/inetd.conf` werden über einen sogenannten TCP–Wrapper aufgerufen. Dieser Wrapper nimmt, bevor er die jeweiligen Prozesse startet, verschiedene Initlialisierungen vor. So sucht der TCP–Wrapper in den Dateien `/etc/hosts.allow` und `/etc/hosts.deny` (Seite 136) nach den jeweiligen Dienst betreffenden Einstellungen und übergibt diese als Parameter an dieses Programm.

Wünscht man mit seinem Rechner auf Netzwerkdienste zuzugreifen oder bietet solche gar an, muß der `inetd` aktiviert werden. Überprüfen läßt sich dies mit:

```
user@sonne > ps eax | grep inetd
  133  ?  S    0:00 /usr/sbin/inetd
```

Fehlt die Ausgabe, kann der Dämon von Hand (als Root) gestartet werden. Um den Start beim nächsten Bootvorgang automatisch zu veranlassen, setzt man die entsprechende Variable in der `/etc/rc.config` (`START_INETD="yes"`) und ruft anschließend `SuSEconfig` auf.

Modifiziert man eine beliebige Konfigurationsdatei im System, so müssen die entsprechenden Dienste neu gestartet werden. Im Unterschied zu manch anderen Betriebssystemen muß Linux nicht zwangsläufig neu gebootet werden. Es genügt, den Prozessen ein Signal SIGHUP zu senden, woraufhin diese ihre Konfigurationsdateien neu einlesen. Bearbeitet man z.B. die `/etc/inetd.conf`, informiert man den `inetd` wie folgt:

```
root@sonne > killall -HUP inetd
```

12.4 Remote Procedure Call

Der RPC ist ein Protokoll, das die Implementierung verteilter Anwendungen – also Netzwerkdienste – vereinfachen soll. Ursprünglich von der Firma Xerox entwickelt, haben sich heute drei Standards etabliert:

- ONC: Open Network Computing (SUN...)
- NCA: Network Computing Architecture (HP...)
- DCE: Distributed Computing Environment (OSF)

Wie der Name schon verdeutlicht, handelt sich sich um eine Dienstleistung, die auf einem entfernten Rechner erbracht wird. Im Unterschied zu lokalen Prozeduraufrufen ergeben sich daraus zahlreiche Probleme:

- Welcher Host bietet einen entsprechenden Dienst an?
- Welches Transportprotokoll (TCP/UDP) soll verwendet werden?
- Was ist bei Ausfall des Dienstanbieters (Server)?
- Semantik des Aufrufs (genau, höchstens oder mindestens einmal)
- Datendarstellung (Wortbreite, big endian, little endian)
- Performance
- Sicherheit

Nun soll hier keine Diskussion über die Grundlagen des RPC erfolgen, dennoch sei das Prinzip eines entfernten Prozeduraufrufs in Abbildung 12.5 kurz verdeutlicht.

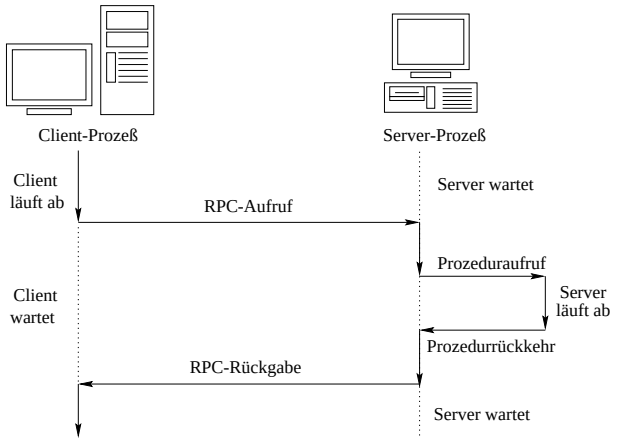

Abbildung 12.5: Ablauf eines RPC–Aufrufs

Bei einem lokalen Prozeduraufruf werden die Parameter über den Stack übergeben. Das Prinzip versagt aber im Falle des RPC, deshalb werden nun die Parameter versendet, zusammen mit der gewünschten Programmnummer, der entsprechenden Version und der jeweiligen Prozedurnummer u.a.

Stellt sich nun die Frage: An wen schicke ich das Paket?

Die Wahl eines entsprechenden Servers kann auf mehreren Wegen erfolgen. So kann in einer lokalen Datei eine entsprechende Server–Adresse vermerkt sein oder es kommt ein per Multicast oder Broadcast arbeitendes Protokoll zum Einsatz.
Kennt der Client die genaue Portnummer des Dienstanbieters nicht, sendet er eine Anfrage an Port 111 des Server–Rechners, wo der sogenannte *Portmapper* wartet.
Jeder auf dem Serverrechner aktive RPC–Server hat sich zuvor beim Portmapper registrieren lassen. Dieser kennt nun die genauen Fähigkeiten der einzelnen Server und kann – falls der Dienst von einem Prozeß erbracht wird – auf die Anfrage mit der korrekten Portnummer antworten und der Client wird seine Aufgaben nun direkt an diesen Port übermitteln (Abbildung 12.6).

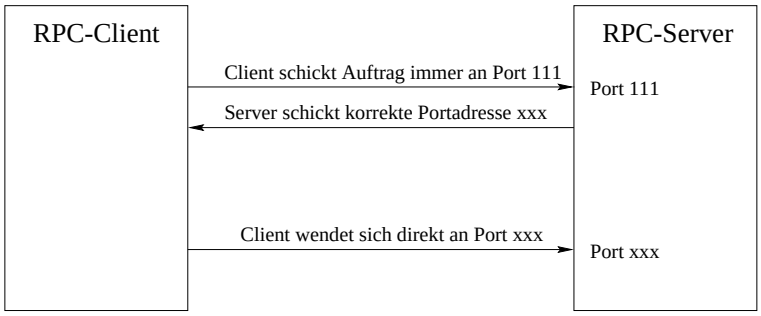

Abbildung 12.6: Anfrage beim Portmapper

Die in den weiteren Abschnitten beschriebenen Netzwerkdienste greifen auf RPC–Dienste zurück, so daß der Portmapper vor dem Start der jeweiligen Dienste gestartet werden muß!

```
user@sonne > ps eax | grep portmap
  585  ?  S    0:00 /sbin/portmap
```

Ist der Portmapper noch nicht aktiv, sollte er durch Eintrag in die `/etc/inetd.conf` bereits beim Systemstart installiert werden.

12.5 Überprüfung des Netzwerkanschlusses

Um die Funktionalität des Netzwerkanschlusses zu testen, existieren mehrere Programme.
Erster Anlaufpunkt ist die Überprüfung des lokalen TCP/IP–Systems:

```
user@sonne > ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.2 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.2 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.2 ms

--- localhost ping statistics ---
```

```
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.2/0.2/0.2 ms
```

Das Kommando `ping` (Abbruch mit (CTRL-C)) schickt die Datenpakete ans loopback–Device. Geht hier schon etwas schief, ist entweder der Kernel nicht richtig konfiguriert oder irgendein Netzwerkdienst wurde nicht gestartet.

Als nächstes wird die Netzwerkkarte getestet:

```
user@sonne > ping sonne
PING sonne.galaxis.de (194.168.10.101): 56 data bytes
64 bytes from 194.168.10.101: icmp_seq=0 ttl=64 time=0.2 ms
64 bytes from 194.168.10.101: icmp_seq=1 ttl=64 time=0.1 ms

--- sonne.galaxis.de ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.2 ms
```

Ein möglicher Fehler hier wäre ein Defekt mit der Karte oder der Verkabelung.

Schließlich "pingt" man einen Rechner seines Netzes an:

```
user@sonne > ping zeus
PING zeus.buero.saxedu.de (194.180.239.20): 56 data bytes
64 bytes from 194.180.239.20: icmp_seq=0 ttl=128 time=0.8 ms
64 bytes from 194.180.239.20: icmp_seq=1 ttl=128 time=0.5 ms
64 bytes from 194.180.239.20: icmp_seq=2 ttl=128 time=0.5 ms

--- zeus.buero.saxedu.de ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.5/0.6/0.8 ms
```

Ein anderes Kommando ist `ifconfig` , das, ohne Parameter aufgerufen, die vorhandenen Netzwerkschnittstellen auflistet:

```
root@sonne > ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Bcast:127.255.255.255
        Mask:255.0.0.0
        UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
        RX packets:112 errors:0 dropped:0 overruns:0 frame:0
        TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0

dummy0 Link encap:Ethernet  HWaddr 00:00:00:00:00:00
        inet addr:194.180.239.167  Bcast:194.180.239.255
        Mask:255.255.255.0
        UP BROADCAST RUNNING NOARP MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0

eth0   Link encap:Ethernet  HWaddr 00:80:C8:47:B3:47
```

```
inet addr:194.180.239.167  Bcast:194.180.239.255
Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:41405 errors:19 dropped:0 overruns:0 frame:29
TX packets:9123 errors:0 dropped:0 overruns:0 carrier:0
collisions:607
Interrupt:15 Base address:0xd000
```

Wichtigstes Device ist hier "eth0", das eigentliche Netzwerkdevice. `ifconfig` dient der Konfiguration dieser Schnittstellen, z.B. könnte man mittels

```
root@sonne > ifconfig eth0 down
```

dieselbe deaktivieren.

Schließlich versucht man noch einen Rechner "outside of the world" zu erreichen:

```
root@sonne > ping 198.41.0.4
PING 198.41.0.4 (198.41.0.4): 56 data bytes
64 bytes from 198.41.0.4: icmp_seq=0 ttl=247 time=159.3 ms
64 bytes from 198.41.0.4: icmp_seq=1 ttl=247 time=157.3 ms
64 bytes from 198.41.0.4: icmp_seq=2 ttl=247 time=189.1 ms
64 bytes from 198.41.0.4: icmp_seq=3 ttl=247 time=163.9 ms

--- 198.41.0.4 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 157.3/167.4/189.1 ms
```

Die verwendete Adresse gehört zu einem Nameserver der Toplevel–Domain. Schlägt dieser Kontaktversuch fehl, stimmt vermutlich etwas mit dem Routing nicht, womit wir beim Kommando `route` angelangt wären. Das Kommando listet die bekannten Interfaces auf und zeigt, wie diese erreichbar sind:

```
root@sonne > route
Kernel IP routing table
Destination       Gateway       Genmask           Flags
   Metric   Ref   Use   Iface
sonne.galaxis.de    *           255.255.255.255   UH
   0        0      0    dummy0
194.180.239.0       *           255.255.255.0     U
   0        0     31    eth0
loopback            *           255.0.0.0         U
   0        0      2    lo
default           gw.galaxis.de 0.0.0.0           UG
   0        0      5    eth0
```

Falls der letzte Eintrag fehlt, scheitert die Vermittlung an unbekannte Adressen, da der Rechner mit einer solchen nichts anzufangen weiß. Der `default`–Eintrag bewirkt, daß Pakete mit unbekannter IP–Adresse an diesen Rechner geleitet werden, in der Hoffnung, daß dieser die Adresse kennt. Fehlt diese Route, fügt man sie am besten hinzu:

```
root@sonne > route add default gw gw.galaxis.de
```

Der Name `gw.galaxis.de` muß lokal auflösbar sein (am besten durch einen Eintrag in der `/etc/hosts`). Danach sollten auch Rechner außerhalb des lokalen Netzes ansprechbar sein.

Falls nicht, liegt der Fehler vermutlich außerhalb unseres Einflußbereiches. Wen es interessiert, der kann mit dem Kommando `traceroute` die genaue Wegewahl des Paketes verfolgen und eventuell das schwache Glied in der Kette isolieren:

```
root@sonne > traceroute www.linux.de
traceroute to www.linux.de (195.254.37.153), 30 hops max,
    40 byte packets
1  base.saxedu.de (194.180.239.1) 1.003 ms  1.539 ms  0.884 ms
2  195.211.141.1 (195.211.141.1) 3.062 ms  3.965 ms  2.505 ms
3  frankfurt1.GIGABELL.NET (195.211.224.1) 26.195 ms 29.581 ms
   16.679 ms
4  hssi4-0.frankfurt2.GIGABELL.NET (195.211.199.30)  19.642 ms
   20.949 ms  19.533 ms
5  194.31.232.65 (194.31.232.65) 21.751 ms 21.525 ms 17.002 ms
6  194.97.193.6 (194.97.193.6)  19.811 ms  27.464 ms 17.253 ms
7  194.97.193.17 (194.97.193.17) 51.076 ms 37.010 ms 28.361 ms
8  194.97.216.19 (194.97.216.19) 31.216 ms 30.439 ms 31.207 ms
9  195.254.37.153 (195.254.37.153)  31.260 ms  32.833 ms
   28.463 ms
```

Falls ein Paket nicht vermittelt werden kann, wird in der letzten Zeile anstelle der Zeitinformationen ein Fehlercode (eingeleitet durch `!`) erscheinen. Z.B. besagt ein `!N`, daß das Netzwerk oder das Protokoll nicht erreichbar ist. Drei Sterne in einer Zeile beschreiben den Transportversuch über ein Gateway, von welchem keine Antwort kam. Nach drei Versuchen wird versucht, ein Paket über ein anderes Gateway zu routen.

