Die Linuxfibel

Thomas Ermer
Michael Meyer

Diese Seiten entstanden mit Unterstützung der **Saxonia Systems AG**.

Und wer vielleicht einen Kurs belegen möchte oder einer professionellen Linux-Zertifizierung bedarf, der findet beim **Saxonia Bildungsinstitut** die richtigen Ansprechpartner.

Neues auf der Seite

Asche auf mein Haupt! Lange ist's her, dass sich etwas Wesentliches in der Linuxfibel tat. Und obwohl vor allem Michael mich immer fort mit neuem Material versorgte, schaffte ich es nicht, dieses in angemessener Zeit in Form zu bringen und damit zumindest einige Löcher in der Linuxfibel zu stopfen. Auch die jetzt vorliegende Fassung glänzt mehr durch ein neues Aussehen als durch neue Inhalte; mal abgesehen vom neuen Kapitel Sicherheit, das sich formals Netzwerk-Sicherheit nannte. Ein paar Sachen wie bspw. der Logical Volume Manager sind dann doch hinzugekommen und auch Aktualisierungen und zahlreiche Korrekturen (Danke an die Leser, die mir diese zusandten!).

Leider werde ich in kommender Zeit eher weniger Aufmerksamkeit der Fibel widmen können (man hat ja noch andere Interessen;-), sodass mir (schreibende) Hilfe sehr willkommen wäre. Wenn sich also jemand zum Autoren berufen fühlt, ein wenig XHTML beherrscht (das unterscheidet sich nur unwesentlich von HTML), den in der Fibel gepflegten (Schreib)Stil mag und dann auch noch in einem der Linuxthemen über einigermaßen fundiertes Wissen verfügt, dann...: **BITTE MELDEN!**

Versuch der Abschätzung des derzeitigen Standes

Die folgende Tabelle versucht den Stand der Dinge widerzuspiegeln. Die Farben symbolisieren dabei folgenden Status:

- (Fast) fertig...
- Wird derzeit verstärkt bearbeitet...
- Wird derzeit kaum bearbeitet...

Des Weiteren gibt die prozentuale Angabe den genauen Bearbeitungsstand an.

Einleitung	100% fertig	System-Administration	95% fertig
Erste Schritte	100% fertig	X-Window-System	15% fertig
Die Bash	100% fertig	Der Kernel	50% fertig
Das Dateisystem	100% fertig	Netzwerk Grundlagen	65% fertig

Nutzerkommandos	100% fertig	Netzwerk Clients	25% fertig
Installation	100% fertig	Netzwerk Server	25% fertig
Shells	50% fertig	Sicherheit	40% fertig
Unix Werkzeuge	100% fertig	Anhang	65% fertig

Die letzten signifikanten Änderungen

Januar 2003	Debian-Paketverwaltung, Vsftp-Server, Netzwerk-Diagnose (teilweise) u.a.m.
Februar 2003	Version 0.8.3 steht zum Download bereit
Januar 2004	Umfassende Layout-Änderungen (Umstellung auf XHTML und CSS), Fehlerkorrekturen, Aktualisierungen und kleinere Erweiterungen
Januar 2004	Version 0.8.4 steht zum Download bereit

Hinweise

Die Navigation im Buch erfolgt entweder mit Hilfe der Menüs auf der linken Seite oder mittels des Kompasses:

Ein Klick auf die Mitte (Haussymbol) bringt Sie zurück auf diese Startseite
Ein Klick oben öffnet die Startseite des folgenden Kapitels
Ein Klick unten öffnet die Startseite des aktuellen Kapitels (bzw. auf der Startseite selbst das vorangegangene Kapitel)
Ein Klick links führt zur vorhergehenden Seite
Ein Klick rechts bringt Sie auf die nächste Seite

Die Druckversionen der HTML-Seiten werden bei Installation des RPM-Pakets unter Linux automatisch mittels eines Skripts erstellt. Unter Linux kann das Skript auch per Hand aufgerufen werden. Das Skript selbst und Hinweis zur Verwendung finden Sie im Anhang Skripte.

Die PDF-Version müssen Sie nicht separat herunter laden. Wie Sie sie aus den HTML-Quellen erzeugen können, ist ebenso im Anhang Skripte beschrieben.

Konventionen

Vereinfacht ausgedrückt existieren in Unix-Systemen zwei Privilegierungsstufen von Benutzern. Zum einen ist das **root** - der Systemadministrator -, dessen Nutzerkennzeichen auf jedem Unix-System existiert. Einige Kommandos sind einzig **root** vorbehalten. Diese werden in allen Beispielen des Buches durch das Prompt

```
root@sonne>
```

symbolisiert.

Kommandos, die jedem Benutzer zur Verfügung stehen, werden durch

```
user@sonne>
```

dargestellt.

Alle Beispiele, Ausschnitte aus Dateien und Shellskripten werden durch einen hervorgehobenen Hintergrund kenntlich gemacht. Innerhalb dieser Markierung werden alle Ausgaben des Systems bzw. der Kommandos in normaler Schrift dargestellt. Eingaben des Nutzers werden durch **fette Schrift** und Kommentare farblich gekennzeichnet. In Skripten und

ähnlichen verwenden wir ggf. weitere Farben, um die Programmstruktur zu unterstreichen.

In Beispielen von Benutzereingaben, die Tastenkombinationen erfordern, umschreiben wir die Tasten mit eckigen Klammern. »[Ctrl]-[A]« bedeutet dabei, das die Tasten [Ctrl] (entspricht [Strg] auf deutschen Tastaturen) und [A] gleichzeitig zu betätigen sind. »[Ctrl]-[M],[X]« meint, dass [Ctrl] und [M] gleichzeitig und [X] nachfolgend zu drücken ist.

Download

Um automatische Download-Werkzeuge auszubremsen (die unnötigerweise sämtliche Pakete laden würden), bitten wir Sie, die URL der Download-Seite von Hand einzugeben: http://www.linuxfibel.de/download.htm.

Stand der Pakete: **07.01.2004**.

...und nochmals die Bitte, mich über Fehler, Ungereimtheiten, Änderungen in Versionen beschriebener Programme usw. zu unterrichten.

Ach ja - Ein Dank an Michael für das Schnüren der RPM-Pakete!

Kontakt

Hinweise, Korrekturvorschläge, Kritiken, Danksagungen an:

Thomas Ermer, Autor und Webmaster

Michael Meyer, Autor

Das Dateisystem - Überblick

Überblick

Schon eine minimale Linux-Installation kommt mit einigen tausend Dateien daher, eine Vollinstallation der »großen« Distributionen dürfte einen Umfang von ca. 500.000 Dateien haben!

Wohin mit all den Daten?

In diesem Kapitel betrachten wir die typische Verzeichnisstruktur eines Unix-Systems. Anhand des Filesystem Hierarchie Standard **FHS** verfolgen wir die Eingliederung der verschiedenen Dateiarten in die Verzeichnisstruktur. Sie werden kennen lernen, wo Sie nach Konfigurationsdateien suchen müssen, welche Verzeichnisse Programme enthalten können, wo die Dokumentationen zu finden sind usw.

Und Sie erfahren, wie Sie auf eine Diskette oder CDROM zugreifen können.

Ziel des Kapitels

Nach dem Studium dieses Abschnitts sollten Sie wissen:

- Welche drei Hierarchiestufen in der Verzeichnisstruktur existieren
- In welchen wesentlichen Verzeichnissen Programme abgelegt sind
- Wo die Konfigurationsdateien liegen
- Was Devices sind
- Was es mit dem Begriff des Mountens auf sich hat
- Wie Sie eine Diskette ins Dateisystem integrieren

Inhalt des Kapitels

Das Dateisystem - Verzeichnishierarchie

Strukturierung ist notwendig

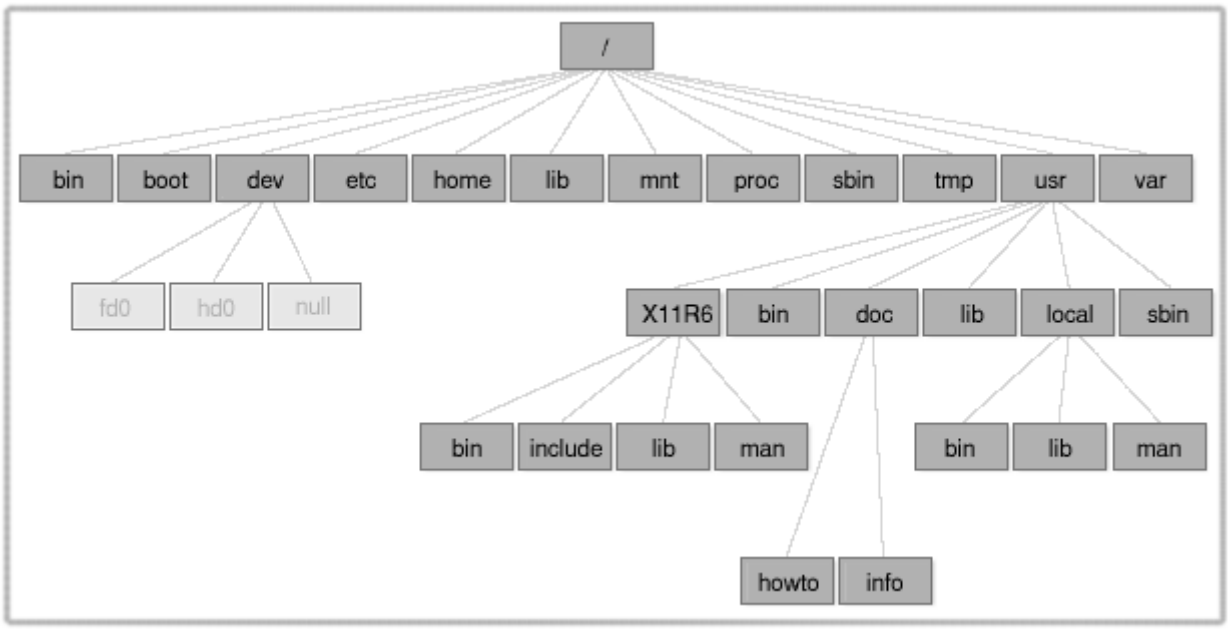


Abbildung 1: Die Linux-Verzeichnisstruktur

Bei der Unmenge von Dateien in Unix-Systemen (auf meinem System sind es 106839!) ist eine hierarchische Struktur unabdingbar. Lange Zeit brachte jedes Unix-Derivat seine eigenen Vorstellungen vom Aufbau seiner Dateiverwaltung mit, aber unterdessen ist sich ein Gremium, bestehend aus den wichtigsten Unix-Distributoren, System- und Paketentwicklern, mehr oder weniger einig geworden und erarbeitete den **Filesystem Hierarchie Standard**, der wichtige Strukturen definiert. Die meisten Distributionen folgen diesen Richtlinien, wobei Abweichungen oft durch unterschiedliche Auslegung des Standards begründet werden.

Neben der Beschreibung der vorgesehenen Verwendung jedes Verzeichnisses werden auch konkrete Kommandos genannt, die mindestens in diesen Verzeichnissen vorhanden sein müssen. Des Weiteren finden systemspezifische Vorgaben und optionale Komponenten Erwähnung.

Ausgangspunkt ist die Wurzel (auch als Root bezeichnet)

Sobald der Kernel aktiv ist, lädt er als erstes das Root-Dateisystem, 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

Verzeichnisse in der Wurzel

Wenden wir uns also zunächst dem Inhalt des Wurzelverzeichnisses zu. Verzeichnisse und Dateien, die als optionale Komponenten im Standard enthalten sind, werden farblich hervorgehoben.

/bin

Die wichtigsten Kommandos, um mit dem System arbeiten zu können, finden Sie hier. Sie dürfen von jedem Benutzer ausgeführt werden. Zu den Kommandos gehören `cat`, `chgrp`, `chmod`, `chown`, `cp`, `date`, `dd`, `df`, `dmesg`, `echo`, `ed`, `false`, `kill`, `ln`, `login`, `ls`, `mkdir`, `more`, `mount`, `mv`, `ps`, `pwd`, `rm`, `rmdir`, `sed`, `setserial`, `sh`, `stty`, `su`, `sync`, `true`, `umount`, `uname` (`sh` ist in Linuxsystemen zumeist ein Link auf `bash`).

/boot

Hier finden Sie die statischen Dateien des Bootmanagers 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 Treiber zur Ansteuerung der gesamten Hardware (Festplatte, Floppy, RAM...). Gemäß der UNIX-Philosophie »Alles ist eine Datei« werden diese Treiber durch Dateien repräsentiert, die so genannten **Gerätedateien** (»device special files« oder »device nodes«). Einige dieser Gerätedateien erfüllen spezielle Aufgaben (Beispiele finden Sie an etlichen Stellen des Buches) und sind mit keiner realen Hardware verbunden. Dennoch stellen sie aus Sicht des Systems Treiber dar.

Wenn Sie sich den Inhalt des Verzeichnisses »/dev« betrachten, entdecken sie unzählige Einträge, deren Anzahl unmöglich der vorhandenen Hardware entsprechen kann. Tatsächlich wurde für jedes denkbare Stück Hardware vorsorglich ein solcher »Eintrittspunkt« geschaffen, dessen definierter Name es Anwendungsprogrammen erst ermöglicht, auf ein konkretes Gerät zuzugreifen. Solange keine entsprechende Hardware und der zugehörige Treiber installiert wurden, sind diese Eintrittspunkte nur »leere Hülsen« und würde eine Anwendung darauf zugreifen, erhielte sie eine Fehlermeldung, dass das angeforderte Gerät nicht verfügbar ist.

Obige Verfahrensweise beinhaltet jedoch einige Schwächen, auf die wir im einzelnen an dieser Stelle nicht eingehen werden. Als Lösung bahnt sich seit Kernelversion 2.4. das so genannte Device File System an, womit erst bei der Registrierung eines Treibers beim Kernel eine entsprechende Gerätedatei unterhalb von »/dev« generiert wird. Noch dazu werden Geräte entsprechend ihrer Klasse (bspw. IDE-Festplatten, Terminals, SCSI-Geräte) in separate Unterverzeichnisse eingeordnet, sodass der Inhalt von »/dev« die tatsächliche Hardware des Systems in struktuierter Form repräsentiert. Im Abschnitt Systemadministration➔> Dateisysteme widmen wir uns konkret diesem neuen Dateisystem.

Noch aktiviert keine Distribution standardmäßig das neue Verfahren und selbst bei dessen Verwendung werden zur Unterstützung älterer Programme die »alten« Gerätedateien zusätzlich verwendet (ein Prozess überwacht hierzu die Anforderungen der Programme und kümmert sich ggf. um das Mapping auf die neuen Einträge unter »/dev«), sodass nachfolgende Aussagen weiterhin ihre Gültigkeit behalten.

Drei Informationen sind für jedes Device relevant:

Art des Zugriffs

```
brw-rw-rw-   1 root     disk       2,   0 Nov  8 20:48 /dev/fd0
crw-rw----   1 root     lp         6,   0 Nov  8 20:48 /dev/lp0
```

- Blockorientiert (b) - gepufferter Zugriff, z.B. Festplatten
- Zeichenorientiert (c) - ungepufferter Zugriff, z.B. Bildschirm, Drucker

Hauptgerätenummer (major device number)

```
brw-rw-rw-   1 root     disk       2,   0 Nov  8 20:48 fd0
crw-rw-rw-   1 root     tty        2,   0 Nov  8 20:48 ptyp0
```

Diese Nummern spezifizieren den zu verwendenden Treiber. Befinden sich im System bspw. zwei IDE-Festplatten, werden beide über einunddenselben Treiber angesprochen. Nicht allein die Nummer benennt den Treiber, sondern die Art des Geräts (zeichen- und blockorientiert) ist ebenso entscheidend. Identische Nummern für zeichen- und blockorientierte Geräte (z.B. Nummer 2 für Pseudoterminals (c) und Floppys (b)) beziehen sich also auf unterschiedliche Treiber!

Die Hauptgerätenummern werden von einer zentralen Instanz verwaltet; der aktuelle Stand bei Auslieferung der Kernelquellen ist in der Datei »/usr/src/linux/Documentation/devices.txt« zu finden.

Nebengerätenummer (minor device number)

```
brw-rw-rw-   1 root     disk      2,  0 Nov  8 20:48 /dev/fd0
```

Diese Nummer dient dem Treiber zu entscheiden, welche Instanz einer Hardware anzusprechen ist. Sind bspw. vier IDE-Festplatten im System installiert, erkennt der Treiber anhand der Nebengerätenummer, ob er den Master oder den Slave des ersten oder zweiten IDE-Controllers anzusteuern hat.

Spezielle Devices

Als erstes Beispiel eines solchen »Pseudotreibers« soll hier `/dev/null`, der »Mülleimer von Unix« angeführt werden. Sie werden bei der täglichen Arbeit mit Linux noch genühend Situationen erfahren, in denen Sie die Bildschirmausgaben von Programmen gern unterdrücken möchten. Im vorhergehende Kapitel zu den Grundlagen der Bash haben Sie die Umleitung der Ausgaben in eine Datei kennen gelernt. Das Resultat wäre eine Datei. Sie könnten sich abschließend löschen oder aber gleich anstatt der regulären Datei den Mülleimer verwenden:

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

Sie verstehen das Beispiel nicht? Dann arbeiten Sie bitte nochmals den Abschnitt Ein/Ausgabe-Umleitung durch.

Wichtige Gerätedateien

cdrom
: Link auf eine entsprechende Datei (z.B. cdu535)

cua*
: (Veraltetes) Devices für serielle Schnittstellen, das für ausgehende Modemverbindungen verwendet wurde (und in manchen Distributionen noch immer wird). Physisch zeigt ein solches Device auf dasselbe Gerät wie /dev/ttys*, jedoch blockiert ein Programm nicht, wenn es das Gerät eröffnet und noch kein Verbindungssignal anliegt. Aktuell sollten die Schnittstellen /dev/ttyS* bevorzugt werden.

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*

(Veraltetes) Device für die seriellen Schnittstellen, das vornehmlich zur Überwachung eingehender Verbindungen genutzt wurde. Ein Programm, das diese Datei eröffnet, wird blockiert, solange das Modem kein »Carrier Detect« meldet. Aktuell sollten die Schnittstellen /dev/ttyS* bevorzugt werden.

ttyS*

(Neues) Device für die seriellen Schnittstellen, das sowohl für eingehende als auch für ausgehende Verbindungen genutzt werden sollte. Die bei /dev/cua* und /dev/ttys* angedeuteten Probleme mit blockierenden Programmen werden vollkommen durch den Kernel behandelt. Somit ist es (ohne Umwege) möglich, auf einer Schnittstelle auf eingehende Verbindungen zu warten. Solange eine solche Verbindung nicht eröffnet wurde, kann auf derselben Schnittstelle eine ausgehende Verbindung eröffnet werden. Für den Zeitraum einer aktiven Verbindung bleibt der jeweils andere Prozess blockiert.

/etc

Enthält alle lokalen Konfigurationsdateien (Tastatur, X, Netzwerk...)

/home

Alle Heimatverzeichnisse der Nutzer finden Sie standardmäßig hier. Nach dem Login landet jeder Benutzer i.d.R. in seinem »Home«. Heimatverzeichnisse können vom Systemverwalter auch an anderer Stelle angesiedelt werden.

/lib

Die beim Systemstart benötigten Bibliotheken stehen hier. Ebenso liegen die Kernelmodule in einem eigenen Unterverzeichnis unterhalb von `/lib`.

/mnt

Mountpunkt für temporäre Partitionen

/opt

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

/root

Heimatverzeichnis des Administrators. In realen Unix-Installationen werden die Heimatverzeichnisse aller Benutzer oft auf einem Server gehalten. Bei einem Ausfall eines solchen Servers sollte aber zumindest Root in der Lage sein, vernünftig mit dem System zu arbeiten. Daher liegt dessen Heimatverzeichnis direkt unterhalb der Verzeichnisbaumwurzel.

/sbin

Wichtige Systemprogramme (beim Booten benötigt; Ausführung erfordert Root-Rechte)

/tmp

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

/usr

Die zweite Hierarchie

/var

Variable Daten

Verzeichnisse unter /usr

Der Filesystem Hierarchie Standard bezeichnet die Verzeichnisse unter `/usr` (user system ressources) als »zweite Hierarchie«. Schauen wir uns an, was dort stehen sollte:

`X11R6`	X Window System (Version 11, Release 6)
`X386`	X Window System (Version 11, Release 5 auf x86er)
`bin`	Die meisten Nutzerprogramme
`games`	Spiele und Lernprogramme
`include`	Headerdateien für C-Programme
`lib`	Allgemeine Bibliotheken (außer X11)
`local`	Lokale Hierarchie. Hier hat der Administrator die Möglichkeit, Nicht-Standard-Pakete einzuspielen.
`sbin`	(weniger wichtige) Systemprogramme
`share`	Architektur-unabhängige Dateien
`src`	Quelldateien zu den Paketen

Links unter /usr

Einige (symbolische) Links sind ebenso vorgeschrieben:

`/usr/spool`	Link auf `/var/spool`
`/usr/tmp`	Link auf `/var/tmp`
`/usr/spool/lock`	Link auf `/var/lock`

Verzeichnisse unter /var

Es gibt eine Menge Daten, die permanenter Veränderung unterliegen oder nur kurze Zeit existieren. Protokollierungen fallen ebenso in diese Kategorie, wie auch Mails, zu druckende Dateien, News, ... Insbesondere auf Servern sollte dem Verzeichnis »/var« eine eigene Partition gegönnt werden.

`account`	Prozessnutzungsprotokoll (falls unterstützt)
`cache`	Zwischenspeicher von Programmen
`crash`	Speicherauszug bei Systemabsturz (falls unterstützt)
`games`	Variable Spieledaten
`lock`	Sperren (Dateien, Geräte, etc.)
`log`	Protokolle über Systemvorgänge
`mail`	Mailboxen der Nutzer
`opt`	Variable Daten der optionalen Programme
`run`	Dateien zu laufenden Prozessen

`spool`	Von Anwendungen gespoolte Daten
`state`	Variable Status Informationen
`tmp`	Temporäre Dateien, die zwischen Reboots erhalten bleiben
`yp`	Dateien des Network Information Systems

Linux Systeme

Speziell für Linux-Systeme definiert der Standard Weiteres:

allgemein	Der Name des Standard-Kernels ist `vmlinux` oder `vmlinuz`
`/dev`	Enthält nur die im Dokument »Linux Allocated Devices« beschriebenen Links (sonst wie oben)
`/proc`	Enthält Kernel- und Prozessinformationen in einem virtuellen Dateisystem
`/sbin`	Enthält zusätzlich Routinen zum `ext2`-Dateisystem und `lilo`
`/usr/src`	Enthält zusätzlich die Kernelquellen

Des Weiteren werden im Standard **Daten nach folgenden Kriterien** unterschieden:

- Statische und variable Daten
- Gemeinsam und exklusiv nutzbare Daten

Die Eingliederung der Speichermedien in die Verzeichnisstruktur

Übersicht

»Jede Datei und jedes Verzeichnis ist ausgehend von der Wurzel erreichbar.« So oder so ähnlich steht es im letzten Abschnitt geschrieben...

Aber wo finde ich denn nun meine zweite und dritte Festplatte wieder? Wo sind die Daten meiner Diskette? Wie navigiere ich auf der CDROM?

Jedes Unix-Dateisystem ist so gestaltet, dass es die physische Struktur der Speichermedien vor dem Anwender verbirgt. Die Partition einer Festplatte beinhaltet dabei das Root-Dateisystem, also die Wurzel selbst. Alle weiteren Partitionen/Festplatten werden in die Struktur integriert, indem ein (fast) beliebiges Verzeichnis auf eine solche Partition/Festplatte verweist.

Nicht nur der Zugriff auf die Partitionen von Festplatten wird über diese Technik realisiert, auch die Wechselmedien wie Diskette und CDROM gliedern sich in dieses Schema ein, ebenso kann der Zugriff auf Daten, die irgendwo im Netz liegen, so realisiert werden.

Für den Anwender geschieht das alles transparent. Er bemerkt es höchstens an der Geschwindigkeit beim Zugriff auf Daten, welches Speichermedium sich tatsächlich hinter einem Verzeichnis verbirgt. Und selbst hier können Techniken des Cachings (Zwischenspeichern von Daten) den Mangel an Tempo vertuschen.

Dennoch besteht die Frage: **»Wenn ich meine Diskette einlege... wie kann ich nun darauf zugreifen?«**.

Indem eine Verbindung zwischen einem existierenden Verzeichnis und dem Gerät (Device), der das Diskettenlaufwerk repräsentiert, hergestellt wird. Und hier kommt der **Begriff des Mountens** ins Spiel. »Mounten« (montieren) bezeichnet den Vorgang, dass ein Gerät mit einem konkreten Verzeichniseintrag verbunden wird.

mount - Verbindung zwischen Gerät und Verzeichnis

Bevor auf die Daten einer Festplatte, Diskette, ... zugegriffen werden kann, muss also die Verbindung zu einem existierenden Verzeichnis hergestellt werden.

Zumindest die Partition, die das Wurzelverzeichnis enthält, wird schon während des Systemstarts vom Kernel »gemountet«. Das Betriebssystem benötigt die dortigen Programme und Bibliotheken für seine Arbeit.

Wurde bei der Installation das System auf mehrere Partitionen verteilt, so werden meist noch weitere Partitionen während des Bootvorganges in die Verzeichnisstruktur integriert. Um welche es sich dabei handelt, steht in der Datei `/etc/fstab`, aber um deren Aufbau soll es an dieser Stelle nicht gehen.

Wichtig ist nur zu wissen, dass man zum Mounten von Geräten die **Rechte** dazu besitzen muss. Im produktiven Einsatzbereich wird dieser doch recht entscheidende Eingriff ins System einzig dem Administrator - also Root - vorbehalten sein, in weniger sicherheitskritschem Umfeld liberalisiert man die Berechtigungen dahin gehend, dass zumindest die Disketten- und CDROM-Laufwerke vom »normalen« Anwender mit gewissen **Einschränkungen** gemountet werden dürfen.

Diskette und CDROM auf der Konsole

Dieses Vorgehen funktioniert auch in den Terminals der verschiedenen Windowmanager.

Der Aufruf des Mount-Kommandos erwartet folgende Syntax:

```
mount [Optionen] Gerät Verzeichnis
```

Von den möglichen Optionen interessieren uns derzeit nur der **Typ des zu mountenden Dateisystems**. D.h. wir müssen `mount` mitteilen, wie die Daten auf dem Medium gespeichert sind. Auf einer CDROM sind die Daten immer nach dem ISO9660-Standard-Format abgelegt, also ist die entsprechende Option `-t iso9660`. Auf Disketten sind die Daten meist in einem Minix-Dateisystem organisiert (Option `-t minix`) oder im MSDOS-Format (Option `-t msdos`).

Welches Gerät ist nun zu mounten? Der Zugriff auf die Hardware erfolgt über so genannte Devices (man kann sich darunter einen Treiber vorstellen). Alle Devices liegen im Verzeichnis `/dev`.

Die Namensgebung bei **Diskettenlaufwerken** ist simpel: das erste Laufwerk verbirgt sich hinter `/dev/fd0`, das zweite heißt `/dev/fd1`, usw. Das Speicherformat der Diskette (Zylinder/Sektoren) sollte in den meisten Fällen automatisch erkannt werden. Gelingt dies einmal nicht, muss das dem Format zugeordnete Device direkt angesprochen werden. Die Namen der Geräte lassen auf das entsprechende Speicherformat schließen:

```
user@sonne> ls / dev/ fd0*
/dev/fd0          /dev/fd0h1600 /dev/fd0u1120 /dev/fd0u1840 /dev/fd0u720
/dev/fd0CompaQ /dev/fd0h360   /dev/fd0u1440 /dev/fd0u1920 /dev/fd0u800
/dev/fd0d360      /dev/fd0h410   /dev/fd0u1600 /dev/fd0u2880 /dev/fd0u820
/dev/fd0h1200     /dev/fd0h420   /dev/fd0u1680 /dev/fd0u3200 /dev/fd0u830
/dev/fd0h1440     /dev/fd0h720   /dev/fd0u1722 /dev/fd0u3520
/dev/fd0h1476     /dev/fd0h880   /dev/fd0u1743 /dev/fd0u360
/dev/fd0h1494     /dev/fd0u1040 /dev/fd0u1760 /dev/fd0u3840
```

CDROM-Laufwerke müssen leider etwas anders angesprochen werden, da es hier mehrere Industriestandards gibt und jeder seinen eigenen Treiber benötigt. Hat der Administrator das System nutzerfreundlich eingerichtet, kann meist das Device `/dev/cdrom` verwendet werden, das ein Verweis auf die eigentliche Gerätedatei sein sollte. Wenn nicht sollten wir die Bauart und den Anschluss des Laufwerkes kennen. »Moderne« IDE-CDROM-Laufwerke (so genannte ATAPI-Laufwerke) werden wie Festplatten behandelt und gliedern sich in deren Namensraum ein. Die erste (E)IDE-Festplatte (sie sollte im »Normalfall« als Master am ersten IDE-Controller betrieben sein) bezeichnet man dabei mit `/dev/hda`, die als Slave am ersten Controller installierte Platte nennt sich `/dev/hdb`. Der Master am zweiten Controller wird über `/dev/hdc` angesprochen, und der Slave letztlich über `/dev/hdd`. Wenn wir jetzt wissen, an welchem Controller das CDROM-Laufwerk angeschlossen ist, kennen wir die benötigte Bezeichnung.
Ein SCSI-CDROM-Laufwerk folgt der Benennung der SCSI-Festplatten. Die erste SCSI-Platte wird mit `/dev/sda` angesprochen, die zweite mit `/dev/sdb`... und irgendwo reiht sich auch das CDROM-Laufwerk ein.
Bei herstellerspezifischen Geräten sucht man nach einem entsprechenden Device (z.B. wird ein Sony-CDROM-Laufwerk als `/dev/sonycd` bezeichnet).

Was jetzt noch fehlt ist das **Verzeichnis**, worin wir das Gerät mounten möchten. Prinzipiell kann jedes Verzeichnis - bei Wahrung der Rechte - verwendet werden. Allerdings werden Daten, die in einem solchen Verzeichnis eventuell vorhanden sind, durch einen Mountvorgang verdeckt. D.h. sie stehen nach dem Mounten bis zum Unmounten nicht zur Verfügung. Aus diesem Grund wird man sich für ein leeres Verzeichnis als Mountpunkt entscheiden.

Wer ein SuSE-System vor sich hat, findet in der Wurzel die beiden (leeren) Verzeichnisse `/cdrom` und `/floppy`, die als Mountpunkte schon vorgesehen sind. Bei anderen Linux-Distributionen nennen sich die Verzeichnisse meist `/mnt/cdrom` bzw. `/mnt/floppy` (Siehe Verzeichnisstruktur). Sind solche Einträge nicht vorhanden, kann man als Systemadministrator auch neue Verzeichnisse erstellen und diese als Mountpunkte verwenden.

Mit all den Erläuterungen sollten die nachfolgenden Beispiele leicht verständlich sein:

```
root@sonne> mount -t msdos / dev/ fd0 / floppy

root@sonne> mount -t iso9660 / dev/ hdc / mnt/ cdrom
```

Wenn Sie einen Blick in die schon erwähnte Datei /etc/fstab werfen und dort Einträge ähnlich dieser:

```
user@sonne> egrep 'floppy| cdrom' / etc/ fstab
/dev/hdc  /cdrom  iso9660  ro,noauto,user,exec  0 0
/dev/fd0  /floppy  auto  noauto,user  0 0
```

vorfinden, dann ist die Eingabe der kompletten Befehlszeile nicht mehr notwendig. Außerdem ist nun das Mounten auch dem »normalen« Anwender gestattet (Eintrag »user« in den Zeilen). Es genügt nun:

```
user@sonne> mount / floppy

user@sonne> mount / dev/ hdc
```

Erklärung: `mount` erkennt die unvollständige Befehlszeile und schaut nun selbst in der Datei `/etc/fstab` nach, ob das angegebene Argument dort auftaucht. Wird das Kommando fündig, ergänzt es die fehlenden Parameter selbsttätig. Als Angabe erwartet `mount` entweder die Bezeichnung des Gerätes oder den Namen des Verzeichnisses, so wie es in der Datei angegeben ist. Der normale Anwender ist nicht berechtigt, die Werte der /etc/fstab »umzubiegen« (z.B. einen anderen Mountpunkt anzugeben).

Diskette und CDROM unter KDE und Gnome

Unter KDE und - bei entsprechender Konfiguration - auch unter Gnome werden die eigentlichen Mechanismen des Mountvorganges vor dem Nutzer verborgen.

Durch Anklicken des entsprechenden Symbols auf dem Desktop mit der linken Maustaste werden das Disketten- bzw. das CDROM-Laufwerk automatisch gemountet (im Hintergrund wird einfach der entsprechende Mount-Aufruf durchgeführt) und deren Inhalte im Dateimanager angezeigt. Ist eines der Geräte gemountet, wird dies bei der üblichen Konfiguration anhand eines anderen Symbols angezeigt. Soll nun das Medium gewechselt werden, ist das Unmounten mit anschließendem erneuten Mounten notwendig.

umount - Eine Verbindung lösen

Spätestens beim Herunterfahren des Systems werden alle Geräte automatisch abgehangen. Macht sich aber ein Wechsel der Diskette oder der CDROM notwendig, so ist dies dem System mitzuteilen.

Auf der Konsole

```
user@sonne> umount / mnt/ cdrom

user@sonne> umount / dev/ fd0
```

Das Kommando `umount` ist das Gegenstück zum Mounten, d.h. die Verbindung zwischen Gerät und Verzeichnis wird aufgehoben. Dem Kommando genügt dabei die Angabe entweder des Gerätes oder des Verzeichnisses.

Unter KDE oder Gnome

Erfolgte das Mounten über die entsprechenden Symbole auf dem Desktop, dann löst man die Verbindung, in dem

man mit der rechten Maustaste auf das Symbol klickt und im aufklappenden Menü den Eintrag »unmounten« anwählt.

Nutzerkommandos

Überblick

Was wäre ein Betriebssystem ohne die zahlreichen nützlichen Programme? Ein Ansammlung kryptischer Systemrufe, mit denen kaum jemand ernsthaft arbeiten könnte.

Unter Nutzerkommandos haben wir versucht, eingeteilt nach dem Verwendungszweck, die wichtigsten Kommandos zusammen zu fassen. **Nutzer** deutet die mit solchen Programmen verbundenen Berechtigungen schon an: sie dürfen von jedem ausgeführt werden. Voraussetzung ist natürlich, dass die entsprechenden Pakete installiert wurden.

Von allen möglichen Optionen der Kommandos beschränken wir uns auf die »Gebräuchlichsten« und davon auch nur jene, die dem »normalen« Benutzer zur Verfügung stehen (die also keine Administratorrechte erfordern). Zumeist schöpfen wir hierbei aus unserer Erfahrung im Umgang mit den verschiedenen Programmen. Beispiele zu allen Kommandos sollen das Verständnis für deren Gebrauch stärken helfen.

Vorab seien zwei Optionen erwähnt, die nahezu alle Kommandos kennen: **--help** (oder manchmal auch »-h«) zeigt, der Name spricht Bände, eine Kurzhilfe zum Gebrauch des Kommando an und **--version** gibt Versionsnummer und eventuelle Compileroptionen preis.

Ziel des Kapitels

Der Leser sollte die wichtigsten Kommandos kennen und sie mit den verschiedenen Optionen anwenden können.

Inhalt des Kapitels

Nutzerkommandos - Archivierung

Einleitung

Werkzeuge zur Archivierung ermöglichen in erster Linie das »Packen« mehrerer Dateien in eine einzige Datei. Darüber hinaus bestehen Möglichkeiten, die Daten zu komprimieren. Damit verringert sich nicht nur die Netzlast bei eventueller Datenübertragung, sondern es spart so manchen Platz auf der heimischen Festplatte. Bei heutigen Festplattenkapazitäten mag dies überflüssig anmuten, aber so manche Datei lässt sich in gepackter Form noch auf eine einzelne Diskette platzieren.

Zur Demonstration der Güte der einzelnen Werkzeuge werden die Dateien der Linuxfibel (eine frühe Version) mit tar archiviert und anschließend mittels gebräuchlicher »Kompressoren« gepackt. Die dabei berechneten Kompressionsraten dienen einzig der Veranschaulichung, letztlich hängen sie stark von der Art der Dateien ab. Die hier getroffenen Aussagen gelten im Wesentlichen für Textdateien.

Die verwendete Archivdatei »Linuxfibel.tar« ist ungepackt knapp 2 Megabytes groß (Stand: Ende 2000).

```
user@sonne> ls -l Linuxfibel.tar
-rw-r--r--   1 user   users     1914880  May 23 14:07  Linuxfibel.tar
```

Die Liste der unterstützten Archivierer und Komprimierungstool ist unter Linux umfangreicher als es dieser Abschnitt vermitteln kann. Alte Bekannte wie »lharc«, »zoo« oder »unrar« wurden auch zwar Linux portiert, haben es jedoch zu keiner nennenswerten Verbreitung gebracht. Wir kehren ihre Diskussion schlicht und einfach unter den Tisch.

bzip2 - Dateien komprimieren

Aufruf: bzip2 [-cdfkqstvzVL123456789] [filenames ...]

Dieses Programm basiert auf dem **Burrows-Wheeler** blockorientierten Kompressionsalgorithmus mit Huffman Kodierung und ist derzeit das effektivste Komprimierungsprogramm unter Linux. Allerdings erreichte es bislang nicht die Verbreitung eines »gzip«, obwohl es in keiner Standardinstallation fehlt. Eine Datei wird mit folgendem Befehl komprimiert:

```
user@sonne> bzip2 Linuxfibel.tar
user@sonne> ls -l Linuxfibel.tar.bz2
-rw-r--r--   1 user   users     772999  May 23 14:07  Linuxfibel.tar.bz2
```

bzip2 versieht seine komprimierten Dateien mit der Endung ».bz2« und erwartet dieses auch im Falle eines Entpackens. Greifen Sie auf das Kommando »file« zurück, falls sich das Format einer Datei einmal nicht anhand der Endung offenbart.

Um die Kompressionsrate zu berechnen, bemühen wir den Taschenrechner bc

```
user@sonne> bc -l
bc 1.05
Copyright 1991, 1992, 1993, 1994, 1997, 1998 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
(1 - 772999 / 1914880) * 100
59.631987383021390375 00
quit
user@sonne>
```

Somit wird eine Kompressionsrate von 59.6% erreicht.