Das Kommando `netstat` dient schließlich der Überwachung aller Netzwerkschnittstellen. Hiermit läßt sich der Status jedes Ports abfragen, auch derer, an denen z.Z. nur ein Dienst "lauscht".
Anhand von Statistiken der Schnittstellen lassen sich u.a. Schwachstellen im Netzwerk erkennen:

```
root@sonne > netstat -i
Kernel Interface table
Iface   MTU Met  RX-OK RX-ERR RX-DRP RX-OVR
TX-OK TX-ERR TX-DRP TX-OVR Flags
lo     3584   0     68      0      0      0
   68      0      0      0 BLRU
dummy  1500   0      0      0      0      0
    0      0      0      0 BORU
eth0   1500   0  13680      4      0      0
 4306      0      0      0 BRU
```

Der Eintrag `eth0` beschreibt die Schnittstelle zum physikalischen Netzwerk:

- `MTU` gibt die maximale Gröse von Paketen an, hier also 1500 Bytes.

- `Met` gibt die Metrik an, die zur Kostenermittlung einer passierten Route vorgesehen ist. In Linux wird hiervon kein Gebracuh gemacht.
- `RX-OK/TX-OK` beschreibt die Anzahl empfangener/gesendeter Pakete
- `RX-ERR/TX-ERR` ist der Zähler für fehlerhafte Pakete. Ein im Verhältnis zu `xx-OK` sehr hoher Wert kann auf eine Überlastung des Netzwerkes hindeuten und des damit vermehrten Auftretens von Kollisionen.
- `RX-DRP/TX-DRP` beinhaltet die Anzahl verworfener Pakete. Der Host ist offensichtlich überlastet und kann die eintreffenden/ausgehenden Pakete nicht schnell genug verarbeiten.
- `RX-DRP/TX-DRP` zählt die Pakete, die einen Überlauf des Puffers hervorgerufen haben. Wahrscheinlich überschreitet die Paketlänge die mögliche MTU der Schnittstelle. Bei der Implementierung eigener Netzwerksoftware ist zu beachten, daß die Linux–IP–Implementierung derzeit keine Fragmentierung unterstützt.
- `FLAGS` beschreibt des Status der Verbindung:

Flags	Bedeutung
A	Empfang von Multicast–Adressen
B	Broadcast erlaubt
D	Device wird debugged
L	Loopback–Device
M	alle Pakete werden empfangen
N	Pakete ohne Trailer senden
O	keine ARP
P	Point–to–Point–Verbindung
R	Device ist in Betrieb
U	Device ist aktiv

12.6 Einige Netzdienste

Zunächst soll es um einige Netzdienste gehen, die weniger Unix–spezifischen Charakter besitzen, sondern von den meisten netzwerktauglichen Betriebsystemen angeboten werden.

12.6.1 Telnet

Telnet realisiert den Zugriff auf entfernte Rechner in Form eine Terminalsitzung. Auf die Ressourcen des anderen Rechners (Software, Hardware) kann somit interaktiv zugegriffen werden, sofern man über die Berechtigung dazu verfügt. *Telnet* basiert auf dem Client–Server–Prinzip (Abbildung 12.7). Client und Server müssen beide dieselbe Terminalemulation verwenden, durch die Tastenbelegung und Darstellungsmodus beschrieben werden.

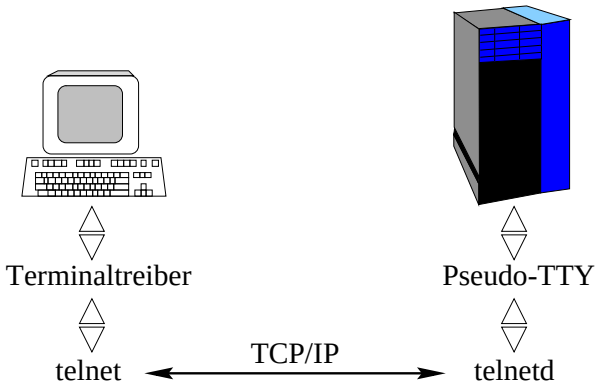

Abbildung 12.7: Ablauf einer Telnet–Sitzung

Das Programm `telnet` ist normalerweise auf jeder Linux–Distribution enthalten und sollte mit dem Basissystem installiert worden sein. Soll der Rechner als Server eingesetzt werden, sind nur die Einträge der Dateien `/etc/services` und `/etc/inetd.conf` zu entkommentieren:

```
root@sonne > less /etc/services
...
telnet          23/tcp
...
root@sonne > less /etc/inetd.conf
...
telnet  stream  tcp   nowait  root  /usr/sbin/tcpd in.telnetd
...
```

Nach Modifizierung einer der Dateien ist der `inetd` neu zu starten:

```
root@sonne > kill -HUP $(cat /var/run/inetd.pid)
```

Jeder Benutzer, der Zugang auf den lokalen Rechner erhalten soll, ist in die `/etc/passwd` einzutragen. Anschließend kann die Installation getestet werden:

```
user@sonne > telnet sonne
Trying 192.168.10.101...
Connected to sonne.galaxis.de
Escape character is '^]'.
Welcome to SuSE Linux 6.0 (i386) - Kernel 2.0.36 (ttyp4).

sonne login: user
Password:
Have a lot of fun...
Last login: Mon Apr 19 13:36:34 on ttyp3 from :0.0.
You have mail.
```

Telnet läßt sich hervorragend zum Verfolgen einiger anderer Internetdienste einsetzen. Am Beispiel des Simple Mail Transfer Protocols (SMTP) zeigt das folgende Beispiel den Ablauf bei einem Mail–Transfer. Ein > am Zeilenanfang kennzeichnet eine Eingabe im Dialog; die anderen Zeilen sind die Antworten:

```
user@sonne > telnet sonne smtp
Trying 192.168.10.101...
Connected to sonne.galaxis.de
Escape character is '^]'.
220 sonne.galaxis.de ESMTP Sendmail 8.8.8/8.8.8; Mon, 19 Apr
1999 14:44:05 +0200
>  HELO galaxis.de
250 sonne.galaxis.de Hello user@sonne.galaxis.de [192.168.10
.101], pleased to meet you
>  HELP
214-This is Sendmail version 8.8.8
214-Topics:
214-    HELO    EHLO    MAIL    RCPT    DATA
214-    RSET    NOOP    QUIT    HELP    VRFY
214-    EXPN    VERB    ETRN    DSN
214-For more info use "HELP <topic>".
214-To report bugs in the implementation send email to
214-    sendmail-bugs@sendmail.org.
214-For local information send email to Postmaster at your site.
214 End of HELP info
>  MAIL FROM:user@galaxis.de
250 user@galaxis.de... Sender ok
>  RCPT TO:root@sonne.galaxis.de
250 root@sonne.galaxis.de... Recipient ok
>  DATA
354 Enter mail, end with "." on a line by itself
>
>  Date: Mon Apr 19 11:11:11 MEST 1999
>  From: <user@sonne.galaxis.de>
>  To: <root@sonne.galaxis.de>
>  Subjects: demonstration
>  this is the mail body.
>  .
250 OAA01356 Message accepted for delivery
```

```
> QUIT
  221 sonne.galaxis.de closing connection
  Connection closed by foreign host.
```

Man benötigt also keinen Mail–Client, um jemandem eine Nachricht zukommen zu lassen :-)

12.6.2 File Transfer

Das Kommando `ftp` dient der Übertragung von Daten von einem Internetrechner zu einem anderen. Die Besonderheit des zugehörigen Protokolls liegt in den getrennten Kanälen für die Daten und die Steuerung (Abbildung 12.8), sowie in der Datenübertragung ohne Verwendung eines Spoolers.
Im RFC 959 ist für FTP TCP–Port 20 als Steuerungskanal und TCP–Port 21 als Datenkanal festgelegt. FTP verwendet als Transportprotokoll immer TCP, da dieses bereits Vorkehrungen für einen sicheren Datentransfer enthält und FTP sich darum nicht zu kümmern braucht.

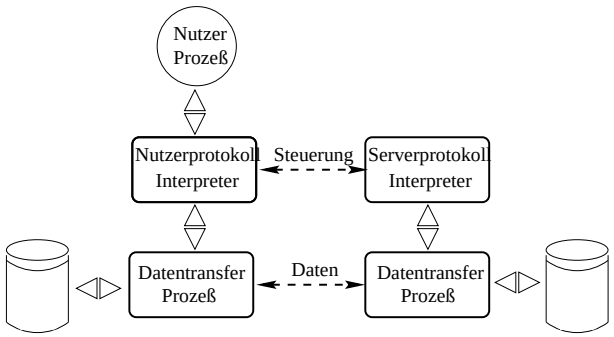

Abbildung 12.8: File Transfer Protokoll

Einrichtung des Servers: Per Voreinstellung sollte das Paket bereits installiert sein: In den Dateien `/etc/inetd.conf` und `/etc/services` müssen die entsprechenden Einträge vorhanden sein:

```
user@sonne > less /etc/inetd.conf
...
# Option -a bewirkt die Auswertung der Datei /etc/ftpaccess
#
ftp   stream  tcp   nowait  root   /usr/sbin/tcpd  wu.ftpd -a
...
user@sonne > less /etc/services
...
ftp          21/tcp
...
```

Beeinflußt wird die Arbeit des FTP–Servers durch drei (vier) Dateien:

- `/etc/ftpaccess`: enthält Optionen (Sicherheit), die den Umfang der Dienstleistungen des Servers festlegen.

- `/etc/ftpconversions`: beschreibt die Behandlung komprimierter Daten.
- `/etc/ftpusers`: enthält die Nutzerkennzeichen, die FTP nicht verwenden dürfen:

```
user@sonne > less /etc/ftpusers
#
# ftpusers
#     This file describes the names of the users that may
#     _*NOT*_ log into the system via the FTP server.
#     This usually includes "root","uucp", "news" and the
#     like, because those users have too much power to be
#     allowed to do "just" FTP...
#
root
lp
news
uucp
games
man
at
mdom
gnats
nobody
# End.
```

  - `/etc/ftpgroups`: analog zu `ftpusers` beschränkt diese Datei den Zugang von Gruppen; diese Datei existiert nicht immer.

Die mit dem Paket installierten Dateien sollten in den meisten Fällen den Ansprüchen genügen. Auf Client–Seite ist keinerlei Konfiguration notwendig. Eine kleine Beispielsitzung soll die Installation testen:

```
user@sonne > ftp sonne
Connected to sonne.galaxis.de.
220 sonnegalaxis.de FTP server (Version wu-2.4.2-academ[BETA-
18](1) Fri Dec 11 19:58:25 /etc/localtime 1998) ready.
Name (sonne:user):
331 Password required for user.
Password:
230 User user logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> pwd
257 "/home/user" is current directory.
ftp> ?
Commands may be abbreviated.  Commands are:

!            debug        mdir         sendport     site
$            dir          mget         put          size
account      disconnect   mkdir        pwd          status
```

```
append       exit         mls          quit         struct
ascii        form         mode         quote        system
bell         get          modtime      recv         sunique
binary       glob         mput         reget        tenex
bye          hash         newer        rstatus      tick
case         help         nmap         rhelp        trace
cd           idle         nlist        rename       type
cdup         image        ntrans       reset        user
chmod        lcd          open         restart      umask
close        ls           prompt       rmdir        verbose
cr           macdef       passive      runique      ?
delete       mdelete      proxy        send
ftp> cd /usr/local
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 10
drwxr-xr-x  10 root    root      1024 Feb 15 15:52 .
drwxr-xr-x  23 root    root      1024 Feb 15 15:54 ..
drwxr-xr-x   2 root    root      1024 Feb 15 15:48 bin
drwxr-xr-x   7 root    root      1024 Apr 20 13:15 ftp
drwxr-xr-x   6 root    root      1024 Dec 12 01:20 httpd
drwxr-xr-x   4 root    root      1024 Feb 15 15:52 httpsd
drwxr-xr-x   2 root    root      1024 Feb 15 15:48 include
drwxr-xr-x   2 root    root      1024 Feb 15 15:05 info
drwxr-xr-x   2 root    root      1024 Feb 15 15:48 lib
drwxr-xr-x  12 root    root      1024 Apr 19 08:47 man
226 Transfer complete.
ftp> bye
221 Goodbye.
```

Der einfachste Weg um *anonymes* FTP zuzulassen (Nutzerkennzeichen `ftp` oder `anonymous`), ist unter SuSE die Installation des Paketes `ftpdir` aus der Serie `n`. Die Schritte zur Realisierung per Hand sollen hier in knapper Form abgehandelt werden.

1. In der `/etc/passwd` muß ein Account unter dem Kennzeichen `ftp` existieren:

   ```
   root@sonne > less /etc/passwd | grep ftp
   ftp:x:40:2:ftp account:/usr/local/ftp:/bin/bash
   ```

   Das eingetragene Homeverzeichnis ist gleichzeitig die Basis für die anzulegenden Dateien und Verzeichnisse.