Entpackt wird diese Datei mittels

```
user@sonne> bunzip2 Linuxfibel.tar.bz2
```

compress - Dateien komprimieren

Aufruf: compress [-f] [-v] [-c] [-V] [-r] [-b bits] [name ...]

Der Komprimierungsklassiker unter UNIX kennzeichnet von ihm gepackte Dateien standardmäßig mit der Endung ».Z«. **compress** beruht auf einer älteren Version des **Lempel-Ziv-Algorithmus**. Seine Kompressionsraten sind um einiges geringer als beim de facto Linux-Standardpacker gzip, dennoch existiert das Programm auf jedem UNIX-System.

```
user@sonne> compress Linuxfibel.tar
user@sonne> ls -l Linuxfibel.tar.Z
-rw-r--r--   1 user    users      1171919  May 23 14:07  Linuxfibel.tar.Z
```

Die erzielte Kompressionsrate beträgt ganze 38.8%, was nicht mehr ganz dem Stand der Technik entspricht...

Entkomprimiert wird mit dem Kommando **uncompress**:

```
user@sonne> uncompress Linuxfibel.tar.Z
```

cpio - Konvertierung von Archivformaten

Aufruf: cpio -o [-0acvABLV] [-C bytes] [-H format] [-M message] [-O [[user@]host:]archive] [-F [[user@]host:]archive] < name-list [> archive]
cpio -i [-bcdfmnrtsuvBSV] ...
cpio -p [-0adlmuvLV] ...

Mittels **cpio** (**copy i**n and **o**ut) lassen sich aus Dateien und Verzeichnisbäumen »cpio-Archive« erzeugen. Dabei liest **cpio** die zu archivierenden Dateien nicht von der Kommandozeile, wie etwa »tar«, sondern von der Standardeingabe. Das hört sich zwar kompliziert an, ist es aber keinesfalls. Denn so etwas lässt sich sehr leicht mit den Kommando ls oder find und einer Pipe realisieren:

Um alle html-Seiten der Linux-Fibel zu archivieren, ist die Option **-o** bzw. **--create** zu benutzen:

```
user@sonne> cd ~user/doc/linuxfibel
user@sonne> ls *.htm | cpio -o > Linuxfibel.cpio
```

Zur Archivierung ganzer Verzeichnishierarchien bietet sich die Kombination mit dem Kommando find an:

```
user@sonne> find ~user/doc/linuxfibel -print | cpio -o > Linuxfibel.cpio
```

Um das Archiv wieder auszupacken, verwendet man die Option **-i** bzw. **--extract**:

```
user@sonne> cpio -i < Linuxfibel.cpio
# oder auch:
user@sonne> cpio -il Linuxfibel.cpio
```

Wie zu erkennen ist, liest »cpio« ebenso seine eigenen Archive von der Standardeingabe.

Die Option **-t** bzw. **--list** ermöglicht das Betrachten des Inhalt eines »cpio-Archives«:

```
user@sonne> cpio -t < Linuxfibel.cpio
```

Alternativ eignet sich »cpio« in Verbindung mit der Option **-p** (»--pass-through«) zum Kopieren von Dateien in ein Verzeichnis:

```
user@sonne> find ~user/doc | cpio -p /tmp/user/doc
```

Letzteres Kommando entspricht einem einfachen Kopierbefehl mittels cp, nämlich »cp -r ~user/doc/* /tmp/user/doc«. Warum also so umständlich? Durch die Kombination mit find ist »cpio« um einiges flexibler. Die vielfältigen Optionen von letzterem Kommando helfen ungemein, um tiefversteckte Dateien schnell und selektiv zu finden. Das wirkt sich dann natürlich vorteilhaft auf »cpio« aus.

Zusätzliche Optionen zu den oben genannten sind die folgenden:

-m bzw. --preserve-modification-time

Zeitstempel bleiben erhalten

-v bzw. --verbose

Anzeige der gerade bearbeitende Datei

-d bzw. --make-directories

Notwendige Verzeichnisse werden erstellt

-H Format bzw. --format=FORMAT

Format des cpio-Archives (-H ustar : POSIX-tar Format)

dpkg - Paketverwaltung a'la Debian

Aufruf: dpkg [options]

Dieses Kommando steht nur den Benutzern von Debian-Linux zur Verfügung. Es erledigt im Wesentlichen dieselbe Funktionalität wie das verbreitete rpm anderer Distributionen ((De)Installation, Versionskontrolle, Paketverwaltung, Update), nur arbeitet es mit ***.deb**-Archiven zusammen.

Das Paketmanagement bleibt dem Administrator vorbehalten. Der »normale« Benutzer muss sich mit Abfragen zu Paketinformationen begnügen.

Einen Überblick über alle installierten Pakete verschafft die Option **-l**, durch Anfügen eines Paketnamens beschränkt sich die Ausgabe auf dieses.

```
user@sonne> dpkg -l bash
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
```

```
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version          Description
+++-==============-================-
=====================================
ii  bash           2.03-6           The GNU Bourne Again SHell
```

Eine Liste der in einem Paket enthaltenen Dateien bringt die Option **-L** zum Vorschein:

```
user@sonne> dpkg -L bash
/.
/bin
/bin/bash
/bin/rbash
/bin/sh
/usr
/usr/bin
/usr/bin/bashbug
/usr/share
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/builtins.1.gz
/usr/share/man/man1/sh.1.gz
/usr/share/man/man1/rbash.1.gz
/usr/share/man/man1/bash.1.gz
/usr/share/man/man1/bashbug.1.gz
/usr/share/man/man3
/usr/share/info
/usr/share/doc
/usr/share/doc/bash
...
```

Und schließlich lässt sich mit der Option **-S** nach dem Paket suchen, in dem die angegebene Datei enthalten ist:

```
user@sonne> dpkg -S mcopy
manpages-de: /usr/man/de/man1/mcopy.1.gz
```

Der Dateiname darf die shellüblichen Metazeichen enthalten, allerdings müssen diese vor der Interpretation durch die Shell geschützt werden (z.B. durch Voranstellen des Backslash).

gzip - Dateien komprimieren

Aufruf: gzip [-acdfhlLnNrtvV19] [-S suffix] [name ...]

Diese GNU-Version des alt bekannten zip ist der Standardpacker unter Linux. Er basiert auf der **Lempel-Ziv**-Kodierung (LZ77) und seine gepackten Dateien enden i.d.R. auf ».gz«.

```
user@sonne> gzip Linuxfibel.tar
user@sonne> ls -l Linuxfibel.tar.gz
-rw-r--r-- 1 user users 833729 May 23 14:07 Linuxfibel.tar.gz
user@sonne> gzip -l Linuxfibel.tar.gz
compressed uncompr. ratio uncompressed_name
    833729  1914880  56.4% Linuxfibel.tar
```

Die Option »-l« aktiviert quasi den »gzip«-eigenen Taschenrechner, der die Kompressionsrate gleich mit ausgibt. In diesem Fall sind es 56.4%.

Entpacken geschieht durch das Kommando »gzip -d« (»--decompress«) oder einfach mit **gunzip**:

```
user@sonne> gunzip Linuxfibel.tar.gz
```

rpm - Der RedHat Package Manager

Aufruf: rpm [options]

Auf den ersten Blick ähneln die »rpm«-Pakete des RedHat Package Managers den hier behandelten tar oder zip-Archiven. Sie bestehen genauso aus einer Sammlung von Dateien und Dateiinformationen. Allerdings steht eine gänzlich andere Philosophie dahinter. Die durch rpm-Pakete installierten Dateien, sprich Programme, Konfigurationsdateien, Bibliotheken u.a., werden in einer Datenbank fest gehalten. Hierdurch wird es erst möglich, Paketzugehörigkeiten, Versionskontrolle, Aktualisierung und saubere (!) Deinstallation zu realisieren. Weiterhin werden Abhängigkeiten von Paketen berücksichtigt.

Installation, Aktualisierung und Löschen von rpm-Paketen ist Aufgabe des Systemadministrators. Genauso fällt das Erzeugen ebensolcher in seinen Aufgabenbereich. In diesem Kapitel, das auf die Belange eines Einsteigers gemünzt ist, soll nur auf die Abfrage von Paketen und Dateien eingegangen werden. Eine Abfrage, eine **--query**, entspricht im Unix98 Standard der Option **-q**. Um bspw. herauszufinden, zu welchen Paket das Programm vi gehört, wird die »Anfrage« mit einer zusätzlichen Option **-f** bzw. **--file** kombiniert:

```
user@sonne> rpm -qf /usr/bin/vi
vim-5.4-13
```

Im Paket »vim« ist dieser Editor zu finden. Dieses Paket selbst ist in der Version 5.4 Release 13 installiert. Bevor genauer auf dieses Beispiel eingegangen wird, werden erst einmal die wichtigsten Optionen zur Paketabfrage aufgelistet:

-l bzw. --list

Auflisten aller Dateien des Paketes

-i

Informationen zum Paket

-R bzw. --requires

Zeigt alle benötigten Programme

-s bzw. --status

Zeigt Status aller Dateien des Paketes (normal, not installed oder replaced)

-a bzw. --all

Zeigt alle installierten Pakete an (ohne zusätzliche Paketangabe)

Welche Dateien sind durch das Paket »vim« installiert? Die Antwort verrät folgende Anfrage:

```
user@sonne> rpm -ql vim
/etc/vimrc
/usr/bin/rview
/usr/bin/rvim
/usr/bin/vi
/usr/bin/view
/usr/bin/vim
/usr/bin/vimtutor
/usr/bin/xxd
...
```

Welche Informationen über das Paket selbst sind verfügbar?

```
user@sonne> rpm -qi vim
Name        : vim                          Relocations: (not relocateable)
Version     : 5.4                             Vendor: SuSE GmbH, Nuernberg, Germany
Release     : 13                           Build Date: Mon 08 Nov 1999 20:48:21 MET
Install date: Don 27 Jan 2000 16:59:31 MET  Build Host: allen.suse.de
Group       : unsorted                       Source RPM: vim-5.4-13.src.rpm
Size        : 3629468                         License: 1999 - not specified
Packager    : feedback@suse.de
Summary     : Vim
Description :
Vim (Vi Improved) is an almost compatible version of the UNIX editor vi
whereby almost every possible command can be performed using only ASCII
characters. Only the 'Q' command is missing (you don't need it). Many new features have
been added: multi level undo, command line history, filename
completion, block operations, editing of binary data, etc.
Vi is available for the AMIGA, MS-DOS, Windows NT, and various versions
of UNIX.
For SuSE Linux, Vim is used as /usr/bin/vi.

Authors:
--------
    Bram Moolenaar <mool@oce.nl>
```

Diese Abfragekommandos können auch auf rpm-Pakte angewendet werden, die als Dateien vorliegen. Mit

```
user@sonne> rpm -qR TolleSoftware-1.00-2.i386.rpm
```

lässt sich erfragen, welche Programme bzw. Pakete notwendig für die Installation dieser »tollen Software« sind.

tar - Das Standard-Archiv-Format

Aufruf: tar [OPTION]... [Dateien]...

Dieses Programm wurde ursprünglich zur Verwaltung von Bandarchiven benutzt, daher auch der Name **tar** (**t**ape **ar**chiver). Der heutige Funktionsumfang ermöglicht ebenso das Schreiben zu archivierender Daten in Dateien oder gar auf unformatierte Disketten (/dev/fd0)... Letzteres hat den großen Vorteil, dass die Daten unabhängig vom Filesystem sind, d.h. jedes UNIX-Betriebssystem wäre befähigt, auf die Daten zugreifen.

Folgende Optionen sind die Wesentlichsten zur Verwaltung von so genannten tar-Archiven:

-c bzw. --compress

Archivieren

-t bzw. --list

Anzeigen

-x bzw. --extract

Extrahieren

-r bzw. --append

Anfügen

-u bzw. --update

Austausch nur neuerer Dateien

--delete

-f DATEI bzw. --file= DATEI

Archivdatei (auch /dev/fd0)

-C VERZEICHNIS bzw. --directory= VERZEICHNIS

Ins VERZEICHNIS wechseln

-v bzw. --verbose

Ausführlichere Anzeige

-M bzw. --multi-volume

Falls ein Band nicht reichen sollte

Um unser obiges Beispielarchiv der Linuxfibel zu erzeugen, wurden folgende Kommandos ausgeführt:

```
# Zuerst in das Quellverzeichnis wechseln
user@sonne> cd ~user/doc/linuxfibel
# Nun alle Dateien archivieren
user@sonne> tar -cf Linuxfibel.tar *.htm images/
```

Mit obigem Kommando werden alle Dateien mit der Endung ».htm« und das Verzeichnis »images« einschließlich Inhalt in der Datei »Linuxfibel.tar« archiviert.

Ob alles geklappt hat, verrät:

```
user@sonne> tar -tf Linuxfibel.tar
```

Bei zusätzlicher Verwendung der Option **-v** im obigen Kommando werden Information über Dateigrößen, Eigentümer, Zeitmarke und Zugriffsrechte mit angezeigt. Wenn nicht anders angegeben, RTFM, bleiben die Dateiattribute erhalten.

Ohne Dateiangaben wird alles extrahiert. Betrifft dies nur bestimmte Dateien, so erfolgt dies wie folgt:

```
user@sonne> tar -xvf Linuxfibel.tar images/*.gif
```

Normalerweise werden Archive mit »tar« nicht komprimiert. Man muss, wie in den Beispielen auf dieser Seite, das Archiv explizit selbst packen. Das GNU-tar jedoch unterstützt folgende Packer mittels der aufgelisteten Optionen:

-z bzw. --gzip

gzip

-Z bzw. --compress

compress

-j bzw. --bzip2

bzip2 (bei älteren Versionen von tar I (großes i) anstatt j!)

```
user@sonne> tar -xzf Linuxfibel.tar.gz images/*.gif
```

Anzumerken ist, dass in einem gepackten Archiv, auch wenn dies durch »tar« selbst komprimiert wurde, keine Dateien hinzugefügt, erneuert oder gelöscht werden können. Dies kann nur im unkomprimierten tar-Archiv

geschehen.

zip - Dateien komprimieren

Aufruf: zip [-AcdDeEfFghjklLmoqrRSTuvVwXyz@$] [-b path] [-n suffixes] [-t mmddyyyy] [-tt mmddyyyy] [zipfile [file1 file2 ...]] [-xi list]

Als letzten Archivierungs- und Kompressionstool sei hier **zip** erwähnt. Es ist kompatibel mit dem unter MS-Windows weit verbreiteten »PKZIP« (Phil Katz ZIP) und kann daher gut zum Datenaustausch genutzt werden.

Um die Linuxfibel zu archivieren, wird folgendes Kommando aufgerufen:

```
user@sonne> cd ~ user/ doc/ linuxfibel
user@sonne> zip -r Linuxfibel.zip *
user@sonne> ls -l Linuxfibel.zip
-rw-r--r--   1  user   users     990163  May 23 14:07  Linuxfibel.zip
```

Auch wenn die Online-Hilfe von zip (»man zip« oder »zip -h«) anderer Meinung ist, klappt das rekursive Archivieren nur mit der Option **-r** und nicht mit »-R«. Groß- und Kleinschreibung ist sowieso ein Problem bei Kompatibilätsversuchen mit MS-DOS. Gut, dass es dafür noch die Optionen »-L« und »-U« gibt, womit Klein- bzw. Großschreibung erzwungen werden kann.

Die Kompressionsrate beträgt im Beispiel 48.3%. Entpacken funktioniert mit **unzip**:

```
user@sonne> unzip Linuxfibel.zip
```

»unzip« ermöglicht ebenso das Betrachten des Archivinhalts. Hierzu ist die Option **-t** zu verwenden, die gleichzeitig auch die einzelnen Dateien des Archivs testet.

```
user@sonne> unzip -t Linuxfibel.zip
```

Nutzerkommandos - Dateien und Verzeichnisse

cd - Verzeichnisse wechseln

Aufruf: cd [-PL] [dir]

Change Directory dient zum Wechsel des aktuellen Verzeichnisses. Wird dem Kommando kein Argument mitgegeben, wechselt »cd« in das durch »$HOME« beschriebene Verzeichnis (HOME ist die Shellvariable, die den Namen des Heimatverzeichnisses eines Benutzers enthält):

```
user@sonne> pwd
/usr/lib
#  Aufruf von »cd« ohne Argument und anschließende Ausgabe des aktuellen Pfadnamens
user@sonne> cd; pwd
/home/user
```

Beginnt das Argument mit einem Schrägstrich **/** , so handelt es sich um eine **absolute** Pfadangabe. »cd« versucht den Pfad ausgehend von der Wurzel zu finden.

Alle anderen Argumente werden als **relative** Pfadangaben behandelt. Ist die Variable »$CDPATH« gesetzt, sucht das Kommando zuerst in allen diesen Verzeichnissen nach dem durch das Argument beschriebene Verzeichnis. Erst wenn es dieses dort nicht findet, sucht »cd« ausgehend vom aktuellen Verzeichnis. Die Verwendung der Variable »$CDPATH« ist ungewöhnlich und kann leicht zu Verwirrung führen.

```
user@sonne> cd / usr/ src; pwd
/usr/src

user@sonne> cd ../ ../ usr; pwd
/usr

user@sonne> export CDPATH= "/ :/ var"
user@sonne> cd tmp; pwd
/tmp
```

Mittels symbolischer Links ist es möglich, auf Verzeichnisse zu verweisen, die an beliebigen Stellen im System liegen. Normalerweise - bzw. mit der Option **-L** - folgt »cd« der logischen Struktur solcher Links auch »rückwärts«:

```
user@sonne> ls -l / usr/ dict
lrwxrwxrwx   1 root     root     10 Jan 27 17:21 /usr/dict -> share/dict

user@sonne> cd /usr/dict/../; pwd
/usr
```

Mit der Option **-P** weist man »cd« an, immer der physischen Verzeichnisstruktur zu folgen:

```
user@sonne> cd -P / usr/ dict/ ../ ; pwd
/usr/share
```

cp - Dateien und Verzeichnisse kopieren

Aufruf: cp [OPTIONEN]... QUELLE ZIEL

Mit dem Kommando **cp** lassen sich eine Datei in eine andere, mehrere Dateien in ein Verzeichnis oder ganze Verzeichnisse kopieren. »cp« erwartet die Angabe von Quelle und Ziel und entscheidet anhand von Optionen und Ziel, wie zu verfahren ist.

Ist Ziel ein existierendes Verzeichnis, so werden alle Quelldateien in dieses hinein kopiert. Ist Ziel ein unbekannter Name, so wird die Quelle (nur eine einzige!) unter dem Namen kopiert. Die Rechte von Quelle werden dabei übernommen.

Die wichtigsten Optionen sind:

-d

Links bleiben als Link erhalten

-f

Existierende Zieldateien werden überschrieben

-i

Nachfrage vor dem Überschreiben existierender Zieldateien

-l

Alle Dateien werden gelinkt, anstatt physisch kopiert

-p

Dateiattribute bleiben erhalten

-r

Dateien werden rekursiv kopiert

-R

Verzeichnisse werden rekursiv kopiert

-u

Existierende Zieldateien werden nur überschrieben, wenn die Quelldatei aktueller ist

```
# Einfaches Kopieren einer Datei
user@sonne> cp datei.txt file.txt
```

```
# Rekursiv mit Unterverzeichnissen, bestehende Dateien überschreiben (force)
user@sonne> cp -Rf dir dest
# Nur neuere Dateien
user@sonne> cp -u dir dest
# Linken anstatt Kopieren
user@sonne> cp -l dir dest
```

file - Dateityp bestimmen

Aufruf: file [-vbczL] [-f namefile] [-m magicfiles] file ...

Das Kommando versucht zu den als Argument übergebenen Dateinamen deren Typ zu bestimmen. Als erstes untersucht **file** die Informationen, die der Systemruf **stat()** liefert. Hierzu zählen der Test, ob die Datei leer ist und der Test, ob es sich um eine spezielle Datei handelt.
Ein zweiter Testlauf überprüft die Existenz einer **Magic Number**. Programmdateien, Bilddateien, Sounddateien, Datenbanken... beinhalten zumeist eine solche Identifizierung. Das Kommando »file« verwendet in der Voreinstellung die Datei "/var/lib/magic" als Sammlung solcher Magic Numbers. Mit der Option **-m < Magic-Datei>** kann eine andere Vergleichsdatei angegeben werden.
Schließlich versucht »file«, insofern es sich um eine ASCII-Datei handelt, die Sprache des Textes zu analysieren. Allerdings ordnet das Kommando so ziemlich alles dem Englischen zu...

Wichtige Optionen sind **-z**, die »file« veranlasst, den Inhalt einer komprimierten Datei zu testen und **-f < Datei>** , wodurch die zu untersuchenden Dateinamen aus der angegebenen Datei entnommen werden (ein Name pro Zeile).

```
user@sonne> file storage.htm
storage.htm: html document text

user@sonne> file werk.tgz
werk.tgz: gzip compressed data, deflated, last modified: Sun May 28 20:55:43 2000, os: Unix

user@sonne> file -z werk.tgz
werk.tgz: GNU tar archive (gzip compressed data, deflated, last modified: Sun May 28 20:55:43 2000, os: Unix)

user@sonne> file / bin/ ls
/bin/ls: ELF 32-bit LSB executable, Intel 80386, version 1, dynamically linked (uses shared libs), stripped
```

find - Dateien suchen

Aufruf: find [PFAD...] [AUSDRUCK...]

find ist ein äußerst mächtiges Instrument zum Suchen von Dateien. Mit PFAD kann dem Kommando ein beliebiges Startverzeichnis für die Suche angegeben werden. Fehlt die Angabe, startet das Kommando im aktuellen Verzeichnis. Allerdings ist der Verzicht auf die Pfadangabe nur bei der GNU-Version (also unter Linux) des Kommandos gestattet, deswegen empfiehlt es sich, die Angabe immer vorzunehmen.

find durchsucht in der Voreinstellung rekursiv alle Unterverzeichnisse. Deshalb lassen sich die Anzahl der Ebenen (Option **--maxdepth Anzahl**) beschränken. »Anzahl=1« bedeutet dabei die Suche einzig im angegebenen Verzeichnis.

Die gebräuchlichste Methode wird die Suche nach Dateien mit einem bestimmten Namen sein. Die Option **-name Dateiname** ermöglicht dies. **Dateiname** kann dabei die Shell-Metazeichen zur Dateinamenserweiterung enthalten, in dem Falle muss der Name vor der Auswertung durch die Shell geschützt werden, indem er in Anführungsstriche (doppelt/einfach) eingeschlossen wird.

```
user@sonne> find / usr -maxdepth 3 -name "XF* "
/usr/X11R6/bin/XF86_Mach64
/usr/X11R6/bin/XF86Setup
```

```
/usr/X11R6/bin/XF86_VGA16
/usr/X11R6/bin/XFCom_Cyrix
/usr/X11R6/bin/XFCom_Matrox
/usr/X11R6/bin/XFCom_Trident
```

Dateien verfügen über viele Eigenschaften. Sie haben einen Besitzer, Rechte, mehrere Zeitstempel, einen Typ, usw. Für jede dieser Eigenschaften existieren Optionen, mit denen sich gezielt Dateien bestimmter Charakteristik herausfischen lassen. Die Optionen sind u.a:

-amin Minuten

Findet Dateien, auf die vor *Minuten* zugegriffen wurden. Um bspw. alle Dateien zu finden, deren Zugriff vor bis 8 Minuten geschah, können die Angaben kombiniert werden: "-amin +2 -amin -8".

-atime Tage

Findet Dateien, auf die vor *Tage*n zugegriffen wurden (zur Angabe von Bereichen siehe Option "amin")

-empty

Findet leere Dateien (Größe 0 Bytes)

-group Gruppe

Findet Dateien, die zur *Gruppe* (Nummer oder Name) gehören

-iname Name

Findet Dateien mit *Name*, wobei Groß- und Kleinschreibung keine Rolle spielen

-links Anzahl

Findet Dateien, auf die *Anzahl* fester Links zeigen

-mtime Tage

Findet Dateien, die vor *Tage*n geändert wurden (zur Angabe von Bereichen siehe Option »amin«)

-name Name

Findet Dateien mit *Name*

-path Name

Findet Dateien mit *Name*, wobei Name auch Pfadangaben enthalten kann ("*/include/*.h")

-perm Modus

Findet Dateien, deren Rechte exakt dem angegebenem Modus entsprechen (Angabe numerisch oder symbolisch)

-perm -Modus

Findet Dateien, bei denen mindestens die durch Modus angegebenen Rechteflags gesetzt sind

-perm +Modus

Findet Dateien, bei denen mindestens eines der durch Modus angegebenen Rechteflags gesetzt ist

-regex Muster

-size Number[Einheit]

Findet Dateien einer bestimmten Dateigröße. In der Voreinstellung ist die »Einheit« 512 Byte große Blöcke, bei Number=1 werden alle Dateien mit einer Größe zwischen 1 und 512 Bytes gefunden. Als Einheit kann stehen: **b** für 512 Byte-Blöcke (Voreinstellung), **c** für 1 Byte-Blöcke, **k** für Kilobyte-Blöcke und **w** für 2 Byte-Blöcke.

-type Typ

Findet Dateien eines bestimmten Types: **b** Blockdevice, **c** Zeichenweise arbeitendes Device, **d** Verzeichnis, **p** Fifo-Datei, **f** "normale" Datei, **l** symbolischer Link und **s** Socket.

-user Nutzer

Findet Dateien des "Nutzers" (UID oder Nutzername)

Betrachten wir Beispiele zur Anwendung einiger der Suchkriterien:

```
# Um die nachfolgende Suche erfolgreich enden zu lassen, manipulieren wir mit Hilfe von "touch" die Zugriffszeiten einiger Dateien, anschließend suchen wir alle Dateien, auf die in der letzten Minute zugegriffen wurde
user@sonne> touch sax* ; find . -amin 1
.
./saxsys.tgz
./sax_logo.gif
# Alle Dateien im Verzeichnis /tmp, die »user« gehören
user@sonne> find /tmp/ -user user 2>/dev/null
/tmp/kfm-cache-500
/tmp/kfm-cache-500/index.html
/tmp/kfm-cache-500/index.txt
/tmp/acout.1364.NHKnJh
/tmp/acin.1364.c36auh
# Dateien im Verzeichnis /usr mit 43 harten Links darauf
user@sonne> find /usr/ -links 43 2>/dev/null
/usr/include
/usr/share
```

In den Ausgaben wurden die Fehler nach "/dev/null" verschoben, da **find** natürlich nur Verzeichnisse unter die Lupe nimmt, zu dessen Zugang der Benutzer berechtigt ist.

Die Fülle der Ausgaben von **find** lässt rasch den Wunsch nach detaillierteren Suchkriterien aufkeimen. Das Kommando hält logische Operatoren bereit, um verschiedene Kriterien miteinander zu verknüpfen. Die Wichtigsten sind:

(Ausdruck)

Zur vorrangigen Bewertung des geklammerten Ausdrucks

! Ausdruck

Negiert das Suchkriterium

Ausdruck1 -a Ausdruck2

Beide Kriterien müssen zutreffen (das "-a" kann entfallen)

Ausdruck1 -o Ausdruck2

Eines der Kriterien muss stimmen, ist »Ausdruck1« wahr, wird »Ausdruck2« nicht ausgewertet

Ein Beispiel soll die Anwendung demonstrieren:

```
# Suche nach ausführbaren Dateien (keine Verzeichnisse), die »user« gehören
user@sonne> find . -maxdepth 1 \! -type d -a -perm +111 -a -user user
./.xinitrc
./.xsession
./dialog
./selfeditor.pl
```

Erläuterung: Die Suche wurde auf das aktuelle Verzeichnis beschränkt "-maxdepth 1", Verzeichnisse ausgeschlossen "\! -type d" (das "!" ist ein Sonderzeichen der Shell, deswegen muss die Auswertung durch die Shell verhindert werden). Es sollen alle Dateien gefunden werden, bei denen mindestens in einer Rechtegruppe das x-Flag gesetzt ist "-perm +111" und die »user« gehören "-user user". Die Suchkriterien sind jeweils per UND zu verknüpfen.

Da **find** rekursiv alle Unterverzeichnisse (eventuell bis zu einer bestimmten Tiefe) durchsucht, kann die Ausführung sehr langwierig werden. Es besteht aber die Möglichkeit, eine Suche abzubrechen, sobald im Verzeichnis- oder Dateinamen eine bestimmtes Textmuster enthalten ist. Dem jeweiligen Suchkriterium ("-name" oder "-path") ist die Option **-prune** hinten an zu stellen:

```
# Anzeige alle Dateien "-print", deren Name nicht mit einem Punkt beginnt
user@sonne> find . -path './.*' -prune -o -print
.
./nsmail
./iglinux.tgz
./linuxbuch
```

Schließlich wünscht man sich, mit der gefundenen Datei etwas anstellen zu können, d.h. die gefundene Datei durch ein Kommando zu schleusen. Mit der Option **-exec Kommando(s) {} ;** wird in jedem Schritt die gefundene Datei der Reihe nach mit den angegebenen Kommandos bearbeitet. Die geschweiften Klammern dienen dabei als Platzhalter, der den aktuellen Dateinamen enthält. Das Semikolon schließt die exec-Option ab:

```
# Suche nach leeren Dateien und zeige diese mittels "ls -l" an
user@sonne> find ./bsp/ -empty -exec ls -l \{\} \;
insgesamt 0
insgesamt 0
-rw-r--r--   1 user   users        0 Jun 16 09:30  ./bsp/lib/bla/file
-rw-r--r--   1 user   users        0 Jun 16 09:30  ./bsp/lib/foo/file
```

Bemerkung: Die Maskierung der geschweiften Klammern und des Semikolons ist entscheidend, da diese sonst von der Shell substituiert werden (ein Semikolon wird bspw. entfernt). Auch muss zwischen schließender Klammer und Semikolon ein Leerzeichen stehen!

locate - Dateien suchen

Aufruf: locate [-d path] pattern...

Das Kommando dient zum schnellen Auffinden von Dateien. Dabei kann die zu suchende Datei die Shell-üblichen Metazeichen "*", "?" und "[...]" enthalten. »locate« schreibt dann alle, dem Suchmuster entsprechenden Dateien auf die Standardausgabe.

Um effizient zu arbeiten, durchsucht »locate« nicht die Verzeichnisse, sondern schaut in einer Datenbank (Voreinstellung ist **/var/lib/locatedb**) nach. Die Aktualisierung der Datenbank kann mit dem Kommando **updatedb** erfolgen. Mit der Option "-d <pfad>" kann ein anderes, die Datenbank enthaltendes Verzeichnis angegeben werden.

```
user@sonne> locate *atedb
/usr/bin/updatedb
/usr/lib/pgsql/bin/createdb
/var/lib/locatedb
```

ln - Dateien linken

Aufruf: ln [OPTION]... TARGET [LINK_NAME]

Mit dem Kommando **ln** erzeugt man »Verweise« auf Dateien. D.h. man erzeugt einen neuen Namen für eine bestehende Datei oder Verzeichnis und greift über diesen auf die Datei bzw. das Verzeichnis genauso zu, als würde man den originalen Namen verwenden.

Der Vorteil von Links liegt sicherlich im Ersparnis an Speicherplatz. Jede Datei wird nur noch einmal auf der Festplatte abgelegt. Mit Links ist es möglich, dass Dateien in einem lokalen Verzeichnis erscheinen, obwohl sie physisch ganz woanders, sogar auf einem entfernten Rechner liegen können.

Vorsicht ist allerdings beim Modifizieren der Dateien geboten. Im Falle einer Kopie bleibt im Falle eines Fehlers das Original erhalten. Bei einem Link manipuliert man immer das einzige Original!

Es existieren **zwei Arten** von Links, die sich in der internen Realisierung durch das Dateisystem unterscheiden und damit Konsequenzen in der Verwendung nach sich ziehen.

Vereinfacht ausgedrückt werden Verwaltungsinformationen aller Dateien in Unix-Systemen in so genannten Inodes gespeichert und die Daten selbst in den Datenblöcken. Diese Inodes besitzen eine für das jeweilige Dateisystem eindeutige Nummer. Ein Verzeichnis ist in Unix eine Datei, deren Daten die enthaltenen Dateinamen zusammen mit ihrem Speicherplatz sind.

Ein **fester Link** ist ein neuer Verzeichniseintrag in einer Verzeichnisdatei, wobei der neue Dateiname gemeinsam mit der Inode-Nummer der originalen Datei gespeichert wird. Feste Links dürfen nur von Root auf Verzeichnisse angewandt werden. Außerdem sind sie nur für Dateien desselben Dateisystem möglich (Eindeutigkeit der Inode-Nummern).

Ein **symbolische Link** wird im Verzeichniseintrag als Linkname und Inode-Nummer eines **neuen** Inodes abgelegt, welcher den vollständigem Zugriffspfad zur originalen Datei beinhaltet, deshalb kann er »überall« hin zeigen.

Um einen Link zu einer Datei eines anderen Verzeichnisses unter dem selben Namen anzulegen, ist dem nur die zu linkende Datei anzugeben:

```
# Fester Link
user@sonne> ln / etc/ XF86Config
# Symbolischer Link
user@sonne> ln -s / etc/ lilo.conf
user@sonne> ls -l XF86Config lilo.conf
-rw-r--r--    2  root    root       3799 Jun  2 12:20  XF86Config
lrwxrwxrwx    1  user    users        14 Jun 16 15:34  lilo.conf -> /etc/lilo.conf
```

Der Unterschied zwischen festem und symbolischem Link wird in der Ausgabe des Kommandos **ls -l** nochmals deutlich. Im ersten Fall (fester Link) zeigen sich exakt die selben Rechte / der selbe Besitzer... wie die originale Datei. Das ist ja auch nicht verwunderlich, werden die Information doch aus dem selben Inode entnommen. Im zweiten Fall hingegen gehört der Link nun user und besitzt alle Rechte. Dies ist legitim, da der Zugriff auf den Link jedem gestattet sei, der Zugriff auf das Original aber anhand dessen verifiziert wird.

Dem Kommando **ln** können mehrere zu linkende Dateien angegeben werden. Dabei muss die letzte Angabe ein Verzeichnis sein. In diesem werden nachfolgend alle Links erzeugt.

Von den Optionen verdienen sich das Prädikat »Besonders nützlich« **-b**, das im Falle existierender Linkziele diese unter dem Namen »<Linkziel>~« sichert, **-d**, das Root ermächtigt, ein Verzeichnis »hart« zu linken und **-f**, um existierende Ziele ungefragt zu überschreiben (das Gegenstück ist "-i").

ls - Verzeichnisinhalte auflisten

Aufruf: ls [OPTION]... [DATEI]...

Ohne Argumente aufgerufen, gibt **ls** eine Liste der »sichtbaren« Dateien und Verzeichnisse des aktuellen Verzeichnisses aus. Sichtbar bedeutet dabei, dass der Dateiname **nicht mit einem Punkt** beginnt (das ist der einzige Weg, um Dateien zu »verstecken«).

Mit der Angabe von »DATEI« kann die Ausgabe auf bestimmte Dateien beschränkt oder für andere Verzeichnisse vorgenommen werden. »DATEI« kann dabei die shellüblichen Metazeichen enthalten.

```
# Dateien im Verzeichnis /usr/doc/howto
user@sonne> ls /usr/doc/howto/
de  en
# Dateien in /etc, die ein X im Namen enthalten
user@sonne> ls /etc/*X*
/etc/XF86Config  /etc/XF86Config.vm

/etc/X11:
wmconfig
```

Das letzte Beispiel zeigt etwas mehr an, als wir vielleicht erwartet hatten... und zwar den Inhalt des Verzeichnisses "/etc/X11". Der Grund ist, dass die Shell die Expansion des Metazeichens ***** vornimmt und dem Kommando **ls** u.a. das Verzeichnis "/etc/X11" als Argument mitgibt. Findet **ls** nun ein Verzeichnis als Argument vor, wird es dessen Inhalt auflisten. Allerdings lässt dich dieses Verhalten mit der Option **-d** unterdrücken.

```
user@sonne> ls -d /etc/*X*
/etc/X11  /etc/XF86Config  /etc/XF86Config.vm
```

Die Optionen beeinflussen in erster Linie die Fülle an Informationen, die das Kommando **ls** zu den Dateien anzeigt. Eine Auswahl an Optionen sei tabellarisch angeführt:

-a

Anzeige aller Dateien (einschließlich der "versteckten")

-d

Zeigt bei Unterverzeichnissen nur deren Namen, nicht deren Inhalt an (Siehe letztes Beispiel)

-i

Anzeige des Speicherplatzes (Inodenummer) einer Datei

-l

Anzeige von Namen, Typ, Rechten, Anzahl der Hardlinks, Besitzer, Gruppe, Größe, Zeitmarke

-t

Anzeige nach Zeitmarke sortiert

-F

Kennzeichnet Dateitypen durch Anhängen eines Symbols. Die Markierung kann bei monochromen Monitoren bzw. X-Servern sinnvoll sein.

***** Ausführbare Datei

/ Verzeichnis

@ Symbolischer Link

| Named pipe (FiFo)

= Socket

nichts "Normale" Datei

-I MUSTER

(Großes i) Dateien, deren Namen mit dem Muster übereinstimmen, werden von der Anzeige ausgeschlosser

-S

Nach Dateigröße sortierte Ausgabe

--color

Farbliche Kennzeichnung des Dateityps. Welcher Typ mit welcher Farbe dargestellt wird, steht in der Datei "/etc/DIR_COLORS".

Zum Abschluss noch einige Anwendungsbeispiele:

```
user@sonne> ls -l / boot/ v*
-rw-r--r--    1  root    root      716916  Dec 14 1999  /boot/vmlinuz
-rw-r--r--    1  root    root      667293  Nov 19 1999  /boot/vmlinuz.old

user@sonne> ls -i / boot/ ?y*
    16 /boot/System.map

user@sonne> ls --color
desktop.gif
filesystem.gif

user@sonne> ls -l 'gif'

user@sonne> ls -F / dev/ | head -5

3dfx
:0@
:1@
:2@
:3@
```

lpc - Druckerstatus abfragen

Aufruf: lpc [command [argument ...]]

Das *Line Printer Control* Programm dient dem Administrator zum Manipulieren der Druckerwarteschlange. Dem »normalen« Benutzer steht einzig die Möglichkeit zur Abfrage derer zur Verfügung.