2. Im FTP–Basisverzeichnis sind folgende Unterverzeichnisse mit den angegebenen Rechten zu erzeugen:

   ```
   root@sonne > ls -l /usr/local/ftp
   total 7
   ```

```
drwxr-xr-x   2 root     root       1024 Apr 20 15:00 bin
drwxr-xr-x   2 root     root       1024 Apr 20 15:00 dev
drwxr-xr-x   2 root     root       1024 Apr 20 15:00 etc
drwxr-xr-x   2 root     root       1024 Apr 20 15:00 lib
drwxr-xr-x   2 root     root       1024 Apr 20 15:00 msgs
drwxr-xr-x   2 root     root       1024 Apr 20 15:00 pub
drwxr-xr-x   3 root     root       1024 Apr 20 15:00 usr
```

3. Ins Unterverzeichnis `./bin` sind die Programme `compress`, `ls` und `tar` zu installieren. Diese Programme werden vom FTP–Dämonen benötigt (minimal reicht bereits `ls`).

   Achtung: Es werden die statisch gelinkten Versionen der Kommandos benötigt:

   ```
   root@sonne > file ls
   ls: ELF 32-bit LSB executable, Intel 80386, version 1,
   statically linked, stripped
   ```

4. Im Unterverzeichnis `./etc` müssen die Dateien `passwd` und `group` mit den Rechten `644` erzeugt werden:

   ```
   root@sonne > vim etc/passwd
   root:x:0:0:Super User:/root:/bin/bash

   root@sonne > vim etc/group
   root:x:0:root
   users:x:100:root
   ```

   Das Kommando `ls` zeigt anhand dieser Dateien die Zuordnung der UIDs und GIDs für die gespeicherten Dateien an.

5. Das Verzeichnis `./lib` bleibt in den meisten Fällen leer. Sollten im Verzeichnis `./bin` dynamisch gelinkte Programme installiert werden, müssen alle von diesen benötigten shared–object– Bibliotheken hier zu finden sein.

6. In `./msgs` existieren die beiden Dateien `msg.dead` und `welcome.msg`, deren Inhalte bei fehlgeschlagendem Login bzw. als Begrüßung angezeigt werden.

7. Das Verzeichnis `./pub` ist normalerweise die Wurzel für alle Daten, die per FTP der Welt zugänglich gemacht werden.

8. Das Verzeichnis `./usr` enhält ein Unterverzeichnis `bin` mit den beiden Kommandos `ls` und `gzip`. Dieses kann aber auch entfallen.

Neben FTP existiert ein weiteres Datentransfer–Protokoll mit minimalen Anforderungen: Trivial File Transfer Protocol. Im Unterschied zum FTP verwendet TFTP das User Datagram Protocol (UDP) zum Transport der Daten.
Als Folge daraus muß das Protokoll sich selbst um Sicherungsfunktionen kümmern. Auch entfällt das Einloggen auf dem Server, so daß TFTP serverseitig

starken Restriktionen unterworfen ist (nur für "weltweit" lesbare und schreibbare Dateien). Einsatzgebiet von TFTP ist das Bootstrapping plattenloser Rechner, da die hierfür notwendigen Daten (Netzwerktreiber, IP, UDP, das Protokoll selbst) in einem EEPROM Platz finden.

12.6.3 News

Die NetNews sind ein internationaler Kommunikationsdienst, vergleichbar mit den Leserbriefen einer Zeitung. Einen Beitrag nennt man nicht von ungefähr auch Artikel und diese sind in zahlreiche Rubriken unterteilt, den sogenannten Newsgruppen.
International existieren derzeit ca. 3500 Rubriken, in Deutschland sind es etwa 200. Um die Flut an Informationen verwalten zu können, sind die Rubriken im sogenannten Usenet[3] hierarchisch angeordnet, wobei in der "Wurzel" die englischsprachigen Toplevel–Gruppen stehen:

- `comp`: Computer: Alles ueber Hard-/Software...
- `sci`: Wissenschaftsthemen
- `humanities`: Geisteswissenschaften
- `news`: Zum NetNews System selber (Anfaenger, Admins, Software...)
- `rec`: Freizeit, Hobbies
- `soc`: Soziale Fragen, Politik
- `talk`: Plaudereien
- `misc`: Diverses verschiedenes
- `school`: Internationale Schulprojekte
- `gnu`: GNU Software
- `alt`: Alternative Newshierarchie

Neben den aufgeführten Gruppen gibt es die Ländergruppen, wobei die deutschen Newsgroups im Zweig `de` angeordnet sind.

Spezialitäten unter den Gruppen sind *Testgruppen*, die – der Name läßt es vermuten – für Systemtests eingesetzt werden und oftmals auf eintreffende Artikel mit einer automatisch generierten Antwort reagieren.

Moderierte Newsgruppen werden von einem Moderator verwaltet, der über den Inhalt der Gruppe wacht.

Die *Einführung neuer Newsgruppen* erfolgt nach einem Voting–Verfahren, dessen Ablauf in den Verhaltensregeln zur Nutzung des Usenet, den Netiquette, festgelegt ist. Prinzipiell werden alle Diskussionsthemen, die sich nicht einer speziellen Gruppe zuordnen lassen, zunächst in allgemeinen Gruppen (`misc`) abgehandelt. Sollte sich eine solche Artikelserie als äußerst ergiebig erweisen, wird irgendwann irgendwer die Auslagerung in eine eigene Gruppe anregen.

[3] Usenet steht für alle an den NetNews beteiligten Rechner.

Dieser "jemand" wird einen Aufruf zur Diskussion (Request for Discussion – RfD) initiieren und nach einer gewissen Zeit (wenige Wochen) wird, falls sein Aufruf auf positive Resonanz stieß, eventuell dieselbe Person einen Aufruf zur Abstimmung (Call for Votes – CfV) folgen lassen. Ist die Wahl erfolgreich, wird der zuständige Moderator einen Server die Einrichtungsnachricht absenden lassen. Dieser Server kann von jedem Netzbenutzer per Einspruchsmail gestoppt werden.
Für das sehr vereinfacht dargestellte Verfahren existiert eine eigene Newsgruppe (für die deutschen Newsgruppen: `de.admin.news.announce`), in der allen interessierten Nutzern die anstehenden Diskussionen/Entscheidungen mitgeteilt werden.

Aufbau eines Artikels

Der Aufbau eines News–Artikels ähnelt stark dem einer Mail:

```
     Subject: <1999> Die Newsgruppen der de.alt-Hierarchie
        Date: 15 Apr 1999 22:00:02 -0000
        From: Sample User <user@sonne.galaxis.de>
    Reply-To: de-newusers-infos@sonne.galaxis.de
Organization: Moderation von de.newusers.infos
  Newsgroups: de.newusers.infos, de.newusers.questions,
              de.admin.news.groups, de.alt.admin
 Followup-To: de.newusers.questions
 ...
```

Subject	Thema des Artikels
Date	Zeitpunkt des Postens des Artikels
From	Absender des Artikels
Reply–To	Antwort als Email dorthin
Organisation	verantwortlicher Moderator
Newsgroup	für welche Gruppe ist der Artikel bestimmt
Followup–To	Antworten in diese Newsgruppen
Distribution	Verbreitung des Artikels `local`, `world`
Keywords	Schlüsselwörter (optional)
Summery	Zusammenfassung (optional)
Path	Weg, den der Artikel genommen hat (automatisch generiert)

Einrichten von News

Um News verwenden zu können, benötigt man Zugriff auf einen entsprechenden Server. Entweder greift man auf den vom Provider zur Verfügung gestellten News–Server zu, oder man sucht sich im Internet einen der frei zugänglichen Server. Die Adressen letzterer findet man z.B. über die verschiedenen Suchdienste im Web (Suche nach Newsserver).

Kennt man einen solchen Server, trägt man seinen Namen in die Datei `etc/nntpserver`. Alternativ setzt man die Variable `NNTPSERVER` in der `/etc/rc.config`

und ruft SuSEconfig auf. Verwendet man keinen lokalen Newserver (siehe weiter unten), muß zum Transport der Artikel der entsprechende Dämon nntpd von inetd gestartet werden[4]:

```
root@sonne.de > vim /etc/nntpserver
java.cs.uni-magdeburg.de

user@sonne.de > less /etc/inetd.conf | grep nntp
nntp  stream  tcp     nowait  news    /usr/sbin/tcpd \
/usr/sbin/leafnode
```

Exportiert man nun noch per Hand die NNTPSERVER kann man z.B. den Newsreader tin aus der Serie n zum Lesen und Schreiben von News benutzen. Zu starten ist er wie folgt:

```
user@sonne > tin -r
         Group Selection (java.cs.uni-magdeburg.de  937)
h=help

     1    10  biz.marketplace.services.non-computer
M    2     5  biz.marketplace.services.computers
     3    16  biz.marketplace.non-computer
     4    27  biz.marketplace.computers.mac
     5    45  biz.marketplace.computers.pc-clone
     6    10  biz.marketplace.computers.other
     7     3  biz.marketplace.computers.discussion
     8     3  biz.marketplace.computers.workstation
M    9    69  biz.comp.accounting
    10     1  biz.comp.telebit                  Support of the T
elebit modem.
    11     4  biz.comp.telebit.netblazer        The Telebit Netb
lazer.
    12    20  misc.kids.computer                The use of compu
ters by child
    13   107  misc.forsale.computers.memory     Memory chips and
 modules for
    14   140  misc.forsale.computers.pc-specif  PC motherboards.
    15   203  misc.forsale.computers.pc-specif  Other PC-specifi
c equipment.
    16    74  misc.forsale.computers.pc-specif  PC expansion car
ds.

    <n>=set current to n, TAB=next unread, /=search pattern, c)a
 tchup, g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=to
 ggle all/unread, s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub
  pattern, y)ank in/out
```

[4]Der hier angegebene Newsserver ist z.Z. frei zugänglich.

Neben dem `emacs` (siehe Seite 129) läßt insbesondere der Netscape komfortabel als Newreader einsetzen, man trage nur in den Preferences den entsprechenden Server (Mail & Newsgroups ⇒Newsgroup Server) ein.

Nun kann man sich zwar an den heißen Diskussionen beteiligen, doch wird man, falls man per Modem am Netz hängt, sich bei Erhalt der nächsten Telefonrechnung freiwillig vom Schwarzen Brett des Internets zurückziehen. Die Kosten... Sich die Megabytes an News online reinzuziehen ist wirklich nichts für sparsame Leute, viel günstiger ist da die Einrichtung eines lokalen News–Servers, das Herunterladen der Daten zum billigen Nachttarif und die Einordnung der Artikel ins lokale Newssystem.

News–Server einrichten

Zum Transport der News ist ein eigenes Programm zuständig. Weit verbreitet sind `CNews` bzw. das neuere `INN`. Desweiteren benötigt man ein Programm, um die News zwischen verschiedenen Newsservern zu versenden. `INN` ist der eigentliche Newsserver und soll uns zunächst interessieren; anschließend betrachten wir die Installation des Programmes `suck` zur Kommunikation zwischen den Servern.
Das Paket `inn` sollte allen neueren Distributionen beiliegen, so daß nach der Installation zur Konfiguration übergegangen werden kann.

Zunächst sind eine Voraussetzungen im System zu überprüfen:

1. Ein Benutzer `news` und eine gleichnamige Gruppe müssen existieren:

   ```
   user@sonne > less /etc/passwd | grep news
   news:x:9:13:News system:/etc/news:/bin/bash

   user@sonne > less /etc/group | grep news
   news:x:13:news
   ```

2. Das Programm `mail` muß installiert sein.

   ```
   user@sonne > which mail
   /usr/bin/mail
   ```

3. Der News–Dämon darf *nicht* vom `inetd` gestartet werden (diesen benötigt man jedoch, falls man auf News ohne die Installation eines lokalen Servers zugreifen möchte):

   ```
   user@sonne > less /etc/inetd.conf | grep nntp
   #nntp    stream  tcp     nowait  news    /usr/sbin/tcpd \
   /usr/sbin/leafnode
   ```

4. In `/etc/aliases` müssen (zumindest) folgende Einträge existieren:

   ```
   user@sonne > less /etc/aliases | grep news
   bin:            root
   news:           root
   ```

```
newsadm:        news
newsadmin:      news
usenet:         news
```

5. Das Verzeichnis `/var/lock/news` muß vorhanden sein, dem Benutzer `news` und der Gruppe `news` gehören und die Rechte `755` haben.