Wird **lpc** ohne Argumente aufgerufen, geht es in den interaktiven Modus und wartet auf weitere Kommandoeingaben. Dieser Modus kann durch Eingabe von **quit** oder **exit** beendet werden. Alle Kommandos, die **lpc** versteht, lassen sich ebenso als Argument auf der Kommandozeile angeben. Das Programm wird den Befehl bearbeiten und anschließend terminieren.

```
user@sonne> lpc
lpc> status
ascii:
        queuing is enabled
        printing is enabled
        no entries
        no daemon present
```

```
lp:
        queuing is enabled
        printing is enabled
        2 entries in spool area
        no daemon present
raw:
        queuing is enabled
        printing is enabled
        no entries
        no daemon present
lpc> help
Commands may be abbreviated.  Commands are:

abort   enable  disable help    restart status  topq    ?
clean   exit    down    quit    start   stop    up
lpc> abort lp
?Privileged command
lpc> exit
user@sonne>
```

Neben dem Beenden und Aufruf der Hilfe steht dem Benutzer nur **status** zur Anzeige der Warteschlangeneigenschaften zur Verfügung.

lpr - Dateien ausdrucken

Aufruf: lpr [-Pprinter] [-#num] [-C class] [-J job] [-T title] [-U user] [-i [numcols]] [-1234 font] [-wnum] [-cdfghlnmprstv] [name ...]

Zum Drucken von Dokumenten von der Konsole aus, dient das Kommando **lpr** (auch grafische Anwendungen nutzen intern dieses Kommando). **lpr** fügt einer Druckerwarteschlange einen neuen Auftrag hinzu und informiert anschließend den Druckerdaemon. Der Druckjob wird in die mit **-P Druckername** angegebene Warteschlange oder, wenn die Option nicht angegeben wurde, in die Standard-Druckerschlange, die mit Hilfe der Umgebungsvariablen »$PRINTER« gesetzt wird, eingereiht.

```
# Default-Drucker
user@sonne> lpr beispiel.txt
# Drucker mit dem Namen »remotelp«
user@sonne> lpr -Premotelp beispiel.txt
```

Das obige Vorgehen funktioniert meist erst nach dem Einrichten des Druckers, das aber nicht Bestandteil dieses Abschnitts sein soll. »Meist« steht hier bewusst, da im Falle, dass die zu druckende Datei Daten in einem für den Drucker verständlichen Format beinhaltet (und der Druckerdaemon aktiv ist), dieses durchaus gelingen kann. Im Normalfall müssen die Daten aber erst aufbereitet werden, was bei geeigneter Druckerkonfiguration für den Benutzer transparent geschieht.

lpr bietet aber eine Reihe von Optionen, die dem Kommando Informationen über das konkrete Datenformat mitteilen, zum Glück muss sich der Benutzer, vorausgesetzt der Administrator hat seine Hausaufgaben erledigt, um solche kryptischen Angaben nicht kümmern.

Dennoch existieren ein paar Optionen, die nützliche Effekte hervorzaubern:

-h

Oft werden intern Dokumente so aufbereitet, dass sie noch eine Statusseite enthalten, die u.a. die Anzahl d Seiten des Jobs, den Auftraggeber, die Druckkosten usw. beschreibt. Der Druck dieser Seite kann mit der Option "-h" unterbunden werden.

-m

Wenn der Auftrag ausgedruckt wurde, wird der Auftraggeber per Mail benachrichtigt... das spart manchen zum Druckerraum...

-s

Die zu druckende Datei wird nicht ins Spoolverzeichnis kopiert sondern symbolisch gelinkt. Das ist hilfreich, wenn die Festplatte mit dem Spoolverzeichnis wenig freien Speicherplatz bietet.

-P

Der Ausdruck erfolgt auf dem angegebenen Drucker ("-Panderer_Drucker") anstatt auf dem default-Drucke

Weitere Optionen ermöglichen den mehrfachen Ausdruck eines Dokuments **-# Anzahl**, die Änderung der Zeilenbreite **-wBreite**, das Einrücken der Ausgabe **-i Spalte** und Manipulationen der Statusseite, z.B. erzwingt **-j Titel einen anderen Titel (sonst erster Dateiname).**

lprm - Druckauftrag löschen

Aufruf: lprm [-Pprinter] [-] [job # ...] [user ...]

lprm ist die einzige Möglichkeit für den normalen Benutzer, einen Druckjob aus der Warteschlange zu entfernen. Mit dem Aufruf lprm - löscht ein Nutzer alle eigenen Aufträge aus dem Spool-Verzeichnis des default-Druckers. Mit -P Druckername kann er einen anderen als den voreingestellten Drucker auswählen. Die Nummer eines Druckjobs verrät u.a. ein Aufruf des Kommandos lpq. Mit dieser Nummer kann ein einzelner Job (oder mehrere) gezielt entfernt werden:

```
user@sonne> lpq
lp is ready and printing
Rank   Owner      Job  Files                 Total Size
active user        14   beispiel.txt          146 bytes
1st    user        15   beispiel.txt          146 bytes
user@sonne> lprm 14
dfA014sonne dequeued
cfA014sonne dequeued
user@sonne> lpq
lp is ready and printing
Rank   Owner      Job  Files                 Total Size
1st    user        15   beispiel.txt          146 bytes
```

lpq - Aufträge in der Druckerwarteschlange auflisten

Aufruf: lpq [-l] [-Pprinter] [job # ...] [user ...]

Line printer queue **ist das Kommando, um die noch ausstehenden Druckaufträge in einer Druckerwarteschlange anzusehen. Ohne Argumente aufgerufen, werden alle Aufträge des »default«-Druckers angezeigt:**

```
user@sonne> lpq
no entries
user@sonne> lpr beispiel.txt; lpq
lp is ready and printing
Rank   Owner      Job  Files                 Total Size
1st    user        15   beispiel.txt          146 bytes
```

Befinden sich weitere Drucker im System (z.B. Netzwerkdrucker) kann mit der Option -P Druckername die Warteschlange des benannten Druckers betrachtet werden. Mit Angabe eines Nutzerkennzeichens werden nur die Aufträge dieses Benutzers angezeigt und die Option -l führt zur Ausgabe »aller« Informationen zum Druckjob, während sonst nur so viele Informationen angezeigt werden, wie auf eine Zeile des Terminals passen.

```
user@sonne> lpq
```

```
no entires
user@sonne> lpr -T "Ein anderer Titel" beispiel.txt; lpq -Plp -l
lp is ready and printing

user: active                [job 014sonne]
      beispiel.txt          146 bytes
user: 1st                   [job 015sonne]
      beispiel.txt          146 bytes
```

mkdir - Verzeichnis erstellen

Aufruf: mkdir [OPTION] DIRECTORY...

Mit mkdir lassen sich neue Verzeichnisse erzeugen. Dabei gelten die durch umask vorgegebenen Rechte, außer, dem Kommando wird mittels der Option -m eine andere Rechtemaske mitgegeben.

Das Kommando weigert sich, wenn ggf. die Elternverzeichnisse nicht existieren. In solchen Fällen kann mit der Option -p das automatische Erzeugen fehlender übergeordneter Verzeichnisse erzwungen werden.

```
# Einfaches Erzeugen eines neuen Verzeichnisses:
user@sonne> mkdir ~/testdir
# Mit anderen Rechten als den durch umask vorgegebenen:
user@sonne> mkdir -m 777 ~/testdir
# Falls übergeordnetes Verzeichnis nicht existiert, soll es angelegt werden:
user@sonne> mkdir -p ~/parent/child/testdir
```

mv - Dateien umbenennen oder verschieben

Aufruf: mv [OPTION]... QUELLE ZIEL

Mit dem Kommando mv lassen sich eine einzelne Dateien und ein einzelnes Verzeichnis umbenennen oder Datei(en) bzw. Verzeichnis(se) in ein Zielverzeichnis verschieben.

mv erkennt aus dem Kontext, welche Aktion vorzunehmen ist. Die Optionen sind:

-b

Existiert eine Zieldatei bereits, wird diese gesichert (Name: "<Zielname>~")

-i

Nachfrage vor Überschreiben einer existierenden Zieldatei

-f

Überschreiben einer existierenden Zieldatei ohne Nachfrage

-u

Überschreiben älterer existierender Zieldateien

Zum Abschluss einige Beispiele:

```
# Einfaches Umbenennen (Ziel ist eine Datei)
user@sonne> mv foo bla
# Einfaches Verschieben (Ziel ist ein existierendes Verzeichnis)
user@sonne> mv bla /tmp
```

```
# Umbenennen und Verschieben (Ziel ist eine Datei in einem existierenden Verzeichnis)
user@sonne> mv /tmp/bla ~/foo
```

pwd - Name des aktuellen Arbeitsverzeichnisses ausgeben

Aufruf: pwd [OPTION]

Das Kommando gibt den vollständigen Namen des aktuellen Verzeichnisses aus. Einzige Optionen sind --help und --version.

```
user@sonne> cd /usr/X11R6/bin
user@sonne> pwd
/usr/X11R6/bin
```

rm - Datei löschen

Aufruf: rm [OPTION]... DATEI...

Das Löschen von Dateien unter Unix reduziert sich mit dem Kommando rm auf des Entfernen des Verzeichniseintrages und Freigabe der von den Dateien allokierten Datenblöcke. Da diese Blöcke nun nicht mehr einzelnen Dateien zugeordnet werden können, ist die Konsequenz, dass einmal entfernte Daten nicht wieder hergestellt werden können (prinzipiell ist eine Restauration möglich, aber das ist Thema des Profiteils).

Der Benutzer sollte also beim Löschen von Daten Vorsicht zum obersten Prinzip erheben.

rm entfernt alle als Argument angegebenen Daten. Besitzt man für eine Datei keine Schreibberechtigung, fordert rm eine nochmalige Bestätigung des Löschens. Ist man der Eigentümer der Datei, oder befindet sich die Datei in seinem Heimatverzeichnis, so kann eine »schreibgeschützte« Datei entfernt werden (Ausnahme: Dateiattribute).

```
user@sonne> ls -l foo
-r--r--r--  1 user  users  0 Jun 17 10:05 foo
user@sonne> rm foo
rm: schreibgeschützte Datei »foo« entfernen? y
```

Mit der Option -f (*force*) kann die obige Nachfrage unterdrückt werden, mit -i (*interactive*) muss jede zu löschende Datei bestätigt werden.

Im Unterschied zum später beschriebenen Kommando rmdir vermag rm in Verbindung mit der Option -r (*recursive*) auch nicht leere Verzeichnisse zu entfernen.

Beginnt ein Dateiname mit einem Minus, würde rm dieses als Option betrachten und vermutlich mit einer Fehlermeldung den Dienst quittieren. Dem Kommando muss in solchen Fällen explizit das Ende der Optionen bekannt gegeben werden, indem zwei Minus die Liste der Dateinamen einleiten:

```
user@sonne> rm -minus_name
rm: Ungültige Option -- »m«
Versuchen Sie »rm --help« für weitere Informationen.
user@sonne> rm -- -minus_name
```

Das im letzten Absatz beschriebene Problem mit den Dateinamen kann bei fast allen Kommandos wie beschrieben gelöst werden.

rmdir - Verzeichnis löschen

Aufruf: rmdir [OPTION]... DIRECTORY...

Das Kommando löscht leere Verzeichnisse. Zum Entfernen nichtleerer Verzeichnisse kann das Kommando rm genutzt werden.

Mit der Option -p können übergeordnete Verzeichnisse entfernt werden. Auch hier gilt, dass diese leer sein müssen.

```
user@sonne> rmdir Nicht_leeres_Verzeichnis
rmdir: Nicht_leeres_Verzeichnis: Das Verzeichnis ist nicht leer

user@sonne> rmdir dir/ subdir/ subsubdir
user@sonne> rmdir -p dir/ subdir
```

whereis - Pfade zu Programmen und Manuals finden

Aufruf: whereis [-bmsu] [-BMS directory... -f] filename ...

whereis gibt zu einem gegebenem Programmnamen den Zugriffspfad, das zugehörige Manual und - falls installiert - das Verzeichnis der Quelldateien an. Das Kommando durchsucht hierzu die typischen Verzeichnisse, wo z.B. ein Kommando gespeichert sein könnte.

```
user@sonne> whereis whereis
whereis: /usr/bin/whereis /usr/man/man1/whereis.1.gz
```

Mit Optionen kann die Ausgabe eingeschränkt werden. So gibt "-b" nur den Pfad zum Kommando an, "-m" zeigt die Hilfedatei und "-s" die Programmsourcen.

```
user@sonne> whereis -s whereis
whereis:
```

Der Suchraum des Kommandos kann eingeschränkt werden. Die Option "-B < Suchpfade> " weist »whereis« an, nur in den angegebenen Verzeichnissen nach Binaries zu suchen. "-M" betrifft die Manual-Suchpfade und "-S" die der Sourcen. Werden diese Optionen verwendet muss hinter der letzten Pfadangabe die Option "-f" stehen, um »whereis« das Ende mitzuteilen.

which - Vollständigen Pfad zu einem Programm ausgeben

Aufruf: which progname ...

Das Kommando zeigt den vollständigen Zugriffspfad zu einem Kommando an, indem es das Kommando anhand der »PATH«-Variable sucht. Das Kommando ist vor allem hilfreich, wenn man sich nicht sicher ist, ob ein Kommando das richtige ist.

```
user@sonne> which whereis
/usr/bin/whereis

user@sonne> which which
user@sonne> echo $?
0
user@sonne> which gibts_nicht
user@sonne> echo $?
1
```

Gibt das Kommando nichts aus, wurde entweder das Kommando nicht gefunden oder es handelt sich um ein Shell-builtin-Kommando (Vergleiche auch Arten von Kommandos). Im ersten Fall ist der

Rückgabewert »-1«, sonst »0«.

Nutzerkommandos - Hilfe

apropos - Manuals inklusive deren Kurzbeschreibung anzeigen

Aufruf: apropos [-dhV] [-e|-w|-r] [-m SYSTEM[,...]] [-M PFAD] MUSTER

Die Manual Pages enthalten stets eine Zeile mit einer Kurzbeschreibung des Inhalts. **apropos** durchsucht diese Beschreibungen und den Namen des Manuals nach dem »Muster« und gibt übereinstimmende Zeilen aus.

Wichtige Optionen steuern die Bewertung des Musters. Das Muster muss exakt übereinstimmen, wenn die Option **--exact** verwendet wird (die Option "-e" funktioniert bei manchen Implementierungen nicht). Es kann mit der Option **-r** Reguläre Ausdrücke enthalten (dies ist die Voreinstellung). Und mit der Option **-w** werden auch enthaltene Wildcards interpretiert.

In welchen Pfaden das Kommando nach den Manuals sucht, wird über die Umgebungsvariable "$MANPATH" oder mittels der Angabe der Suchpfade auf der Kommandozeile ("-M Pfad(e)") bestimmt.

In heterogenen Netzwerkumgebungen sind häufig Manual-Beschreibungen zu Befehlen verschiedener Unix-Systeme installiert. Mit "-m System[, System]" kann **apropos** angewiesen werden, die Manuals zu diesem Unix-System zu durchsuchen. Wird die Option nicht verwendet, wird der Inhalt der Variable »$SYSTEM« benutzt.

```
user@sonne> apropos --exact print*
print*: nichts passendes.

user@sonne> apropos --exact printf
printf (1)          - format and print data
printf (3)          - formatted output conversion
printf (1)          - format and print data
printf (3)          - formatted output conversion

user@sonne> apropos --exact -w [fs]printf
fprintf (3)          - formatted output conversion
sprintf (3)          - formatted output conversion
fprintf (3)          - formatted output conversion
sprintf (3)          - formatted output conversion
```

info - Betrachten der Info-Seiten

Aufruf: info [OPTION]... [INFO-FILE [MENU-ITEM...]]

Die Hilfe im Info-Format soll(te) einmal die Manual Pages ablösen. Ihr Vorteil sind die enthaltenen Verweise zu andern Info-Seiten, denen man in der Manier eines Hyperlinks folgen kann. Eine Info-Seite bezeichnet man auch als Knoten (Node). Über einen Verweis gelangt man dann zu einem anderen Knoten...

Von der verschiedenen Optionen seien nur **--directory < PFAD>** und **--output DATEI** erwähnt. Erstere veranlasst **info**, neben den Pfaden der Variablen »$INFOPATH« auch die angegebenen Pfade nach Einträgen zu durchsuchen. Die zweite Option schreibt die Ausgabe des Knotens in die »DATEI«. In dem Fall besteht keine Möglichkeit zur Interaktion mit dem Programm.

Innerhalb des Info-Programms navigiert man mit verschiedenen Tasten. Die Wichtigsten seien tabellarisch zusammengestellt:

h

?

Listet alle Kommandos von `info` auf (Ende mit "l")

[Leertaste]

Im Text eine Seite nach unten scrollen

[Backspace]

Im Text eine Seite nach oben scrollen

n

Springt zum nächsten Thema

p

Kehrt zum letzten Thema zurück

m

Thema auswählen:

1. Cursor auf einen Menüeintrag des aktuellen Themas setzen und "m" eingeben
2. "m" eingeben gefolgt vom gewünschten Thema ([Ctrl]-[g] verwirft die Eingabe)

q

`info` beenden

Zu empfehlen ist der Aufruf der Info-internen Hilfe, um die Bedienung kennen zu lernen. Eine typische Darstellung einer Info-Seite veranschaulicht folgende Abbildung:

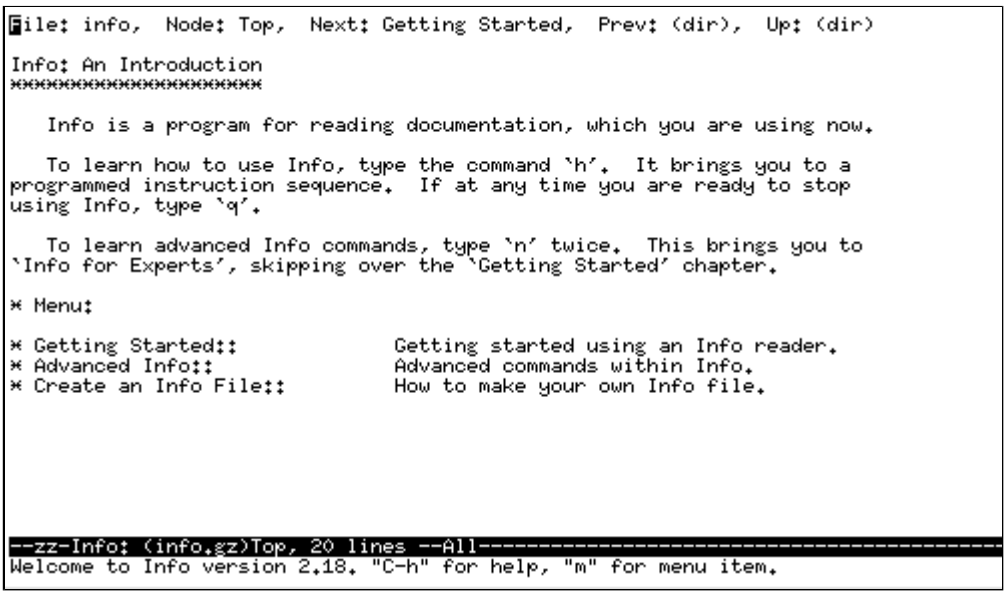


Abbildung 1: Anzeige von »info«

man - Betrachten der Manual Pages

Aufruf: man [-c|-w|-tZT device] [-adhu7V] [-m system[,...]] [-L locale] [-p string] [-M path] [-P pager] [-r prompt] [-S list] [-e extension] [[section] page ...] ...

man ist eine Interface zur Unix-Online-Hilfe der Manual Pages und wird zur Betrachtung dieser eingesetzt.

Mit der Option **-k** arbeitet "man" wie apropos, mit der Option **-f** wie das Kommando whatis.

Um die Arbeitsweise des Kommandos zu verstehen, sollte man die Eingliederung der Manuals in Sektionen (nach ihrem Verwendungszweck) und ihren internen Aufbau kennen. Eine Diskussion hierzu findet man im Abschnitt Hilfe in der Einführung. An dieser Stelle möchten wir uns auf die Beschreibung wichtiger Optionen beschränken:

-a

Das Kommando zeigt nun nacheinander alle Manuals an, die den gesuchten Begriff behandeln. Die aktuelle Seite beendet man durch Eingabe von **q** und wird nachfolgend aufgefordert, die nachfolgende Anzeige zu beginnen, zu überspringen oder das Programm zu beenden.

-M Pfad

Die Suchpfad für Manuals werden erweitert. Normal such man nur in den in der Variable »$MANPATH« enthaltenen Pfaden.

-P Pager

Hier kann ein anderer Pager angegeben werden, Voreinstellung ist »$PAGER«.

-S Liste

»Liste« sind die durch Kommas getrennten Nummern der Sektionen, in denen das Kommando suchen soll.

n

»n« ist die Nummer der Sektion, in der das Kommando suchen soll.

-w

Es wird nur der vollständige Pfad zum Manual und nicht dessen Inhalt angezeigt.

-u

Die Index-Datenbank (meist /var/cache/man/index.bt) wird aktualisiert.

```
user@sonne> man -w printf
/usr/man/man1/printf.1.gz /var/cache/man/fsstnd/cat1/printf.1.gz

user@sonne> man -P "od -x" whoami |  head -4
0000000 0a0a 570a 4f48 4d41 2849 2931 2020 2020
0000020 2020 2020 2020 2020 2020 2020 2020 2020
0000040 2020 5346 2046 2020 2020 2020 2020 2020
0000060 2020 2020 2020 2020 2020 5720 4f48 4d41

user@sonne> man printf |  sed -n '4p'
PRINTF(1)                 FSF                 PRINTF(1)

user@sonne> man 3 printf |  sed -n '4p'
PRINTF(3)          Linux Programmer's Manual          PRINTF(3)
```

whatis - Manual Kurzbeschreibungen anzeigen

Aufruf: whatis [-dhV] [-r|-w] [-m SYSTEM[,...]] [-MPFAD] NAME ...

Im Unterschied zu apropos sucht **whatis** nur in den Manual Page Namen, ob diese das Muster »NAME« enthalten, schreibt aber ebenso die Kurzbeschreibung zu den gefundenen Manuals auf die Standardausgabe.

Das Kommando verwendet für die Suche eine Index-Datenbank "/var/cache/man/index.bt".

Mit den Optionen **-w** werden enthaltenen Metazeichen und mit der Option **-r** Reguläre Ausdrücke im Muster ausgewertet.

```
user@sonne> whatis printf
printf (1)          - format and print data
printf (3)          - formatted output conversion
user@sonne> whatis [vf]printf
[vf]printf: nichts passendes.

user@sonne> whatis -w [vf]printf
fprintf (3)          - formatted output conversion
vprintf (3)          - formatted output conversion
```

Nutzerkommandos - Kommunikation

irc - Online diskutieren

Aufruf: irc [OPTIONEN] [nickname [serverlist]]

Der *Internet Relay Chat* ist eine verbreitete Möglichkeit, um seine Gedanken online einer Gruppe von Diskussionspartnern mitzuteilen. Dazu existieren die IRC-Server, die nahezu in Echtzeit die Nachrichten in einem »Channel« an andere Server weiterleiten und sie letztlich auch an die Clients durchreichen. Das Programm **irc** ist ein sehr einfacher Client, der den Zugang zu den »Klatschstuben« ermöglicht. Alle Kommandos des **irc** zu beherrschen, erfordert eine langwierige Studienarbeit. Aber das ist zum Glück nicht notwendig, die Kenntnis einiger Grundlagen genügt vollauf.

Bevor wir uns ins Gewühl stürzen, seien einige Optionen der Kommandozeile erwähnt. Alles was mit einem Minus »-« eingeleitet wird, sind »normale« Optionen. Einige erfordern ein weiteres Argument, andere nicht. Das erste nicht mit einem Minus eingeleitete Argument ist der »Nickname«, also der Name, unter dem wir am Schwarzen Brett erscheinen. Alle weiteren Angaben betreffen IRC-Server, mit denen wir die Kontaktaufnahme wünschen. Die wichtigsten Optionen sind **-c Kanal**, die uns gleich nach dem Kommandostart mit dem angegebenen Kanal verbindet und **-p Portnummer**, falls der Server nicht am üblichen Port 6667 lauert. Anstatt der Angabe von Kommandozeilenargumenten lassen sich alle Parameter auch zur Laufzeit setzen. Genau dieses Vorgehen wollen wir im weiteren Verlauf betrachten:

```
user@sonne> irc
*** Connecting to port 6667 of server irc.undernet.org
*** Looking up your hostname
*** Checking Ident
*** Couldn't look up your hostname
*** No ident response
*** user Nickname is already in use. (from NewYork.NY.US.Undernet.org)
*** You have specified an illegal nickname
*** Please enter your nickname



Nickname: tux4ever
```

Da dem Programm kein Server mitgeteilt wurde, hat es sich mit einem der Standard-IRC-Server verbunden. Dieser versucht nun etwas über unseren Rechner zu erfahren, scheitert aber, da dieser in einem privaten Netz liegt und somit keinen gültigen DNS-Eintrag besitzt. Ebenso versucht **irc**, unser Benutzerkennzeichen als Nickname durchzubringen, doch protestiert der Server, da der Name schon in Verwendung ist. So müssen wir einen neuen Spitznamen wählen. Dieser lässt sich jederzeit mit **/ nick NeuerName** ändern und ein anderer IRC-Server kann über **/ server ServerName** angesprochen werden.

Jetzt befinden wir uns sozusagen im Gebäude mit den vielen Konferenzsälen und sollten eines der Diskussionsforen besuchen. Wir »raten« mal einen Kanalnamen und versuchen es mit »linux.de«:

```
/ join # linux.de
*** tux4ever (~user@192.168.100.99) has joined channel # linux.de
*** # linux.de 962032614 (from Seattle.WA.US.Undernet.org)
```

Anmerkung: Mit der oben angegebenen IP-Adresse wird die Kontaktaufname im Internet nie gelingen, hier benötigt man einen Proxy.

Existiert kein Kanal unter diesem Namen, so wird er angelegt. Allerdings nützt das einem herzlich wenig, solange man allein im Kanal ist. Egal was man schreibt... es ist niemand da, der einem zuhört. Informationen über die Teilnehmer gewinnt man mit **/ names Kanal** (nur die Nicknamen), **/ who Kanal** (Nicknamen und Rechner, an denen die Leute sitzen) und **/ whois Nickname** (alle Informationen, die zu einem angemeldeten Benutzer erhältlich sind) bzw. **/ whowas Nickname** im Falle eines nicht mehr angemeldeten Teilnehmers.

Alles, was man innerhalb des **irc** eintippt, und das nicht gerade mit einem Slash beginnt, ist für alle anderen im Kanal angemeldeten Benutzer sichtbar. Manchmal möchte man aber gezielt einem Partner eine Nachricht zukommen lassen. Diese geschieht mit **/ msg Nickname Nachricht**.

Wer letztlich öfter »chattet«, der findet weitere Informationen zu Kommandos in der Hilfe, die über **/ help** erreicht wird. Mit dem Fragezeichen erreicht man innerhalb der Hilfe die Anzeige alle verfügbaren Kommandos.

Verlassen wird **irc** entweder mit **/ quit** oder mit **/ exit**, wobei den beiden Befehlen nachstehender Text als Grund für das Ausscheiden den anderen Kanalinsassen mitgeteilt wird.

Bei vielen Diskussionen fallen immer mal wieder Leute aus dem Rahmen und entpuppen sich als lästige Störenfriede. Solche Personen können vom Operator des Kanals »heraus gekickt« werden, indem dieser ihre Sitzung mit **/ kick Kanal Nickname [Nachricht an die Nervensäge]** beendet. Noch drastischer wirkt **/ mode Kanal + b Nickname!Loginname@Rechner**, womit der genannte Teilnehmer keinen Zugang mehr zum Kanal erhält.

Und wer ist der Operator eines Kanals? Immer derjenige, der den Kanal erzeugte oder eine Person, der vom aktuellen Operator mittels **/ op Nickname** zum Neuen ernannt wird.

mail - Ein einfacher Mailclient

Aufruf: mail [-iInv] [-s subject] [-c cc-addr] [-b bcc-addr] to-addr...

Mail ist ein sehr einfacher Kommandozeilenclient zum Versenden und Lesen von Nachrichten. Sein Vorteil liegt ganz klar in der Eignung zum Einsatz in Shellskripten, wo eine automatische Mailgenerierung erwünscht wird. Aber auch auf der Kommandozeile ist es eine effiziente Möglichkeit, um mal »eben schnell« seine Gedanken versenden zu können oder die neue Nachricht ohne das kryptische **pine** zu inspizieren.

Ist neue Mail eingetroffen, ermöglicht der Aufruf von **mail** das Lesen dieser:

```
user@sonne> mail
"/var/spool/mail/user": 2 messages 1 new 2 unread
 U  1 tux@galaxis.de      Tue Jun 20 08:18  14/446   "Neuigkeiten"
>N  2 tux@galaxis.de      Tue Jun 20 08:19  11/411   "Probemail"
& p
Message 2:
From tux@galaxis.de Tue Jun 20 08:19:08 2000
Date: Tue, 20 Jun 2000 08:19:08 +0200
From: Tux dem Pinguin <tux@galaxis.de>
To: user@galaxis.de
Subject: Probemail

Alles bleibt beim Alten.

& q
Saved 2 messages in mbox
user@sonne>
```

Um die erste **neue** Nachricht zu lesen, ist **p** einzugeben, mit **+** gelangt man zur nächsten und mit **-** zur vorhergehenden (neuen oder ungelesenen) Nachricht. Durch Eingabe einer Ziffer kann eine Nachricht gezielt gewählt werden.
Um eine **Nachricht zu löschen**, ist **d Nummer** einzugeben, wobei Nummer eine einzelne Ziffer, mehrere, durch Leerzeichen getrennte Ziffern oder ein Bereich (2-7) sein kann. Solange **mail** nicht beendet wurde, kann eine gelöschte Mail mit **u Nummer** gerettet werden.

Zum **Antworten auf eine Mail** ist **r** einzugeben.
Beendet wird mail durch Eingabe von **q**.

Nach dem Ende von **mail** werden alle gelesenen Mails von der Mailbox in die Datei "~/mbox" verschoben, mit der Kommandozeilenoption **-f** liest **mail** deren Inhalt ein.

Zum **Senden einer Nachricht** ist das Kommando mit der email-Adresse des Empfängers aufzurufen. Gleichzeitig kann der Grund der Mail ("Subject") mit **-s Subject** und weitere Empfänger **-c Adresse[, Adresse] angegeben werden. Bei Verzicht auf die Option -s wird das Programm zur Eingabe des Subjects auffordern:**

```
user@sonne> mail tux@galaxis.de
Subject: Antwort
Alle was nun eingegeben wird, bildet den Inhalt der Mail, bis zum Ende
mit [Ctrl]+[D] oder der Eingabe eines einzelnen Punktes auf einer neuen
Zeile.
.
EOT
user@sonne>
```

mail ist nicht in der Lage, Attachements zu behandeln. Für solche Fälle ist u.a. metamail geeignet.

talk - Mit anderen Benutzer Nachrichten austauschen

Aufruf: talk person [ttyname]

Das Kommando talk ermöglicht die Kommunikation mit einem anderen Benutzer über Rechnergrenzen hinweg. talk kopiert die Eingaben auf dem lokalen Terminal in den unteren Bereich des Terminals des Partners. Umgekehrt werden die Eingaben des Gegenübers im oberen Bereich unseres Terminals sichtbar.

Eine talk-Sitzung initiiert man mit dem Aufruf

```
user@sonne> talk <Benutzer>[@remote_host]
```

Wird die Angabe des entfernten Rechners weg gelassen, so wird der Aufbau einer Verbindung zum lokalen »Benutzer« versucht. So etwas ist durchaus sinnvoll, da ja bekanntlich mehrere Benutzer gleichzeitig mit einem System arbeiten können.

Wünschen wir einen kleinen Schwatz mit den Benutzer tux, der am Rechner linux.arktis.eis sitzt, so geben wir Folgendes ein:

```
user@sonne> talk tux@linux.arktis.eis
[Waiting for your party to respond]
[Ringing your party again]
```

Mit »[Waiting for your party to respond]« wird angedeutet, dass der Talk-Dämon den Aufbau der Verbindung versucht. Reagiert der Partner nicht schnell genug, wird nach jeweils ca. 30 Sekunden der Aufruf wiederholt »[Ringing your party again]«.
Tux am Rechner linux.arktis.eis bemerkt den Wunsch nach einer Verbindung anhand einer Meldung auf seinem Terminal und gleichzeitigem Erklingen des System-Klingeltons:

```
user@sonne>

Message from Talk_Daemon@sonne at 7:20 ...
talk: connection requested by user@sonne.galaxis.de.
talk: respond with: talk user@sonne.galaxis.de
```

Tux reagiert mit dem Aufruf:

```
tux@linux> talk user@sonne.galaxis.de
```

Und nun kann die Unterhaltung stattfinden, solange, bis eine Partei den Talk mit [Ctrl]-[C] beendet. Die Eingaben von »user« finden in der oberen Hälfte dessen Terminals statt und werden gleichzeitig in der unteren Hälfte des Terminals auf Tux' Rechner dargestellt. Abbildung 1 zeigt die Ausgaben auf den Terminals von »user« (oben) und »Tux« (unten).

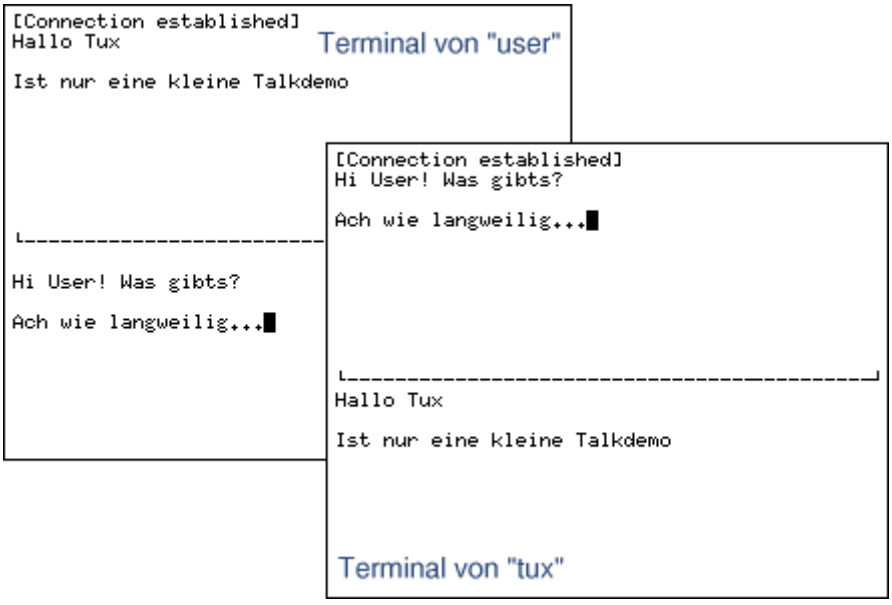


Abbildung 1: Aussehen der 'talk'-Fenster von 'user' und 'tux'

tin - Ein einfacher Newsreader

Aufruf: tin [OPTIONEN] [newsgroup[,...]]]

tin und rtin sind auf der Kommandozeile arbeitende Newsclients. Während tin zum Lesen lokal gespeicherter News (/ var/ spool/ news) verwendet wird, lassen sich mit rtin bzw. tin -r die Archive eines Newsservers durchforsten. Beide Programme weisen starke Ähnlichkeiten in Gestaltung und Bedienung mit dem Mailclient elm auf; wer also dieses Programm beherrscht, sollte sich mit den Newsreaders leicht anfreunden können.

Beim bloßen Aufruf versucht sich tin mit dem nächstgelegenen Newsserver zu verbinden, der per default unter news.< meine domaine> zu finden sein sollte. Wenn auf diesem Wege kein Server erreichbar ist, hilft die explizite Angabe eines Newsservers durch die Kommandozeilenoption tin -g < servername> .

```
user@sonne> tin -g news.tin.org
                Group Selection (news.tin.org  0)              h=help




   <n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unread,
  s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out

                        *** No groups ***
```

Greifen Sie erstmalig auf einen Newsserver zu, so startet dieser mit einer leeren Liste (in älteren

Versionen wurde eine Liste alle Newsgruppen angezeigt). Bei späteren Kontakten werden nur die zuletzt abonnierten Gruppen angezeigt.

Abonnement einer Gruppe

Um nun an die Liste der auf dem Server vorhandenen Newsgruppen zu gelangen, kann entweder über Eingabe von s (subscribe) mit nachfolgendem Gruppennamen eine solche direkt markiert werden oder aber Sie fordern mittels S mehrere Gruppen an. Hierbei lässt sich mittels Wildcards die Auswahl einschränken:

```
# nach Eingabe von S
Enter wildcard subscribe pattern> *
```

Analog zur Shell steht * für alle Gruppen; der Newsserver antwortet mit folgender Ausgabe:

```
          Group Selection (news.tin.org 12)          h=help
M  1 1631 tin.bugs       bugs mailinglist <tin-bugs@tin.org> (Moderated)
M  2   4 tin.announce     announce mailinglist <tin-announce@tin.org> (Moder
M  3 101 tin.diffs       diffs mailinglist <tin-diffs@tin.org> (Moderated)
M  4 1678 tin.dev        developpers mailinglist <tin-dev@tin.org> (Moderat
M  5 784 tin.users       users mailinglist <tin-users@tin.org> (Moderated)
M  6 238 lists.inn-workers
M  7   4 lists.inn-patches
M  8   1 lists.inn-announce
M  9  30 lists.inn-bugs
M 10 180 lists.inn-committers
M 11  54 lists.debian-mips
M 12  24 lists.ircnet


  <n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unread,
 s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out

              End of groups
```

Die einzelnen Spalten bedeuten von links nach rechts:

- **Status der Gruppe**
- **Gruppennummer**
- **Anzahl ungelesener Beiträge in der Gruppe**
- **Gruppentitel**
- **Kurzbeschreibung zum Inhalt der Gruppe**

Bei den auf manchen Newsservern angehäuften Datenmengen, wird es schnell lästig, die komplette Gruppenliste herunter zu laden. Es empfiehlt sich daher, die Auswahl der Abonnements auf das wirkliche Interessengebiet einzuschränken. Eine bereits erwählte Gruppe lässt sich daher mittels u (unsibscribe) abbestellen. Das Kündigen des Abonnements mehrerer Gruppen erfolgt über U, wobei wiederum Wildcards zulässig sind. Als Beispiel sollen alle mit »lists« beginnende Gruppen abbestellt werden:

```
# nach Eingabe von U
Enter wildcard unsubscribe pattern> lists*
```

Als Reaktion erscheint die eingeschränkte Liste:

```
          Group Selection (news.tin.org 5)          h=help

M 1 1631 tin.bugs    bugs mailinglist <tin-bugs@tin.org> (Moderated)
M 2   1 tin.announce announce mailinglist <tin-announce@tin.org> (Mode
M 3 101 tin.diffs    diffs mailinglist <tin-diffs@tin.org> (Moderated)
```

```
M 4 1678 tin.dev     developpers mailinglist <tin-dev@tin.org> (Modera
M 5 784 tin.users    users mailinglist <tin-users@tin.org> (Moderated)




   <n>=set current to n, TAB=next unread, /=search pattern, c)atchup,
g)oto, j=line down, k=line up, h)elp, m)ove, q)uit, r=toggle all/unread,
  s)ubscribe, S)ub pattern, u)nsubscribe, U)nsub pattern, y)ank in/out

                unsubscribed from 7 groups
```

Lesen der Beiträge

Um nun an die Artikel einer Gruppe zu gelangen, ist der Selektionsbalken mit Hilfe der Pfeiltasten auf diese zu setzen und die Auswahl durch Eingabe von [Enter] zu bestätigen. Man erhält die Liste der Beiträge, welche letztlich auf gleiche Art und Weise ausgewählt werden.