Konfigurationsdateien des innd

/var/lib/news/active

Diese Datei enthält die Newsgruppen, die auf dem Newsserver aktiv sein sollen. Interessiert man sich für nur wenige Gruppen, kann man diese gleich von Hand eintragen. Der Aufbau eines Eintrages ist:

```
<newsgroup><high_mark><low_mark><flag>
```

`high_mark` bzw. `low_mark` bezeichnen die höchste bzw. kleinste Artikelnummer, die mit dieser Newsgruppe verbunden wird. Da die Datei später vom `innd` selbst verwaltet wird, sind hier am Anfang je 10 Nullen einzutragen. Das Flagfeld kann folgende Einträge enthalten:

Flag	Wirkung
`y`	lokales Posten gestattet
`n`	lokales Posten nicht gestattet
`m`	moderierte Gruppe, Posten nur über den Moderator
`j`	Artikel werden nicht in dieser Gruppe gespeichert
`x`	Artikel können nicht in diese Gruppe gepostet werden
`=<group>`	Artikel werden lokal in die Gruppe `group` gespeichert

Beispiel:

```
control 0000000000 0000000000 y
control.cancel 0000000000 0000000000 y
junk 0000000000 0000000000 y
alt.linux 0000000000 0000000000 y
alt.os.linux 0000000000 0000000000 y
de.markt.comp.hardware 0000000000 0000000000 y
```

Bei kompletter Spiegelung eines Newsservers, holt man sich die Liste mit dem Programm `getlist`:

```
news@sonne > /ust/lib/news/bin/getlist -h <newsserver> \
> active > /tmp/active.server
```

In dieser Datei sind die Nummern (siehe oben) allerdings auf die Werte des externen Servers eingestellt. Um die Artikel auch lokal herunterzuladen, müssen diese auf Null gesetzt werden. Dazu kann folgendes `sed`–Skript benutzt werden:

```
news@sonne > cd /var/lib/news
news@sonne > sed 's/ [0-9]* [0-9]* / 0000000000 0000000001 /'\
> /tmp/active.server > active
```

Nach dem Start des `innd` sollte diese Datei nicht mehr von Hand modifiziert werden!

/etc/news/expire.ctl

Die Datei beschreibt die Verweildauer der News auf dem Server. Nach Ablauf der Zeit werden die Nachrichten automatisch gelöscht. Eine zu lange Speicherzeit kann den Plattenplatz enorm strapazieren.

Es existieren zwei Formen von Einträgen:

- `/remember/:days` gibt an, wielange sich der Server die Nachrichtennummer merken soll, auch wenn diese schon verfallen ist. Neben der Angabe der Tage (auch Gleitkommazahl möglich) ist auch das Schlüsselwort `never` zulässig.
- `<newsgroup>:<flag>:<minimum>:<default>:<maximum>` bestimmt die Verfahrensweise für die einzelnen Newsgruppen:
  - `newsgroup` Welche Gruppen betrifft der Eintrag, es können die Shell–üblichen Wildcards verwendet werden.
  - `flag` Einschränkung der Gruppen:
    - A alle Gruppen
    - M nur moderierte Gruppen
    - U nur unmoderierte Gruppen
- `minimum` minimale Verweildauer eines Artikels auf dem Server
- `default` Verweildauer eines Artikels, falls keine Verfallszeit angegeben ist
- `maximum` maximale Verweildauer eines Artikels

Beispiel:

```
news@sonne > less /etc/news/expire.ctl
## Remember for 14 days, if articles expire before 14 days
/remember/:14

## Keep all markt-groups for 14 days
de.markt.comp.*:A:2:7:14

## Never delete linux specific news
de.comp.os.linux:M:never::

## for internal use required
junk:A:1:1:2
control:A:1:1:2
```

Zur Automatisierung des Löschens veralteter Artikel läßt sich ein Eintrag in die `crontab` vornehmen:

```
news@sonne > crontab -e
SHELL=/bin/bash
MAILTO=news
```

```
#run at 8:00 everyday news.daily
0 18 * * *   /usr/lib/news/bin/news.daily \
>>$HOME/news_daily.log 2>&1
```

/etc/news/host.nntp

Hier wird der Host eingetragen, auf dem der `innd` läuft:

```
news@sonne > vim /etc/news/hosts.nntp
localhost:
```

/etc/news/inn.conf

Hier wird Einiges betreffend des Newstransportes festgelegt; drei Zeilen sollten an die lokalen Gegebenheiten angepaßt werden:

```
news@sonne > vim /etc/news/inn.conf
...
# Name unserer Organisation, so wie sie im Header erscheinen
# soll
organisation:   Galaktische Nachrichtenagentur

# Wer ist der Newsserver (nntp)?
server:         localhost

# Domain des lokalen Hosts
domain:         galaxis.de
```

/etc/news/newsfeeds

Hier wird die Verteilung der Newsgruppen auf die verschiedenen Server definiert. Der Aufbau einer Zeile hat die Form:

```
site[/exclude,exclude...]\
        :pattern,pattern...[/distrib,distrib...]\
        :flag,flag...\
        :param
```

- `site`: Newserver, an den die Artikel geschickt werden sollen. Es ist eine Art default–Eintrag für den Fall, daß im `path`–Feld des Artikels kein Server angegeben ist. Mittels der `exclude`–Liste können Artikel, die von bestimmten Rechnern kommen (stehen im Feld `path`) vom Posting ausgeschlossen werden.

- `pattern`: Das sind die Newsgruppen, die zum Newsserver gesendet werden sollen; die üblichen Wildcards können benutzt werden. Wird z.B. ein Artikel nach `de.markt.comp.*` gepostet, würden diese in die beiden Gruppen `de.markt.comp.hardware` und `de.markt.comp.misc` verteilt werden.

Eine solche Mehrfachverteilung kann durch die `distrib`–Einträge eingeschränkt werden.

- `flags`: Hier werden über umfangreiche Parameter Sachen wie die maximale Artikelgröße, Auswertung des Headers, Ablage auf dem Server ... spezifiziert. Man informiere sich in der Manual Page zu `newsfeeds`. Wir verwenden hier nur `Tf`, was die Speicherung des Artikels in einer Datei veranlaßt und `Wnm`, wodurch die Speicherung der Headerinformationen auf Message-ID und relativen Pfadnamen beschränkt wird.
- `param`: Die Bedeutung des Feldes ist stark vom verwendeten Eintrag (feed) abhängig und soll hier nicht erläutert werden.

Als erster Eintrag in einer Datei `newsfeeds` erscheint ein Server mit dem Namen `ME`. Die hier getroffenen Einstellungen gelten für alle weiteren Einträge. Eine `overview`–Zeile dient der Generierung eines Inhaltsverzeichnisses durch das Programm `overchan`.

Beispiel:

```
##  $Revision: 1.17 $
##  newsfeeds - determine where Usenet articles get sent
##  Format:
##      site[/exclude,exclude...]\
##              :pattern,pattern...[/distrib,distrib...]\
##              :flag,flag...\
##              :param
##  ...
##
##  Note that refusing articles means you won't offer them
##  to sites you feed

ME:*::

# for NOV overview database
OVERVIEW!:*,!junk,!control*:Tc,WO:/usr/lib/news/bin/overchan

java.cs.uni-magdeburg.de\
        :*,!junk,!control*\
        :Tf,Wnm:java.cs.uni-magdeburg.de
```

/var/lib/news/newsgroups

Hier werden Kurzbeschreibungen zu den Newsgruppen hinterlegt. Die Datei besitzt nur informativen Charakter und ist für den Betrieb des Newsservers nicht notwendig. Analog zur Datei `active` läßt sich mittels `getlist` eine komplette Liste vom Server holen.

```
control                         News server internal group
junk                            News server internal group
de.markt.comp.hardware          Hardwaremarkt
```

```
de.markt.comp.misc              Computermarkt (diverses)
```

/etc/news/nnrp.access

Liste der Rechner, die Zugriff auf den Newsserver erhalten und Art des Zugriffs.

```
news@sonne > vim /etc/news/nnrp.access
# Datei nnrp.access
#
# Format:
#    <Hostname|Netzname><Permissions><Nutzername>
#    <Passwort><Newsgroup-Liste>
#
localhost:Read Post:::*
erde.galaxis.de:Read:::*
```

Im Verzeichnis `/etc/news` existiert noch eine Fülle weiterer Konfigurationsdateien. Zu den meisten existieren Manual Pages, andere finden in den Howtos Erwähnung. Für den Aufbau eines einfachen lokalen Newsservers reichen jedoch die oben bechriebenen Dateien aus.

Der `innd` sollte nun gestartet werden. Um ihn in Zukunft bereits beim Systemstart zu aktivieren, ist in der Datei `/etc/rc.config` die Variable `START_INN=yes` zu setzen.
Vorerst können wir den Dämonen per Hand aktivieren:

```
root@sonne > /usr/lib/news/bin/inndstart
```

Konfiguration von suck

Mit dem Paket `suck` werden zwei Skriptvorlagen zum Holen bzw. Senden der News geliefert, die an die lokalen Gegebenheiten anzupassen sind.
Zunächst legen wir ein Basisverzeichnis zur Installation der Skripten fest. Vorgeschlagen wird für SuSE das Verzeichnis `/var/spool/news/suck`. Existiert dieses noch nicht, legen wir es an (Besitzer `news:news`) und kopieren die beiden Skripte in dieses:

```
news@sonne > cp /usr/doc/packages/suck/get.news.innxmit \
> /var/spool/news/suck/get.news
news@sonne > cp /usr/doc/packages/suck/put.news \
> /var/spool/news/suck/put.news

news@sonne > chmod +x *.news
```

get.news

Das Skript `get.news` ist nun zu bearbeiten, die zu bearbeitenden Zeilen sind durch `->` gekennzeichnet, im Beispiel ist nur der Header der Datei aufgeführt:

```
   news@sonne > vim get.news
   #!/bin/sh

   # BEFORE USING - check to ensure all the paths  defined below
   # are correct!!

   # NOTE: this script probably needs to be  run by  root.  Most
   # systems will not let a normal user run ctlinnd

-> REMOTE_HOST=java.cs.uni-magdeburg.de
-> LOCAL_HOST=localhost

-> SPOOLDIR=/var/spool/news     # base directory for articles to
                                # be rposted
-> NEWSDIR=/usr/lib/news        # base directory for news
                                # binaries
-> BASEDIR=/var/lib/news/sucks  # base directory for scripts and
                                # data files

   CTLINND=${NEWSDIR}/bin/ctlinnd       # location of binary
   SHLOCK=${NEWSDIR}/bin/shlock         # location of binary

   TMPDIR=${BASEDIR}            # location for suck.* files
   MSGDIR=${BASEDIR}/Msgs       # where to put MultiFile messages
                                # when getting them

-> SITE=java.cs.uni-magdeburg.de    # name of site from newsfeeds
                                    # file

   OUTGOING=${SPOOLDIR}/out.going/${SITE}  # location of the list
                                           #of articles to upload
   # ...

   TESTHOST=testhost
   RPOST=rpost
   SUCK=suck

   # if we are already running, abort
   # ...
```

Das Skript `put.news` sollte in der ursprünglichen Version funktionieren:

```
   news@sonne > less put.news
   #!/bin/sh

   # this is just a simple script to use sed to strip off the
   # NNTP_Posting_Host and Xref headers that my ISP's newsfeed
   # doesn't like.  this could be written as a one liner
   # sed -e SEDCMD1 $1 | sed SEDCMD2 > $2
```

```
if [ $# -ne 2 ]; then
     echo
     echo "Usage ‘basename $0‘ infile outfile "
     echo
     exit -1
fi

SEDCMD="/^NNTP-Posting-Host/d"
SEDCMD2="/^Xref/d"
OUTFILE=$2
INFILE=$1

if [ -f ${INFILE} ]; then
   sed -e ${SEDCMD} ${INFILE} | sed -e ${SEDCMD2} > ${OUTFILE}

   if [ $? -ne 0 ]; then
           echo "Error"
           exit -1
   fi
else
     echo "$1 does not exist"
     exit -1
fi
```

Bevor es ans Herunterladen der Artikel geht, beschreiben wir in einer Datei `sucknewsrc`, welche Newsgruppen wir überhaupt lokal speichern wollen:

```
news@sonne > vim sucknewsrc
alt.linux 0
alt.os.linux 0
de.markt.comp.hardware 0
```

Die Nummern hinter den Newsgruppen geben an, bis zu welcher Nummer die Artikel schon lokal vorliegen. Nach dem Aufruf von `get.news` entsprechen diese dann der höchsten heruntergeladenen Artikelnummer.
Um nicht eine Unmenge an Artikeln zu ziehen, sollte man die Nummern initial auf `-1` setzen und einmal `get.news` aufrufen. Die Datei `sucknewsrc` enthält anschließend die auf dem Newsserver liegenden Artikelnummern:

```
news@sonne > less sucknewsrc
alt.linux 7777
alt.os.linux 64476
de.markt.comp.hardware 106330
```

Durch Herabsetzen der Nummern um den Wert `x` erreicht man beim nächsten Aufruf von `get.news` das Herunterladen der letzten `x` Artikel.

12.6.4 Mail

12.7 Network File System

Das NFS kann durch drei Merkmale charakterisiert werden:

- Es ermöglicht die gemeinsame Benutzung von im Netzwerk verteilten Daten.
- Es arbeitet meist zuverlässig.
- Es eröffnet eine Sicherheitslücke.