Auf einen Beitrag folgen i.A. ein oder mehrere Antwortartikel, die ihrerseits wiederum Reaktionen hervorrufen können. Die Gesamtheit aller Artikel, die sich auf einen Beitrag beziehen bezeichnet man als Thread und in der Statuszeile eines Artikels werden die Anzahl der ungelesenen Beiträge des Threads und dessen Titelthema angezeigt.

```
Die, 25 Jul 2000 18:26:51       lists.inn-bugs       Thread   2 of  12
Lines 17            Re: supported systems        RespNo  1 of  1
Katsuhiro Kondou <kondou@nec.co.jp>           at ka-nus - ka's not Usenet

Newsgroups: lists.inn-bugs


In article <20000724154351.A7782@work.fantomas.sk>,
    "Matus \"fantomas\" Uhlar" <uhlar@fantomas.sk> wrote;

} inn2.2.3 supports freebsd4.0 although it's not mentioned in INSTALL file...
}
}  FreeBSD 2.2.x, 3.x
}
} 4.x should be there too ;)

How about INN2.3 which will be released very soon?
If so, I'll add it.

  <n>=set current to n, TAB=next unread, /=search pattern, ^K)ill/select,
    a)uthor search, B)ody search, c)atchup, f)ollowup, K=mark read,
    |=pipe, m)ail, o=print, q)uit, r)eply mail, s)ave, t)ag, w=post

                              --More--(98%) [1428/1452]
```

Mit n bzw. [Enter] am Ende eines Beitrages erreicht man die nächste Antwort, mit p gelangt man zurück. Befindet man sich in den Tiefen eines solchen Threads, bringt q einen in die Liste der Beiträge zurück. Mit weiteren Eingaben von q hangelt man sich die Ebenen zurück. Irgendwann erreicht man somit die Gruppenauswahl, wo eine nochmalige Eingabe von q zum Programmende führt.

Erstellen eigener Beiträge

Bei vielen Servern ist diese Möglichkeit abgeschaltet, dennoch soll das Vorgehen demonstriert werden. Durch Eingabe von w im Toplevel einer Gruppe erstellt man einen neuen Beitrag. Befindet man sich dagegen in einem Beitrag, erstellt man eine Antwort auf diesen. Ist man zum schreiben nicht berechtigt, erntet man die Ausgabe ** Posting not allowed ****.**

Verfolgen wir die Vorgänge beim Posten eines neuen Beitrages auf Gruppenebene. Nach Eingabe von w werden Sie zur Angabe eines Titels für den Artikel aufgefordert:

```
          junk (7T(B) 7A 0K 0H R)          h=help

  1  +     9 test news2mail                Hans Wurst
  2  +     9 und noch ein test             Hans Wurst
  3  +     9 und noch eines....            Hans Wurst
  4  +     9 der 5.te



<n>=set current to n, TAB=next unread, /=search pattern, ^K)ill/select,
   a)uthor search, B)ody search, c)atchup, f)ollowup, K=mark read,
   |=pipe, m)ail, o=print, q)uit, r)eply mail, s)ave, t)ag, w=post

Post subject []> Demonstration
```

```
From: Newbie News <newbie@galaxis.de>
Subject: Demonstration
Newsgroups: junk
Organization: Linuxfibel Autoren Team
Summary: Wie postet man einen Artikel
Keywords: Editor, vi, Posten

Es öffnet sich der in der Shell-Variablen EDITOR gespeicherte Editor bzw. der vi. Alle Felder können nachfolgend bearbeitet
werden. Wichtig ist, dass im Header die vollständige Mailadresse angegeben wird. Mit dem Speichern des Textes und Beenden
des Editors erscheint, falls kein Fehler auftrat, das nachfolgende Ausgabefenster.
```

```
          Check Prepared Article


Your article:
"Demonstration"
will be posted to the following newsgroup:
junk Articles for missing newsgroups - DO NOT REMOVE




q)uit, e)dit, g) pgp, p)ost, p(o)stpone: p
```

Post "p" ist bereits voreingestellt und muss nur noch durch [Enter] bestätigt werden.

Sonstiges

Weitere wichtige Kommandos innerhalb von tin sind h zur Anzeige einer Hilfe und M, das Hinweise zur Konfiguration des Newsreaders hervorruft.

Bei heutigem Mitteilungsbedürfnis mancher Zeitgenossen wächst die Beitragsflut mancher Newsgruppen in beträchtliche Dimensionen. Und da jeglicher Netzwerkverkehr im Internet von irgend jemand zu bezahlen ist, verwundert es kaum, dass es nur noch wenige "freie" Newsserver im Netz gibt. Üblich ist die Beschränkung des Zugangs auf autorisierte Benutzer. Um in einem solchen Fall das Authentifizierungsprozedere nicht jedes Mal neu über sich ergehen zu lassen, hilft eine Datei .newsauth im Heimatverzeichnis des Benutzers:

```
user@sonne> cat ~/.newsauth
# Aufbau: nntpserver password [user]
news.galaxis.de News4u
```

***user* ist nur anzugeben, wenn sich der Benutzername auf dem Server vom lokalen Benutzerkennzeichen unterscheidet. Vergessen Sie nicht, die Leserechte der Datei auf den Eigentümer zu beschränken, da diese das Passwort im Klartext enthält (das Abonnement mancher Newsserver könnte dem Benutzer in Rechnung gestellt werden!).**

Bei Exchange-Servern u.a. kann es bei entsprechender Konfiguration dieser vorkommen, dass der Newsclient unbedingt eine Authentifizierung anfordern sollte, da er sonst nur einen leeren Newsserver zu sehen bekommt. Die Anforderung geschieht durch die Option "-A".

write - Nachrichten schreiben und versenden

Aufruf: write user [ttyname]

Mittels `write` lässt sich eine Kommunikation zu anderen Benutzern (die am selben Rechner eine Sitzung offen haben müssen) aufbauen, indem eigene Eingaben auf das Terminal des Gegenübers kopiert werden.

Wird `ttyname` nicht angegeben, und der andere Benutzer ist mehrfach angemeldet, so wählt das Kommando das Terminal mit der geringsten Idle-Zeit (Stillstandszeit - das Terminal, an dem der Partner zuletzt eine Eingabe getätigt hat).

Um mit dem Benutzer namens Tux, der am Terminal `pts/3` angemeldet ist, zu kommunizieren, geben wir Folgendes ein:

```
user@sonne> write tux pts/3
Das ist eine Nachricht von user.
[Ctrl]+[D]
```

Auf Tux's Terminal schaut das Ganze so aus:

```
tux@sonne>
Message from user@sonne on pts/1 at 09:58 ...
Das ist eine Nachricht von user.
EOF
```

Selbstverständlich kann Tux sich vor unliebsamen Nachrichten schützen, indem er mit

```
tux@sonne> mesg n
```

die Darstellung von Nachrichten abstellt (Aktivierung mittels `mesg y`). Macht der Partner unserer Korrespondenz von dieser Möglichkeit Gebrauch, würde ein Aufruf von `write` die folgende Fehlermeldung fabrizieren:

```
user@sonne> write tux pts/3
write: tux has messages disabled on pts/3
```

Nutzerkommandos - Netzwerk

ftp - Einfacher FTP-Client

Aufruf: `ftp [OPTIONEN] [host]`

Das **File Transfer Protocol** ermöglicht den Austausch von Daten zwischen einem FTP-Server und den Clients. Beim Kommando **ftp** handelt es sich um einen einfachen, kommandozeilen-orientierten FTP-Client, der jeder Distribution beiliegen sollte.

Der Zugang auf einen FTP-Server erfordert prinzipiell die Angaben von Nutzerkennzeichen und einem Passwort. Alle Server, die allgemein zugängliche Daten zur Verfügung stellen, ermöglichen daher die Anmeldung als Nutzer **ftp** oder **anonymous**. Als Passwort ist die Angabe der eigenen email-Adresse üblich (aber nicht unbedingt erforderlich).

ftp startet, wird es ohne Angabe eines Zielrechners aufgerufen, im interaktiven Modus und erwartet weitere Eingaben. Als eine der ersten Aktionen ist die Verbindung zu einem Server mit dem Kommando **open Servername** sicherlich sinnvoll:

```
user@sonne> ftp
ftp> open localhost
Connected to localhost.
220 sonne.galaxis.de FTP server (Version 6.2/OpenBSD/Linux-0.11) ready.
Name (localhost:user): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Zur Login-Aufforderung gelangt man unmittelbar, wenn der Zielrechner per Kommandozeile angegeben wird.

Die Kommandos von **ftp** lassen sich grob in drei Gruppen einordnen: Kommandos zur Zugriffssteuerung, Kommandos zur Vereinbarung der Übertragungsparameter und Kommandos zum Datentransfer, wobei in diesem Abschnitt einzig elementare Kommandos zur Datenübertragung von Interesse sein sollen.

Den Inhalt des aktuellen Verzeichnisses (Kommando **pwd**) auf dem FTP-Server zeigen **ls** oder **dir** an. Um das Verzeichnis auf dem Server zu wechseln, gibt man **cd [Verzeichnisname]** ein. Ohne Angabe des Verzeichnisses landet man im FTP-Basisverzeichnis des Servers.

Datenübertragungen finden nun zwischen aktuellem Serververzeichnis und aktuellem Verzeichnis auf dem lokalen Rechner statt. Letzteres kann mit dem Befehl **lcd [Verzeichnisname]** geändert werden. Dateien vom Server lädt man mit **get Datei** herunter, möchte man Metazeichen im Dateinamen verwenden, nutzt man **mget Datei_mit_Metazeichen**:

```
user@sonne> ftp localhost
Connected to localhost.
220 spitz.maus.de FTP server (Version 6.2/OpenBSD/Linux-0.11) ready.
Name (localhost:user): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
```

```
ftp> cd ftp1
250 CWD command successful.
ftp> pwd
257 "/pub/ftp1" is current directory.
ftp> lcd /tmp
Local directory now /tmp
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
foo1    foo2    foo3
226 Transfer complete.
ftp> mget f*
mget foo1? y
200 PORT command successful.
150 Opening BINARY mode data connection for 'foo1' (0 bytes).
226 Transfer complete.
mget foo2? n
mget foo3? n
ftp> bye
221 Goodbye.
```

Die Gegenrichtung, das **Senden von Dateien** zum Server, ist bei anonymem FTP meist untersagt. Möglich ist es mit den Kommandos **put Datei** bzw. **mput Datei_mit_Metazeichen. Auch ein Anlegen von Verzeichnissen kann bei entsprechender Konfiguration erlaubt sein mkdir [Verzeichnisname].**

ifconfig - Status der Netzwerkinterface betrachten

Aufruf: ifconfig [interface]

Das Kommando ifconfig dient dem Administrator zur Konfiguration der Netzwerkschnittstellen. Der "normale" Benutzer kann sich einzig die aktuellen Konfigurationseinstellungen anzeigen lassen. Ohne Argumente aufgerufen, listet ifconfig die Einstellungen aller Schnittstellen auf, mit dem Namen einer Schnittstelle nur deren Charakteristiken:

```
# Der Pfad /sbin ist selten in $PATH eines Nutzers enthalten, deswegen Aufruf mit
vollständigem Pfad
user@sonne> /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:90:27:8F:FC:86
          inet addr:192.168.99.127  Bcast:192.168.99.255  Mask:255.255.255.0
          UP brOADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8203 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7363 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0x8000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Die interessanteren Informationen zu einer Schnittstelle sind sicherlich die IP-Adresse (inet addr), die Broadcast-Adresse (Bcast), die Subnetzmaske (Mask). "UP" besagt, dass die Schnittstelle aktiv ist, unter "RX" und "TX" sind einige Informationen über empfangene und gesendete Pakete abzulesen.

ping- Erreichbarkeit von Rechnern testen

Aufruf: ping [OPTIONEN] host

Ob ein Rechner derzeit im Netzwerk ansprechbar ist, verrät ping. Das Kommando sendet kleine

Pakete an den Zielrechner und misst die Antwortzeiten. Der Rechner kann mittels seiner IP-Adresse oder als symbolischer Name angegeben werden. In letzterem Fall muss dieser aber auflösbar sein (irgendein Mechanismus muss zum Namen die entsprechende Adresse liefern können.

```
user@sonne> ping erde
PING erde.galaxis.de (192.168.100.111): 56 data bytes
64 bytes from 192.168.100.111: icmp_seq=0 ttl=253 time=23.118 ms
64 bytes from 192.168.100.111: icmp_seq=1 ttl=253 time=9.612 ms
64 bytes from 192.168.100.111: icmp_seq=2 ttl=253 time=11.366 ms[Ctrl]+[C]
--- erde.galaxis.de ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 9.612/14.698/23.118 ms
```

ping wiederholt den Sendevorgang bis zum expliziten Abbruch durch Eingabe von [Ctrl]-[C]. Anschließend verrät es eine Statistik mit u.a. verlorenen Paketen (Zeitüberschreitung beim Empfang der Antwort), und den Zeiten der schnellsten/ durchschnittlichen/ langsamsten Übertragung.

Mit der Option -c Anzahl lässt sich die Anzahl der Testdurchläufe von vornherein beschränken. Ist man an der Aufzeichnung des Weges, den das Paket nimmt, interessiert, ist die Option -R nütze. Die weiteren Optionen betreffen im Wesentlichen die Manipulation spezieller Felder des IP-Headers sowie das Aussehen der Testpakete (zum Ändern der Paketgröße (normal 56 Bytes) wähle man -s Bytes):

```
user@sonne> ping -R -c 1 -s 512 www.linuxfibel.de
PING www.linuxfibel.de (195.211.141.227): 512 data bytes
520 bytes from 195.211.141.227: icmp_seq=0 ttl=125 time=30.403 ms
RR:     sonne.galaxis.de (192.168.99.127)
        erde.galaxis.de (192.168.51.1)
        mars.galaxis.de (192.168.5.2)
        aura.saxsys.de (195.211.141.226)
        www.linuxfibel.de (195.211.141.227)
        base.saxsys.de (192.168.5.1)
        amor.saxsys.de (192.168.51.10)
        192.168.99.2
        sonne.galaxis.de (192.168.99.127)
        0.0.0.89        (truncated route)
--- www.linuxfibel.de ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 30.403/30.403/30.403 ms
```

netstat - Allerlei Netzwerkstatistiken...

Aufruf: `netstat [OPTIONEN]`

netstat ist ein Kommando, um Fehler oder Schwachpunke in der Netzwerkkonfiguration zu lokalisieren. Die Behebung der Fehler bleibt zwar dem Administrator vorbehalten, aber die verschiedenen Informationen werden auch dem normalen Benutzer nicht vorenthalten.

Aus der Fülle der Optionen möchten wir nur einige wenige vorstellen.

Ohne Option gerufen, listet netstat den Status aller geöffneter Sockets (Kommunikationsendpunkte) auf. Etwas Licht ins Dunkel bringt da -p, das die Namen der Programme anzeigt, die den Socket eröffnet haben:

```
user@sonne> netstat -p | head -7
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node PID/Program name    Path
```

```
unix  1      [ N ]       StrEAM     CONNECTED     2675   -                    @000004b8
unix  1      [ N ]       StrEAM     CONNECTED     2271   394/netscape         @000003ee
unix  1      [ ]         StrEAM     CONNECTED     623    360/kpanel           @000000bf
```

Eine Statistik der aktiven Verbindungen erhält man mit der Option -a

```
user@sonne> netstat -a | head -10
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address      Foreign Address    State
tcp        0      0 *:6000             *:*                LISTEN
tcp        0      0 *:auth             *:*                LISTEN
tcp        0      0 *:ssh              *:*                LISTEN
tcp        0      0 *:swat             *:*                LISTEN
tcp        0      0 *:http-rman        *:*                LISTEN
tcp        0      0 *:finger           *:*                LISTEN
tcp        0      0 *:pop3             *:*                LISTEN
tcp        0      0 *:login            *:*                LISTEN
```

Die Routing-Tabelle bringt die Option -r zum Vorschein:

```
user@sonne> netstat -r
Kernel IP routing table
Destination   Gateway      Genmask       Flags  MSS Window  irtt Iface
loopback      *            255.0.0.0     U        0 0          0 lo
```

Und die Statistiken bezüglich der verschiedenen Protokolle sollen auch noch Erwähnung finden (Option -s):

```
user@sonne> netstat -s
Ip:
    580 total packets received
    0 forwarded
    0 incoming packets discarded
    0 incoming packets delivered
    580 requests sent out
Icmp:
    ...
Tcp:
    35 active connections openings
    0 passive connection openings
    0 failed connection attempts
    0 connection resets received
    0 connections established
    580 segments received
    580 segments send out
    0 segments retransmited
    0 bad segments received.
    0 resets sent
Udp:
    ...
TcpExt:
```

nslookup - Den Nameserver befragen

Aufruf: `nslookup [-option ...] [host-to-find | -[server]]`

Wie erhält man zu einem Rechnernamen dessen IP-Adresse? Oder man kennt die Adresse und würde gern den zugehörigen Namen erfahren? Dann sollte man den Domain Name Service befragen und mit dem Werkzeug nslookup eine Anfrage stellen.

```
user@sonne> nslookup www.gnu.org
Server:  localhost
Address:  127.0.0.1
```

```
Name:    www.gnu.org
Address:  198.186.203.18

user@sonne> nslookup 198.186.203.18
Server:  localhost
Address:  127.0.0.1

Name:    gnudist.gnu.org
Address:  198.186.203.18
```

Im interaktiven Modus (Aufruf ohne Angabe eines Rechnernamens) lassen sich dem Server eine Reihe weiterer Informationen entlocken. Da die damit verbundenen Möglichkeiten den Rahmen des Erträglichen sprengen würden, schauen wir uns nur an, welche Information zu "www.gnu.org" verfügbar sind:

```
user@sonne> nslookup
Default Server:  localhost
Address:  127.0.0.1

> set type=any
> www.gnu.org
Server:  localhost
Address:  127.0.0.1

Non-authoritative answer:
www.gnu.org     internet address = 198.186.203.18

Authoritative answers can be found from:
gnu.org nameserver = ns1.gnu.org
gnu.org nameserver = sfi.santafe.edu
gnu.org nameserver = nic.cent.net
gnu.org nameserver = ns2.cent.net
ns1.gnu.org     internet address = 158.121.106.18
> exit
user@sonne>
```

Mit "Non-authoritative answer" deutet uns das Programm schon an, dass andere DNS-Server eventuell genauere Informationen liefern könnten. Also versuchen wir eine Anfrage bei "ns1.gnu.org", indem wir den Servernamen auf der Kommandozeile einem Minus/ Leerzeichen folgen lassen:

```
user@sonne> nslookup - ns1.gnu.org
Default Server:  ns1.gnu.org
Address:  158.121.106.18

> set type=any
> www.gnu.org
Server:  ns1.gnu.org
Address:  158.121.106.18

www.gnu.org     CPU = INTEL-686 OS = GNU/LINUX
www.gnu.org     preference = 10, mail exchanger = mescaline.gnu.org
www.gnu.org     preference = 20, mail exchanger = tug.org
www.gnu.org     internet address = 198.186.203.18
gnu.org nameserver = ns1.gnu.org
gnu.org nameserver = sfi.santafe.edu
gnu.org nameserver = nic.cent.net
gnu.org nameserver = ns2.cent.net
mescaline.gnu.org       internet address = 158.121.106.21
tug.org internet address = 158.121.106.10
ns1.gnu.org    internet address = 158.121.106.18
sfi.santafe.edu internet address = 192.12.12.1
nic.cent.net    internet address = 140.186.1.4
ns2.cent.net    internet address = 140.186.1.14
> exit
user@sonne>
```

Jetzt wissen wir, dass Linux den Gnu-Rechner dirigiert, ein Pentium (oder Klone) in seinem Inneren den Takt angibt und wir kennen die verantwortlichen Mailserver und...

telnet - Entfernte Terminalsitzung

Aufruf: `telnet [OPTIONEN] [host [port]]`

telnet ermöglicht eine Terminal-Sitzung auf einem entfernten Rechner, d.h. man arbeitet bei bestehender Verbindung (fast) genauso, als würde man auf dem lokalen Rechner die Befehle eintippen.

telnet wird aus Sicherheitsgründen (z.B. fehlender Passwort-Verschlüsselung) häufig deaktiviert, aber dieses Vorgehen wie auch die Möglichkeiten zur Konfiguration sind Bestandteil des Netzwerkkapitels zu Telnet.

Ohne Argumente gerufen, wechselt das Kommando in den interaktiven Modus und deutet die Bereitschaft zur Entgegennahme der Kommandos an:

```
user@sonne> telnet
telnet> help
Commands may be abbreviated.  Commands are:

close           close current connection
logout          forcibly logout remote user and close the connection
display         display operating parameters
mode            try to enter line or character mode ('mode ?' for more)
open            connect to a site
quit            exit telnet
send            transmit special characters ('send ?' for more)
set             set operating parameters ('set ?' for more)
unset           unset operating parameters ('unset ?' for more)
status          print status information
toggle          toggle operating parameters ('toggle ?' for more)
slc             change state of special charaters ('slc ?' for more)
z               suspend telnet
!               invoke a subshell
environ         change environment variables ('environ ?' for more)
?               print help information
telnet> quit
user@sonne>
```

Um die Verbindung zu einem Rechner herzustellen, ist nun open Rechnername [Port] einzugeben.

```
telnet> open localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Welcome to Linux (i386) - Kernel 2.2.16 (pts/3).

sonne login:
```

Als Nutzerkennzeichen ist nun ein auf dem entfernten Rechner existierendes Kennzeichen zu wählen. Nach anschließender Verifizierung des Passwortes eröffnet man eine Sitzung, auf der analog zu einem lokalen Konsolen-Login verfahren wird. Die Sitzung beendet man mittels logout oder exit.

telnet kann dem Zielrechner auch per Kommandozeilenargument mitgeteilt werden, dann wird unverzüglich mit dem Verbindungsaufbau begonnen und mit Beenden der entfernten Terminalsitzung ist auch telnet beendet. Der Telnet-Dienst verwendet Port 23, durch Angabe eines alternativen Ports kann auch mit anderen Diensten kommuniziert werden, als Beispiel führen wir eine kurzen Dialog mit dem sendmail-Mailserver (am Port 25):

```
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
<ENTER>
Date: Mon Apr 1 11:11:11 MEST 2000
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.
```

Von der Optionen sind -l Nutzer (kleines L) und -x interessant. Ersteres teilt telnet mit, dass wir eine Verbindung unter dem Nutzerkennzeichen "Nutzer" wünschen. Die "login"-Aufforderung entfällt damit. Und "-x" ermöglicht eine verschlüsselte Datenübertragung, allerdings nur, wenn Client und Server diese unterstützen.

traceroute - Den Weg eines Paketes aufzeichnen

Aufruf: `traceroute [OPTIONEN] host [Paketgröße]`

traceroute verfolgt die Route, die ein Paket zu einem Zielrechner (anzugeben als IP-Adresse oder symbolischer Name) nimmt. Das Kommando nutzt die Tatsache, dass ein jedes IP-Paket nur eine bestimmte Lebensdauer haben kann (TTL time to live). Ist diese Dauer abgelaufen, wird der Rechner, bei dem der Verfall auftrat, eine Fehlernachricht an den Absender schicken.

traceroute erhöht nun in jedem Schritt das TTL-Feld des Paketes, so dass es vom jeweils nächsten Vermittlungsrechner verworfen wird. Pro Knotenrechner wird der Vorgang drei Mal wiederholt (Ändern mit -q Anzahl) und die Zeiten gemessen. Auf die Reaktion eines Rechners wird 3 Sekunden gewartet (Ändern mit -w Sekunden).

```
user@sonne> /usr/sbin/traceroute www.gnu.org
traceroute to www.gnu.org (198.186.203.18), 30 hops max, 40 byte packets
 1  erde.galaxis.de (194.180.239.1)  0 ms  0 ms  1 ms
 2  sphere.galaxis.de (192.168.5.1)  1 ms  1 ms  1 ms
 3  195.211.141.1 (195.211.141.1)  3 ms  3 ms  3 ms
 4  br2.frankfurt.gigabell.net (195.211.224.12)  18 ms  19 ms  18 ms
 5  cr1.frankfurt.gigabell.net (195.211.199.30)  17 ms  19 ms  17 ms
 6  atm1-145.cr1.newyork.gigabell.net (195.211.114.30)  111 ms  113 ms  113 ms
 7  usa-gate1.cr2.newyork.lightning.net (216.66.2.77)  110 ms  112 ms  110 ms
 8  a4-0-0.br1.nycmny.us.lightning.net (216.66.3.1)  188 ms  173 ms  139 ms
 9  a2-0-0.br1.snvaca.us.lightning.net (216.66.3.6)  179 ms  177 ms  186 ms
10  s0-1-0.br1.plalca.us.lightning.net (216.66.2.90)  182 ms  186 ms  180 ms
```

```
11  paix2.via.net (198.32.175.80)  185 ms  180 ms  183 ms
12  209.81.23.54 (209.81.23.54)  185 ms  180 ms  181 ms
13  gnudist.gnu.org (198.186.203.18)  184 ms  182 ms  186 ms
```

ypwhich - NIS-Server und Dateien lokalisieren

Aufruf: ypwhich [OPTIONEN]

ypwhich soll hier stellvertretend für die Kommandos des Network Information Systems stehen. Dieses System ist ein Verzeichnisdienst, mit dem bestimmte Konfigurations- und Verwaltungsdateien für einen Bereich des Netzwerkes (die NIS-Domäne) gleichsam zur Verfügung gestellt werden können. Somit ist es z.B. möglich, dass sich die Benutzer an einem beliebigen Rechner eines NIS-Bereiches anmelden und überall dieselbe Umgebung vorfinden.

Ohne Argumente aufgerufen, meldet das Kommando den konfigurierten NIS-Server, der für den Rechner verantwortlich ist. Mit der Option -x werden alle Dateien aufgelistet, die anstatt lokal vom NIS-Server bezogen werden:

```
user@sonne> ypwhich
nis.galaxis.de
user@sonne> ypwhich -x
Benutze "ethers"        für Map "ethers.byname"
Benutze "aliases"       für Map "mail.aliases"
Benutze "services"      für Map "services.byname"
Benutze "protocols"     für Map "protocols.bynumber"
Benutze "hosts"         für Map "hosts.byname"
Benutze "networks"      für Map "networks.byaddr"
Benutze "group"         für Map "group.byname"
Benutze "passwd"        für Map "passwd.byname"
```

Nutzerkommandos - Nutzerinformationen

groups - Anzeige der Gruppenzugehörigkeit

Aufruf: groups [OPTION]... [USERNAME]...

Anzeige der Gruppen, denen ein Nutzer angehört.

```
user@sonne> groups
users fibel

user@sonne> groups root
root : root bin shadow dialout nogroup audio
```

id - Anzeige der effektiven Nutzer-/Gruppen-ID

Aufruf: id [OPTION]... [USERNAME]

Das Kommando kann die reale sowie die effektive Nutzer/Gruppen-Nummer anzeigen. Wesentliche Optionen sind:

-g

Ausgabe der default-Gruppen-ID des Nutzers

-G

Ausgabe aller Gruppen-IDs, denen ein Nutzer angehört

-n

Ausgabe von Namen anstelle von Nummern (in Verbindung mit `gGu`)

-r

Ausgabe der realen anstelle der effektiven Nutzer/Gruppen-Nummer

-u

Ausgabe der Nutzer-ID des Nutzers

```
user@sonne> id
uid=500(user) gid=100(users) Gruppen=100(users),102(fibel)

user@sonne> id -G
```

```
100 102

user@sonne> id -nG
users fibel
```

last - Informationen zu den zuletzt angemeldeten Nutzern

Aufruf: last [-R] [-num] [-n num] [-adox] [-f file] [name...] [tty...]

Das Kommando listet alle in der Datei **/ var/ log/ wtmp** enthaltenen Nutzer auf, die sich seit Bestehen der Datei am System angemeldet haben sowie die Systemstarts, Shutdowns und Runlevelwechsel (Option -x). Mit der Option **-f** kann dem Kommando **last** eine andere Datei als Quelle seiner Informationen mitgegeben werden.

Durch Angabe von **-n Nummer** wird die Ausgabe auf "Nummer" Zeilen beschränkt. Ebenso lässt sich die Ausgabe auf bestimmte Nutzer (Angabe des Nutzernamens) und bestimmte Terminals (Terminalname oder bei TTY's nur die Nummer "tty0" oder einfach "0") reduzieren.

```
user@sonne> last -n 3
user     tty1                  Thu Jun  8 07:02   still logged in
reboot   system boot               Thu Jun  8 07:01
user     tty1                  Wed Jun  7 08:39 - crash  (08:33)

wtmp begins Wed May 17 09:58:15 2000

user@sonne> last 3
root     tty3                  Fri Jun  2 12:06 - crash  (01:52)
newbie   tty3                    Wed May 31 10:49 - 10:53  (00:03)

wtmp begins Wed May 17 09:58:15 2000
```

logname - Anzeige des Loginnamens des Nutzers

Aufruf: logname [OPTION]...

Als Optionen sind nur eine kleine Hilfe und die Ausgabe der Versionsnummer verfügbar.

```
user@sonne> logname
user
```

finger - Informationen über Nutzer

Aufruf: finger [-lmMspho] [user ...] [user@host ...]

Das Kommando versucht aus verschiedenen Quellen Informationen über einen oder mehrere Nutzer zu ermitteln. Im Unterschied zu who vermag **finger** auch Informationen von Nutzern auf anderen Rechnern einzuholen. Aus diesem Grund wird der zugehörige Dämon meist abgeschaltetapropos --exact -w [fs]printf, so dass die Anfragen nur auf dem lokalen Rechner fruchten.

Ohne Argumente oder konkretem Nutzernamen gerufen, werden Loginname, vollständiger Nutzername, das Terminal des Login, die verstrichene Zeit, seit der Nutzer das letzte Mal aktiv war (Idle-Time), die Zeit des Anmeldens und, falls die Anmeldung von einen anderen Rechner aus erfolgte, der Name des entfernten Rechners oder Displays.

```
user@sonne> finger @mond.galaxis.de
[mond.galaxis.de]
```

```
Welcome to Linux version 2.3.39 at mond.galaxsis.de !

  1:35pm up 7 days, 21:30, 23 users,  load average: 0.52, 0.85, 0.62

Login      Name              Tty      Idle  Login Time   Where
user       Beispielnutzer     1         2d     Tue 07:15
tux        Tux der Pinguin    pts/3   23:06     Wed 14:19 :0.0
alf        Der Außerirdische  pts/9     -      Thu 13:29 melmac.outside.all
```

Wird ein bestimmter Nutzername (auch mehrere) angegeben, so werden zusätzlich zu den oben genannten Informationen die Login-Shell des Nutzers angezeigt, dessen Heimatverzeichnis, Büronummer und private Telefonnummer (immer vorausgesetzt, dass diese Information im GCOS-Feld der /etc/passwd auch gesetzt sind). Und schließlich erfährt man auch noch, wann der Nutzer zuletzt seine Mailbox abgefragt hat und ob er einen Termin hat.

```
user@sonne> finger user root
Login: root                         Name: root
Directory: /root                    Shell: /bin/bash
Last login Tue Jun 6 07:41 (MEST) on pts/3 from tty3
New mail received Wed May 17 08:03 2000 (MEST)
    Unread since Mon Feb 14 14:30 2000 (MET)
No Plan.

Login: user                         Name: Beispielnutzer
Directory: /home/user               Shell: /bin/bash
Office: 333, x2-3444                Home Phone: 5
On since Thu Jun  8 07:02 (MEST) on tty1, idle 6:39
Mail last read Thu Jun  8 07:34 2000 (MEST)
No Plan.
```

Die verfügbaren Optionen beeinflussen nur die Anzahl der dargestellten Informationen.

users - Anzeige alle am lokalen Rechner angemeldeten Nutzer

Aufruf: users [OPTION]... [FILE]

Die aktuell angemeldeten Nutzer werden angezeigt. Die Informationen werden aus `FILE` bzw. aus `/var/run/utmp` entnommen, falls `FILE` nicht angegeben wurde:

```
user@sonne> users
berta root user
```

w - Wer ist angemeldet und was tut er?

Aufruf: w - [husfV] [user]

Das Kommando zeigt die uptime des Systems, die angemeldeten Nutzer und was sie tun an.

```
user@sonne> w
  4:45pm  up  9:32,  2 users,  load average: 0.01, 0.04, 0.03
USER     TTY    FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
user     tty1   -                7:13am  9:31m  9:21   0.02s  sh /usr/X11R6/bin/startx
root     tty2   -                4:44pm 11.00s  0.06s  0.04s  -bash
```

who - Wer ist angemeldet?

Aufruf: who [OPTIONEN]... [DATEI | ARG1 ARG2]

Das Kommando liefert eine Reihe von Information über Nutzer, die zurzeit am lokalen System angemeldet sind. Angezeigt werden der Nutzername, der Name des Terminals, an dem der Nutzer angemeldet ist, die Zeit des Anmeldens und, falls der Nutzer von einem anderen Rechner agiert, den Namen des Rechners oder des X-Displays.

Zusätzliche Informationen liefern die Optionen "-i" bzw. "-u", die die Zeit ausgeben, die der Nutzer am betreffenden Terminal inaktiv war (Idle-Time) und "-T" bzw. "-w", die den Status des Terminals anzeigen, ob Schreiben auf dieses mittels write oder talk erlaubt ist ("+": Schreiben erlaubt, "-": Schreiben untersagt, "?": Status unbekannt).

```
user@sonne> who
tux      tty1     Jun  6 07:15 01:40
tux      pts/0    Jun  6 07:15 01:39
tux      pts/1    Jun  6 07:15 01:20
alf      pts/2    Jun  6 07:32 00:01
user     pts/4    Jun  6 07:36   .

user@sonne> who -Hiw
BENU     MESG LEIT     LOGIN-ZEIT   RUHIG VON
tux       +   tty1     Jun  6 07:15 01:47
tux       +   pts/0    Jun  6 07:15 01:47
tux       -   pts/1    Jun  6 07:15 01:27
alf       -   pts/2    Jun  6 07:32 00:09
user      +   pts/4    Jun  6 07:36   .
```

whoami - Wer bin ich?

Aufruf: whoami [OPTIONEN]...

Falls das Gedächtnis mal eine Pause macht...

```
user@sonne> whoami
user
```

Nutzerkommandos - Prozesssteuerung

bg - Prozess im Hintergrund weiterlaufen lassen

Aufruf: bg [job_spec]

Das Kommando lässt einen zuvor gestoppten Prozess im Hintergrund weiterlaufen. Die Ausführung eines Prozesses wird durch Senden des Signals 19 (SIGSTOP) gestoppt. Befinden sich mehrere Prozesse im Zustand "Stopp", kann mittels `%n` der gewünschte Prozess über seine Jobnummer angesprochen werden (siehe jobs).

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

Bemerkung: Die Eingabe von [Ctrl]-[Z] bewirkt in der Bash das Senden des Signals SIGSTP an den aktiven Prozess.

fg - Hintergrundprozess in den Vordergrund holen

Aufruf: fg [job_spec]

Ein im Hintergrund laufender Prozess kann in den Vordergrund geholt werden. Soll nicht gerade der zuletzt im Hintergrund gestartet Prozess gewählt werden, ist die Angabe der Jobnummer erforderlich, die man mit dem Kommando `jobs` ermitteln kann.

```
user@sonne> sleep 100& sleep 500&
[1] 1177
[2] 1178
user@sonne> fg % 2
```

jobs - Anzeige aktiver Jobs

Aufruf: jobs [-lnprs] [jobspec ...]

Mit dem Kommando lassen sich alle Hintergrundprozesse ("Jobs") der aktiven Shell anzeigen.

Als Optionen versteht **jobs**:

-l

Zusätzliche Anzeige der PID

-n

Nur Prozesse anzeigen, deren Status sich seit dem letzten Aufruf von jobs geändert hat

-p

Nur die PIDs anzeigen

-r

Nur aktive Prozesse anzeigen

-s

Nur gestoppte Prozesse anzeigen

```
user@sonne> sleep 100&
[1] 1177
user@sonne> jobs
[1]+  Running                 sleep 10000 &
user@sonne> jobs -n
[1] 1177
```

kill - Prozessen Signale senden

Aufruf: kill [-s signal | -n signum | sigspec] pid ...

Mit dem Kommando lassen sich Signale an Prozesse versenden. Die Angabe der Prozesse erfordert deren Prozessnummer (PID) oder die Jobnummer (%n). Fehlt die Angabe des Signals, wird Signal 15 (SIGTERM) angenommen.

Mit der Option `-l` gibt `kill` eine Liste der möglichen Signale aus:

```
user@sonne> kill -l
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGtrAP      6) SIGAbrT      7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR
```

Die gebräuchlichsten Signale sind dabei 1 (SIGHUP), das vor allem verwendet wird, um Prozesse zum erneuten Einlesen ihrer Konfigurationsdateien zu zwingen, 15 (SIGTERM), das einer höflichen Aufforderung an einen Prozess gleichkommt, er solle doch seine Arbeit beenden (ein Prozess kann dieses Signal ignorieren) und 9 (SIGKILL), das für den betreffenden Prozess das sofortige Ende bedeutet.

```
user@sonne> kill -TERM 255
```

```
user@sonne> kill -15 236

user@sonne> sleep 100&
[1] 5926
user@sonne> kill -KILL % 1
```

Bei der Signalangabe in symbolischer Form wird der Signalname ohne die Vorsilbe SIG benutzt!

killall - Prozessen Signale senden

Aufruf: killall [-egiqvw] [-signal] name ...

Der wesentlichste Unterschied zum Kommando `kill` ist die Spezifizierung der Prozesse über den Namen der Kommandos, die sie ausführen oder mittels ihrer Gruppennummer (`-g` *GID*).
`killall` sendet allen Prozessen Signale, die das Kommando ausführen, mit der Option `-i` kann aber eine nochmalige Rückfrage vor dem tatsächlichen Senden erzwungen werden:

```
user@sonne> killall -i -15 bash
Kill bash(280) ? (y/n) n
Kill bash(350) ? (y/n) n
Kill bash(351) ? (y/n) n
Kill bash(352) ? (y/n) n
bash: no process killed
```

Mit der Option `-w` wartet `killall` so lange, bis der letzte der angegebene Prozess seine Arbeit beendet hat. Das Kommando schickt hierzu periodisch (jede Sekunde) erneut das Signal.

nice - Prozess mit anderer Priorität starten

Aufruf: nice [OPTION]... [COMMAND [ARG]...]

Ein zu startendes Kommando kann durch `nice` eine andere als die voreingestellte Priorität gegeben werden. D.h. im Vergleich zur "normalen Priorität" erhält ein solcher Prozess prozentual weniger/mehr Rechenzeit zugeteilt.
Ein Wert von -20 bedeutet dabei die höchste Priorität; ein Wert von 20 die geringste. Ein normaler Nutzer darf die Priorität eines Prozesses nur verringern (also den Wert erhöhen), nur Root kann diese erhöhen (den Wert verringern).

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

Mit renice kann die Priorität eines laufenden Prozesses beeinflusst werden.

nohup - Prozess vom Elternprozess abnabeln

Aufruf: nohup COMMAND [ARG]...

Das von `nohup` gestartete Kommando läuft unabhängig von der aktiven Shell. D.h. ein so gestartetes Kommando arbeitet auch nach dem Beenden der Sitzung (logout) weiter. Die Ausgaben von `nohup` werden ggf. in eine Datei `nohup.out` umgeleitet. Kann diese im aktuellen Verzeichnis nicht erzeugt werden, wird sie im Heimatverzeichnis angelegt. Scheitert auch dies, beendet `nohup` seine Tätigkeit.
Ein über `nohup` gestartetes Kommandos erhält eine um 5 erhöhte Priorität.

```
user@sonne> bash
user@sonne> ./ sleepproc&
[1] 776
```

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

Anmerkung: Im Beispiel wird in einer Subshell ein Skript "sleepproc" gestartet und die Shell beendet. Wie zu erwarten war, wurde der in der Shell gestartete Prozess mit dem Ende der Shell beendet. In einem zweiten Schritt wird das Skript "sleepproc" unabhängig von der Shell gestartet... es existiert auch nach Beendigung der Shell weiter.

ps - Prozesse anzeigen

Aufruf: ps [options]

Das Kommando gibt eine Momentaufnahme der Prozesse aus. Über Optionen lassen sich zum einen die anzuzeigenden Prozesse selektieren und zum anderen das Ausgabeformat steuern.
Achtung: das Kommando verwendet mehrere Stile der Kommandoeingabe, die Option a besitzt eine andere Bedeutung als die Option `-a`!

Wichtige Optionen zur Selektion von Prozessen sind:

-A oder -e

Wirklich alle Prozesse anzeigen

a

Alle Prozesse anzeigen, die ein Terminal kontrollieren

-U bzw. -G

Auswahl nach UID bzw. GID

N

Negation der Auswahl

p

Nur den angegebenen Prozess (PID ist anzugeben)

r

Nur laufende Prozesse

x

Auch Prozesse anzeigen, die kein Terminal kontrollieren

Optionen, die das Ausgabeformat steuern:

f

Prozesshierarchie als Baum anzeigen

-l

Langformat

e

Anzeige der Umgebungsvariablen des Prozesses

m

Anzeige aller Threads

```
user@sonne> ps
  PID TTY          TIME CMD
  280 tty1     00:00:00 bash
  287 tty1     00:00:00 startx
  288 tty1     00:00:00 tee
  297 tty1     00:00:00 xinit
  344 tty1     00:00:02 konsole
  345 tty1     00:00:00 konsole
  351 pts/1    00:00:00 bash
  352 pts/0    00:00:00 bash
  958 pts/2    00:00:00 ps

user@sonne> ps -p 280 -l
  F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
100 S   500   280   273  0  60   0 -   580 wait4  tty1     00:00:00 bash

user@sonne> ps -x -f
  PID TTY      STAT   TIME COMMAND
  280 tty1     S      0:00 -bash
  287 tty1     S      0:00 sh /usr/X11R6/bin/startx
  297 tty1     S      0:00  \_ xinit /home/user/.xinitrc --
  309 tty1     S      0:07      \_ kwm
  344 tty1     S      0:02          \_ konsole -restore konsolerc.1 -icon konsol
  351 pts/1    S      0:00          |   \_ /bin/bash
  345 tty1     S      0:00          \_ konsole -restore konsolerc.4 -icon konsol
  352 pts/0    S      0:00          |   \_ /bin/bash
  669 pts/0    S      0:00          |       \_ BitchX tec2.saxsys.de
  346 tty1     S      0:01          \_ konsole -restore konsolerc.5 -icon konsol
  354 pts/2    S      0:00          |   \_ /bin/bash
  985 pts/2    R      0:00          |       \_ ps -x -f
  348 tty1     S      0:19          \_ webmaker -restore webmakerrc.2
  288 tty1     S      0:00 tee /home/user/.X.err
  339 tty1     S      0:02 kpanel
  375 tty1     S      0:00 /usr/X11R6/bin/xload
  336 tty1     S      0:00 krootwm
  333 tty1     S      0:01 kbgndwm
  327 tty1     S      0:01 kfm -d
```

pstree - Prozessvererbung anzeigen

Aufruf: pstree [-a] [-c] [-h|-Hpid] [-l] [-n] [-p] [-u] [-G|-U] [pid|user]

Das Kommando stellt die aktiven Prozesse in einer Baumstruktur dar, welche die Prozessvererbung symbolisiert.

```
user@sonne> pstree
init-+-actived
     |-atd
     |-cron
     |-gpm
     |-httpd---httpd
     |-in.identd---in.identd---5*[in.identd]
     |-inetd
```

```
    |-kflushd
    |-klogd
    |-kpiod
    |-kswapd
    |-kupdate
    |-lockd---rpciod
    |-2*[login---bash---ssh]
    |-login---sh
    |-2*[mingetty]
    |-nscd---nscd---5*[nscd]
    |-portmap
    |-syslogd
```

Das Beispiel verdeutlich, dass `init` der Elternprozess aller Prozesse ist. `pstree` versucht per Voreinstellung identische Prozesse in der Darstellung zusammen zu fassen, so bedeutet `2*[mingetty]`, dass 2 Prozesse das Kommando `mingetty` ausführen. Mit der Option `-c` wird diese Zusammenfassung abgeschaltet:

```
...
 |-in.identd---in.identd-+-in.identd
    |                  |-in.identd
    |                  |-in.identd
    |                  |-in.identd
    |                  `-in.identd
...
```

renice - Prozesspriorität ändern

Aufruf: renice priority [[-p] pid ...] [[-g] pgrp ...] [[-u] user ...]

Prozessen kann mittels `renice` nachträglich eine andere Priorität zugeteilt werden. Wie auch bei `nice` gilt, dass einzig Root die Priorität erhöhen darf:

```
user@sonne> renice +10 23258
23258: Alte Priorität: 0, neue Priorität: 10
user@sonne> renice -10 23258
renice: 23258: setpriority: Keine Berechtigung
```

Das Kommando erlaubt das gleichzeitige Verändern der Prioritäten mehrerer Prozesse. Mögliche Angaben sind:

- Mehrere Prozess-IDs
- `-g` *`GID`* Die Gruppennummer von Prozessen
- `-u` *`Nutzer`* Der Besitzer der Prozesse

top - Prozesse sortiert anzeigen

Aufruf: top [-] [d delay] [q] [c] [S] [s] [i] [n] [b]

top listet Prozesse, sortiert nach ihrem Anteil an CPU-Zeit, auf. Nach Voreinstellung wird diese Liste aller 5 Sekunden aktualisiert, mit der Option **-d *Zeit*** kann ein anderes Intervall eingestellt werden. Eine Option **-q lässt das Kommando die Liste so oft wie möglich aktualisieren, mit -n *Anzahl* kann die Anzahl der Refreshs eingeschränkt werden. Anschließend wird `top` seine Arbeit beenden.**

```
user@sonne> top
  4:02pm  up  3:16,  9 users,  load average: 0.27, 0.16, 0.11
100 processes: 95 sleeping, 4 running, 1 zombie, 0 stopped
CPU states:  4.5% user,  4.7% system,  0.0% nice, 90.7% idle
```

```
Mem:  127812K av, 122576K used,    5236K free,  77260K shrd,    2112K buff
Swap: 963816K av,    384K used, 963432K free                   59184K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  413 root      11   0 17784  17M  2064 R       0  3.5 13.9   5:16 X
  489 user      10   0  1216 1216   996 R       0  2.3  0.9   5:15 xosview.bin
 1818 root       4   0  1008 1008   732 S       0  1.5  0.7   0:09 ssh
 3549 user       7   0  1064 1064   840 R       0  1.5  0.8   0:01 top
   98 root       0   0   364  364   244 S       0  0.1  0.2   0:00 scanlogd
    1 root       0   0   204  204   172 S       0  0.0  0.1   0:03 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kupdate
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kpiod
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kswapd
    6 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
   90 bin        0   0   400  400   316 S       0  0.0  0.3   0:00 portmap
  104 root       0   0   696  696   548 S       0  0.0  0.5   0:00 syslogd
  108 root       0   0   856  856   384 S       0  0.0  0.6   0:00 klogd
```

Als Überschrift zeigt top die Uptime, die Anzahl der Prozesse, eingeteilt nach ihrem Status, die Auslastung von CPU, Speicher und Swap an.
Es folgen die Informationen zu den einzelnen Prozessen (die Auflistung enthält nur die Beschreibung der Standard-Informationen):

PID

Prozessnummer

USER

Nutzer, mit dessen Rechten der Prozess ausgeführt wird

PRI

Priorität, mit der der Prozess läuft

NI

Der nice-Faktor, mit dem der Prozess läuft

SIZE

Speichergröße des Prozesses inklusive Stack

RSS

Verbrauch an physischen Speicher

SHARE

Größe des Speichers, der auch von anderen Prozessen genutzt wird

STAT

Status des Prozesses

LIB

Speicherverbrauch der Bibliotheken des Prozesses (bei ELF-Prozessen wird diese Größe mit be SHARE aufgeschlagen)

% CPU

% MEM

Speicherverbrauch (in Prozent)

COMMAND

Kommando, das der Prozess ausführt.

Interaktive Arbeit mit top

Während das Kommando in periodischen Abständen das Terminal mit neuen Informationen überschwemmt, lassen sich verschiedenste Aktionen vornehmen. Folgende Eingaben (Auswahl) bewirken folgende Reaktion:

[Leertaste]

Sofortige Aktualisierung der Anzeige

f

Hierüber kann die Anzeige der Informationen eingestellt werden. Es erscheint eine Auflistung Informationsfelder, wobei ausgewählte Felder mit einem Stern * gekennzeichnet sind. Vor jed Feld steht ein Bezeichner (Buchstabe), durch dessen Eingabe die Auswahl umgeschalten wird. Rückkehr zur Ausgabe von top durch [Enter].

h bzw. ?

Anzeige einer Kurzhilfe zu den verschiedenen Kommandos

k

Zum Senden von Signalen an einen Prozess. Es wird zur Angabe der PID und des zu sendender Signals aufgefordert.

n bzw. #

Zum Ändern der Anzahl angezeigter Prozesse. Man wird zur Eingabe der neuen Anzeige aufgefordert.

o

Ändern der Reihenfolge der Darstellung der Felder. In der oberen Zeile der erscheinenden Ausgabe ist die Reihenfolge symbolisch dargestellt, wobei ein gewähltes Feld durch einen Großbuchstaben markiert ist. Durch Eingabe des entsprechenden Feldbezeichners als Kleinbuchstabe, wird der Eintrag in der Liste "nach hinten" befördert; mittels des Großbuchsta nach vorn. Rückkehr zur Ausgabe von top durch [Enter].

```
 Current Field Order: bAcDgHIjklMnoTPrqsuzVYEFWX
Upper case characters move a field to the left, lower case to the right

* A: PID      = Process Id
  B: PPID     = Parent Process Id
  C: UID      = User Id
* D: USER     = User Name
* E: %CPU     = CPU Usage
* F: %MEM     = Memory Usage
  G: TTY      = Controlling tty
* H: PRI      = Priority
* I: NI       = Nice Value
...
```

r

Ändern der Priorität eines Prozesses

q

Beendet top

Nutzerkommandos - System-Informationen

df - Freien Plattenplatz anzeigen

Aufruf: df [OPTIONEN]... [DATEI]...

Ohne Argumente aufgerufen, zeigt das Kommando die Belegung (Gesamt/Belegt/Frei) aller gemounteten Dateisysteme an:

```
user@sonne> df
Filesystem     1k-blocks      Used Available Use% Mounted on
/dev/hda5        2035606   1882570     47812  98% /
/dev/hda6        2035606   1127568    802814  58% /usr
/dev/hda4          15022      1657     12564  12% /boot
```

Um auch Dateisysteme zu erfassen, die keine Blöcke belegt haben (z.B. "proc", "devpts"), muss die Option **-a** angegeben werden. Ebenso kann ein Dateisystemtyp von der Anzeige ausgeschlossen werden (**-x < Typ>**) bzw. die Ausgabe auf einen Typ beschränkt (**-t < Typ>**) werden:

```
user@sonne> df -a -x ext2
Filesystem     1k-blocks      Used Available Use% Mounted on
proc                   0         0         0   -  /proc
devpts                 0         0         0   -  /dev/pts
```

Die etwas kryptische Angabe der Blockgrößen kann mit **-h** automatisch abgekürzt oder durch **-m** in 1 MByte bzw. **-k** in 1 kByte -Blockgröße erzwungen werden. Anstelle der Anzeige der Datenblöcke lassen sich auch die Inode-Belegungen **-i** auswerten:

```
user@sonne> df -hi -t ext2
Filesystem     1k-blocks      Used Available Use% Mounted on
/dev/hda5            514k      39k     475k    8% /
/dev/hda6            514k      75k     439k   15% /usr
/dev/hda4            7.8k      33     7.8k    0% /boot
```

du - Speicherverbrauch von Verzeichnissen und Dateien anzeigen

Aufruf: du [OPTIONEN]... [DATEI]...

Ohne die Angabe eines Verzeichnisses, zeigt das Kommando den belegten Plattenplatz vom aktuellen Verzeichnis und allen enthaltenen Unterverzeichnissen an. Die Option **-h** bringt die Angabe der belegten Blöcke in eine besser lesbare Form:

```
user@sonne> du -h
152k    ./images/.xvpics
816k    ./images
4.6M    .
```

Die wichtigen Optionen sind: **-b**, **-k** und **-m** zur Ausgabe der Blockgröße in Byte, Kilobyte bzw. Megabyte. **-s** zeigt nur die Information für jedes als Argument benannte Verzeichnis an (keine separate Auflistung der Unterverzeichnisse).

```
user@sonne> du -s
4.6M    .
```

free - Belegung des Hauptspeichers anzeigen

Aufruf: free [-b | -k | -m] [-o] [-s delay] [-t] [-V]

Das Kommando zeigt eine Tabelle zur Belegung des physischen und Swap-Speichers an. Die einzelnen Einträge bedeuten:

- **total** Verfügbarer Speicher insgesamt
- **used** Verwender Speicher
- **free** Freier Speicher
- **shared** Speicher, der von mehreren Prozessen gleichzeitig benutzt wird
- **buffer** Speicher, der für Puffer reserviert wurde oder genutzt wird (z.B. für Pipes)
- **cached** Speicher, der Daten enthält, die "eventuell" demnächst gebraucht werden könnten. Z.B. werden alle Programme, die bereits benutzt wurden, aber ihre Arbeit beendet haben, solange wie möglich im Hauptspeicher gehalten, in der Annahme, dass sie kurzfristig wieder aufgerufen werden. "cached"-Speicher wird verworfen, sobald der Speicherbereich benötigt wird.

```
user@sonne> free
             total       used       free     shared    buffers     cached
Mem:        127748     113836      13912      68768       3848      60496
-/+ buffers/cache:      49492      78256
Swap:       136544          0     136544
```

Die Angabe der Speichergröße erfolgt in Kilobyte, kann aber mit der Option **-b** in Bytes und mit **-m** in Megabytes erfolgen. Eine Zusammenfassung des Swap- und Hauptspeichers liefert die Option **-t** und eine permanente Anzeige erhält man mittels **-s <Sekunden>** , wobei »Sekunden« das Zeitintervall der Aktualisierung bestimmt.

```
user@sonne> free -tms 3
             total       used       free     shared    buffers     cached
Mem:           124        121          3         68          0         72
-/+ buffers/cache:         47         77
Swap:          133          0        132
Total:         258        121        136
3 Sekunden später
             total       used       free     shared    buffers     cached
Mem:           124        121          3         68          0         72
-/+ buffers/cache:         47         77
Swap:          133          0        132
Total:         258        121        136
[Ctrl]+[C]
```

ps - Prozessstatus anzeigen

Siehe unter Nutzerkommandos → Prozesssteuerung.

top - Liste der aktivsten Prozesse anzeigen

Siehe unter Nutzerkommandos → Prozesssteuerung.

uname - Verschiedene Systeminformationen anzeigen

Aufruf: uname [OPTION]...

Ohne Argumente aufgerufen, gibt das Kommando den Namen des Betriebssystems aus. Weitere Informationen entlockt man »uname« mit:

a

Anzeige aller Informationen

m

Typ der Hardware

n

Rechnername

r

Versionsnummer des Betriebssystems

s

Name des Betriebssystems (wie ohne Argumente)

p

Prozessortyp

v

Version des Betriebsystems (Zeit der Kernel-Übersetzung)

```
user@sonne> uname -a
Linux sonne 2.2.14 #1 Mon Nov 8 15:51:29 CET 1999 i586 unknown

user@sonne> uname -p
unknown
```

uptime - Laufzeit des Systems anzeigen

Aufruf: uptime

Das Kommando gibt folgende Informationen aus:

- Aktuelle Zeit
- Zeit, die das System bereits läuft
- Anzahl aktuell angemeldeter Nutzer
- Mittlere Systemauslastung der letzten 5, 10 und 15 Minuten

```
user@sonne> uptime
  4:12pm  up 25 days,   6:22, 11 users,   load average: 0.20, 0.40, 0.45
```

Nutzerkommandos - Textbearbeitung

cat - Dateien verketten

Aufruf: cat [OPTIONEN] [DATEI]...

cat verkettet seine Argumente (Dateiinhalte) und schreibt das Ergebnis auf die Standardausgabe. Werden dem Kommando keine Argumente übergeben, erwartet **cat** seine Eingaben von der Standardeingabe (eine solche Eingabe wird mit [Ctrl]-[D] (EOF) abgeschlossen):

```
user@sonne> cat
Ohne Argument liest »cat« von der Standardeingabe [Enter]
Ohne Argument liest »cat« von der Standardeingabe
Jede durch [Enter] abgeschlossene Zeile gibt »cat« sofort aus[Enter]
Jede durch [Enter] abgeschlossene Zeile gibt »cat« sofort aus
Beendet wird die Ausgabe durch [Ctrl]-[D][Enter]
Beendet wird die Ausgabe durch [Ctrl]-[D]
[Ctrl]-[D]
user@sonne>
```

Mit der Option **-n** nummeriert **cat** die Zeilen in der Ausgabe:

```
user@sonne> cat -n testdatei
     1 Eine erste Zeile.
     2 Noch eine Zeile.
     3 Eine dritte Zeile.
```

csplit - Dateien an definierten Stellen zerlegen

Aufruf: csplit [OPTIONEN]... DATEI MUSTER...

csplit zerlegt die angegebene Datei an allen dem Suchmuster entsprechenden Zeilen und schreibt die einzelnen Teile in die Dateien *xx00*, *xx01*, *xx02*, ...

Als Muster kann Folgendes stehen:

Ganze Zahl

Die Trennung erfolgt **vor** der durch die Zahl spezifizierten Zeile

/ Regulärer Ausdruck/ [OFFSET]

Die Trennung erfolgt **vor** der Zeile, die das Muster enthält

% Regulärer Ausdruck% [OFFSET]

Die Zeilen bis zur Zeile, die das Muster enthält, werden übersprungen

{Ganze Zahl}

Wiederholt die vorhergehende Mustersuche so oft wie angegeben

{*}

Wiederholt die vorhergehende Mustersuche so oft wie möglich

[OFFSET]

+/-n: trennt n Zeilen hinter/vor der durch das Muster spezifizierten Zeile

In unserer Beispieldatei steht Folgendes:

```
user@sonne> cat testdatei
Eine erste Zeile.
Noch eine Zeile.
Eine dritte Zeile.
```

Zunächst soll **csplit** an Zeilen, die das Muster "Zeile" enthalten, aufgeteilt werden:

```
user@sonne> csplit testdatei / Zeile/
0
51
```

Das Kommando trennt wie erwartet **vor** der Zeile mit dem Muster - allerdings betrachtet **csplit** nach der ersten erfolgreichen Suche seine Arbeit als erledigt. Die beiden Dateien *xx00* und *xx01* enthalten 0 bzw. 51 Bytes. Dies sind die Ausgaben des Kommandos.
Soll nun nach jeder Zeile mit dem Muster getrennt werden, hilft diese Eingabe:

```
user@sonne> csplit testdatei / Zeile/  {*}  -f part -z
17
16
18
```

Mittels **-f *part*** werden die Zieldateien mit *part00*, *part01*, ... benannt. **-z** entfernt leere Dateien.

cut - Abschnitte aus jeder Zeile einer Datei extrahieren

Aufruf: cut [OPTIONEN]... [DATEI]...

cut kann verwendet werden, um bestimmte Bytes, Felder oder Zeichen aus den Zeilen einer Datei zu extrahieren.

Extrahieren von Bytes: Mit der Option **-b *Bereich*** lassen sich die durch Bereich benannten Bytes aus der Eingabedatei herausfiltern. Als Beispiel soll die Verwendung in Verbindung mit dem Pipelining demonstriert werden:

```
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
```

cut bezieht seine Eingabe von der Ausgabe des Kommandos **ls -l**. Daraus werden die Bytes 1-11 (Zugriffsrechte) sowie alle Bytes ab dem 56. bis zum Zeilenende (Dateiname) extrahiert.

Extrahieren von Feldern: cut arbeitet mit der Option **-f *Feldnummern* mit Feldern. Per Voreinstellung werden Tabulatoren als Feldbegrenzer angenommen. Mit der Option -d *Zeichen* kann ein beliebiger anderer Begrenzer angegeben werden:**

```
user@sonne> cut -d " " -f 1,3 testdatei
Eine Zeile.
Noch Zeile.
Eine Zeile.
```

Extrahieren von Zeichen: Diese Option arbeitet analog zu -b, da ein Zeichen ein Byte groß ist.

diff - Dateiinhalte vergleichen

Aufruf: diff [OPTIONEN] DATEI_1 DATEI_2

Nicht selten stößt man bei der Suche im Dateisystem auf Dateien mit gleichem Namen. Handelt es sich nun um Kopien oder entspringt die Namensverwandtschaft dem Zufall? diff hilft im Falle von Textdateien weiter, indem es einen zeilenweisen Vergleich vornimmt und die Unterschiede in den Dateien protokolliert. diff lässt sich selbst von eingefügten oder entfernten Zeilen nicht aus der Ruhe bringen. Das Werkzeug erkennt derartige Passagen und vermag unabhängig von der Zeilennummerierung den Vergleich zu führen.

Ein Beispiel

Anhand zweier Dateien soll die Arbeitsweise des Kommandos demonstriert werden.

```
user@sonne> cat Datei1
Abschnitt 1
Abschnitt 2
Abschnitt 3
  Unterabschnitt 3.1
  Unterabschnitt 3.2
Abschnitt 4
```

In der zweiten Datei bleibt die Grobstruktur erhalten:

```
user@sonne> cat Datei2
Abschnitt 1
  Unterabschnitt 1.1
Abschnitt 2
Abschnitt 3
Abschnitt 4
  Unterabschnitt 4.1
```

Zum Verstehen der Ausgabe von diff bedarf es sicher einiger Übung:

```
user@sonne> diff Datei1 Datei2
1a2
>   Unterabschnitt 1.1
4,5d4
<   Unterabschnitt 3.1
<   Unterabschnitt 3.2
6a6
>   Unterabschnitt 4.1
```

Der Report ist aus Sicht der zweiten Datei aus der Eingabe verfasst. Eine mit > beginnende Zeile, deutet an, dass sie hinzugekommen ist. < leitet eine entfernte Zeile ein. Die weiteren Angaben geben Auskunft über die Lage der Änderungen. a (append) steht für Einfügungen; d (delete) für gelöschte Zeilen. Die Nummern vor dem Buchstaben beschreiben hierbei die Positionen in der Originaldatei, die nachfolgenden Nummern die der zweiten Datei.

Neben dem obigen (Standard)Format steuern Optionen die Ausgabe von diff. In Zusammenhang mit patch ist das »unified«-Format verbreitet:

```
user@sonne> diff --unified Datei1 Datei2
--- Datei1      Fri Apr 13 18:28:51 2001
+++ Datei2      Fri Apr 13 18:32:49 2001
@@ -1,6 +1,6 @@
 Abschnitt 1
+  Unterabschnitt 1.1
 Abschnitt 2
 Abschnitt 3
-  Unterabschnitt 3.1
-  Unterabschnitt 3.2
 Abschnitt 4
+  Unterabschnitt 4.1
```

Von den Optionen seien nur -b und -B genannt, mit welchen diff enthaltene Leerzeichen bzw. Leerzeilen beim Vergleich außer Acht lässt.

Werden anstatt Dateinamen die Namen von Verzeichnissen angegeben, vergleicht diff die darin enthaltenen Dateien in alphabetischer Reihenfolge. Das Verfahren kann rekursiv (Option -r) auf alle Unterverzeichnisse angewendet werden, womit sich die Unterschiede ganzer Verzeichnishierarchien auswerten lassen.

In Skripten interessiert häufig nicht die konkrete Ausgabe von diff sondern einzig, ob sich zwei Dateien unterscheiden oder nicht. Hierfür kann der Rückgabewert des Kommandos betrachtet werden:

- **Rückgabewert 0: Dateiinhalte sind identisch**
- **Rückgabewert 1: Dateiinhalte sind unterschiedlich**
- **Rückgabewert 2: Ein Fehler ist aufgetreten**

expand - Tabulatoren durch Leerzeichen ersetzen

Aufruf: expand [OPTIONEN]... [DATEI]...

Die Voreinstellung der Tabulatorschrittweite ist 8 und kann mittels -t *Anzahl* geändert werden.

Die Beispieldatei wurde in einem Editor erstellt und enthält Tabulatoren, deren Anzahl durch eine vorangestellte Nummer markiert ist. Das Kommando cat kann zur symbolischen Anzeige der Tabulatoren genutzt werden:

```
user@sonne> cat -t tabdatei
1^I2^I
1^I2^I3^I4
1^I
```

Nun die Ausgaben von expand mit normaler Schrittweite des Tabulators und mit geänderter Schrittweite:

```
user@sonne> expand tabdat
1       2
1       2       3       4
```

```
1

user@sonne> expand -t 3 tabdat
1   2
1   2   3   4
1
```

Hinweis: Das Kommando unexpand ersetzt Leerzeichen durch die entsprechende Anzahl Tabulatoren.

fmt - Texte formatieren

Aufruf: fmt [-w Länge] [OPTIONEN]... [DATEI]...

fmt ist ein sehr einfacher Textformatierer. Er ermöglicht das beliebige Auffüllen und Umbrechen der Zeilen von ASCII-Texten.

```
user@sonne> cat beispiel.txt
Der Inhalt der Beispieldatei soll zunächst nach der 30. Position umgebrochen werden.
Dann demonstrieren wir die Wirkung der Optionen "-s" und "-t".

user@sonne> fmt -w 30 beispiel.txt
Der Inhalt der Beispieldatei
soll zunächst nach der
30. Position umgebrochen
werden. Dann demonstrieren
wir die Wirkung der Option
"-s" und "-t".
user@sonne> fmt -w 30 -s beispiel.txt
Der Inhalt der Beispieldatei
soll zunächst nach der
30. Position umgebrochen
werden.
Dann demonstrieren wir die
Wirkung der Option "-s" und "-t".

user@sonne> fmt -w 30 -t beispiel.txt
Der Inhalt der Beispieldatei    soll zunächst nach der    30. Position umgebrochen    werden. Dann demonstrieren wir die
   Wirkung der Option "-s"    und "-t".
```

Im Vergleich zu fold werden ganze Wörter im Text niemals zersäbelt. Genauso wird das Wort nach einem Punkt "." nicht abgetrennt. -w Spalten setzt die maximale Länge der Zeilen; mit der Option -s werden umgebrochene Zeilen nicht aufgefüllt (mit der nachfolgenden Zeile kombiniert). Und die Option -t verdeutlicht den Absatzanfang, indem alle weiteren Zeilen eingerückt werden.

Weitere Optionen sind:

-c bzw. --crown-margin

Erhält die Einrückung der ersten beiden Zeilen

-u bzw. --uniform-spacing

Kürzt auf ein Leerzeichen zwischen Worten, zwei nach Sätzen

fold - Lange Zeilen umbrechen

Aufruf: fold [OPTIONEN]... [DATEI]...

Das Kommando bricht nach der Voreinstellung lange Zeilen in der Eingabe nach dem 80. Zeichen um. Das entspricht den Optionen -bw 80. Sollten die Zeilen an Whitespaces getrennt werden, muss das

fold mitgeteilt werden -s. Mit -w Nummer kann eine beliebige Umbruchposition angegeben werden:

```
user@sonne> cat beispiel.txt
Der Inhalt der Beispieldatei soll zunächst nach der 30. Position umgebrochen werden.
Dann demonstrieren wir die Wirkung der Option "-s".

user@sonne> fold -w 30 beispiel.txt
Der Inhalt der Beispieldatei s
oll zunächst nach der 30. Posi
tion umgebrochen werden.
Dann demonstrieren wir die Wir
Wirkung der Option "-s".

user@sonne> fold -w 30 -s beispiel.txt
Der Inhalt der Beispieldatei
soll zunächst nach der 30.
Position umgebrochen werden.
Dann demonstrieren wir die
Wirkung der Option
"-s".
```

grep - Zeichenmuster in Dateien suchen

Aufruf: grep [-[AB] NUM] [-CEFGVbchiLlnqsvwxyUu] [-e MUSTER | -f DATEI] files...

Zu grep existieren noch die verwandten Kommandos egrep und fgrep, die sich nur durch unterschiedliche Interpretation des Musters unterscheiden. Die Kommandos im allgemeinen durchsuchen die angegebenen Datei(en) (oder die Standardeingabe) nach Zeilen, die das Muster enthalten und schreiben im einfachsten Fall die übereinstimmenden Zeilen auf die Standardausgabe.

Da uns der Umgang mit dem Kommando als immens wichtig erscheint, haben wir der grep-Familie einen eigenen Abschnitt im Kapitel Werkzeuge (grep) spendiert und werden dort eine ausführliche Diskussion folgen lassen. Einige Beispiele sollen dennoch den Gebrauch des Kommandos schulen:

```
user@sonne> grep root / etc/ passwd
root:x:0:0:root:/root:/bin/bash

user@sonne> ps ax | grep bash
  287 tty1     S      0:00 -bash
  355 pts/0    S      0:00 /bin/bash
  438 pts/4    S      0:00 -bash
 4341 pts/1    S      0:00 grep bash

user@sonne> ps x | grep bash | grep -v grep
  287 tty1     S      0:00 -bash
  355 pts/0    S      0:00 /bin/bash
  438 pts/4    S      0:00 -bash
```

less - Dateiinhalte betrachten

Aufruf: less [OPTION]... [DATEI]...

less dient wie auch das nachfolgend beschriebene Kommando more zur Anzeige der Inhalte von Dateien. Allerdings ermöglicht less das Zurückblättern selbst in Texten, die aus einer Pipe gelesen werden. Auch beendet sich das Kommando nicht selbsttätig beim Erreichen des Dateiendes.

Als Weiterentwicklung von more kennt less dieselben Tasten zur Navigation (Siehe Tabelle bei more). Wichtige Erweiterungen sind die Rückwärtssuche von Text mit ?Muster, sowie die Kommandos g und G, die zur ersten/ letzten Bildschirmzeile springen. Wird vor den Kommandos eine Zahl *n* eingegeben, so wird zur benannten Zeile gewechselt. Im Falle von G erfolgt die Zählung vom

Dateiende aus.

Darüber hinaus stehen unzählige Kommandozeilenoptionen bereit, die in der Praxis so ziemlich niemals zum Einsatz gelangen.

more - Dateiinhalte betrachten

Aufruf: more [-dlfpcsu] [-num] [+/ Muster] [+ Zeile] [DATEI ...]

more dient der seitenweisen Anzeige des Inhalts von Dateien. Nach jeder Seite wird die Ausgabe angehalten und auf eine Eingabe des Benutzers gewartet. Ist das Dateiende erreicht, beendet sich more selbsttätig. In einer Statuszeile zeigt more die prozentuale Position der dargestellten Seite innerhalb der Datei an.

Das Kommando beginnt mit der Darstellung der ersten Seite, es sei denn, mit + Zeilennummer wird explizit die erste darzustellende Zeile angegeben. Alternativ kann die Ausgabe mit +/ Muster 2 Zeilen vor der ersten Zeile mit dem Auftreten eines bestimmten Musters begonnen werden.

Von den Kommandozeilenoptionen sind *-n* nützlich, das die Anzahl darzustellender Zeilen pro Seite auf den Wert *n* setzt, *-d* zur erweiterten Anzeige der Statuszeile:

```
user@sonne> more -d datei.txt
...
--Mehr--(44%)[Leertaste zum Fortfahren, «Q» zum Beenden.]
```

Die Option *-s* fasst eine Folge von Leerzeilen zu einer einzigen zusammen und *-u* unterdrückt die Darstellung von unterstrichenen Text.

Die nachfolgende Liste zeigt die wichtigsten Eingaben zur Navigation in den Seiten auf:

[Leerzeichen]

Nächste Bildschirmseite, wird zuvor eine Zahl *n* eingegeben, so werden die folgenden *n* Zeilen angezeigt

d

Ausgabe der nächsten 11 Zeilen

***n*s**

Überspringen der nächsten *n* Zeilen

***n*f**

Überspringen der nächsten *n* Bildschirmseiten

***n*b**

Springt um *n* Bildschirmseiten zurück

q

Beenden des Programms

=

Anzeige der aktuellen Zeile

v

Der Editor vi wird mit der aktuellen Datei gestartet

***n*/ Muster**

Springt zum *n*-ten Auftreten des Musters

n

Sucht das letzte Muster erneut, durch Voranstellen einer Zahl *n* wird das *n*-te Auftreten gesuch

,

Springt zum Ausgangspunkt der letzten Suche

n

Springt an den Anfang der nächsten Datei (falls mehrere in der Kommandozeile angegeben wurden), durch Voranstellen eine Zahl *n* wird um die entsprechende Anzahl Dateien weiter gegangen

p

Springt an den Anfang der aktuellen Datei, falls mehrere in der Kommandozeile angegeben wurden, gelangt man in die vorherige Datei aus der Liste

nl - Zeilen nummerieren

Aufruf: nl [OPTION]... [DATEI]...

Das Kommando schreibt seine Eingabe auf die Standardausgabe, wobei vor jeder Zeile deren Nummer ausgegeben wird:

```
user@sonne> nl testdat
   1  eine erste Zeile
   2  eine zweite Zeile
   3  eine einzige Zeile
```

nl kann in der Eingabe logische Einheiten erkennen und für jede eine eigene Nummerierung vornehmen. In diesem Zusammenhang spielte das Kommando früher bei der Textformatierung eine Rolle. Heute wird es vor allem zur Nummerierung von Programm-Quelltexten genutzt.

Als Optionen seien hier nur genannt:

-i *n*

Inkrementiert die Zeilennummer in jedem Schritt um den Wert *n*

-s *Zeichenkette*

Schreibt *Zeichenkette* zwischen jede Zeilennummer und den Text der Zeile

-v *n*

Startet mit Zeilennummer *n*

```
user@sonne> nl -v 10 -i 10 -s ". Zeile =  " testdat
   10. Zeile =  eine erste Zeile
```

```
20. Zeile = eine zweite Zeile
30. Zeile = eine einzige Zeile
```

od - Dateiinhalte in verschiedenen Formaten darstellen

Aufruf: od [OPTIONEN]... [DATEI]...

od lässt sich den Pagern zuordnen, da das Kommando nur das Betrachten von Dateien gestattet und nicht deren Modifikation. Das Standardausgabeformat ist oktale Bytes, kann aber wie folgt geändert werden:

-a bzw. -t a

ASCII-Format

-b bzw. -t oC

Oktale Bytes

-c bzw. -t c

ASCII und Escape-Squenzen (bei nichtdruckbaren Zeichen)

-d bzw. -t u2

Unsigned Short (dezimal)

-f bzw -t fF

Floats

-h bzw. -t x2

Short (hexadezimal)

-i bzw. -t d2

Shorts (dezimal)

-l bzw. -t d4

Long (dezimal)

-o bzw. -t o2

Shorts (oktal)

-x bzw. -t x2

Short (hexadezimal)

Der vorrangige Nutzen des Kommandos liegt sicherlich im Betrachten von Speicherauszügen. Optionen, die die Eingabe beeinflussen sind u.a., -j bytes zum Überspringen der ersten Bytes der Eingabedatei(en), -N bytes zum Begrenzen der Ausgabe auf eine anzugebende Anzahl Bytes und -w bytes zum Begrenzen der Länge einer Ausgabezeile.