Ein als NFS–Server ausgezeichneter Rechner bietet verschiedene Dateisysteme zum Export an; ein Client kann diese in sein lokales Filesystem importieren. Der Vorgang geschieht transparent, d.h. ein Anwendungsprogramm bemerkt keinen Unterschied, ob es auf lokale Daten zugreift oder ob die Daten irgendwo im Netzwerk liegen. Und genau hier beginnen die Probleme. Wie wir wissen, gehören alle Daten irgendjemandem. Anders ausgedrückt, besitzt jede Datei einen numerischen Wert, der den Nutzer beschreibt. Während auf einem Rechner gesichert ist, daß zwei Nutzer niemals einunddieselbe ID besitzen, ist dies in einem Netzwerk nicht mehr gegeben. Ein Nutzer auf dem einem Rechner könnte durchaus die Dateien eines fremden Nutzers mit gleicher ID manipulieren und Unix würde nichts bemerken... Um solchen Problemen zu entgehen, lernen wir im nächsten Abschnitt das Network Information System kennen.

Um sich Ärger zu ersparen, sollte man sich erst einmal der Unterstützung des Kernels vergewissern:

```
user@sonne > cat /proc/filesystems
        ext2
        umsdos
        msdos
        vfat
nodev   proc
nodev   nfs
        iso9660
        hpfs
```

Wenn die Zeile *nodev nfs* fehlt, muß der Kernel entsprechend konfiguriert werden.

12.7.1 Der NFS–Server

In der Datei /etc/exports sind alle zu exportierenden Dateisysteme zusammen mit den Zugriffsrechten enthalten:

```
# Aufbau einer /etc/exports
#
/home (rw, no_all_squash)
/projects erde.galaxis.de(rw) zeus(ro)
/usr/bin (ro, root_squash)

#
```

Der Server exportiert das Home–Verzeichnis an alle Rechner. Auf diesem Verzeichnis ist Lesen und Schreiben gestattet. Auf `/projects` darf nur von den Rechnern `erde.galaxis.de` (lesend/schreibend) und `zeus` (nur lesend) zugegriffen werden und `/usr/bin` kann von allen Rechnern gemountet werden (nur lesend), wobei dem Systemadministrator des Client–Rechners seine Sonderrechte versagt werden [5].
Per default werden in neueren Konfigurationen auch die Rechte normaler Nutzer auf den Nutzer `nobody` beschränkt. Deswegen muß das Heimatverzeichnis mit der Option `no_all_squash`.

Weitere Attribute entnehme man den Manuals (man -S 8 exports).

Voraussetzungen:

- Der Portmapper muß aktiv sein:

```
user@sonne > ps eax | grep portmap
 85    ?    S   0:00 /sbin/portmap
```

- Der NFS–Dämon muß aktiv sein:

```
user@sonne > ps eax | grep rpc.nfsd
 91    ?    S   0:00 /usr/sbin/rpc.nfsd
```

- Der Mount–Dämon muß aktiv sein:

```
user@sonne > ps eax | grep rpc.mountd
 89    ?    S   0:00 /usr/sbin/rpc.mountd
```

- Wurden Änderungen in `/etc/exports` vorgenommen, müssen diese bekanntgegeben werden:

```
root@sonne > killall -HUP rpc.mountd
root@sonne > killall -HUP rpc.nfsd
```

Hat man alle Dämonen gestartet, sollte man die Installation testen:

```
user@sonne > rpcinfo -p
   program  vers  proto  port
    100000    2    tcp    111  portmapper
    100000    2    udp    111  portmapper
    100005    1    udp    690  mountd
    100005    2    udp    690  mountd
    100005    1    tcp    693  mountd
    100005    2    tcp    693  mountd
    100003    2    udp   2049  nfs
    100003    2    tcp   2049  nfs
 545580417    1    udp    693  bwnfsd
 545580417    1    tcp    695  bwnfsd
```

[5] Seine ID 0 wird intern auf -2 (nobody) gesetzt.

12.7.2 NFS–Client

Einzige Voraussetzung ist ein laufender Portmapper, sowie ein entsprechend konfigurierter Kernel (mit NFS–Unterstützung).

Zum Mounten der Dateisysteme benötigt man *root*–Rechte:

```
mount -t nfs <NFS_Server>:<Remote_Pfad> <Lokaler_Pfad>
```

Beispiele:

```
root@erde > mount -t nfs sonne:/usr/bin /usr/bin
root@erde > mount -t nfs 192.168.10.101:/home /home
```

Entfernt werden die NFS–Dateisysteme wie normale Filesysteme:

```
root@erde > umount /usr/bin
root@erde > umount /home
```

Meist ist es sinnvoll, einige Dateisysteme unmittelbar während des Systemstarts zu importieren; hierzu trägt man eine entsprechende Zeile in die Datei `/etc/fstab` ein:

```
# /etc/fstab auf erde.galaxis.de
#
/dev/hdb1       /          ext2    defaults           1  1
/dev/hdc3       /usr       ext2    defaults           1  2
sonne:/usr/bin  /usr/bin   nfs  rsize=1024,wsize=1024 0  0
```

Optimierungen

In der Datei `/etc/fstab` setzen wir die Blockgröße für Lese– und Schreiboperationen auf 1024 Bytes. Welcher Wert tatsächlich "optimal" ist, läßt sich wie folgt testen (Das Verzeichnis `/mnt` ist vom NFS–Server importiert worden):

```
user@sonne > time dd if=/dev/zero of=/mnt/testfile bs=16k \
> count=4096
user@sonne > time dd if=/mnt/testfile of=/dev/null bs=16k \
> count=4096
```

Abbildung 12.9 zeigt den internen Ablauf einer Mount–Anforderung über das Network File System.

12.8 Das Network Information System

Dieses ursprünglich von der Firma Sun entwickelte **N***etwork* **I***nformation* **S***ystem* (**NIS**)[6] ermöglicht die Verteilung verschiedenster Informationen (z.B. Paßwörter) an die Rechner eines Netzwerkes.
Der Hintergrund ist der, daß ein Unix- bzw. Linux-System meist an ein Netzwerk angeschlossen ist und viele Rechner in diesem Netzwerk gleiche Eigenschaften

[6] Ursprünglich der Name **Y***ellow* **P***ages* (**YP**) für das System vorgesehen. Da dieser Name aber ein Markenzeichen der British Telecom war, wurde er gegen NIS ersetzt; die zugehörigen Protokolle faßt man dennoch unter YP–Protokolle zusammen.

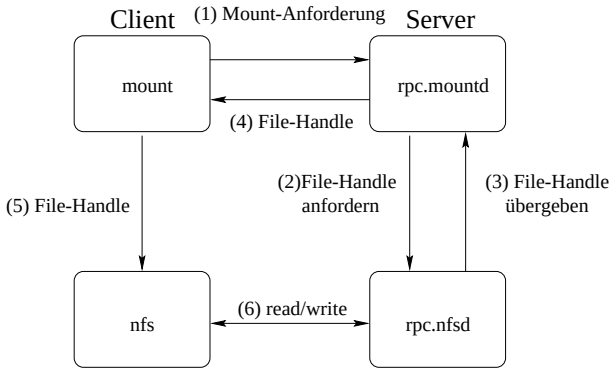

Abbildung 12.9: NFS–Mount–Ablauf

besitzen. Vorstellbar ist zum Beispiel, daß ein Nutzer an den verschiedenen Rechnern eines Netzwerkes unter dem selben Account und damit dem selben Paßwort arbeiten will. Würde der **NIS**-Service nicht existieren, müßte der Nutzer auf jedem Rechner extra eingerichtet werden und sein Paßwort, zum Beispiel bei einer Änderung, von ihm auf allen Rechnern geändert bzw. die Datei "/etc/passwd" auf alle Rechner kopiert werden. Ein sehr umständliches Verfahren, wie sich unschwer erkennen läßt. Nahezu alle systemweit geltenden Einstellungen lassen mittels NIS an zentraler Stelle verwalten, ob es die Zuordnung von Hostnamen zu IP–Adressen sind (/etc/hosts), die /etc/services oder aber global wirksame Alias–Einstellungen. Wichtigster Einsatz sind dennoch das Management von /etc/passwd und /etc/groups an zentraler Stelle, um Nutzer– und Gruppen–Identifikationen netzweit eindeutig zu halten (Vergleiche Sicherheitsprobleme beim NFS, Seite 165).

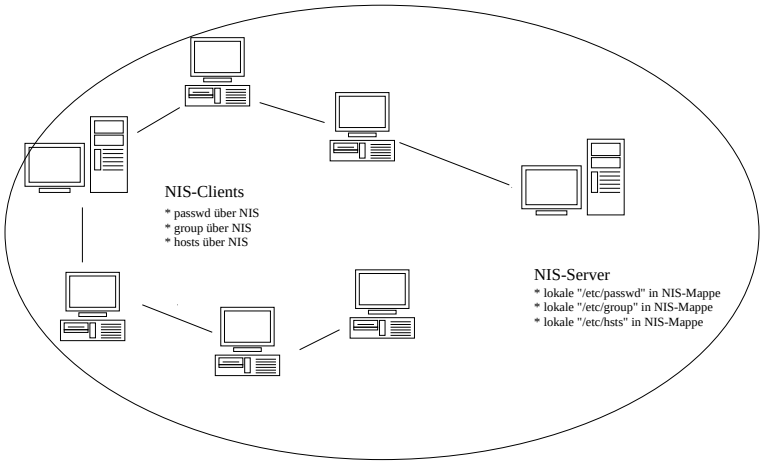

Abbildung 12.10: Beispiel eines NIS–Domain

12.8.1 Der NIS–Server

Zunächst muß dem Server mitgeteilt werden, in welchem Verwaltungsbereich er zuständig ist, d.h. es muß ein sogenannter Domainname gesetzt werden, der nichts mit dem Internet–Domainnamen zu tun hat (aber durchaus gleich heißen könnte).

```
# domainname
(none)
# domainname nis.obst
```

Alle Clients, die auf diesen Server zugreifen möchten, müssen ihrerseits denselben Domainnamen setzen! Bei SuSE läßt sich der Domainname durch Eintrag in die /etc/rc.config (YP_DOMAINNAME="nis.obst") und anschließendem Aufruf von SuSEconfig (dieses Tool ist bei jeder manuellen Änderung der rc.config aufzurufen) bereits zum Systemstart setzen.
Der NIS–Server verwaltet alle zu verteilenden Daten in Dateien, die in ein spezielles Datenbankformat (dbm) übersetzt wurden. Welche Dateien per NIS verwaltet werden sollen, wird in */var/yp/Makefile* festgelegt:

```
# Ausschnitt aus /var/yp/Makefile
# Alle nach <all> stehenden Maps werden generiert
#
all:  passwd group rpc services netid

ethers:    ethers.byname ethers.byaddr
hosts:     hosts.byname hosts.byaddr
networks:  networks.byaddr networks.byname
protocols: protocols.bynumber protocols.byname
rpc:       rpc.byname rpc.bynumber
services:  services.byname
passwd:    passwd.byname passwd.byuid
group:     group.byname group.bygid
shadow:    shadow.byname
netid:     netid.byname
netgrp:    netgroup netgroup.byhost netgroup.byuser
publickey: publickey.byname
mail:      mail.aliases
```

Ein Aufruf von

```
# (cd /var/yp; make)
```

erzeugt in /var/yp ein Unterverzeichnis mit den Namen der NIS–Domain, in welchem letztlich die NIS–Maps abgelegt werden.

In einer Datei */etc/ypserv.conf* läßt sich das Verhalten des NIS–Servers einstellen; es genügt zunächst eine leere Datei mit diesem Namen zu erzeugen. Für die möglichen Einstellungen – die vor allem die Sicherheit betreffen, durchsuche man die Manpages *man -S 5 ypserv.conf* bzw. *man -S 5 ypserv*.
Hat man den Portmapper am Laufen, startet man mittels

```
# ypserv
# /usr/sbin/rpc.yppasswdd
```

den Serverprozeß und den Dämon zur Verwaltung des Paßwortes, da ein Nutzer nun nicht mehr über *passwd* sein Paßwort ändern kann (dann würde dieses nur in der lokalen Datei geändert werden), sondern über *yppasswd*.

Zur Kontrolle überprüfen wir den Portmapper:

```
# rpcinfo -u localhost ypserv
program 100004 version 2 ready and waiting
```

Um die Maps immer auf dem aktuellen Stand zu halten, sollte man eine entsprechende Anweisung in die Master–crontab aufnehmen:

```
*/10 * * * * root make -s -C /var/yp
```

In großen Netzwerken ist es aus Performance– und Verfügbarkeitsgründen üblich, neben einem Master–Server noch mehrere Slave–Server zu starten. Diese Slave–Server erhalten in regelmäßigen Abständen vom Master die aktuellen NIS–Maps und beantworten ihrerseits Anfragen von Clients. Neben einer Entlastung des Masters erreicht man hierdurch eine Sicherheit gegen Ausfall eines Servers.