Als Beispiel werfen wir einem Blick auf den Masterbootsektor Mbr der ersten IDE-Festplatte:

```
root@sonne> od -x -N 512 / dev/ hda |  tail -5
```

```
od -x -N 512 /dev/hda | tail -5
0000700 0201 fe06 067f 7d82 0000 fac5 003f 0000
0000720 1841 fe05 ffff a318 0044 1d2b 00b7 0000
0000740 0741 fe82 177f 7847 0040 2ad1 0004 0100
0000760 0001 fe83 013f 003f 0000 7d43 0000 aa55
0001000

root@sonne> od -a -N 512 / dev/ hda |  tail -5
0000700 soh stx ack   ~ del ack stx   } nul nul   E   z   ? nul nul nul
0000720   A can enq   ~ del del can   #   D nul   +   gs   7 nul nul nul
0000740   A bel stx   ~ del etb   G   x   @ nul   Q   * eot nul nul soh
0000760 soh nul etx   ~   ? soh   ? nul nul nul   C   } nul nul   U   *
0001000
```

sort - Dateiinhalte sortieren oder mischen

Aufruf: sort [OPTIONEN]... [DATEI]...

Das Kommando sort verkettet alle Eingabedateien und schreibt das Ergebnis im default-Modus sortiert auf die Standardausgabe. Die weiteren Modi sind das Mischen sortierter Daten mittels der Option -m und das Prüfen, ob eine Datei sortiert ist -c.

sort vergleicht, wenn nicht anders angegeben, die gesamte Zeile. Das Sortierkriterium kann auf Feld (er) eingeschränkt werden, wobei die Feldgrenze zwischen letztem Nicht-Whitspace und Whitespace voreingestellt ist. Das impliziert, dass ein Feld führende Whitespace enthalten kann, aber keine nachfolgenden. Um ein Feld zu spezifizieren, sollte die POSIX-konforme Notation -k Feld1[,Feld2] verwendet werden, wobei die Nummerierung der Felder bei 1 beginnt. Um die Ausgabe des Kommandos ls -l nach der Dateigröße zu sortieren, benötigen wir folgende Kommandozeile:

```
user@sonne> ls -l / boot/  |  sort -k 5
-rw-r--r--   1  root     root          620 Feb 24 18:26  os2_d.b
-rw-r--r--   1  root     root       667293 Nov 19  1999  vmlinuz.old
-rw-r--r--   1  root     root       716916 Dec 14  1999  vmlinuz
-rw-------   1  root     root         7680 Jan 13 09:04  chos.map
-rw-r--r--   1  root     root          812 Nov  9  1999  chos.loader-linux
```

Das Ergebnis entspricht vermutlich nicht ganz dem Erwarteten... sort sortiert, wenn man ihm nichts anderes mitteilt, gemäß dem ASCII-Zeichensatz. Und da bspw. die Zeichenkette "71" alphabetisch kleiner ist als "76", wird "716916" vor "7680" eingeordnet... Um eine numerische Sortierung zu erzwingen, ist die Option -n zu verwenden, allerdings kann sort auch dann nicht mit Vorzeichen und Gleitkommazahlen (wie "1.0e-34") umgehen. Letzteres ermöglicht erst die Option -g.

```
user@sonne> ls -l / boot/  |  sort -nk 5
-rw-r--r--   1  root     root          620 Feb 24 18:26  os2_d.b
-rw-r--r--   1  root     root          812 Nov  9  1999  chos.loader-linux
-rw-------   1  root     root         7680 Jan 13 09:04  chos.map
-rw-r--r--   1  root     root       667293 Nov 19  1999  vmlinuz.old
-rw-r--r--   1  root     root       716916 Dec 14  1999  vmlinuz
```

Wichtige Optionen beeinflussen den Vergleich der Felder, so ignoriert -i nichtdruckbare Zeichen und die Option -d bewirkt eine Sortierung wie im Telefonbuch (alle Zeichen außer Leerzeichen, Ziffern und Buchstaben werden übergangen). Soll die Groß- und Kleinschreibung keine Rolle spielen, erreicht man das mit der Option -f. Auch ein anderer Feldtrenner ist über -t Separator angebbar. Eine Ausgabe der Datei / etc/ passwd, sortiert nach der Gruppennummer erhält man mit:

```
user@sonne> sort -t : -gk 4 / etc/ passwd |  head
+::::::
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
```

```
ftp:x:40:2:ftp account:/usr/local/ftp:/bin/bash
man:x:13:2::/var/catman:/bin/bash
mysql:x:60:2:MySQL Database User:/var/mysql:/bin/false
postgres:x:26:2:Postgres Database Admin:/var/lib/pgsql:/bin/bash
amanda:x:37:6:Amanda Admin:/var/lib/amanda:/bin/bash
lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash
```

Schließlich lässt sich die Sortierung umkehren -r und die Ausgabe in eine -o Datei umlenken.

split - Dateien in gleichmäßige Portionen zerlegen

Aufruf: split [OPTIONEN] [INPUT [PREFIX]]

Soll eine sehr große Datei auf Disketten kopiert werden, kommt man um eine Zerlegung der Datei nicht umhin. Das Kommando split erzeugt aus einer Eingabedatei die benötigte Anzahl Ausgabedateien, wobei die Größe der Einzeldateien und deren Präfix-Name konfigurierbar sind.

Mit der Option -b size weist man split an, das Original in Einzeldateien von jeweils size-Bytes Umfang (bis auf die letzte) aufzuteilen, deren Namen dann *Prefix*aa, *Prefix*ab,..., *Prefix*az, *Prefix*ba, ... lauten. Wird "PREFIX" nicht angegeben, lautet dieses x.

```
user@sonne> ls -l
-rw-r--r--   1 user     users     2099200 Jun  8 07:04  linuxfibel.tar
user@sonne> split -b 1440k linuxfibel.tar
-rw-r--r--   1 user     users     2099200 Jun  8 07:04  linuxfibel.tar
-rw-r--r--   1 user     users     1474560 Jun 13 08:22  xaa
-rw-r--r--   1 user     users      624640 Jun 13 08:22  xab
```

Obiges Beispiel trennt exakt an der angegebenen Byteposition. Bei ASCII-Dateien kann es mitunter sinnvoll sein, eine bestimmte Anzahl Zeilen des Originals jeweils in eine neue Datei zu schreiben. Die Option -l number (kleines "L") erzwingt dieses Vorgehen. Der Nachteil sind die sicherlich stark unterschiedlichen Speichergrößen der einzelnen Teildateien, die stark von den Längen der Zeilen der Ausgangsdatei abhängen. Deshalb existiert -C size, wobei split am jeweils letzten Zeilenende trennt, so dass die entstehende Datei maximal size Bytes groß wird.

tac - "Verkehrtes cat"

Aufruf: tac [OPTIONEN]... [DATEI]...

Das Kommando verkettet (mehrere) Dateien und schreibt das Ergebnis in umgekehrter Reihenfolge auf die Standardausgabe. Es arbeitet im einfachsten Fall wie ein "verkehrtes cat":

```
user@sonne> cat testdat
eine erste Zeile
eine zweite Zeile
eine erste Zeile
eine zweite Zeile
eine einzige Zeile

user@sonne> tac testdat
eine einzige Zeile
eine zweite Zeile
eine erste Zeile
eine zweite Zeile
eine erste Zeile
```

Im Unterschied zu cat arbeitet das Kommando allerdings auf Feldern. Da der voreingestellte Feldtrenner der Zeilenumbruch (Newline) ist, bewirkt tac das cat-ähnliche Verhalten.

Der Feldtrenner kann überschrieben werden. Mit der Option -b wird der Inhalt des ersten Feldes der Eingabedatei als Feldtrenner verwendet. Mit der Option -s *Trenner* kann dem Kommando ein beliebiges Trennmuster angegeben werden. Diese Muster kann ein regulärer Ausdruck sein, wenn das Kommando zusätzlich mit der Option -r aufgerufen wird:

```
user@sonne> tac -r -s ei[nt] testdat
zige Zeile
e eine Zeile
eine zweite erste Zeile
eine Zeile
eine zweite erste Zeile
einein
```

uniq - Doppelte Zeilen ausblenden

Aufruf: uniq [OPTIONEN]... [INPUT [OUTPUT]]

uniq entfernt mehrfach auftretende Zeilen in einer sortierten Datei.

```
user@sonne> cat testdat
eine erste Zeile
eine zweite Zeile
eine erste Zeile
eine zweite Zeile
eine einzige Zeile

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

Wichtige Optionen sind:

-c

Anzeige, wie oft die Zeile existiert:

```
user@sonne> sort testdat | uniq -c
1 eine einzige Zeile
2 eine erste Zeile
2 eine zweite Zeile
```

-d

Nur mehrfach exstierende Zeilen werden angezeigt:

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

-f *N* / -s *N*

Die ersten *N* Felder/ Zeichen werden in den Vergleich nicht mit einbezogen:

```
user@sonne> sort testdat | uniq -f 2
eine einzige Zeile
```

-i

-u

Nur einfach vorkommende Zeilen werden ausgegeben:

```
user@sonne> sort testdat |  uniq -u
eine einzige Zeile
```

wc - Zeilen, Wörter, Zeichen zählen

Aufruf: wc [OPTIONEN]... [DATEI]...

Das Kommando zählt die enthaltenen Zeilen (Option -l), Wörter (-w) und Zeichen (-c) in seiner Eingabe. Mit der Option -L wird die Anzahl Zeichen der längsten Eingabezeile ausgegeben:

```
user@sonne> wc testdat
     5      15      89 testdat

user@sonne> wc -c testdat
    89 testdat

user@sonne> wc -L testdat
    18 testdat
```

Nutzerkommandos - Weiteres

Job zu bestimmter Zeit starten - at

Aufruf: at [-V] [-q queue] [-f file] [-mldbv] ZEIT

Mit **at** lassen sich Kommandos zu einem späteren Zeitpunkt ausführen.

Der Zeitpunkt lässt sich in verschiedenen Formaten angeben:

17:23

17.23 Uhr des heutigen Tages

midnight

0.00 Uhr

noon

12.00 Uhr

teatime

16.00 Uhr

10:30pm

22.30 Uhr, mit dem Suffix **am** anstatt »pm« 10:30 Uhr

1204

Am 4. Dezember dieses Jahres, alternative Angaben sind **12.04** und **12/04**

7/12/05

Am 12. Juli 2005, alternative Angaben sind **7.12.05** und **71205**.

Zeitpunkt + Zeitspanne

Als **Zeitpunkt** kann jede oben beschriebene Angabe stehen und **now** (»jetzt«), als **Zeitspanne** kommt ei Wert gefolgt von **minutes** (Minuten), **hours** (Stunden), **days** (Tage) und **weeks** (Wochen) in Frage.

tomorrow, today

Spezifizieren den Zeitpunkt »Morgen« und »heute«.

Die Berechtigung zur Benutzung von **at** kann durch die beiden Dateien **/etc/at.allow** und **/etc/at.deny** beeinflusst werden. Existieren die Dateien nicht, sind alle Nutzer zum Aufruf des Kommandos berechtigt. Im Falle einer existierenden **/etc/at.allow** sind nur die dort aufgeführten Nutzer zur Benutzung zugelassen, die **/etc/at.deny** wird dann nicht ausgewertet. Gibt es nur letztere Datei, so sind die enthaltenen Nutzer von der Verwendung des Kommandos ausgeschlossen.

at liest die zu startenden Kommandos von der Standardeingabe oder aus einer Datei, falls eine solche mit der Option "-f" angegeben wurde. Alle Kommandos werden in eine Warteschlange eingereiht, deren Name aus einem einzelnen Buchstaben besteht. Die Voreinstellung "a" kann mit der Option "-q" überschrieben werden. Die alphabetische Reihenfolge der Queues gibt die Priorität ihrer Bearbeitung vor.

Nach Beendigung eines Jobs versendet **at** die Ausgaben des Kommandos per Mail (sendmail muss installiert sein!) an den Auftraggeber. Für Kommandos, die keine Ausgaben erzeugen, kann mit der Option "-m" eine Mail-Benachrichtigung über den erfolgten Abschluss der Bearbeitung erzwungen werden.

```
user@sonne> at teatime tomorrow
at> echo "Wieder mal Feierabend"
at> [Ctrl]-[D] <EOT>
warning: commands will be executed using /bin/sh
job 3 at 2000-06-07 16:00

user@sonne> at now + 30 weeks
at> mail -s "Wunschliste" Weihnachtsmann@weissnicht.wo < liste.txt at> [Ctrl]-[D] <EOT>
warning: commands will be executed using /bin/sh
job 5 at 2001-01-02 09:12
```

at-Jobliste betrachten - atq

Aufruf: atq [-V] [-q queue]

Der Inhalt der Warteschlange ausstehender at-Jobs wird angezeigt. Sollen nur die Aufträge einer bestimmten Queue betrachtet werden, muss mit der Option "-q" der Name der Warteschlange angegeben werden.

```
user@sonne> atq
1 2000-06-06 16:00 a
3 2000-06-07 16:00 a
4 2001-01-02 09:07 a
5 2001-01-02 09:12 a
6 2000-06-11 10:16 b
```

Die Informationen sind Jobnummer, Zeitpunkt des Auftrags und Name der Warteschlange. Eine analoge Ausgabe liefert **at -l**.

at-Job abbrechen - atrm

Aufruf: atrm [-V] job [job...]

Aufträge können anhand ihrer Jobnummer gelöscht werden:

```
user@sonne> atrm 3 4 5 6
user@sonne> atq
1 2000-06-06 16:00 a
```

at -r arbeitet analog.

Job starten, wenn der Rechner ruht - batch

Aufruf: batch [-V] [-q queue] [-f file] [-mv] [ZEIT]

Batch arbeitet analog zum Kommando at mit dem einzigen Unterschied, dass **batch** den Start des Kommandos soweit zurückstellt, bis die Auslastung des Systems eine gewisse Grenze (load average < 0.8) unterschritten hat.

Das Kommando wird man bspw. dazu benutzen, eine langwierige und ressourcenintensive Berechnung erst nach Feierabend zu starten (wenn normalerweise die Rechner nicht mehr benötigt werden). Sollte aber dennoch ein Kollege Überstunden scheffeln (das soll es wohl geben;-) und den Rechner benutzen, wird die Bearbeitung des Jobs zurückgestellt...

Verwendung und Optionen des Kommandos sind exakt wie bei at beschrieben.

Taschenrechner - bc

Aufruf: bc [-lwsqv] [long-options] [file ...]

bc ist ein Kommandozeilenrechner, der mit beliebiger (!) Genauigkeit zu rechnen vermag. Die Bedienung des Rechners erinnert stark an die Programmiersprache C. Für komplexere Berechnungen wird man deshalb den Algorithmus in eine Datei schreiben und diese als Argument übergeben.

Der **bc** wird mit der Option **-l** angewiesen, die Standard-Mathematik-Bibliothek zu verwenden, mit **-q** verschwindet die lästige Copyright-Ausgabe. Die anderen Optionen betreffen die Kompatibilität zum POSIX-Standard, **-w** gibt Warnungen bei Nicht-POSIX-konformen Erweiterungen und **-s** lässt nur POSIX-konforme Funktionen zu.

Ohne eine Dateiangabe im Argument, startet der **bc** im interaktiven Modus und erwartet die Eingaben. Mit "quit" kann der Rechner wieder verlassen werden:

```
user@sonne> bc -l
bc 1.05
Copyright 1991, 1992, 1993, 1994, 1997, 1998 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
scale=20
variable=5625
sqrt(variable)
75.00000000000000000000
quit
user@sonne>
```

Im Beispiel wurde die Genauigkeit der Ausgabe auf 20 Nachkommastellen gesetzt **scale=20**, anschließend einer Variablen ein Wert zugewiesen **variable=5625** und die Quadratwurzel dieser Variable berechnet **sqrt(variable)**.

Um den Rechner auf der Kommandozeile benutzen zu können, muss er seine Eingaben aus einer Pipe erhalten. Das obige Beispiel hätte man auch so formulieren können (die Skalierung der Ausgabe wurde bewusst weggelassen):

```
user@sonne> echo "variable=1764;sqrt(variable)" | bc -ql
42
user@sonne>
```

42 ist die Lösung! Wie lautete eigentlich die Frage?

Zahlen werden immer im Format *Vorkommastellen.Nachkommastellen* angegeben, wobei die Vorkommastellen für sich allein stehen können und diese im Falle des Wertes "0" auch entfallen dürfen. Gültige Angaben sind somit: **12.476212**, **42**, **0.12** oder kurz **.12**.

Variablen werden durch beliebig lange Kombinationen aus Buchstaben, Ziffern und dem Unterstrich benannt, wobei das erste Zeichen ein Buchstabe sein muss. Es gibt vier spezielle Variablen, **scale** setzt die Genauigkeit der Ausgabe, **ibase** die Basis der Eingabe und **obase** die Basis der Ausgabe. **last** enthält immer den zuletzt berechneten Wert.

Um z.B. eine binäre Zahl in eine hexadezimale umzurechnen, geben wir Folgendes ein:

```
user@sonne> echo "obase=16;ibase=2;1100;" | bc -l
C
```

Für komplexere Algorithmen sind **Kommentare** sicherlich hilfreich. Alle Eingaben des **bc**, die zwischen **/ *** und *** /** eingeschlossen sind, werden von diesem ignoriert.

Zur Berechnung stehen zahlreiche **Ausdrücke** zur Verfügung. An dieser Stelle kann nur auf einige wenige eingegangen werden. Wer sich mit C auskennt, wird die Feinheiten der Verwendung mancher Ausdrücke beurteilen können.

Nachfolgend kann **expr** eine Variable, eine Zahl oder wiederum ein Ausdruck sein. **var** bezeichnet immer eine Variable.

-expr

Negation

++var, var++

Inkrement-Operatoren

--var, var--

Dekrement-Operatoren

expr [+ -* /] expr

Addition / Subtraktion / Multiplikation / Division

expr % expr

Modulo-Operator

expr ^ expr

Potenzieren

(expr)

Klammerung zur Änderung der Vorangregeln

var = expr

Einfache Zuweisung

var <Operation>= expr

Andere Schreibweise für "var = var <Operation> expr

expr1 < expr2

Kleiner-als Vergleich, analog dazu ">", "<=" (Kleiner oder gleich), ">=", "==" (Gleichheit), "!=" (Ungleichh

!expr

Negation

expr && expr

Logisches UND, analog "||" (logisches ODER)

length(expr)

Anzahl Zeichen in expr

read ( )

Liest einen Wert von der Standardeingabe

sqrt(expr)

Quadratwurzel von expr

Weiterhin stehen die bedingte Ausführung **if (expr) tu_etwas [else tu_etwas]**, die Schleifen **while (expr) tu_etwas und for ([expr1] ; [expr2]; [expr3]) tu_etwas zur Verfügung. Die weiteren Steuerkonstrukte (return, break, continue, halt) sind exakt wie in C zu handhaben.**

Bevor einige Beispiele das Verständnis fördern sollen, folgt noch eine Auflistung mathematischer Funktionen, die verfügbar sind, wenn der bc mit der Option -l gestartet wird:

s (x)

Den Sinus von x

c (x)

Den Cosinus von x

a (x)

Den Arcustangens von x

l (x)

Den natürlichen Logarithmus von x

e (x)

e hoch x

j (n,x)

Besselfunktion

Und nun noch einige Beispiele, die auch eine Funktionsdefinition beinhalten:

```
user@sonne> echo "scale=50; 4*a(1)" | bc -l
3.14159265358979323846264338327950288419716939937508

user@sonne> bc -l
define f (x) {
    if (x <= 1) return (1);
    return (f(x-1) * x);
    }
f(7)
5040
quit
```

Kalender - cal

Aufruf: cal [-mjy] [month [year]]

Der Standardkalender glänzt zwar nicht gerade mit einer besonderen Optik, dafür beherrscht er die Berechnung der Tage vom 1.1.0001 bis 31.12.9999.

Per Voreinstellung wird der Sonntag als erster Tag der Woche dargestellt, mit der Option -m kann es der Montag sein. Wer an den julianischen Kalenderdaten interessiert ist, der wähle die Option -j und den Kalender für ein ganzes Jahr zaubert -y hervor.

Vergessen Sie nicht, das Jahr mit 4 Ziffern zu benennen, sonst landen Sie knappe 2000 Jahre in der Vergangenheit. Wird als Argument nur eine Zahl angegeben, so wird diese als Jahr interpretiert, bei zweien steht die erste Zahl für den Monat und die zweite fürs Jahr:

```
user@sonne> cal 7 2000
     Juli 2000
So Mo Di Mi Do Fr Sa
                1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
user@sonne> cal 9 1752
   September 1752
So Mo Di Mi Do Fr Sa
      1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Datumsanzeige - date

Aufruf: date [OPTION]... [+FORMAT]

Mit dem Kommando lässt sich die Systemzeit anzeigen und (falls man die Rechte des Administrators besitzt) ändern.

```
user@sonne> date
Fre Jun 9 14:08:51 MEST 2000
```

Als Optionen sind u.a. nützlich: die Anzeige der Modifikationszeit einer Datei mit der Option -r Dateiname, die Anzeige der »Universal Time« -u, das Setzen der Systemzeit -s und die Formatierung der Ausgabe mittels der Option -d Format.

Um die Ausgabe, vor allem für die Verwendung in Shellskripten, zu manipulieren, stehen eine Reihe von Platzhaltern zu Verfügung. Ein solcher Formatstring muss durch ein Plus + eingeleitet werden:

```
user@sonne> date "+Es ist der %d Tag im Monat %m des Jahres %Y."
Es ist der 26 Tag im Monat 11 des Jahres 2000.
```

Solche Platzhalter gibt es sprichwörtlich »wie Sand am Meer«, die Interessanteren enthält nachfolgende Tabelle:

%%

Das Prozentzeichen selbst

%n

Zeilenumbruch

%t

% H bzw. % I

Stunden 24 bzw. 12 Stunden-Format

% M

Minuten

% r bzw. % X

Zeit im 12 bzw. 24 Stunden-Format (Stunden/ Minuten/ Sekunden [am| pm])

% s

Sekunden seit dem 01.01.1970

% S

Sekunden der aktuellen Minute

% b bzw. % B

Name des Monats mit 3 Stellen bzw. ausgeschrieben

% c

Datum und Zeit im lokalen Format

% d

Tag des Monats

% j

Tag im Jahr

% m

Monat des Jahres (numerisch)

% U

Nummer der Woche im Jahr

% w bzw. % W

Wochentag (0..6) erster Tag ist Sonntag bzw. Montag

% y bzw, % Y

Jahreszahl 2- bzw. 4-stellig

Diskettenzugriff - Die Mtools

Der Austausch von Daten zwischen Windows-Systemen und Linux erfordert die Festlegung auf ein von beiden Seiten verstandenes Format. Und da sich Windows mit der Unterstützung von Fremdformaten etwas bedeckt hält, ist es an Linux, die Daten entsprechend abzuspeichern.

Für den Zugriff auf MSDOS-Dateien entwickelte man eine ganze Kollektion von Werkzeugen, die sich in Sachen Bedienung stark an den (fast gleichnamigen) DOS-Kommandos orientieren, daher auch der Begriff der m-Tools (MSDOS-Tools). Bedeutung erlangt die Sammlung vor allem beim Zugriff auf DOS-Disketten; die nachfolgenden Beispiele beziehen sich auf dieses Medium.

mattrib

Dient zum Ändern der Attribute einer DOS-Datei. Ein führendes Plus "+ " setzt eine Option; ein Mi "-" entfernt sie.

Als Attribute sind a (Archivbit), r (Nur-Lese-Bit), s (Systembit) und h (Hidden-Bit) erlaubt. Ohne Angabe einer Option zeigt das Kommando die Attribute der angebenen Dateien an. Um Attribute rekursiv auf Unterverzeichnisse anzuwenden, kann die Option -/ verwendet werden.

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

mbadblocks

Überprüft eine DOS-Diskette auf fehlerhafte Blöcke. Solche werden in der FAT (File Allocation Tab markiert und bei einer späteren Neuformatierung (mit mformat) ausgeblendet.

mcd

Wechselt das Verzeichnis in einem DOS-Dateisystem; bzw. zeigt - bei Aufruf ohne Argument - das aktuelle Arbeitsverzeichnis an:

```
# Ohne Angabe des Dateisystems wird immer A: angenommen
user@sonne> mmd tmp
user@sonne> mcd tmp
user@sonne> mcd
A:\tmp
```

mcopy

Dateien werden mit diesem Kommando kopiert, dabei können sowohl Quelle(n) als auch Ziel DOS oder Unix-Dateien sein. Eine DOS-Datei muss daher explizit durch die Laufwerksangabe spezifizie werden. Die Angabe des Ziels kann entfallen, dann wird das aktuelle Verzeichnis angenommen. D wichtigen Optionen sind -/ zum rekursiven Kopieren von Verzeichnisinhalten und -t zur Transformation der DOS-Dateienden in das Unixformat.

```
# Datei(en) auf Diskette im Laufwerk a:
user@sonne> mcopy foo.* a:
# Alle Dateien von Diskette ins aktuelle Verzeichnis
user@sonne> mcopy a:*
```

mdel

Löscht Dateien auf einem DOS-Dateisystem:

```
# Datei auf Diskette B löschen:
user@sonne> mdel b:foo.bar
# Alle Dateien auf Diskette A löschen:
user@sonne> mdel \*
```

Anmerkung: Das Quoten des Metazeichens (*) ist notwendig, damit es nicht von der lokalen Shell ausgewertet wird.

mdeltree

mdir

Zeigt den Inhalt eines Verzeichnisses auf einem DOS-Dateisystem an:

```
user@sonne> 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
```

mdu

Das Kommando zeigt - analog zum Unixkommando "du" - die Belegung des DOS-Dateisystems an Die Angabe erfolgt in so genannten Clustern, deren Größe u.a.m. das Kommando minfo offenbart.

mformat

Erzeugt ein DOS-Dateisystem auf einer Diskette. In den meisten Fällen genügt die Angabe des Laufwerks; anhand des Laufwerktyps werden Default-Werte verwendet. Benötigt man abweicher Werte, genügt die Angabe dieser. Für alle nicht spezifizierten Parameter werden automatisch die Voreinstellungen gewählt. Das Kommando selbst versteht (fast) dieselben Optionen, wie das Kommando mkdosfs. Im Zusammenhang mit der Systemadministration (Dateisysteme anlegen) v näher darauf eingegangen.

minfo

Zeigt die Parameter eines DOS-Dateisystems an.

mkmanifest

Der aus Anwendersicht wohl gravierendste Unterschied zwischen klassischem DOS- und Unix-Dateisystemen ist die DOS-Beschränkung der Dateinamen auf bestimmte Zeichen und Länge (auc unter FAT16 und FAT32 bestehen Unterschiede bez. Unix-Dateisystemen).

Die verschiedenen Mtools-Werkzeuge mappen nun Unixnamen, die nicht in das 8:3-DOS-Schema passen, auf einen unter DOS gültigen Namen. Lesen Sie diese Diskette später wiederum mit mtoo aus, werden die alten Namen rekonstruiert. Ein Problem haben Sie jedoch, falls Sie solche Dateie einem Zwischenschritt mit einem Nicht-Mtools-Werkzeug kopieren, da die Information zur Namensumsetzung nun verloren geht. Und genau hier setzte mkmanifest an.

Das Kommando erzeugt aus den eingegebenen Namen ein kleines Shellskript, mit dem sich die "alten" Dateinamen auf jeden Fall rekonstruieren lassen:

```
user@sonne> mkmanifest Ein_langer_Dateiname Zu.viele.Punkte illegal: okay.c prn.dev > manifest
user@sonne> cat manifest
mv ein_lang Ein_langer_Dateiname
mv zuxviele.pun Zu.viele.Punkte
mv illegalx illegal:
mv xrn.dev prn.dev
```

Die Datei "okay.c" ist nicht enthalten, da sie der DOS-Konvention für Dateinamen genügt und somit nicht umbenannt wird. Kopieren Sie die Datei "manifest" ebenso auf Diskette, dann sind Sie jederzeit in der Lage, die originalen Dateinamen wieder herzustellen.

mlabel

Setzt den Namen eines DOS-Dateisystems bzw. liest diesen aus:

```
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 auf einem DOS-Dateisystem.

```
user@sonne> mmd b:\ directory
```

mmount

Es ist eine Erweiterung des Unix-Mount-Befehls und "mountet" ein DOS-Dateisystem. Alle Option die "mount" versteht, sind zulässig. Für weitere Informationen lese man den Abschnitt Mounten eines Dateisystems (Systemadministration-Dateisysteme).

mmove

Verschiebt Dateien in einem DOS-Dateisystem und/ oder benennt sie um. Werden mehrere Dateie angegeben, muss das letzte Argument ein Verzeichnis sein (reines Verschieben). Es lassen sich ebenso Verzeichnisse samt Inhalt verschieben.

mrd

Entfernt ein leeres Verzeichnis aus einem DOS-Dateisystem.

```
user@sonne> mrd b:\ directory
```

mread

Kopiert eine Datei eines DOS-Dateisystems in ein Unix-Dateisystem. Mit -t wird das Format automatisch konvertiert (CF/ LF nach LF).

mren

Dient zum Umbenennen von Dateien oder Verzeichnissen.

```
user@sonne> mren a:foo.bla a:foo.old
```

mtoolstest

Mit diesem Kommando kann die Konfiguration der Mtools überprüft werden. Die Ausgabe des Kommandos kann als Ausgangspunkt für eigene Konfigurationsdateien dienen.

mtype

Dient zum Betrachten des Inhalts einer DOS-Datei.

Installation - Übersicht

Überblick

Letztlich kommt wohl niemand um die Installation eines Linux-Systems umhin... Und genau hier beginnen oft die Probleme.

Schon die Frage nach der »besten Linux-Distribution« wird von eingefleischten Linuxern sehr kontrovers und nicht selten mit zweifelhaften Argumentationen diskutiert. Wir wollen uns hier dem allgegenwärtigen Geplänkel nicht anschließen und verweisen ausdrücklich darauf, dass die gewählte Reihenfolge der Darstellung einzig der alphabetischen Anordnung geschuldet ist. Dennoch stellen wir einige der verbreiteten Distributionen kurz vor.

Wichtigstes Ansinnen dieses Kapitels ist natürlich eine von konkreten Distributionen weitest gehend unabhängige Auseinandersetzung mit den notwendigen Schritten zu einer befriedigenden Linux-Installation. So sind die Abschnitte *Vorbereitungen* und *Bootmanager* allgemein gültig und diskutieren im ersteren Fall die theoretische Basis.

Trotz aller Ähnlichkeiten der Distributionen steckt der Teufel oft im Detail. So kommen wir nicht umhin, die Installation letztlich an realen Linux-Distributionen zu vollziehen. Aus dem stetig wachsenden Angebot haben wir uns für Debian-Linux entschieden, weil Debian zu den wenigen Distributoren zählt, die einzig auf freie Software setzen. SuSE kommt schon wegen seiner starken Verbreitung im europäischen Raum in Frage und auf RedHat schließlich basieren eine Reihe der erfolgreichsten Distributionen (Mandrake, Helloween,...).

Moderne Distributionen bringen meist die Werkzeuge mit, um eine Installation automatisch ablaufen zu lassen. »Diskette rein und Kaffeetrinken - fertig ist die Installation«, ist kein leerer Slogan, sondern durchaus konfigurierbar. Wie? Auch dies wollen wir Ihnen zeigen.

Ziel des Kapitels

Das Kapitel versetzt Sie in der Lage:

- Die notwendige Hardware für den von Ihnen bevorzugten Einsatzfall abzuschätzen
- Eine optimale Partitionierung der Festplatte zu planen
- Die komplette Installation einer Linux-Distribution vorzunehmen
- Eine automatische Installation zu konfigurieren
- Die möglichen Probleme mit Bootmanagern zu beheben
- Die Bootmanager Lilo und Chos zu installieren
- Mehrere Betriebssystem parallel zu installieren

Inhalt des Kapitels

Installation - Distributionen

Welche Distribution?

Vor der ersten Installation steht der Linux-Neuling vor der Wahl einer so genannten Distribution. Unter dieser versteht man einfach eine Zusammenstellung verschiedenster Software-Pakete inklusive irgendwelcher Hilfsmittel, die die Administration des Systems vereinfachen sollen.

Prinzipiell ist das Aufsetzen eines Linuxsystems auch aus den Quellen möglich (*Linux from the scratch*). Neben Zeit und Erfahrung erfordert das Vorgehen allerdings auch eine gewisse Experimentierfreude und ist für den Neu- oder Umsteiger keinesfalls zu empfehlen. Bequemer ist da die Inanspruchnahme der Leistungen eines (zumeist kommerziellen) Distributors, der sein Geld damit verdient, die Software rund um Linux zu einem lauffähigen System zusammen zu schweißen, die eine oder andere (in Eigenregie) entwickelte Administrationshilfe hinzu zu fügen und das Gesamtpaket für ein gewisses Entgeld zu vertreiben. Bezahlt wird also nicht die Software (bei Open Source ist das rein rechtlich auch nicht ohne Weiteres möglich), sondern die Dienstleistung des »Erzeugens der Distribution«. Bei einigen Distributionen erwirbt man mit dem Kauf darüber hinaus noch einen (zeitlich begrenzten und kostenlosen) Installationssupport. Vielfach stehen die Pakete auch auf FTP-Servern (des Herstellers) zum kostenlosen Herunterladen zur Verfügung. Das per Modem zu versuchen, kann die Kosten für die Verbindung auch rasch über die des käuflichen Erwerbs anwachsen lassen...

Weit mehr als 150 verschiedene Distributoren werben um die Gunst der Kunden. Jede Distribution hat ihre Besonderheiten und ist oft auf ein bestimmtes Anwendungsfeld hin optimiert. In der nachfolgenden Aufzählung sollen die Produkte einiger ausgewählter Anbieter gegenüber gestellt werden.

Caldera

- Einfachste Installation
- Erschwerte nachträgliche Installation
- Nur stabile (»hinreichend« getestete) Software enthalten

Debian

- Grundsystem mit höchster Sicherheit
- Eigene Paketverwaltung
- Upgrade übers Internet möglich
- Strikte Trennung von GPL- und anderer Software

Mandrake

- Deutschsprachig
- Basierend auf RedHat (wovon in der neueren Versionen kaum mehr etwas zu merken ist)
- Im Basispaket ist nur GPL Software enthalten
- Kommerzielles Erweiterungspaket

RedHat

- Auch deutschsprachig erhältlich
- Einer der Marktführer
- Das von RedHat entwickelte Installationswerkzeug (anaconda) wird von vielen Distributoren verwendet
- Einfache Installation

Slackware

- Das »Urgestein« unter den Distributionen
- Hat an Bedeutung verloren

SuSE

- Marktführer in Europa
- Eigenes Installationswerkzeug Yast (in Version Yast1 (Text) und Yast2 (Grafik))
- Arbeitet mit dem XFree86-Projekt zusammen (neueste Grafikkartentreiber)
- Umfassendste Software-Zusammenstellung
- Zeitlich begrenzter und kostenloser Installationssupport

Wie gesagt, die Auswahl an Distributoren ist schier unerschöpflich, und ich bitte um Vergebung, weil ich den favorisierten Anbieter des einen oder anderen kurzerhand unter den Tisch habe fallen lassen. Zu erwähnen sind vielleicht noch die so genannten Minimaldistributionen, die schon auf einer einzigen Diskette Platz finden und beim Booten einen Rechner schnell mal in einen kleinen Netzwerkserver verwandeln können (zur Diagnose ist solch ein Vorgehen oft vorteilhaft).

Gemeinsamkeiten

Eine pessimistisch geprägte Aussage wäre: »Alle Distributionen nennen sich Linux.«.

Im Bewusstsein steht »Linux« für das gesamte installierte System. Auf Basis dieses Trugschlusses reduzieren sich die Gemeinsamkeiten aktueller Distributionen auf die identische Bedienung von Programmen. Und selbst das nur mit Gewissheit, falls diese in einunderselben Version vorliegen.

»Linux« mit »Linux« zu vergleichen, verlangt allerdings die Beschränkung auf das, was »Linux« eigenlich ist, nämlich den Betriebssystemkern (»Kernel«). Und hier unterschiedliche Versionen herzunehmen, hieße, Äpfel mit Birnen zu vergleichen. Der kleinste gemeinsame Nenner aller aktuellen Distributionen ist somit der Kernel. Liegt einer Distribution *A* der Kernel in der Version 2.4.3 bei, so ist dieser austauschbar mit dem 2.4.3er Kernel der Distribution *B*.

Austauschbar heißt nicht zwingend *identisch*. Denn nicht selten integrieren Distributoren Eigenschaften in die Kernel, die (noch) nicht Bestandteil des offiziellen Kernels sind. Deshalb kann der Einsatz eines »Fremdkernels« sowohl eine Einschränkung als auch eine Erweiterung der Fähigkeiten bedeuten. Aber funktionieren sollte ein solcher Kernel mit jeder Distribution (insofern auch Module, Konfigurationsdateien usw. ausgetauscht werden).

Damit erschöpfen sich auch schon die Gemeinsamkeiten... Ein problemloser Austausch einzelner Programme ist selten gegeben. Als Achillesferse entpuppen sich zumeist die erforderlichen Bibliotheken, die in der einen Distribution in jener Version, in der nächsten aber schon in einer neueren vorliegen. Aber auch die Lage der Konfigurationsdateien mancher Programme im Dateisystem kann einen Aufruf scheitern lassen. Zumindest letzteres Manko versuchen die Richtlinien der **Linux Standard Base** zu beseitigen.

Unterschiede

Die eine Distribution kommt mit einer CD daher, während die andere derer 10 beinhaltet. Ist »die Andere« die Bessere?

Bei der heutigen Fülle an Software muss ein Distributor schon Sorgfalt walten lassen, um die gängigen Programme auf eine einzige CD zu packen. Aber auch hier finden Sie reichlich Distributionen, die den »normalen« Bedarf bestens abdecken. Mit 2 oder 3 CD-ROMs sollte eigentlich jedes erprobte Stück freier Software zur Verfügung stehen, wozu dann derer 10?

Vor allem die »großen« Linux-Produzenten legen ihren Paketen Demo-Versionen verschiedenster kommerzieller Software bei. Da werden Spiele, CAD-Programme, Netzwerktools, ... mitgeliefert, deren beschränkte Laufzeit oder Funktionalität den Nutzen mehr als in Frage stellt. Schließlich erhalten Sie unterdessen von nahezu allen Software-Herstellern Evaluierungs-Versionen ihrer Produkte, für die Sie maximal die Download-Kosten tragen.

Den »sparsameren« Distributionen liegen nur die Quellen zum Kernel bei. Andere liefern jedes Stück Quelltext auf separaten Medien mit. Interessant ist der Programmcode wohl in erster Linie für einen sachkundigen Programmierer, der eigene Funktionen zu einem bestehenden Programm hinzufügen möchte.

Und CD's mit einem Live-System braucht man nun wirklich nie. Warum sollte ich 10 CDs kaufen, wenn ich Linux nur ausprobieren möchte? Zum Kennenlernen von Linux empfehlen sich die Evaluierungs-CDs, die regelmäßig den verschiedensten Computer-Zeitschriften beiliegen.

Kompatibel?

Aus der Sicht des Anwenders stellt sich die Frage, wie es um den Austausch der Programmpakete zwischen den einzelnen Distributionen bestellt ist. »Eher mäßig«, lautet die ernüchternde Antwort.

Wer über das nötigen Wissen und auch etwas Zeit verfügt, kann sicherlich die Quell-Pakete des einen Distributors an die Gegebenheiten seines Systems anpassen und sich lauffähige Programme selbst kompilieren. Aber in vielen Fällen ist eine gehörige Portion Programmiererfahrung von Nöten...

Da sich in den letzten Jahren das rpm-Format als Archivierungsform für Programmpakete etabliert hat (Ausnahme Debian), sollte man meinen, diese wären ohne Einschränkung auf jedem Linux-System einsetzbar. Doch weit gefehlt, der simple Transfer klappt höchstens zwischen den so genannten RedHat-basierenden Distributionen ohne Komplikationen. Das Problem ist die unterschiedliche Auslegung des File System Hierarchie Standards und die damit verbundene Ablage der Programm- und Konfigurationsdateien im Dateisystem. Leider wandelt hierbei ausgerechnet die in Europa weit verbreitet SuSE-Distribution auf eigenen (und dennoch Standard-konformen!) Pfaden (wobei mit der Version 7.1 eine erste Angleichung erfolgte)...

Aber ganz so düster sieht die Realität dann doch nicht aus. Ein bewährtes Programm wird binnen kurzer Zeit für jede gängige Distribution verfügbar werden, so dass die Entscheidung für eine Distribution heute wohl neben Preis vor allem die spezifischen Installationshilfen und etwas Lokalpatriotismus bestimmen.

Installation - Vorbereitungen

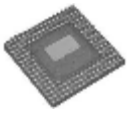
Hardware-Voraussetzungen

Einer der großen Vorzüge von Linux gegenüber anderen verbreiteten Betriebssystemen ist seine relative Bescheidenheit hinsichtlich der Hardware-Anforderungen.

Zwar dreht sich die Spirale auch hier mit jeder neuen System-Version nach oben, aber mit einem abgespeckten Kernel und bei Verzicht auf speicherintensive Anwendungen, lässt sich so mancher alte Rechner noch vernünftig in ein kleines Netzwerk integrieren (so genügt z.B. ein 486er als Firewall vollauf).

An dieser Stelle betrachten wir nun, welche Hardware für welche Zwecke tatsächlich benötigt wird.

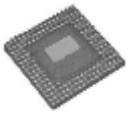
Minimalsystem

 386er  2..4 MB  80 MB (davon 16 Swap)

Ein solcher Rechner taugt vermutlich nur noch zum Testrechner. Wer also daheim ein Netzwerk aufbauen möchte, der kann durchaus auf solch ein betagtes System zurück greifen. Der beschränkte Plattenplatz macht es aber schier unmöglich, die »bekannten« aktuellen Distributionen zu installieren. Der Installationsvorgang selbst wird auf Grund des mangelnden Hauptspeichers beim Anlegen der Ramdisk scheitern. Suchen Sie im Internet, so werden Sie auf mehrere »Mini-Distributionen« stoßen, die sich auch mit solchen begrenzten Kapazitäten begnügen.

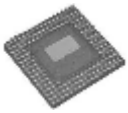
Linux wurde sogar auf 286er Systeme portiert, allerdings gehen damit einige Grundzüge von Unix verloren (Multiuser, Multitasking, Paging,...).

Komfortables Arbeiten unter X

 schneller 486er  16 MB  250 MB (davon 64 Swap)

Mit dieser Festplattengröße lassen sich nur eine beschränkte Auswahl an X-Software-Paketen installieren, auch muss einer der »bescheidenen« Windowmanager (z.B. fvwm, olwm) genügen. Versuchen Sie nicht erst, Gnome oder KDE zum Laufen zu bringen, denn diese Desktop-Umgebungen verlangen nach deutlich schnelleren Prozessoren (komfortable wird's ab 250MHz und 256MB RAM) Für *OpenOffice* sollte gar noch mehr RAM her....

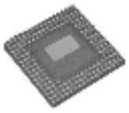
Entwicklersystem

 Pentium-Klasse  > 32 MB  1 GB (davon mind. 64 Swap)

Sieht man einmal von rechenintensiven Grafikmanipulationen o.Ä. ab, genügt ein PC dieser Klasse als Einzelplatzrechner immer. Aber auch hier wird ein knapper Speicherausbau bei der Arbeit mit modernen Desktop-

Umgebungen keine Freude aufkommen lassen. Drastisch erhöht sich der Speicherbedarf, wenn Sie neben KDE oder Gnome Entwicklungsumgebungen wie *kdevelop* oder *Eclipse* einsetzen. Dann stellen 256 MB RAM das absolute Minimum dar.

Serversystem

 > Pentium  > 128 MB  > 1GB (davon mind. 128 Swap)

An einen Fileserver in einem kleinen lokalen Netzwerk werden sicherlich andere Anforderungen gestellt als an einen Webserver für stark frequentierte Seiten...

Tipp

Beim Kauf neuer Hardware sollten Sie sich zuvor in der Hardware-Kompatibiltitätsliste erkundigen, dass diese auch von Linux unterstützt wird. Lassen Sie sich nicht von der auf die Taktfrequenzen der Prozessoren fokussierten Werbung leiten; selbst die langsamsten im Handel erhältlichen Prozessoren (ca. 1000MHz) sind für alle Anwendungen schnell genug. Investieren Sie lieber in einen großzügigen Hauptspeicherausbau (mind. 512 MB) und eine schnelle Festplatte und Grafikkarte.

Partitionieren der Festplatte(n) - Allgemeines

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 beherbergen, aber es gibt viele Gründe, eine Aufteilung vorzunehmen:

Der Swap

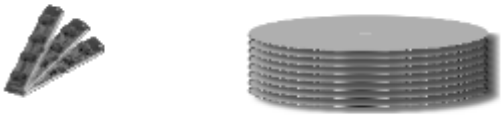


Abbildung 1: Swap »verlängert« den verfügbaren RAM

Linux benötigt eine Auslagerungsdatei. Hierbei handelt es sich um eine »Verlängerung« des Hauptspeichers. Falls im RAM kein Platz mehr frei sein sollte, werden Teile der Daten auf die Festplatte geschrieben. Eine eigene »Swap-Partition« ist einer »Swap-Datei« vorzuziehen, da der Datentransfer auf erstere ohne die Funktionen der Dateiverwaltung und somit beschleunigt vonstatten geht. »Swap-Dateien« sind ein Kompromiss, falls bspw. keine eigene Partition mehr verfügbar ist oder Swap erst nachträglich eingerichtet wird.

BIOS-Probleme

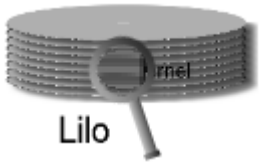


Abbildung 2: 1024-Zylinder-Problem

Möglicherweise kann das BIOS die Geometrie der installierten Festplatte(n) nicht korrekt ermitteln (INT13h). Als Resultat erscheint die Kapazität der Festplatte kleiner als sie tatsächlich ist. Während des Bootvorgangs stehen zur Adressierung der Daten auf den Platten aber häufig nur die Mechanismen des BIOS zur Verfügung, sodass wichtige Daten (bspw. der Kernel selbst) zwingend im zugänglichen Bereich liegen müssen. Diese Problematik ist als 1024-Zylinder-Problem bekannt.

Plattenfehler

Abbildung 3: Plattenfehler

Bei einem Plattenfehler ist möglicherweise nicht das gesamte Areal betroffen. Bei der häufigsten Ursache fehlerhafter Sektoren sind höchstens die Daten der betroffenen Partition(en) hinüber.

Optimierungen

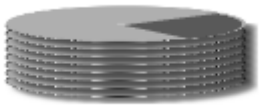


Abbildung 4: Geschwindigkeitsunterschiede beim Zugriff

Aus alten DOS-Zeiten sind dem einen oder anderen Leser sicherlich noch die Bestrebungen in Erinnerung, mittels Defragmentierung der über die Partition verstreuten Dateibestandteile dem System neuen Schwung zu verleihen. Zwar implementieren die meisten Unix-Dateisysteme zuverlässige Algorithmen zur Unterbindung der Fragmentierung auf Dateiebene, aber die relative Lage der Dateien untereinander liegt außerhalb des Einflusses des Dateisystems. Genau hier liegt oft unbeachtetes Potential zur Optimierung, da bspw. zahlreiche Anwendungen aus mehreren Dateien bestehen. Lägen diese nun räumlich nahe beieinander, müsste der Schreib-Lese-Kopf der Festplatte nicht unnötig weite Wege zurücklegen. Das Laden verliefe schneller.

Über die Verknüpfung eines Verzeichnisses mit einer Partition (»Mounten«) ist sichergestellt, dass alle Dateien unterhalb dieses Verzeichnisses in ein und derselben Partition landen. Derartige Überlegungen empfehlen sich, wenn das Linuxsystem auf mehrere Festplatten verteilt werden soll.

Aber selbst eine gezielte Verteilung der Daten auf die Partitionen einer einzelnen Festplatte ergibt einen Sinn, da die Schreib- und Lesegeschwindigkeit nicht über den gesamten Bereich der Platte gleich ist. War bei älteren Festplatten die Anzahl der Sektoren pro Spur noch konstant, so verfügen Neuere auf den äußeren Spuren über eine höhere Anzahl Sektoren als auf den inneren. Somit werden auf äußeren Zylindern pro Umdrehung mehr Daten gelesen als auf den inneren.

Aktualisierung

Abbildung 5: Update

Indem Daten und System eigene Partitionen erhalten, ist eine Systemaktualisierung einfacher zu realisieren. Nur die Systempartition ist zu modifizieren.

Partitionieren der Festplatte(n) - Das 1024 Zylinder Problem

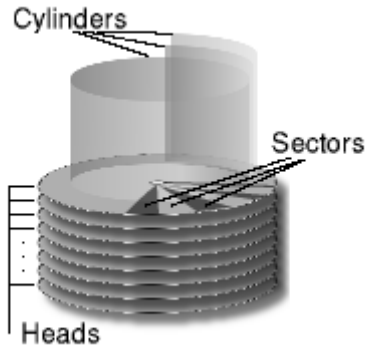


Abbildung 6: Aufbau einer Festplatte

Um die Ursachen des 1024-Zylinder-Problems zu beschreiben, ist die Kenntnis des prinzipiellen Aufbaus von Festplatten nützlich. Festplatten bestehen aus einem Stapel starr verbundener Scheiben mit beidseitig magnetisierter Oberfläche. Zum Schreiben und Lesen der in den Magnetisierungsmustern kodierten Daten greifen seitlich pro Scheibe zwei Schreib-Lese-Köpfe ein. Alle Schreib-Lese-Einheiten sitzen auf einem Kamm, sodass sie stets synchron über die Oberflächen bewegt werden. Ein einzelner Schreib-Lese-Kopf wird als **Head** (englisch für »Kopf«) bezeichnet. Der während einer Umdrehung der Scheiben von einem Head überstrichene Bereich heißt Spur. Da jeder Kopf zu einem Zeitpunkt dieselbe Spurnummer betrachtet, formen die übereinander liegenden Spuren einen **Cylinder** (Zylinder). Eine Spur wiederum ist in einzelne **Sectors** (Sektoren) untergliedert. Ein solcher Sektor ist die kleinste Adressierungseinheit einer Festplatte und seine Koordinaten berechnen sich aus Zylindernummer, Kopfnummer und Sektornummer. Dieses klassische Schema zur Lokalisierung eines Sektors nennt sich **CHS** (Cylinder,Head,Sector).

Der Interrupt 13h des BIOS

Zumindest während des Hochfahrens des Rechners ist der Zugriff auf die Festplatten allein durch die Mechanismen des BIOS realisierbar. Noch aus DOS-Zeiten dienen die Aufrufe des INT13h des BIOS zum Ansprechen jeder IDE-Festplatte im System. Dessen Aufgabe ist sowohl das Auslesen der Geometrie der Festplatte (CHS), als auch die Adressierung des Sektors, den eine Platte als nächstes anzufahren hat.

Die »betagteren« BIOS-Implementierungen stellten für die CHS-Adressierung ganze 20 Bit zur Verfügung. Mit 10 Bit für die Zylinder (2^{10}=**1024**), 6 Bit für die Sektoren (63! [Zählung ab 1]) und 4 Bit für die Köpfe (16) war der Sektor mit der Nummer 1048576 der höchste adressierbare Datenblock. Oder anders ausgedrückt: Bei einer Standard-Sektorgröße von 512 Bytes war bei einer Festplattenkapazität von 528MB das Ende der Fahnenstange erreicht. Aber wer hat schon noch ein BIOS aus den 80er Jahren?

Das oben beschriebene Szenario fand sich in der IDE-Spezifikation wider. Die folgende EIDE-Spezifikation reduzierte die Bits der Sektoren (-2) zu Gunsten der Zylinderanzahl (+2), sodass Platten bis zu 2.1GB Größe gehandhabt werden konnten. Die nächste Erweiterung erfuhr das BIOS, das nun 24 Bit für CHS verwendete: 14 Bit

gönnte man den Zylindern (65536), 8 Bit den Sektoren (255) und die Köpfe gingen bei der Erhöhung wieder einmal leer aus. Für sie standen weiterhin rare 4 Bit zur Verfügung. Wieder unter der Annahme der 512Byte-Sektoren resultiert hieraus die Begrenzung von Festplatten auf 8,4GB.

Also doch kein 1024er Problem?

Die Verteilung der 24 Bit auf CHS ist rein willkürlich. Wählen Sie bspw. 10 Bit für die Zylindernummern (**1024**), 6 Bit für die Sektoren (63) und 8 Bit für die Köpfe (256), so ergibt die Berechnung der adressierbaren Festplattenkapazität dieselbe Schranke, nämlich 8,4GB. Das Problem der fossilen 20 Bit und der noch verbreiteten 24 Bit Adressierung beruht also auf denselben Ursachen und als Begriff hierfür hat sich »das 1024-Zylinder-Problem« etabliert.

Aus Sicht der Festplatte ist die Berechnungsvorschrift ohnehin unerheblich, denn sie weiß selbst, wo der Sektor mit der Nummer 34788 ist, und ihr ist egal, wie das BIOS diese Nummer berechnete.

Die neue Adressierung: LBA

CHS als Berechnungsmodell zur Adressierung deckte sich mit den Geometrien früherer Festplatten. Im Zuge der Miniaturisierung und der höheren Packungsdichten der Daten beschritten die Hersteller neue Wege. Nehmen Sie sich ein Blatt Papier zur Hand, schneiden einen Kreis aus und beginnen, diesen Kreis von außen nach innen in konzentrischen Kreisen zu beschreiben. Sie werden bemerken, dass stets weniger Text auf einen Kreis passt, je näher Sie zum Mittelpunkt gelangen. Für »fossile« Festplatten bedeutete dies, dass die Anzahl der Daten, die auf eine Spur untergebracht werden konnten, durch die Kapazität der innersten Spur begrenzt war. Der logische Schritt war die Abkehr vom starren Zylinder-Sektoren-Verhältnis. Tatsächlich nimmt die Anzahl Sektoren pro Zylinder bei modernen Festplatten mit zunehmender Zylinderzahl stetig ab (Zylinder werden von »außen« nach »innen« nummeriert).

Nur greift nun die Adressierung per CHS nicht mehr... Deshalb findet heute nahezu ausschließlich die Adressierung mit **LBA** *Linear Block Addresses* Verwendung, wobei die Sektoren der Reihe nach, bei 0 und »außen« beginnend, durchnummeriert sind. Die Abbildung von LBA auf CHS wird von der Logik der Festplatte selbst vorgenommen.

Ein BIOS sollte eine Festplatte heute immer per LBA adressieren. Zum einen entspricht das CHS-Format der Festplatte nicht mehr dem vereinfachen Schema eines konstanten Sektoren-Zylinder-Verhältnisses, womit ein BIOS den kompletten Berechnungsmechanismus implementieren müsste (was für jeden Festplattentyp schier unmöglich ist). Zum anderen werden für die Adressierung dieselben Bits verwendet, die sonst das CHS-Format beschreiben, sodass letztlich dieselbe Anzahl Sektoren erschlossen wird.

Durch die Einführung der Festplatten mit einer unterschiedlichen Sektorenanzahl auf den Zylindern musste einem »alten« BIOS weiterhin ermöglicht werden, auch solche Platten anzusteuern. Die Industrie entschied sich, allen Platten dieser Bauart einheitlich die Dimension 16383 Zylinder, 16 Köpfe, 63 Sektoren/Zylinder zu verleihen. Eine Abfrage der Geometriedaten per INT13h durch das BIOS liefert somit immer diese Daten. Fazit: Wir sind noch immer beim 1024-Zylinder-Problem...

Alles halb so wild...

Anlass zur Entwarnung kann aus drei Gründen gegeben werden.

1. Die »post 1998« Konvention erweitert die Interrupt 13 Spezifikation. Ein zu dieser Konvention konformes BIOS verwendet statt der 24 Bit zur Adressierung nun 28 Bit. Hiermit ergibt sich als Schranke der Plattenkapazität 128 GB. Eine nochmalige Aufstockung auf 64 Bit ist bereits angedacht, sodass derartige Probleme auf lange Sicht der Vergangenheit angehören dürften. BIOS-Versionen ab ca. 1999 sollten somit der 1024-Zylinder-Grenze nicht mehr unterliegen.
2. Zahlreiche Hersteller von Festplatten bieten zusätzlich eine »Diskmanagement«-Software, die, im Master Boot Record platziert, die BIOS-Mechanismen zum Ansprechen der Platten durch eine eigene Implementierung ersetzt. Sobald das BIOS den Mbr startet, wird die Software aktiv und alle Festplattenzugriffe laufen ab sofort über diese.
3. Der Linux-Kernel spricht die Festplatten ohne Zugriff auf die BIOS-Routinen an, sodass, ist der Kernel erst einmal aktiv, die volle Kapazität der Platte angesprochen werden kann.

Womit wir nochmals auf das reale 1024-Zylinder-Problem stoßen. Ein modernes Betriebssystem benötigt zum Laden einen Bootloader (nur mit Hilfe eines solchen ist die parallele Installation mehrerer Systeme möglich). Solche Bootmanager verfügen jedoch in den seltensten Fällen über eigene Mechanismen, um eine Festplatte zu adressieren. Sie bedienen sich zumeist der BIOS-Aufrufe, was auch bedeutet, dass sie genau jenen Bereich der Platte ansprechen können, den das BIOS anzusprechen vermag. Zum Laden des Kernels muss ein Bootloader diesen innerhalb der vom BIOS gegebenen Grenzen vorfinden!

Um von jeglichen Zwängen befreit zu sein, benötigen Sie neben dem aktuellen BIOS auch eine geeignete Implementierung des Bootmanagers. Bspw. kann der Lilo erst ab Version 32 (Revision 3) Systeme »hinter« dem Zylinder 1024 booten.

Partitionieren der Festplatte(n) - Wie viele Partitionen sind mindestens notwendig?

Die weiteren Überlegungen werden zeigen, dass die Verteilung der Installation auf mehrere Partitionen sich in zahlreichen Fällen vorteilhaft auswirkt. Im Augenblick liegt der Fokus allerdings auf dem absoluten Minimum. Es ist unerheblich, auf wie viele Festplatten die Partitionen verteilt werden. Nur ihre Existenz zählt.

Eine Partition für jedes System

Für jedes zu installierende System (Linux, Windows, BSD,...) ist mindestens eine eigene Partition zu veranschlagen. Dies gilt ebenso für mehrere Linuxinstallationen (falls Sie verschiedende Distributionen gleichzeitig verwenden möchten), da jedes Linuxsystem seine eigene Root-Partition benötigt.

Aus diesem Aspekt ist die Summe der verwendeten Systeme die untere Schranke der Partitionenanzahl.

Einmal Swap für alle

Aus Gründen der Effizienz benötigt ein Linuxsystem mindestens zwei (primäre oder logische) Partitionen. Eine dient zur Aufnahme der Root-Partition, 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 (RAM) sollten Sie mindestens 256 MB (Minimalinstallationen begnügen sich bei Verzicht auf speicherhungrige Anwendungen auch mit 64MB) veranschlagen. Mit einem entsprechenden Kernel (ab 2.1.117) darf eine einzelne Swap-Partition bis zu 2GB groß sein. Allerdings unterstützen nicht alle Installationswerkzeuge das Anlegen eines Swap-Bereiches mit mehr als 128 MB. In einem solchen Fall können bis zu 8 Swap-Partitionen (63 Partitionen bei entsprechendem Patch) gleichzeitig verwaltet werden.

Bei mehreren installierten Linuxsystemen genügt eine einzige Swap-Partition; es kann ohnehin zu einem Zeitpunkt nur ein System gebootet werden.

Altes BIOS?

Falls auf Ihren Rechner das 1024-Zylinder-Problem zutrifft, müssen Sie sicher stellen, dass der Linuxkernel und die Daten des Bootmanagers im »vorderen« Bereich der Festplatte liegen. Die Optionen hierfür sind das Erzeugen einer Linux-Root-Partition so, dass sie definitiv vor dem 1024. Zylinder endet. Oder aber Sie reservieren zuvorderst eine kleine Partition (10-20MB), die das /boot-Verzeichnis des Linuxsystems aufnehmen wird.

Partitionieren der Festplatte(n) - Wie viele Partitionen sind für Linux ratsam?

Im speziellen Fall wird man sicher bessere Lösungen finden, als die, die ich hier angebe. Aber bei der Verteilung der Partitionen das Optimum herauszuholen, bedeutet auf jegliche Hardwarekonstellation und jeden Einsatzbereich einzugehen. Deswegen sollten die nachfolgenden Tipps nur als Faustregeln verstanden werden.

Gesteigerte Geschwindigkeit mit mehreren Festplatten

Wer mehrere Festplatten sein Eigen nennt, der sollte eine Verteilung der Installation über diese in Betracht ziehen.

In einem Linuxsystem bewerben sich etliche Prozesse um die knappen Hardwareressourcen und nicht wenige davon greifen konkurrierend auf die Festplatte(n) zu. Liegen die Daten zweier Prozesse auf ein und derselben Festplatte, dann wird der Schreib/Lesekopf im Sinne der Gleichbehandlung der Anforderungen mal ein Stück der einen Datei, dann wieder eine Portion der anderen lesen. D.h. zusätzlich zur reinen Zeit für das Lesen der Dateien geht noch Zeit für die Kopfpositionierung drauf. Lägen die Daten auf verschiedenen Festplatten, gelängen die Zugriffe wesentlich schneller...

Zwar weiß niemand mit Sicherheit, welcher Prozess zu welcher Zeit auf welche Datei zugreift, aber mit ein paar simplen Überlegungen lassen sich einige Verzeichnisse herauspicken, die besser auf getrennten Festplatten/Partitionen angelegt werden sollten:

- Swap-Partition und das Root-Verzeichnis sind sichere Kandidaten... Gerade bei Rechnern mit geringem Hauptspeicherausbau (< 128 MB) ist der Swap heiß begehrt...
- `/usr` und `/usr/lib` sollten auf verschiedene Platten: Die meisten Programme erfordern das dynamische Nachladen von Bibliotheken. Wird also ein Programm aus `/usr/bin` gestartet, werden meist noch eine Reihe von Bibliotheken aus `/usr/lib` geladen. Das geht schneller, wenn beide Verzeichnisse auf getrennten Platten liegen...
- Ähnliche Überlegungen legen eigene Bereiche für `/var` und `/tmp` ... nahe.

Gesteigerte Geschwindigkeit auch bei einer Festplatte

Aber selbst wenn nur eine Festplatte für Linux verfügbar ist, lassen sich einige Optimierungen treffen.

Die Zugriffsgeschwindigkeit bei neueren Festplatten hängt stark von der Zylindernummer ab. So sollten die stark frequentierten Partitionen außen (kleine Zylindernummer) liegen und die weniger gebräuchlichen auf den inneren Zylindern.

Eine Aufteilung folgender Art trägt den Überlegungen Rechnung:

1. `/boot` (1024-Zylinder-Problem)
2. Swap
3. Root-Verzeichnis
4. ...
5. `/home`

Aus Gründen der Sicherheit

Gänzlich andere Überlegungen empfehlen die Trennung der statischen von den dynamischen Daten. So werden die Daten im Verzeichnis `/var` auf einem als Server eingesetzten System binnen kürzester Zeit expandieren. Und die Heimatverzeichnisse der Benutzer vom eigentlichen System zu trennen, ist immer eine gute Idee. Letzteres ermöglicht eine spätere Aktualisierung des Systems ohne die privaten Daten sichern zu müssen und es gibt noch einen wichtigeren Grund...

Angenommen wir haben eine simple Installation vorgenommen und die Platte nur in die Bereiche für den Swap-Speicher und eine Partition für den Rest aufgeteilt. Viel freier Speicher ist nicht mehr übrig geblieben, dafür laden sich die Nutzer fleißig die dicksten Movies aus dem Netz... Nicht mehr lange und der Vorrat an Speicherkapazität ist erschöpft.

Was wird passieren? Niemand wird mehr vernünftig mit dem System arbeiten können, da kein Prozess mehr vermag, seine Statusdaten abzulegen. Der Systemverwalter will sich anmelden und per Hand Platz schaffen... und scheitert, da nicht mal ihm Ressourcen zur Verfügung stehen. Unter Umständen muss das System neu gestartet werden.

Es geht noch schneller

Was hier nicht erwähnt wurde... ist die Möglichkeit des Einsatzes von Raid-Systemen (*Redundant Array of Independent Disks*), wobei mehrere (baugleiche) Festplatten zu einer Einheit zusammengefasst werden. Verschiedene Raid-Verfahren zielen hierbei sowohl auf eine erhöhte Datensicherheit (Redundanz) als auch auf einen beschleunigten Zugriff auf die Daten ab (indem zusammen gehörige Daten über die Platten verteilt gespeichert werden). Raid ist eher im Serverbereich von Interesse...

Keine Partition mehr frei?

Man muss kein Prophet sein, um vorauszusehen, dass die meisten Linuxneulinge bereits Erfahrung im Umgang mit Windows haben. Und man wird das gewohnte Betriebssystem nicht gleich über Bord werfen, ohne das Neue auf Herz und Nieren geprüft zu haben.

Was also tun, wenn man Linux installieren möchte, die Festplatten aber mit dem alten System vollgestopft sind?

Die von manchen Distributoren angebotene Live-CD zu verwenden, taugt wirklich nur für den ersten Kontakt. Linux eine neue Platte spendieren, scheitert meist an den Kosten. Bleibt nur, die existierenden Festplatten - sofern noch Speicherkapazität verfügbar ist - umzupartitionieren und somit Linux den notwendigen Bereich einzuräumen.

Eine bestehende Partition frei räumen

Belegt das bestehende System mehrere Partitionen (unter Windows spricht man von Laufwerken), so ist es einen Versuch wert, durch Umkopieren der Dateien eine Partition frei zu räumen und auf diese Weise Platz zu gewinnen. Sind alle Daten gesichert (Vorsicht bei versteckten Dateien!), steht nun ein Bereich für Linux zur Verfügung.

Auch in diesem Fall wird man um eine nachfolgende Umpartitionierung nicht herum kommen, da Linux erst mit mindestens zwei Partitionen (die zweite ist für den Swap) effektiv arbeiten kann.

Befindet sich die soeben bereinigte Partition innerhalb einer erweiterten Partition, so ist ihre Teilung in zwei Partitionen leicht möglich. Hat man neben dieser Partition noch einen weiteren Eintrag für eine primäre Partition zur Verfügung oder falls noch keine erweitere Partition existiert, ist das Einrichten zweier neuer Partitionen ebenso machbar.

Eine bestehende Partition verkleinern

Voraussetzung ist natürlich, dass auf einer solchen Partition wirklich noch ausreichend freier Speicherplatz vorhanden ist (mind. 500MB für eine **schlanke** Linux-Installation).

Als **freies Werkzeug** liegt den meisten Distributionen das DOS-Programm **fips.exe** bei. Das Programm versucht durch **Verkleinerung der letzten Partition (muss DOS oder Windows sein!) Platz für eine weitere Partition zu schaffen. Zu empfehlen ist die vorherige Defragmentierung dieser Partition, da »fips.exe« nur den ungenutzten Speicherbereich am Ende der Partition verwenden kann.**

Manchen Distributionen liegt eine Variante des bekannten PartitionMagic bei. Dieses »Linux Prep Tool« ist speziell auf die Belange der Installation von Linux zugeschnitten, falls schon ein anderes Betriebssystem auf dem Rechner residiert.

Zur Verwendung des Werkzeuges ist eine bestehende Windows-Installation (95,98,NT) notwendig, die zum Einrichten des PartitionMagic auf Festplatte (95,98) bzw. auf Diskette (NT) dient. Nach erneutem Booten startet automatisch die grafische Oberfläche zum Erzeugen einer Linux-Partition.

Im Unterschied zu »fips« ist »Linux Prep Tool« in der Lage, eine beliebige Partition zu verkleinern (immer vorausgesetzt, es ist auch genügend freier Speicherplatz vorhanden).

Partitionieren mit fdisk

Während des Installationsvorganges der einzelnen Distributionen werden Sie meist mit einer um eine grafische Oberfläche erweiterten Fassung von fdisk konfrontiert. Die damit verbundene Änderung der Bedienung ist in den Abschnitten zur Installation enthalten. fdisk kann auch außerhalb des eigentlichen Installationsprozesses nützlich sein, z.B. bei der nachträglichen Installation einer weiteren Festplatte.

Das Standardwerkzeug zum Partitionieren von Festplatten (und sogar Disketten) unter Linux ist fdisk. Zwar existiert unter DOS/ Windows ein gleichnamiges Programm, jedoch ist die Linux-Version wesentlich liberaler und unterstützt nahezu alle Typen von Partitionen (selbst die von Windows...).

Das Partitionieren ist ein tiefer Eingriff in Ihr System. Mit unüberlegten Handlungen könnten Ihre Dateien unwiderruflich verloren sein! Zwar ist es möglich mit dem Befehl

```
root@sonne> dd if=/dev/hda of=/dev/fd0 bs=512 count=1
```

die alte Partitionstabelle (im Beispiel der ersten IDE-Festplatte) auf Diskette zu sichern und diese später durch Vertauschen von In- und Outfile (if=/dev/fd0 bzw. of=.../dev/hda) zurück zu spielen, jedoch macht dies nur Sinn, solange Sie keine Formatierung der neuen Partitionen vorgenommen haben. Ein Backup der wichtigsten Daten sollte einer Umpartitionierung stets voraus gehen!

Wird fdisk ohne Argumente aufgerufen, so verwendet es die erste Festplatte im System, möchten Sie andere Festplatten/ Disketten partitionieren, ist die Angabe des entsprechenden Devices obligatorisch (manche Versionen von fdisk erfordern immer die Angabe der Partition).

Die interessanteste Option des Kommandos ist -l, die die Ausgabe der Partitionstabelle der angegebenen oder aller Festplatten bewirkt. Bei Verzicht auf Optionen startet fdisk im interaktiven Modus, wo sich (kurioserweise???) hinter dem Befehl m ein Hilfebildschirm verbirgt:

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

Befehl (m für Hilfe): m
Befehl  Bedeutung
   a    (De)Aktivieren des bootbar-Flags
   b    bsd disklabel« bearbeiten
   c    (De)Aktivieren des DOS Kompatibilitätsflags
   d    Eine Partition löschen
   l    Die bekannten Dateisystemtypen anzeigen
   m    Dieses Menü anzeigen
   n    Eine neue Partition anlegen
   o    Eine neue leere DOS Partitionstabelle anlegen
   p    Die Partitionstabelle anzeigen
   q    Ende ohne Speichern der Änderungen
   s    Einen neuen leeren »Sun disklabel« anlegen
   t    Den Dateisystemtyp einer Partition ändern
   u    Die Einheit für die Anzeige/Eingabe ändern
   v    Die Partitionstabelle überprüfen
   w    Die Tabelle auf die Festplatte schreiben und das Programm beenden
   x    Zusätzliche Funktionen (nur für Experten)
Befehl (m für Hilfe):
```

Bevor Sie Änderungen vornehmen, sollten Sie sich die bisherige Tabelle mit Hilfe des Kommandos p betrachten:

```
Kommando (m für Hilfe): p
Festplatte /dev/hdb: 15 Köpfe, 62 Sektoren, 899 Zylinder
Einheiten: Zylinder mit 930 * 512 Bytes
   Gerät    Booten   Anfang    Ende     Blöcke  ID Dateisystemtyp
/dev/hdb1              1        23      10664   83 Linux
/dev/hdb2             24       305     131130   82 Linux Swap
/dev/hdb3            306       899     276210   83 Linux
Kommando (m für Hilfe):
```

Eine Partition löschen Sie mit dem Kommando d

```
Kommando (m für Hilfe): d
Partitionsnummer (1-4): 3

Kommando (m für Hilfe):
```

Eine erweiterte Partition kann erst entfernt werden, wenn zuvor alle enthaltenen logischen Partitionen entfernt wurden.

Mit n wird eine neue Partition erzeugt. Sie werden aufgefordert, zwischen erweiterter, logischer oder primärer Partition (soweit vorhanden) zu wählen. Eine logische Partition kann nur angelegt werden, wenn bereits eine erweiterte Partition existiert, ebenso kann es nur eine erweiterte Partition geben und die Summe erweiterter und primärer Partitionen darf 4 nicht übersteigen. Im nachfolgenden Beispiel legen wir eine erweiterte und in dieser zwei logische Partitionen an:

```
Kommando (m für Hilfe): n
Kommando  Aktion
   e      Erweiterte
   p      Primäre Partition (1-4)
e
Partitionsnummer (1-4): 3
Erster Zylinder (306-899) [Standardwert: 306]:
Benutze den Standardwert 306
Letzter Zylinder oder +Größe, +GrößeK oder +GrößeM (306-899) [Standardwert: 899]:
Benutze den Standardwert 899

Kommando (m für Hilfe): n
Kommando  Aktion
   l      Logische Partition (5 oder größer)
   p      Primäre Partition (1-4)
l
Erster Zylinder (306-899) [Standardwert: 306]:
Benutze den Standardwert 306
Letzter Zylinder oder +Größe, +GrößeK oder +GrößeM (306-899) [Standardwert: 899]: +100M

Kommando (m für Hilfe): n
Kommando  Aktion
   l      Logische Partition (5 oder größer)
   p      Primäre Partition (1-4)
l
Erster Zylinder (507-899) [Standardwert: 507]:
Benutze den Standardwert 507
Letzter Zylinder oder +Größe, +GrößeK oder +GrößeM (507-899) [Standardwert: 899]:
Benutze den Standardwert 899

Kommando (m für Hilfe):
```

Nachdem Erzeugen einer neuen Partition ist dessen Typ immer Linux (ext2). Ein anderer Typ muss explizit über das Kommando t gesetzt werden.

```
Kommando (m für Hilfe): t
Partitionsnummer (1-6): 5
Hex code (L um eine Liste anzuzeigen): 93
Der Dateisystemtyp der Partition 5 ist nun 93 (Amoeba)

Kommando (m für Hilfe):
```

Die Angabe des Partitionstyps erfolgt hexadezimal, eine Liste der bekannten Partitionen zeigt das Kommando L an.

Manche Betriebssysteme (u.a. DOS) benötigen zum Start von einer Partition eine spezielle Kennung dieser, das bootable Flag. Ein solches kann mit b gesetzt werden.

Um die Partitionstabelle auf die Festplatte zu schreiben und das Programm zu beenden, ist das Kommando w einzugeben, zum Beenden ohne Speichern kann q gewählt werden.

Das soeben besprochene Verfahren wird Ihnen in ähnlicher Form während der Installation eines Linuxsystems begegnen. Als nächster Schritt folgt dem Partitionieren das Formatieren, also das Anlegen eines Dateisystems auf dieser Partition. Im Laufe einer Installation geschieht dies jedoch für den Nutzer transparent, so dass wir diese Problematik erst im Abschnitt Dateisysteme der Systemadministration besprechen werden.

Installation Debian

Übersicht

Wer als Neuling das erste Mal den Einstieg in Linux angeht, der sei vor der Debian-Distribution gewarnt. Wem Begriffe wie Partitionen, Swap und Mountpoint nicht geläufig sind, dem sei, hat er die Wahl, dringend zu einer »anfängerfreundlichen« Distribution geraten. Auch ist die rein englischsprachige Menüführung nicht jedermanns Sache...

Zielgruppe des Debian-Projekts ist eindeutig der erfahrene Linuxer. Die Kenntnis der Hardwarekonfiguration ist zwingend erforderlich. Wer sich allerdings durch den Dschungel der Installation und Konfiguration hindurch gefunden hat, der erhält ein durchdachtes System mit höchster Sicherheit.

Warum wir dann dennoch der Distribution einen solchen Stellenwert in unserem Buch einräumen? Weil Debian eine der ausgereiftesten Zusammenstellungen an freier Software darstellt. Sie ist die Distribution mit dem umfangreichsten Angebot an GPL-Software, sie ist sauber strukturiert, ausgiebig getestet und zu einhundert Prozent manuell konfigurierbar.

Sie ist nicht selten die von Puristen favorisierte Distribution.

Und nicht zuletzt weichen Installation und Konfiguration stark von den Vorgehen bei RedHat und SuSE ab.