12.8.2 NIS–Clients

Zunächst muß dem Client dieselbe Domain zugewiesen werden wie dem Server. Desweiteren erhält der Client eine Liste der IP–Adressen von NIS–Servern, entweder durch Setzen der Einträge in der /etc/rc.config (YP_SERVER="192.168.1.8") oder durch Eintrag in eine Datei /etc/yp.conf:

```
# Aufbau einer /etc/yp.conf
#
ypserver 192.168.1.8
ypserver Zeus.edu.saxedu.de
```

Bei Eintrag eines symbolischen Namens muß dieser durch die lokale /etc/hosts auflösbar sein!
Welche Dateien durch NIS abgedeckt werden sollen, wird in der Datei /etc/nsswitch.conf festgelegt:

```
# Aufbau der Datei /etc/nsswitch.conf
# <Servicename> <Mechanismen>
#
passwd:     files nis
shadow:     files nis
group:      files nis
hosts:      files dns nis
services:   nisplus [NOTFOUND=return] files
networks:   nisplus [NOTFOUND=return] files
protocols:  nisplus [NOTFOUND=return] files
rpc:        nisplus [NOTFOUND=return] files
ethers:     nisplus [NOTFOUND=return] files
netmasks:   nisplus [NOTFOUND=return] files
```

```
bootparams: nisplus [NOTFOUND=return] files
netgroup:   nisplus
publickey:  nisplus
automount:  files nisplus
aliases:    files nisplus
```

Die einzelnen Einträge bedeuten:

files	–	mittels lokaler Dateien (z.B. "/etc/passwd")
dns	–	mittels DNS
nis oder yp	–	mittels NIS
nisplus oder nis+	–	mittels NIS+ (Erweiterung von NIS)
[**NOTFOUND=return**]	–	bewirkt, daß die Suche gestoppt wird, wenn der vorhergehenden Mechanismus zu keinem Ergebnis kommt

In der Datei /etc/hosts.conf sollte ein Hinweis auf NIS aufgenommen werden:

```
# /etc/host.conf
#
order hosts nis bind
multi on
```

Existiert jetzt noch das Verzeichnis /var/yp, dann kann man mittels

```
# /usr/sbin/ypbind
```

den für die Kontaktierung zum Server zuständigen Dämon starten.
Wiederum überzeugen wir uns von der erfolgreichen Installation:

```
# rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    691  mountd
    100005    2   udp    691  mountd
    100005    1   tcp    694  mountd
    100005    2   tcp    694  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
 545580417    1   udp    694  bwnfsd
 545580417    1   tcp    696  bwnfsd
    100007    2   udp    923  ypbind
    100007    2   tcp    925  ypbind
```

Ob der Server richtig erkannt wird, läßt sich schnell erfragen:

```
# ypwhich
192.168.1.8
```

Änderungen in den lokalen Dateien

Als Beispiel betrachten wir nun eine typische /etc/passwd auf einem Client.

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
+apfel:
+banane:::::/usr/banane:
+@games:NOLOGIN:
-mango
```

Dabei bedeuten:

+ Es handelt sich um einem YP–Eintrag

@ Die Zeile bezieht sich auf einen Netzwerkgruppe

- Diese Zeile wird von der NIS–Routine übergangen

Eintrag Der Wert des Eintrags überschreibt den Wert der Map

Zumindest der Zugang für den Systemverwalter *root* sollte durch die lokale /etc/passwd weiterhin möglich sein, sonst wäre im Falle eines Serverausfalls kein Einloggen im Netz mehr möglich!

Zur Abfrage der in den NIS–Mappen gespeicherten Informationen stehen mehrere Kommanods zur Verfügung:

- **ypcat** Gibt eine Map ganz aus:

```
#  ypcat -x
Use "passwd" for "passwd.byname"
Use "group" for "group.byname"
Use "networks" for "networks.byaddr"
Use "hosts" for "hosts.byname"
Use "protocols" for "protocols.bynumber"
Use "services" for "services.byname"
Use "aliases" for "mail.aliases"
Use "ethers" for "ethers.byname"

#  ypcat group
users:*:100:
```

- **ypmatch** Gibt eine Map zeilenweise aus

```
#  ypmatch users group
users:*:100:
```

- **yppasswd** Ändern des Paßworteintrages in der Paßwort–Map
- **ypwhich** Abfrage der in *ypbind* gespeicherten IP–Adresse u.a.
- **ypset** Ändern der Einstellung in *ypbind*

```
# hostname
Mango
# ypwhich
Mango
# ypset Apfel
# ypwhich
Apfel
```

- **yppoll** Diagnose der Map

12.9 Domain Name Service

12.9.1 Hintergrund

Zur Kommunikation mit anderen Rechnern sprechen wir diese gewöhnlich mit ihrem symbolischen Namen an. Die Umsetzung der symbolischen Rechnernamen in die entsprechenden IP–Adressen erfolgt in der lokalen Datei /etc/hosts.
In den Anfängen des Internets existierte im *Network Information Center* ein Rechner, der eine Datei *HOSTS.TXT* per FTP anbot, die alle bekannten Rechnernamen und Adressen enthielt. Von Zeit zu Zeit lud jeder ans Internet angeschlossenen Rechner diese Datei herunter und aktualisierte seine lokale /etc/hosts.
Heute hat die Zahl der weltweit vernetzten Rechner soweit zugenommen, daß ein einzelner Rechner kaum über die Speicherkapazitäten verfügt, um eine allumfassende /etc/hosts zu speichern, ganz davon abgesehen, daß eine solche Datei zu keinem Zeitpunkt als up–to–date angesehen werden kann.
Die Lösung des Problems bestand in der Verlagerung des Datenbestandes weg von einer Zentrale und hin zum Verursacher, also in einem hierarchisch aufgebauten Informationsdienst.

12.9.2 Namensraum

Das gesamte Netz ist in einzelne Bereiche aufgeteilt, den *Domains*. Auf oberster Stufe (unterhalb einer Wurzel) existieren die sogenannten *Top Level Domains*, die wiederum in Unterbereiche gegliedert sind.

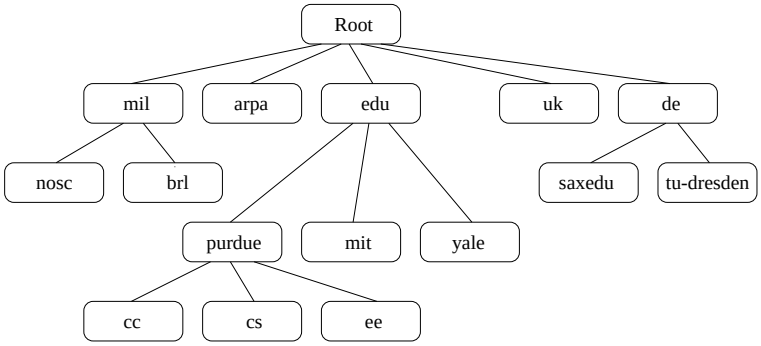

Abbildung 12.11: Struktur des Namensraums

12.9.3 Der DNS–Server

Ein Server verwaltet eine Zone, die an einem Knoten des DNS–Baumes beginnt und alle darunter liegenden Zweige beinhaltet (Abbildung 12.11).
Für die ihm zugeteilte Zone kann ein Server die Verantwortung für Teilbereiche an andere Server delegieren, die ihrerseits eine Subdomain kontrollieren. Jeder Server kennt jeweils seine Nachbarserver der nächsthöheren und nächsttieferen Zone. Aus Sicherheitsgründen gibt es in jeder Zone mindestens zwei aktive Server ("primary" und "secondary"), wobei beide dieselbe Datenbasis verwalten.
Informationen halten die Server in den *Resource Records (RR)*, die folgende Felder enthalten

- Besitzer der Information
- Typ
- Klasse (IN für Internet, ISO...)
- Dauer der Gültigkeit (Zeitpunkt der nächsten automatischen Aktualisierung)
- Daten

Folgende RR–Typen sind definiert:

- **SOA (Start Of Authority)** Zonendefinition
- **A (Address)** IP–Adresse eines Rechners
- **NS (Name Server)** Server–Name für eine Domain
- **HINFO (Host Information)** Angaben zum Betriebssystem, Hardware...
- **CNAME (Canonical Name)** Aliasname für einen Rechner
- **WKS (Well Known Services)** Liste angebotener Dienste
- **MX (Mail Exchanger)** Angabe von Postverwaltungsrechner(n) für die Domain
- **PTR (Gateway Pointer)** Adressen von Gateways

12.9.4 Die Client–Seite

Eine Instanz des *Resolvers* übernimmt für den Client (i.A. ein Anwendungsprogramm) die Anfrage beim Nameserver. Der Resolver speichert einmal ermittelte Informationen (Caching); die Dauer der Speicherung ist vom Eintrag im RR abhängig. Desweiteren ist der für Linux verfügbare Resolver in der Lage, *iterative Queries* abzusetzen, d.h. bei gescheiterten Abfragen kontaktiert er weitere Server, die eventuell die Information liefern können.
Um einen Rechner als DNS–Client zu aktivieren, ist neben der allgemeinen Netzwerkfunktionalität nur ein Eintrag in die /etc/host.conf vorzunehmen (Bind *Berkeley Internet Name Domain Server* ist die unter Linux gebräuchliche Implementierung):

```
#
# /etc/host.conf
#
order hosts bind nis
multi on
```

und eine Datei /etc/resolv.conf anzulegen:

```
# /etc/resolv.conf
#
# zustaendige Domain (optionale Angabe)
domain informatik.tu-chemnitz.de

# mit welchen Domains soll ein Hostname expandiert werden
search tu-chemnitz.de informatik.tu-chemnitz.de
       hrz.tu-chemnitz.de

# Adressen bekannter Nameserver
nameserver 192.168.85.1
nameserver 134.109.192.18
```

In der Datei /etc/nsswitch.conf sollte folgende Zeile aufgenommen werden:

```
# ...
hosts: files dns
# ...
```

12.9.5 Die Server–Seite

Wir beschränken uns zunächst auf die Beschreibung der Konfigurationsdateien für die bind–Versionen 4.xxx.
Der Domain–Server wird mittels *named* gestartet. Dieser Dämon wertet beim Start die Datei */etc/named.boot* aus, deren Aufbau wir uns näher betrachten wollen:

```
directory /var/named

cache      .                        root.cache
primary    0.0.127.in-addr.arpa     pz/127.0.0
primary    10.168.192.in-addr.arpa  pz/192.168.10
primary    galaxis.de               pz/galaxis.de
```

Die `directory`–Zeile veranlaßt den Server, sein Arbeitsverzeichnis entsprechend dem Eintrag zu wechseln.
Die `cache`–Zeile spezifiziert den Inhalt der Datei *root.cache* als Backup–Cache. Die Datei selbst enthält Informationen zu den Wurzel–Domain–Servern und wird bei SuSE mitgeliefert. Da die Informationen dieser Datei aber nicht mehr dem aktuellen Stand entsprechen müssen, kann von FTP.RS.INTERNIC.NET ein Update geholt werden.

```
# Ausschnitt aus root.cache
#
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
```

Zur Verwendung dieser Datei verfolgen wir die Auflösung einer Anfrage nach der IP–Adresse des Rechners "example.edu.saxedu.de".
Wir erinnern uns, daß die Datenbasis des DNS hierarchisch strukturiert ist, also werden wir zuerst die Lage von ".de" in Erfahrung bringen wollen.
Wer kennt einen Server, der sich mit "de" auskennt? Ein Server der Wurzel, eben einer der in `root.cache` spezifizierten Rechner. Also wird zunächst eine Anfrage an den ersten Server aus der Liste gestellt. Antwortet dieser nicht (time–out), wird der nächste Server kontaktiert, bis – hoffentlich – ein Server die Information liefert. Diese Information wird die Adresse eines Nameservers sein, der "de" verwaltet. Die nächste Anfrage nach "saxedu" richtet sich an letzteren Server, der uns weiter an einen Server vermittelt, der etwas über "saxedu" zu berichten vermag...
Letztlich steigen wir in der Hierarchie zu einem Server ab, der die IP–Adresse des gesuchten Hosts kennt.

Nun zur ersten `primary`–Zeile. In ihr wird die Datei pz/127.0.0 als 0.0.127.in-addr.arpa definiert. *pz* (steht für primary zone; es kann ein beliebiger Name gewählt werden) ist hierbei ein Unterverzeichnis in */var/named* (Vergleiche *directory*), welches die Dateien mit den Resource Records enthält. 127.0.0 ist die Netzwerkadresse des Loopback–Devices und wird hier benutzt, um einen Nur–Cache–Server zu aktivieren, d.h. einen DNS–Server, der Anfragen zu anderen Servern weiterleitet und die gewonnenen Informationen lokal zwischenspeichert, um zukünftige Anfragen direkt beantworten zu können.

in-addr.arpa ist eine besondere Domain, die ermöglicht, den Hostnamen zu einer gegebenen IP–Adresse zu ermitteln. Üblicherweise werden symbolische Namen

so angegeben, daß die Top–Level–Domain rechts steht (snoopy.gnu.org). *arpa* ist aus einer Top–Level–Domain und steht demzufolge ganz rechts. Schaut man sich die Suche nach einem Hostnamen an, wird auch die "verdrehte" IP–Adresse klar:
Gesucht wird der Hostname zur (fiktiven) Adresse 192.168.85.12. Intern sucht DNS nach 12.85.168.192.in–addr.arpa. Also sucht DNS zunächst den Server, der "arpa" bedient, dann den Server für "in–addr.arpa", nun den Server "192.in–addr.arpa"..., also wird ausgehend von der Wurzel die Adresse aufgelöst.

Werfen wir einen Blick auf die Datei 127.0.0:

```
@       IN  SOA  mango.obst.de root.mango.obst.de.
              (
              1       ; serial
              10800   ; refresh
              3600    ; retry
              604800  ; expire
              86400  ); minimum TTL
;
            NS mango.obst.de.
;
1           PTR localhost.
```

Eine andere Schreibweise für dieselbe Datei ist:

```
0.0.127.in-addr.arpa. IN  SOA  mango.obst.de root.mango.obst.de.
              (
              1       ; serial
              10800   ; refresh
              3600    ; retry
              604800  ; expire
              86400  ); minimum TTL
;
0.0.127.in-addr.arpa.   IN NS mango.obst.de.
;
1.0.0.127.in-addr.arpa. IN PTR localhost.
```

Bezugnehmend auf letztere Version folgt nach SOA der Hostname und der Verantwortliche für diesen Host, also root. Genau genommen müßte root als `root@mango.obst.de` angegeben werden, allerdings ist @ ein Sonderzeichen (siehe erste Version), das das Orignal beschreibt und dieses kann hier nicht nochmals verwendet werden.
NS teilt DNS mit, wer für 0.0.127.in–addr.arpa der Nameserver ist, PTR erklärt, daß 1.0.0.127.in–addr.arpa als localhost bekannt ist.
Entfernt man in /etc/named.boot die beiden weiteren *primary*–Zeilen (ein Semikolon zu Beginn der Zeile leitet einen Kommentar ein) und startet den Name–Server:

```
# ndc start
```

sollte man, falls man alles richtig konfiguriert hat, in /var/log/messages Einträge ähnlich der Folgenden finden:

```
# tail /var/log/messages
Nov 25 07:34:49 Haus named[185]: starting
Nov 25 07:34:49 Haus named[185]: cache zone "" loaded (serial 0)
Nov 25 07:34:49 Haus named[185]: primary zone "0.0.127.in-addr.arpa"
loaded (serial 1)
Nov 25 07:34:49 Haus named[186]: Ready to answer queries.
```

Eine erste Abfrage läßt sich mit *nslookup* tätigen:

```
# nslookup
Default Server:  localhost
Address:  127.0.0.1

> 127.0.0.1
Server:  localhost
Address:  127.0.0.1

Name:    localhost
Address:  127.0.0.1

> exit
```

Hat soweit alles geklappt, wenden wir uns der nächsten *primary*–Zeile zu und schauen uns die zugehörige Datei an:

```
85.168.192.in-addr.arpa. IN SOA  mango.obst.de root.mango.obst.de. (
                         1       ; serial
                         10800   ; refresh
                         3600    ; retry
                         604800  ; expire
                         86400   );
;
85.168.192.in-addr.arpa. IN NS mango.obst.de.
;
10.85.168.192.in-addr.arpa. IN PTR mango.obst.de.
22.85.168.192.in-addr.arpa. IN PTR apfel.obst.de.
23.85.168.192.in-addr.arpa. IN PTR birne.obst.de.
24.85.168.192.in-addr.arpa. IN PTR pflaume.obst.de.
25.85.168.192.in-addr.arpa. IN PTR erdbeere.obst.de.
```

Diese Datei ermöglicht uns das Ermitteln des Hostnamens aus einer gegebenen IP–Adresse. Für die umgekehrte Auflösung sorgt der dritte *primary*–Eintrag:

```
obst.de.   IN SOA  mango.obst.de. root.mango.obst.de. (
           1       ; serial
           10800   ; refresh
           3600    ; retry
           604800  ; expire
           86400   );
;
;
obst.de. IN NS mango.obst.de.
```

```
;
localhost.obst.de.      IN A 127.0.0.1
mango.obst.de.          IN A 192.168.85.10
apfel.obst.de.          IN A 192.168.85.22
birne.obst.de.          IN A 192.168.85.23
pflaume.obst.de.        IN A 192.168.85.24
erdbeere.obst.de.       IN A 192.168.85.25
```

Es sei besonders auf den abschließenden Punkt nach den Hostnamen hingewiesen. Dieser teilt dem Server mit, daß es sich um einen vollständigen Namen handelt, der nicht erweitert werden muß!

```
# nslookup
Default Server:  localhost
Address:  127.0.0.1

> set q=any
> obst.de
Server:  localhost
Address:  127.0.0.1

obst.de
        origin = mango.obst.de
        mail addr = root.mango.obst.de
        serial = 1
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        minimum ttl = 86400 (1 day)
obst.de nameserver = mango.obst.de
obst.de nameserver = mango.obst.de
mango.obst.de internet address = 192.168.85.10
> exit
```

Jetzt provozieren wir einmal einen Fehler, indem wir den Punkt "vergessen":

```
obst.de.   IN SOA  mango.obst.de. root.mango.obst.de. (
           1       ; serial
           10800   ; refresh
           3600    ; retry
           604800  ; expire
           86400   );
;
;
obst.de. IN NS mango.obst.de.
obst.de  IN MX 10 mail.mango.obst.de ; da fehlt der Punkt!
;
localhost.obst.de.      IN A 127.0.0.1
mango.obst.de.          IN A 192.168.85.10
apfel.obst.de.          IN A 192.168.85.22
birne.obst.de.          IN A 192.168.85.23
```

```
pflaume.obst.de.        IN A 192.168.85.24
erdbeere.obst.de.       IN A 192.168.85.25
```

Das Ergebnis sieht man beim nächsten nslookup (Ausschnitt):

```
# nslookup
> set q=any
> obst.de
 ...
obst.de nameserver = mango.obst.de
obst.de preference = 10, mail exchanger=mail.mango.obst.de.obst.de
obst.de nameserver = mango.obst.de
...
```

Zum abschluß schauen wir uns noch einige mögliche Zusatzeinträge an, die z.B. Informationen zur Hardware liefern können (Datei obst.de):

```
@          IN SOA  mango.obst.de. root.mango.obst.de. (
           1       ; serial
           10800   ; refresh
           3600    ; retry
           604800  ; expire
           86400   );
;
;
         NS mango
         MX 10 mail.mango.obst.de.
         TXT "mango.obst.de, your DNS Server"
mango    MX 10 mail
         HINFO "K6-233" "Linux 2.0.35"

;
localhost.obst.de.      IN A 127.0.0.1
mango.obst.de.          IN A 192.168.85.10
apfel.obst.de.          IN A 192.168.85.22
birne.obst.de.          IN A 192.168.85.23
pflaume.obst.de.        IN A 192.168.85.24
erdbeere.obst.de.       IN A 192.168.85.25
```

nslookup erlaubt uns auf solche Informationen zuzugreifen (es existieren weitere Optionen, man informiere sich in den Manuals):

```
# nslookup
> ls obst.de
ls obst.de
[localhost]
obst.de.                server = mango.obst.de
apfel.obst.de           192.168.85.22
mango                   192.168.85.10
birne.obst.de           192.168.85.23
pflaume.obst.de         192.168.85.24
erdbeere.obst.de        192.168.85.25
```

```
localhost               127.0.0.1

> ls -t HINFO obst.de
[localhost]
mango                   K6-233    Linux 2.0.35

> exit
```

12.9.6 Änderungen in der Version 8

Ab SuSE 5.3 wird neben dem oben beschriebenen Bind–Paket auch die neue, in ihrer Funktionalität erweiterte Version 8.1 ausgeliefert. An dieser Stelle soll kurz auf Änderungen eingegangen werden, die die Administration betreffen.

Die erste Neuerung betrifft die Eingliederung des Paketes in das Dateisystem. Per Voreinstellung wird das Verzeichnis *named* nun in */etc* angelegt und nicht mehr in */var*.
Die zweite – und wesentliche – Neuerung betrifft das Konfigurationsfile *named.boot*, welches nun *named.conf* heißt und mit einer neuen Syntax aufwartet. Der Aufbau der Datenbankenfiles bleibt identisch. Schauen wir uns nun die neue *named.conf* genauer an:

```
# /etc/named.conf (bind 8.1)
#
options {
        directory "/etc/named";
};
#
zone "0.0.127.in-addr.arpa" {
        type master;
        file "pz/127.0.0";
};
#
zone "85.168.192.in-addr.arpa" {
        type master;
        file "pz/192.168.85";
};
#
zone "obst.de" {
        type master;
        file "pz/obst.de";
};
#
zone "." {
        type hint;
        file "root.cache";
};

logging {
     channel my_syslog {
          syslog local6;
          severity info;
     };
```

```
     channel named-info-file {
          file "/var/log/named-info";
          severity info;
     };
     channel lame-server-file {
          file "/var/log/named-lame-server";
          severity info;
     };
     channel named-stats-file {
          file "/var/log/named-statistics";
          severity info;
     };

     category default  { my_syslog; };
     category panic    { my_syslog; named-info-file; default_syslog; };
     category security { my_syslog; named-info-file; default_syslog; };
     category lame-servers { lame-server-file; };
     category cname    { lame-server-file; };
     category statistics { named-stats-file; };
};
```

Offensichtlich sind die "primary"–Zeilen durch entsprechende Zonen mit dem Eintrag "type master" ersetzt worden. Ein "secondary"–Eintrag würde nun einem "type slave" entsprechen. Mit "options" wird das Arbeitsverzeichnis des Servers festgelegt und "zone ." leitet die "cache"–Zeile der alten Version ein.
Bis hierher hat sich nur die Syntax geändert, tatsächlich neu sind aber die Spezifikation zur Protokollierung von Meldungen des Servers. Eine solche Festlegung wird durch das Schlüsselwort "logging" eingeleitet. Zunächst werden durch "channel" die verschiedenen Protokolldateien definiert, um anschließend durch "category" die vordefinierten Arten von Meldungen in die entsprechenden Dateien umzuleiten. Ohne diesen "logging"–Mechanismus würde der Server alle Meldungen in die Datei "/var/log/messages" schreiben.

Anhang A

Shell–Kommandos der Bash

A.1 Die Sonderzeichen

`;`	Trennt Kommandos,
`:`	"Dummy"–Kommando, tut nichts.
`.`	Kommando ohne eigene Subshell ausführen (eine Art "include").Beispiel: `. datei` – der Punkt im Shell–Skript wird durch den Inhalt von Datei ersetzt.
`#`	Kommentar (bis zum Newline).
`#!/bin/sh`	Das Shell–Programm wird von der angegebenen Shell ausgeführt. Für die Shell selbst ist diese Zeile ein Kommentar, der Kernel erkennt hieraus das Dateiformat.
`&`	Programm im Hintergrund starten.
`&&`	bedingte Kommandoausführung. Beispiel: `com1 && com2` – com2 wird nur ausgeführt, wenn com1 erfolgreich war.
`|`	verbindet Ausgabe eines Kommandos mit der Eingabe eines anderen (Pipe).
`||`	bedingte Kommandoausführung. Beispiel: `com1 && com2` – com2 wird nur ausgeführt, wenn com1 einen Fehler liefert.
`*`	Jokerzeichen, beliebig viele (auch Null) beliebige Zeichen.
`?`	Jokerzeichen, genau ein beliebiges Zeichen.
`[abc]`	Jokerzeichen, eines aus "abc".
`[^abc]`	Jokerzeichen, alle außer "abc".
`[ ausdruck ]`	Andere Schreibweise für "test".
`~`	Steht für das Homeverzeichnis.
`>`	Umleitung der Ausgabe in eine Datei (vorhandene Datei ersetzen).
`>>`	Umleitung der Ausgabe in eine Datei (an Datei anhängen).
`>&`	Umleitung von Standardausgabe und –fehler (auch `&>`).
`<`	Umleitung der Eingabe (z.B Lesen aus Datei).
`<< ende`	Lesen aus Datei bis "ende".
`(...)`	Kommandos innerhalb der Klammern werden in einer Shell ausgeführt.

`{...}`	Kommandos gruppieren.
`{ , , }`	Zeichenketten zusammensetzen.
`$`	Inhalt von Variablen (`$PATH`).
`$* oder $@`	Liste der an das Shell–Programm übergebenen Parameter.
`$#`	Anzahl der übergebenen Parameter.
`$0`	Name des Shell–Programms.
`$?`	Rückgabewert des letzten Kommandos.
`$!`	PID des letzten Hintergrundprozesses.
`$$`	PID der aktuellen Shell.
`$1 .. $9`	Parameter 1 bis 9.
`$(...)`	Kommandosubstitution.
`${...}`	Funktionen zur Manipulation von Zeichenketten.
`$[...]`	Arithmetische Berechnung.
`"..."`	Auswertung ausgewählter Sonderzeichen verhindern (z.B. behält `$` seine Wirkung).
`'...'`	Auswertung aller Sonderzeichen verhindern.
`` `...` ``	Kommandosubstitution.
`\`*zeichen*	Hebt Wirkung des Sonderzeichens auf.

A.2 Kommandos

`alias` Definiert Abkürzung für ein Kommando.

`basename` Liefert den Dateinamen des übergebenen Pfades, Beispiele:

```
# basename /home/mango/linux.ps
linux.ps
# basename /home/mango/linux.ps ps
linux.
```

`break [n]` Bricht Schleifen vorzeitig ab, durch Angabe eines Zahlenwertes `n` wird mit erster Anweisung nach der n–ten Schleifenebene fortgefahren.

`case` Bedingte Mehrfachverzweigung. Es findet ein Mustervergleich zwischen Zeichenketten statt. Die Anweisungen nach dem ersten übereinstimmenden Muster bis zum Auftreten von `;;` werden ausgeführt. Mehrere Muster können durch `|` zusammengefaßt werden.

```
case $i in
  muster1 | muster2 ) kommando1;
                      kommando2;;
  muster3           ) kommando;;
  ...
esac
```

`cat << ende` Liest aus aktueller Datei bis zum Auftreten der Zeichenkette “ende”.

`continue [n]` Fährt mit Ausführung des nächsten Schleifendurch–

laufs fort; der optionale Parameter `n` dient zum Sprung in eine äußere Schleifenebene.

`declare` Weist der einer Shellvariablen einen neuen Wert zu oder ändert ihre Eigenschaften, Beispiel:

`declare -r PATH` – Variable darf nur gelesen werden.
`declare -x global=5` – Variable ist global definiert.

`dialog` Zur Anzeige von Dialogen. Mögliche Optionen sind:

`--clear` – entfernt Dialog nach dessen Ende vom Bildschirm.
`--title text` – zeigt text als Dialogtitel an.
`--msgbox text x y` – zeigt text an, der mit RETURN bestätigt werden muß. x gibt die Höhe und y die Breite an.
`--infobox` text x y – zeigt text an, ohne Bestätigung.
`--yesno text x y` – Ja/Nein–Entscheidungsbox.
`--inputbox text x y` – Eingabe einer Textzeile.
`--textbox datei x y` – Ausgabe des Dateiinhalts.
`--menu text x y ...` – Auswahl einer Option aus mehreren.
`--checklist text x y ...` – Auswahl mehrerer Optionen.

`dirname` Pfadname eines vollständigen Dateinamens.

`dirs` Liste der durch `pushd` gespeicherten Verzeichnisse.

`echo` Ausgabe einer Zeichenkette auf dem Bildschirm.

`eval $var` Interpretiert Variableninhalt als Kommandozeile und führt diese aus (inklusive Substitution):

```
# homepwd="cd ; pwd"
# $homepwd
bash: cd;: command not found
# eval $homepwd
/home/mango
```

`exec` Startet ein angebenes Kommando.

`exit` Beendet ein Shell–Programm, ein Rückgabewert kann angegeben werden.

`export` Deklariert eine globale Variable, mit Option `-n` wird eine Umgebungsvariable wieder zu einer normalen Shellvariablen.

`expr` Dient zur Auswertung arithmetische Ausdrücke oder zum Vergleich von Zeichenketten.
Als Ergebnis liefert expr entweder die Anzahl der Zeichen zurück, die dem Muster entsprechen, oder , falls Teile des Musters in `\(...\)` eingeschlossen sind, die passende Zeichenkette.

```
# expr abcdefghi : a.g
7
# expr abc_egf_hij : .*_\(.*\)_.*
efg
```

`file` Versucht den Dateityp der übergebenen Datei zu bestimmen, mit Option `-z` wird versucht, den Typ einer komprimierten Datei zu erkennen.

`for` Bildet eine Schleife über eine Liste von Zeichenketten:

```
for i in a b c; do
    echo $i
done
```

Wird for innerhalb eines Shellprogrammes ohne "`in ...`" aufgerufen, wird die Variable mit der Liste der dem Programm übergebenen Parameter verbunden.

`function name()` Definiert eine Funktion.

`if ...; then` Bedingte Verzweigung, der jeweilige Block wird nur ausgeführt, wenn die Bedingung erfüllt ist. Bedingungen können verschachtelt werden, ein optionaler `else`–Zweig wird behandelt, wenn keine Bedingung zutrifft.

```
if bedingung; then
   kommandos
elif bedingung; then
   kommandos
else
   kommandos
fi
```

`local` Definiert innerhalb einer Funktion eine lokale Variable:

```
function Beispiel() {
   local a
   local text="Beispiel"
...
}
```

`popd` Wechselt in das zuletzt in `pushd` gespeicherte Verzeichnis.

`printf` Formatierte Ausgaben ähnlich zu C (man -S 3 printf).

`pushd` Speichert aktuelles Verzeichnis und wechselt in ein angegebenes Verzeichnis.

`read` Zur Eingabe von Text.

`readonly` Zeigt die schreibgeschützten Variablen der Shell an.

`setterm` Ändern der Einstellungen des Terminals.

`shift [n]` Verschiebt die Liste der Parameter um `n` Einträge. Das Kommando ist notwendig, wenn einem Shellprogramm mehr als 10 Parameter übergeben werden.

`sleep n` Programm "schläft" für n Sekunden.

`source` Führt angegebene Datei aus, als würde sie im Programmtext stehen (analog zu "include" in C).

`test` Dient zur Formulierung von Bedingungen.
test liefert "0", wenn Bedingung erfüllt ist und "1" sonst. Eine alternative Schreibweise ist `[ ausdruck ]`.

Zeichenkettenvergleiche:
`[ zk ]` – 0, wenn Zeichenkette zk nicht leer ist.
`[ -z zk ]` – 0, wenn Zeichenkette zk leer ist.
`[ zk1 = zk2 ]` – 0, wenn Zeichenketten übereinstimmen.
`[ zk1 != zk2 ]` – 1, wenn Zeichenketten übereinstimmen.

Für zwei Zahlen existieren folgende Vergleiche:
`-eq` – gleich
`-ne` – ungleich
`-gt` – größer als
`-ge` – größer gleich
`-lt` – kleiner als
`-le` – kleiner gleich

Eigenschaften von Dateien können getestet werden:
`[ -d datei ]` – 0, wenn datei ein Verzeichnis ist.
`[ -e datei ]` – 0, wenn datei existiert.
`[ -f datei ]` – 0, wenn datei eine "normale" Datei ist.
`[ -L datei ]` – 0, wenn datei ein symbolischer Link ist.
`[ -r datei ]` – 0, wenn datei gelesen werden kann.
`[ -s datei ]` – 0, wenn datei mind. 1 Byte lang ist.
`[ -w datei ]` – 0, wenn datei geschrieben werden kann.
`[ -x datei ]` – 0, wenn datei ausgeführt werden kann.

Bedingungen lassen sich verknüpfen:
`[ !bed ]` – 0, wenn Bedingung falsch ist.
`[ bed1 -a bed2 ]` – 0, wenn beide Bedingungen erfüllt sind.
`[ bed1 -o bed2 ]` – 0, wenn mind. eine Bedingung erfüllt ist.

`trap` Führt ein Kommando in Abhängigkeit eines aufgetretenen Signals aus:

`trap $(echo "SIGHUP") 1` – erzeugt die Ausgabe, falls das Signal "SIGHUP" aufgetreten ist.
`trap -l` listet alle bekannten Signale auf.

`ulimit` Begrenzt die verfügbaren Systemressourcen.

`ulimit -d 4096` – Der Prozeß darf maximal 4k für sein Datensegment verwenden.

`unalias` Entfernt einen Alias.

`unset` Löscht eine Variable.

`until` Bedingte Ausführung einer Schleife:

```
until bedingung; do
  kommandos
done
```

`wait [nr]` Wartet auf Terminierung des angegebenen Hinergrund–prozesses. Ohne Angabe von `nr` wird auf alle Hinter–grundprozesse gewartet.

`while` Bedingte Ausführung einer Schleife:

```
while bedingung; do
  kommandos
done
```

Anhang B

Der Editor vi

B.1 Wichtige Optionen im Kommandomodus

Beginnt in nachfolgende Tabelle die Option nicht mit einem Doppelpunkt, dann wird diese mittels

```
:set [Option]
```

gesetzt und mit

```
:set no[Option]
```

wieder abgeschalten.

Option	Bedeutung
:se[opt]	Abfrage des Wertes einer Option.
:se all	Anzeige der Werte aller Optionen.
ap	autoprint: Änderungen werden am Bildschirm angezeigt (default).
aw	Beim Beenden wird automatisch der Puffer zurückgeschrieben.
bf	beautify: Entfernen aller Steuerzeichen aus dem Text.
ht	Tabulator–Schrittweite.
ic	ignorecase: Groß– und Kleinschreibung werden nicht unterschieden.
nu	Zeilennumerierung einschalten.
scroll	Bestimmt den Scrollbereich.
sh	Bestimmt die Shell, deren Kommandos innerhalb des Editors aufgerufen werden können.
wa	Erlaubt das Zurückschreiben des Puffers in die Originaldatei.

B.2 Starten und Beenden

Der *vi* kann auf verschiedene Arten gestartet werden:

Kommando	Bedeutung
vi	Start ohne Dokument.
vi dok	Start mit dem angegebenem Dokument. Existiert *dok* nicht, wird die Datei neu angelegt.
vi +dok	Der vi arbeitet mit einer Kopie von *dok* und öffnet die Datei selbst erst beim Beenden und Zurückschreiben.
vi -r dok	Nach einem Systemabsturz wird die Abarbeitung an der Stelle fortgesetzt, an der die Unterbrechung stattfand. Laut Voreinstellung speichert der vi aller 30 Sekunden die Puffer ab.
vi -R dok	Eröffnet *dok* nur zum Lesen.

Das Beenden kann u.a. auf folgende Art und Weise erfolgen:

Kommando	Bedeutung
:q	Verlassen ohne zu speichern. Funktioniert nur, falls das Dokument nicht verändert wurde.
:q!	Verlassen ohne zu speichern auch bei modifiziertem Dokument.
:wq	Schreiben des Puffers und Verlassen.
:wn	Der Puffer wird geschrieben und das nächste Dokument geladen.
:ZZ	wie :wq.
:x	wie :wq.

B.3 Positionierung des Cursors

Kommando	Positioniert Cursor auf
^	erstes Zeichen der aktuellen Zeile
0	erste Spalte der aktuellen Zeile
$	letztes Zeichen der aktuellen Zeile
h	ein Zeichen nach links
l	ein Zeichen nach rechts
k	ein Zeichen nach oben
j	ein Zeichen nach unten
w	Anfangs des nächsten Wortes
b	Anfang des vorhergehenden Wortes
e	Ende des aktuellen Wortes
CTRL n	gleiche Spalte, nächste Zeile
CTRL p	gleiche Spalte, vorhergehende Zeile
nG	n–te Zeile des Dokuments
+	Textanfang der nächsten Zeile
-	Textanfang der vorhergehenden Zeile
H	erste Zeile des Bildschirms
M	mittlere Zeile des Bildschirms
L	letzte Zeile des Bildschirms
(	Satzanfang
)	Satzende
{	Absatzanfang
}	Absatzende

B.4 Editieren

Kommando	Einfügen und Ersetzen
i	Einfügen links vom Cursor
I	Einfügen am Zeilenanfang
a	Einfügen rechts vom Cursor
A	Einfügen am Zeilenende
o	neue Zeile hinter der aktuellen einfügen
O	neue Zeile vor der aktuellen einfügen
rc	Ersetze Zeichen unter Cursor durch c
R	Überschreiben ab Cursorposition
s*text*	Ersetze ein Zeichen durch *text*
S*text*	ersetze ganze Zeile durch *text*
ns*text*	Ersetze n Zeichen durch *text*
cw*text*	Ersetze Wort durch *text*
cc*text*	wie S*text*

B.5 Text kopieren

Kommando	Kopierfunktion
yy	kopiert aktuelle Zeile in Puffer
ny	kopiert n+1 Zeilen in Puffer
yw	kopiert ein Wort in Puffer
P	Fügt Inhalt des Puffers vor der aktuellen Zeile ein
p	Fügt Inhalt des Puffers nach der aktuellen Zeile ein

B.6 Text löschen

Kommando	Löscht
x	Zeichen unter Cursor
nx	n Zeichen ab Cursorposition
dd	aktuele Zeile
ndd	n Zeilen ab aktueller Zeile
dL	bis zum unteren Bildschirmrand
dw	ein Wort
d)	bis Absatzende
D	bis Zeilenende

B.7 Suchen und Ersetzen

/muster	Suche vorwärts nach Muster
/	Wiederholt vorwärts
?muster	Suche rückwärts
?	Wiederholt rückwärts
n	Wiederholt letztes Suchkommando
:s/alt/neu	Sucht und ersetzt alt durch neu (einmal)
:s/alt/neu/g	Sucht und ersetzt alle alt und neu in aktueller Zeile
:1,$s/alt/neu	Ersetzen im gesamten Dokument

Für Informationen zu weiteren Kommandos rufe man im *vi* ":h" auf.

Index