Erstes Booten

Bei neuerer Hardware sollte das Starten von CDROM unterstützt werden. Legen Sie also die erste **CDROM** der Debian Distribution ein und legen im BIOS ihres Rechners die Bootreihenfolge entsprechend fest.

Sie können ebenso eine **Diskette** (1.2 oder 1.44 MByte) verwenden, falls obiges Vorgehen scheitert. Kopieren Sie dazu die Datei **rescue.bin** (sie befindet sich auf der CDROM im Verzeichnis dists/stable/main/disks-i386/current/disk-1.44) auf eine Diskette. Verwenden Sie hierzu ein Low-Level-Kopierwerkzeug, wie »dd« unter Unix oder »rawrite.exe« unter DOS/Windows. Letzteres Programm finden Sie ebenso auf der CD.

Als weitere Alternative können Sie das System auch von DOS/Windows aus booten, indem Sie das Skript »start.bat« aus dem Verzeichnis »install« der CDROM starten. Dieses Skript ruft implizit das Programm »loadlin.exe« auf.

Lassen Sie sich von den Fehlermeldungen, die vermutlich während des Installationsvorganges hin und wieder über den Bildschirm flimmern, nicht entmutigen. In den meisten Fällen handelt es sich um die Statusausgaben fehlgeschlagener Tests und sind Bestandteil des Installtionsprozesses. Ein versierter Linuxer vermag aus ihnen seine Schlüsse ziehen...

Nach dem Booten begrüßt Sie der Bildschirm aus Abbildung 1, den Sie mittels [Enter] verlassen.

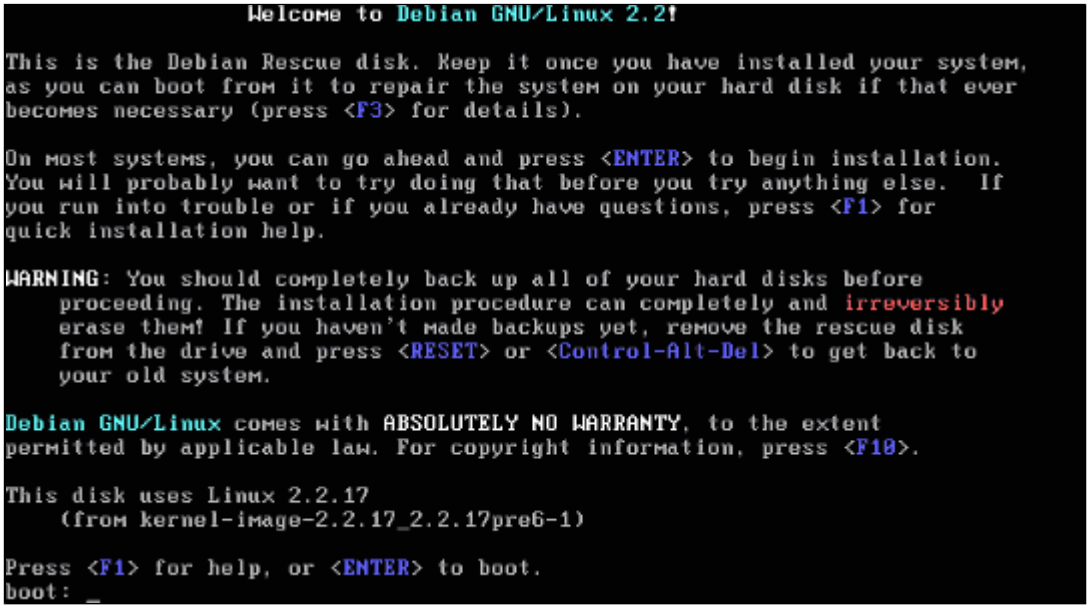


Abbildung 1: Debian 2.2 Begrüßungsbildschirm

Nach dem Hochfahren des Linuxkernels startet die Oberfläche des eigentlichen Installationsprogrammes (Abbildung 2).

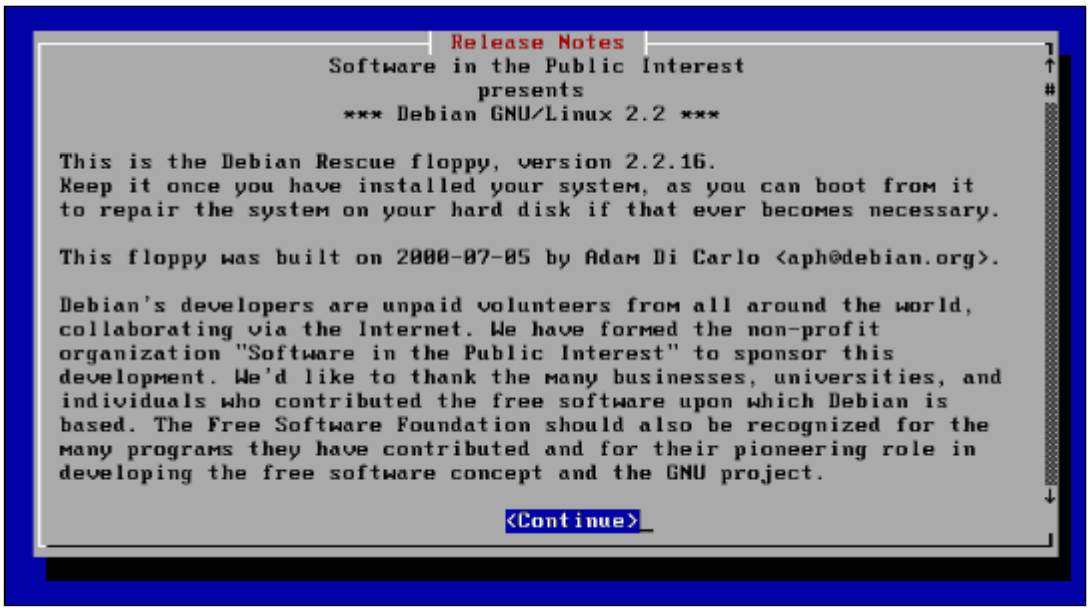


Abbildung 2: Debian 2.2 Installationsprogramm

Die nachfolgenden Eingabemasken können Sie von jedem Punkt der Installation aus wiederholt aufrufen. Zahlreiche Einstellungen lassen sich zudem ebenso zu einem späteren Zeitpunkt vornehmen oder korrigieren.

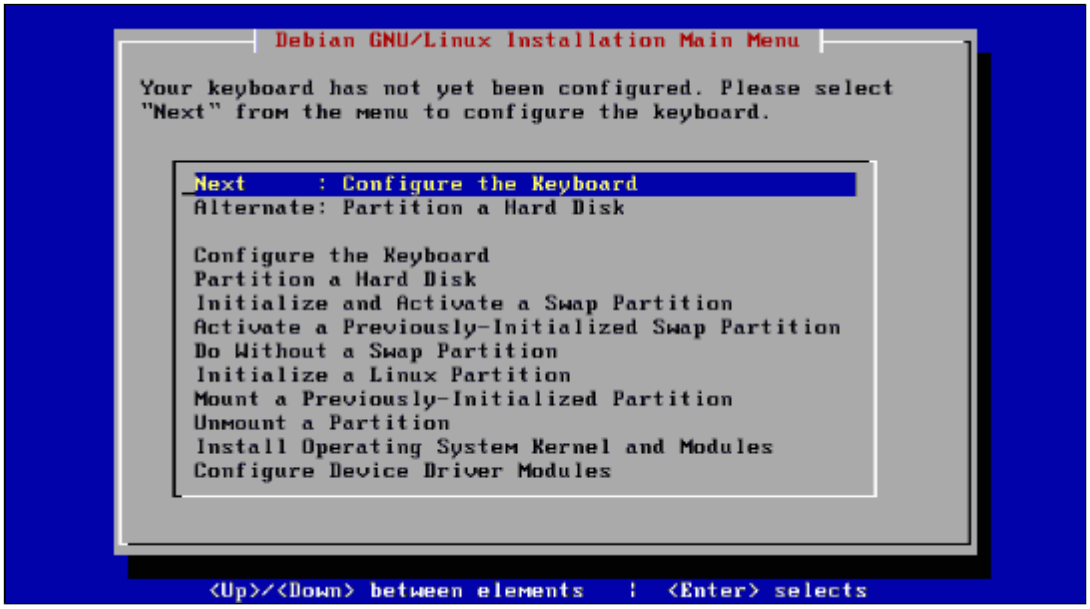


Abbildung 3: Hauptmaske des Installationsprogramms

Die voreingestellten Werte der einzelnen Entscheidungsmasken sollten Sie übernehmen, falls Sie nicht genau wissen, was Sie mit einem alternativen Wert erreichen. Wir folgen im weiteren Text der Installation in der von Debian vorgeschlagenen Reihenfolge.

Anmerkung: Die weiteren Masken zum Vorschlag des jeweils nächsten Konfigurationsschrittes unterscheiden sich nur geringfügig von der zuletzt gezeigten Abbildung und werden deshalb nicht mit dargestellt.

Schritte zum Basissystem

Tastaturauswahl

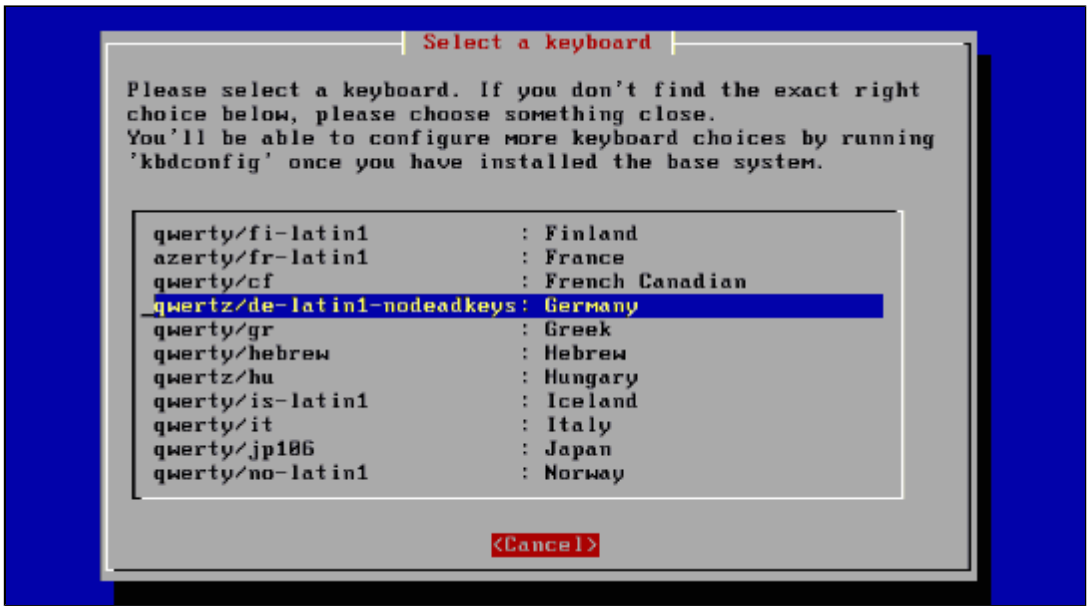


Abbildung 4: Tastaturauswahl

Es geht hierbei um die Festlegung auf eine konkrete Tastaturtabelle. Diese beschreibt, welcher Tastencode beim Kernel welche Reaktion hervorrufen soll. Für deutsche Tastaturen bietet sich »de-latin1-nodeadkeys« an. Bei der späteren Einrichtung der Laufzeitumgebung wird man eventuell einigen Tasten eine abweichende Bedeutung einräumen wollen (bspw. die auf vielen Tastaturen vorhandenen Windows-Tasten mit nützlichen Funktionen belegen). Dies ist Gegenstand des Abschnitts Tastatur im Kapitel Systemverwaltung/Hardwareinstallation.

Mit Bestätigung einer Tastaturtabelle wird diese sofort wirksam.

Partitionierung der Festplatte(n)

Dieser Schritt findet nur bei einer Erstinstallation statt, falls noch keine Linux-Partitionen existieren (sonst wird sogleich zur Festlegung der Mountpoints übergegangen).

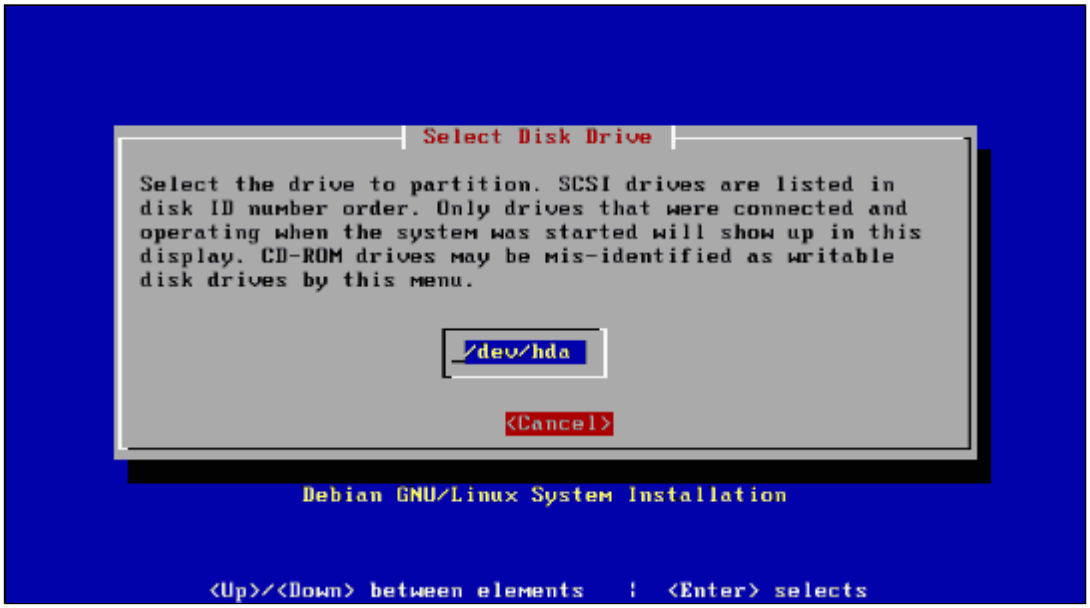


Abbildung 5: Auswahl der zu partitionierenden Festplatte

Die vom Installationsprogramm identifizierten Festplatten ihres Systems werden in einer Auswahlbox dargestellt. Die Namensgebung wurde schon mehrfach im Buch angesprochen, deshalb kurz und bündig: die erste IDE-Festplatte Ihres System wird als "/dev/hda" angesprochen (entspricht unter Windows dem Laufwerk C:). Eine zweite Festplatte hieße "/dev/hdb", die dritte "/dev/hdc" usw. SCSI-Platten werden anhand ihrer ID in das Schema eingereiht; nur nennen diese sich nun "/dev/sda", "/dev/sdb"...

Setzen Sie den Cursor auf die für Linux zu partitionierende Festplatte und bestätigen Ihre Wahl mit [Enter].

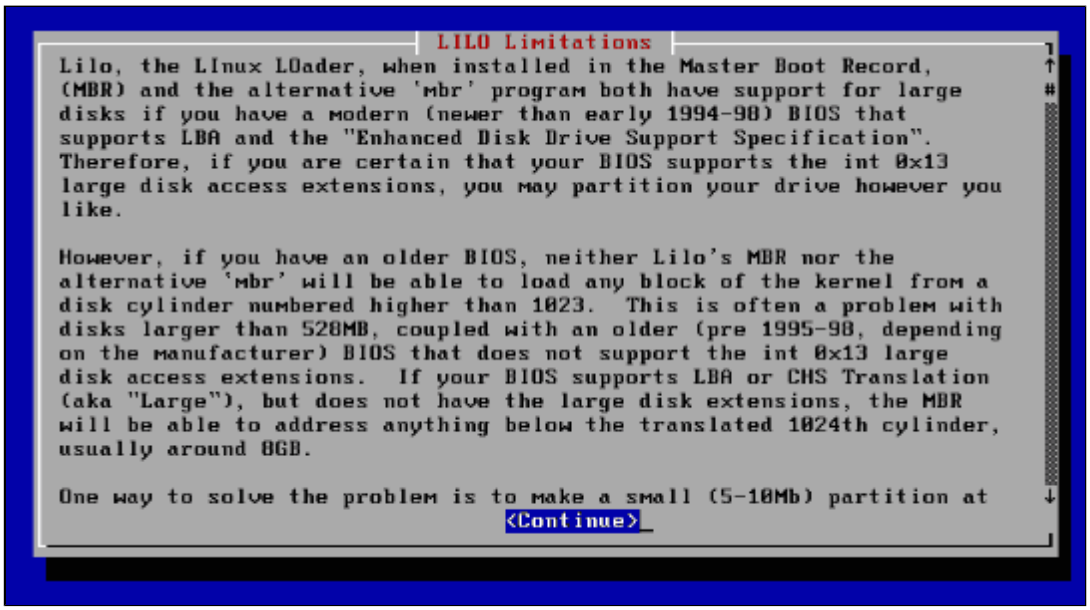


Abbildung 6: Das 1024-Zylinder-Problem lässt grüßen

Der folgende, rein informative Bildschirm erinnert uns nochmals an das so genannte 1024-Zylinder-Problem und der damit verbundenen Einschränkung, dass der Kernel - also das eigentliche Linux - innerhalb der ersten 1023

Zylinder auf der Festplatte liegen muss, um vom Bootmanager (exakt von den Mitteln des BIOS) adressiert werden zu können. Bei neueren Mainboards mit entsprechendem BIOS (Faustregel: ab 1998) besteht diese Restriktion nicht mehr. Sind Sie sich nicht sicher, so versuchen Sie in den anschließenden Schritten der Partitionierung, die erste Linuxpartition so zu positionieren, dass sie vollständig innerhalb der ersten 1024 Zylindern Ihrer Festplatte liegt. Gelingt dies nicht (bspw. weil andere Partitionen zwingend den vorderen Bereich okkupieren), so heißt das Prinzip Hoffnung, dass Ihr BIOS zu den fähigeren seiner Zunft zählt.

Abbildung 7: Partitionieren mit cfdisk

Das von Debian favorisierte Partitionierungswerkzeug ist **cfdisk**. Es unterscheidet sich im Wesentlichen durch eine abweichende Bedienung vom Quasi-Standard-Tool **fdisk**, die Funktionalität ist dieselbe.

Stehen mehrfache Partitionen zur Auswahl, können Sie mittels ⇑ bzw. ⇑ ⇓ zwischen diesen wählen. Die jeweilige Aktion markieren Sie mittels [Tab]. Bevor kritische Aktionen vollzogen werden, müssen Sie diese in einem nochmaligen Dialog bestätigen.

Eine Partition anlegen erfolgt mittels **[New]**. Das Kontextmenü mit diesem Eintrag ist jedoch nur sichtbar, wenn der Cursor sich auf dem mit **Free Space** bezeichneten Eintrag in der Partitionstabelle befindet, d.h. wenn tatsächlich noch Platz für eine neue Partition zur Verfügung steht.

Beim **Erzeugen** einer Partition stehen Sie vor der Wahl, ob diese eine logische oder eine primäre Partition sein soll. (Im Unterschied hierzu muss die erweiterte Partition, die zur Aufnahme logischer Partitionen erforderlich ist, nicht erst explizit angelegt werden. Dies geschieht intern.)
Beachten Sie, dass Sie entweder maximal 4 primäre oder maximal 3 primäre + 15 (SCSI) bzw 63 (IDE) logische Partitionen anlegen können (siehe auch Partitionstabelle (Glossar))!

Des Weiteren müssen Sie die Größe der neuen Partition und ihre relative Lage (unmittelbar nach der vorherigen bzw. unmittelbar vor der nächsten Partition) angeben. Bedenken Sie, dass Sie mindestens 2 Partitionen benötigen (Swap).

Abbildung 8: Partitionstyp setzen

Zu jeder auf diese Weise generierten Partition, die nicht vom Typ *Linux* sein soll, muss nachfolgend deren Typ beschrieben werden. Wählen Sie hierzu den Punkt **[Type]** und suchen in der eingeblendeten Liste die Nummer des entsprechenden Typs. Für *Linux* sollte dies 83 und für *Linux swap* 82 sein.

Haben Sie die Partitionen nach Ihren Bedürfnissen angepasst, können Sie das Schreiben der Partitionstabelle über **[Write]** veranlassen. Alternativ kann der Vorgang über **[Quit]** abgebrochen werden.

Formatieren der Festplatte

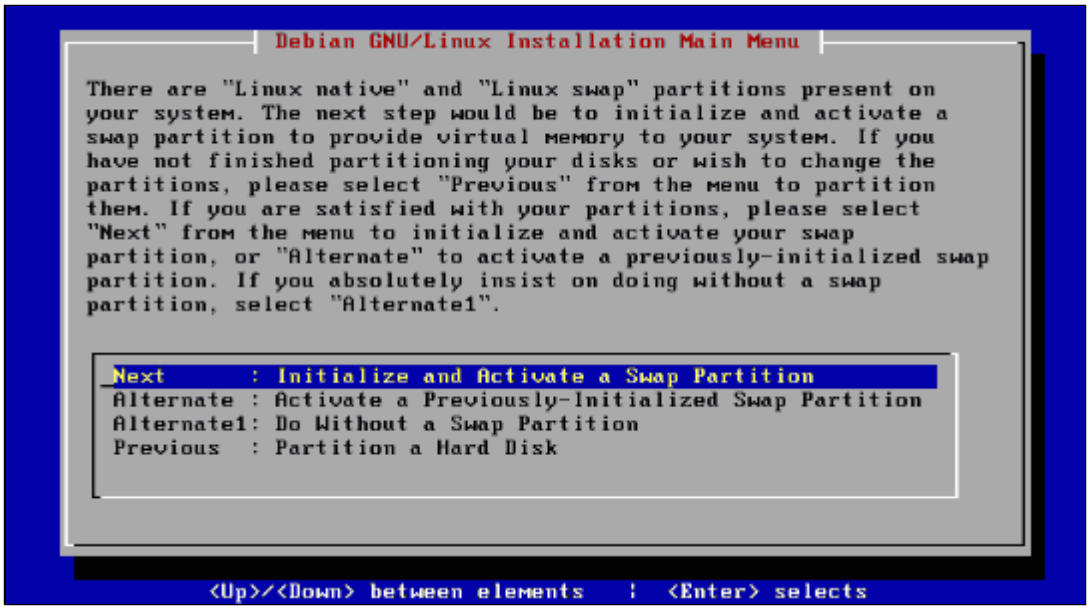


Abbildung 9: Aktivieren der Swap-Partition

Die folgende Maske fordert zur Aktivierung der Swap-Partition auf. Da Sie mindestens eine solche verwenden sollten, ist der Schritt obligatorisch. Die Annahme des Angebots, die Festplatte auf defekte Sektoren hin zu überprüfen, schadet nichts, ist aber sehr zeit intensiv und kann, wenn Sie sich der Unversehrtheit der Platte sicher sind, übersprungen werden, ebenso den folgenden Hinweis, dass alle Daten der Partition durch die Formatierung verloren gehen.

Weiter geht es mit der Formatierung der Linux-Partitionen. Da die Zeiten der Kernelversionen < 2.2 schon ein Weilchen zurück liegen, können Sie auf Kompatibilität mit diesem verzichten. Ihr Linux-Dateisystem wird dadurch

vor allem dünn besetzte Dateien (sparse files, Dateien mit Blöcken, die nur Nullen enthalten) effizienter speichern. Es folgen erneut Abfragen zum Test der Beschaffenheit der Festplatte und die Warnung vor dem endgültigen Überschreiben eventuell vorhandener Daten.

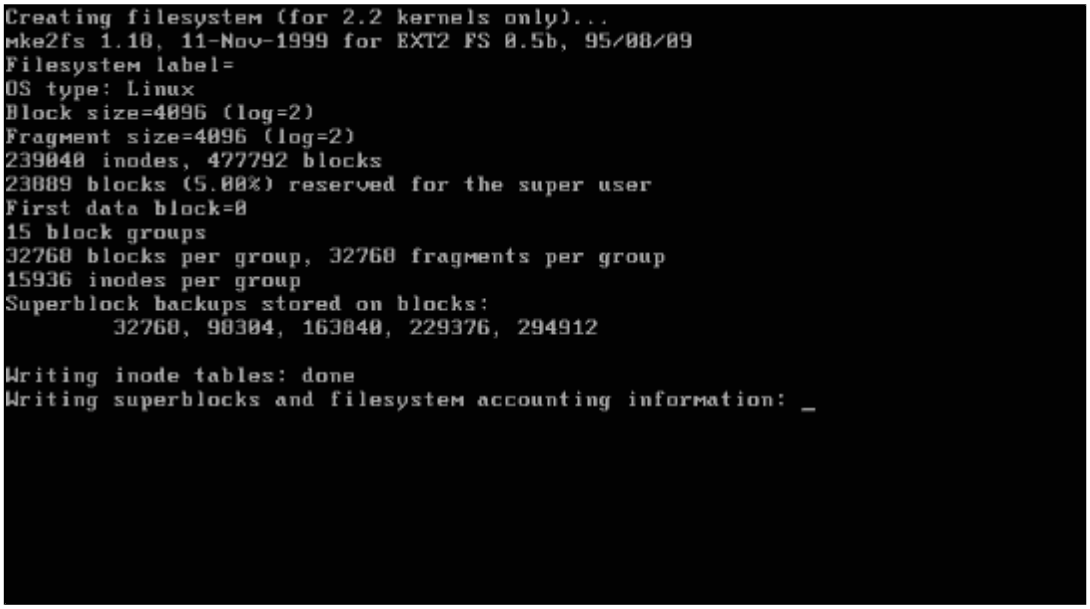


Abbildung 10: Statusausgaben während des Formatierens

Je nach Größe des zu formatierenden Dateisystems kann der Vorgang einige Zeit in Anspruch nehmen. Den jeweiligen Stand der Dinge können Sie anhand der Ausgaben verfolgen.

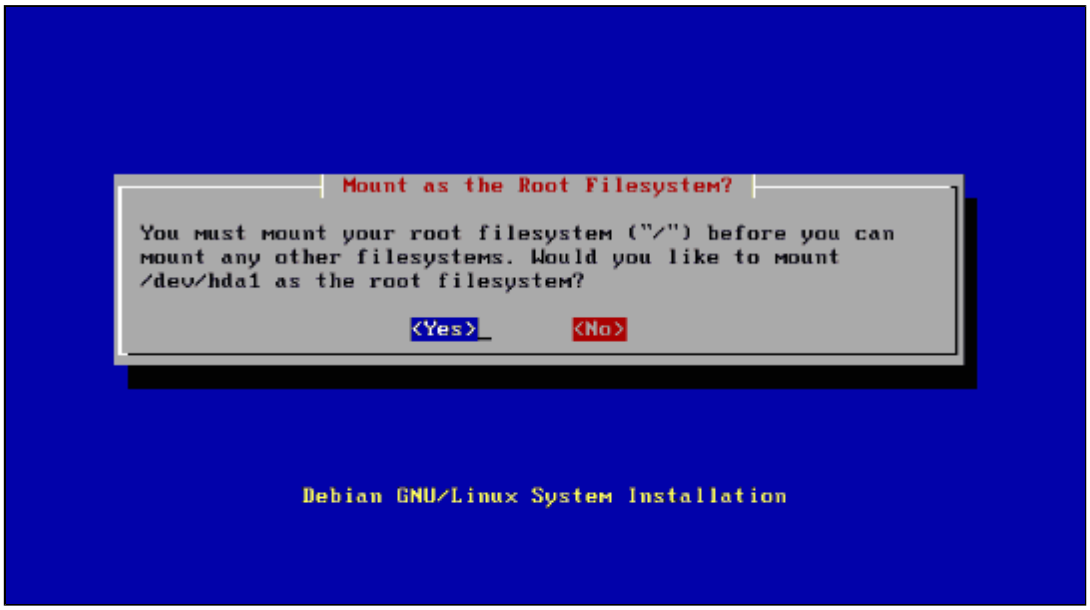


Abbildung 11: Aufforderung zum Mounten der Wurzel

Solange Sie noch keine Partition als Ihre Root-Partition ausgewiesen haben, erscheint nun die Aufforderung, diese Partition als Root zu mounten. Bestätigen Sie den Dialog nur, wenn diese Partition tatsächlich das Wurzelverzeichnis aufnehmen soll

Der letzte Schritt der Formatierung wiederholt sich für jede vorhandene, unformatierte Linuxpartition.

Als nächster Schritt wird die Installierung der Kerneldateien empfohlen. Sie sollten jedoch zunächst alle Linuxpartitionen (sofern Sie mehrere verwenden) den ihnen zugedachten Mountpunkten zuordnen. Prinzipiell ist dies auch später möglich, jedoch wird durch die frühzeitige Aufteilung die sofortige Verteilung der Installation auf diese angeschoben.

Wählen Sie also »Alternate1; Mount a Previously-Initialized Partition«, so erscheint eine Liste der möglichen Partitionen. Neben den lokalen Linuxpartitionen wird auch das Mounten eines NFS-Verzeichnisses angeboten. Dies soll uns hier jedoch nicht interessieren. Die typischen Mountpunkte für eine verteilte Installation werden nachfolgend angeboten. Über »Other...« können Sie ihre eigenen Vorstellungen realisieren.

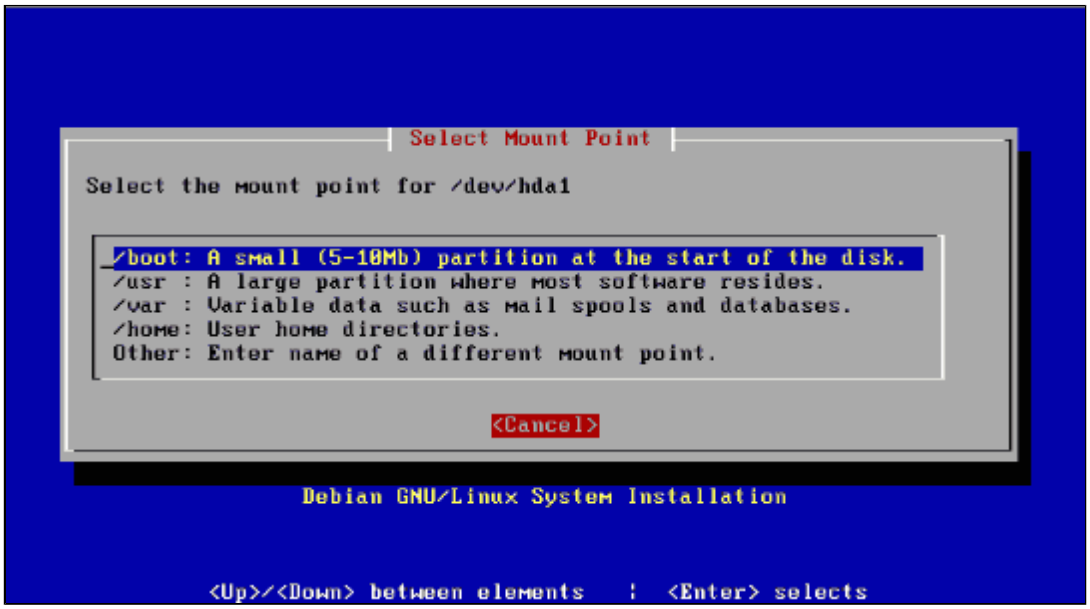


Abbildung 12: Festlegen der Mountpoints

Kernel-Installation

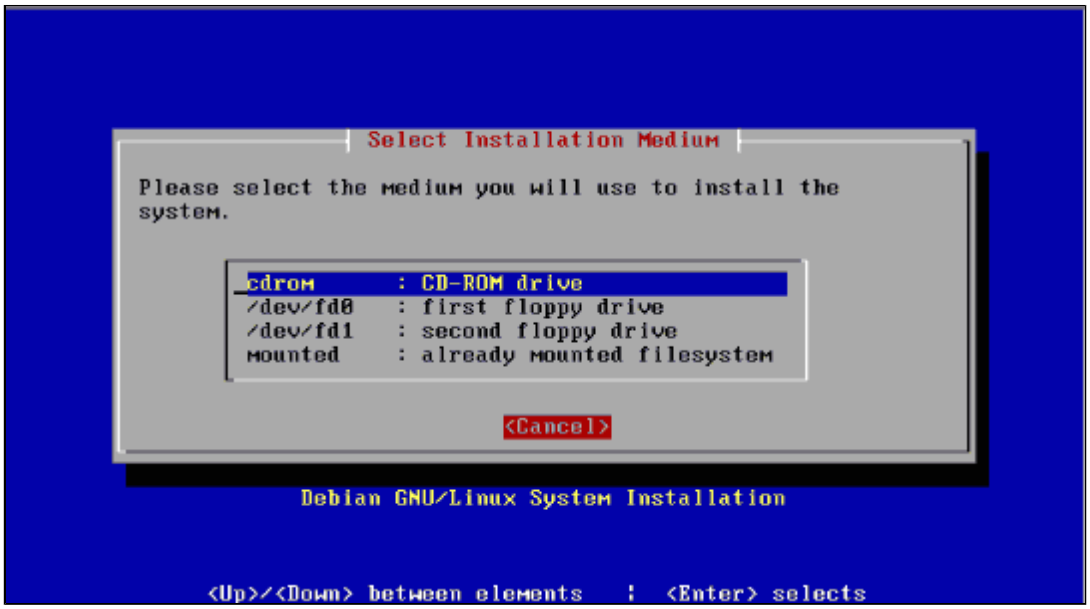


Abbildung 13: Quellmedium des Basissystems

Da Sie vermutlich ohnehin von der CDROM installieren, genügt die Bestätigung der Voreinstellung in den kommenden Masken. Alternativ können Sie einen Kernel (also das eigentliche Linux) und die Module (Erweiterungen des Kernels, die bei Bedarf dynamisch zur Laufzeit hinzugelinkt werden) von einem anderen Medium aus installieren. Denkbar ist, dass Sie eine aktuellere Version bevorzugen, als die, die Debian auf die CDROM presste. Als Medien werden Disketten und Partitionen lokaler Festplatten unterstützt. Des Weiteren müssen Sie den Pfad zu den Kernelpaketen angeben; die Voreinstellung ist bei der Installation von den originalen Medien korrekt.

Optional können Sie im Anschluss weitere Module von anderen Medien installieren.

Treiberkonfiguration

Es geht in diesem Schritt um die Konfiguration Ihrer Hardware. Damit diese funktioniert, müssen die richtigen Treiber mit den richtigen Parametern installiert werden. Diese Treiber liegen in Form von Modulen vor. Module lassen sich auch nachträglich in ein laufendes System integrieren, sodass es nicht tragisch ist, wenn Sie die eine oder andere Einstellung vergessen.

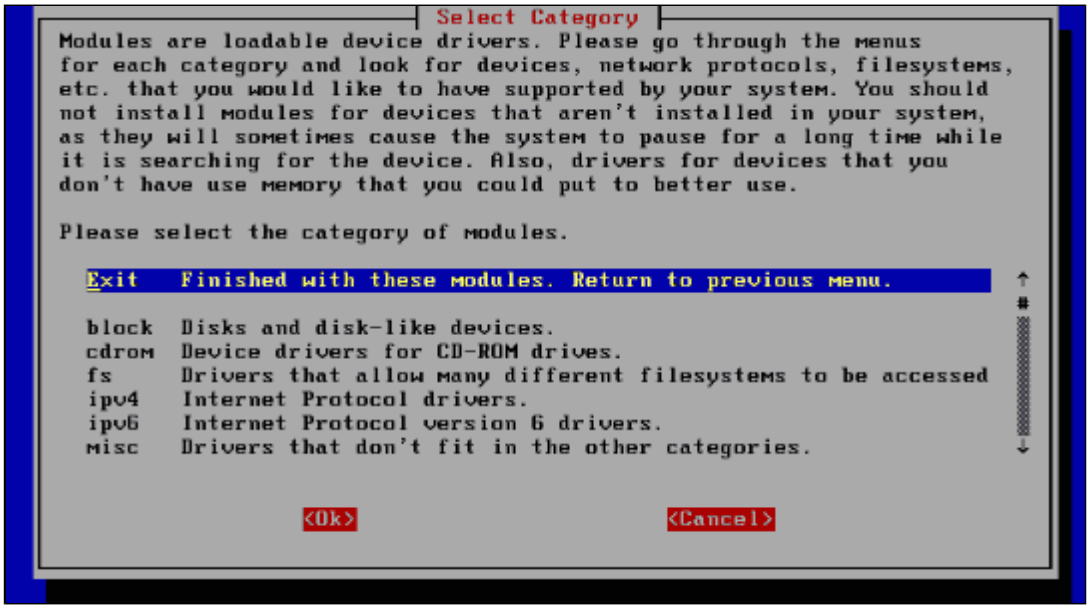


Abbildung 14: Treiberkonfiguration

Die dargebotene Liste gruppiert alle Module nach dem Gerät, das sie bedienen. Und genau hier setzt das Konfigurationsprogramm ein enormes Insiderwissen vom Benutzer voraus. Er muss wissen, welche Hardware im System steckt. Er sollte wissen, welcher Dateisysteme es bedarf. So müssen die Module für NSF installiert werden, wenn der Rechner einmal das Network File System benutzen soll.

Werfen Sie einmal einen Blick in die einzelnen Verzeichnisse. Anhand der Kurzbeschreibung fällt die Entscheidung sicher leichter, was Sie benötigen und was nicht. Auch können Sie gefahrlos die Aufnahme eines Moduls versuchen, von dem Sie nicht wissen, ob es von Ihrer Hardware akzeptiert wird. Bevor das Modul tatsächlich Einzug ins System hält, wird seine Funktion getestet. Scheitert dies, wird das Laden des Moduls abgelehnt. Prinzipiell lässt sich somit die gesamte Hardware einbinden - sofern es einen Treiber dafür gibt -, aber das reine Testen ist mehr als zeitintensiv...

Bei Modulen, die Parameter akzeptieren, können Sie diese angeben. So können einer Netzwerkkarte ein bestimmter Interrupt zugeordnet werden oder einer Soundkarte eine konkrete IO-Adresse und DMA-Kanal (bei ISA-Karten ist eine solche Angabe erforderlich). Hinweise zu solchen Parametern finden Sie u.a im Abschnitt Module des Kernel-Kapitels

Rechnername angeben

Ihr Rechner bedarf eines Namens, anhand dessen er im Netzwerk identifiziert wird. Naja, das ist nicht ganz korrekt, da die IP-Adresse das eigentliche Personaldokument eines Rechners ist. Aber Namen lassen sich dann doch einfacher merken, als kryptische Zahlenkolonnen.

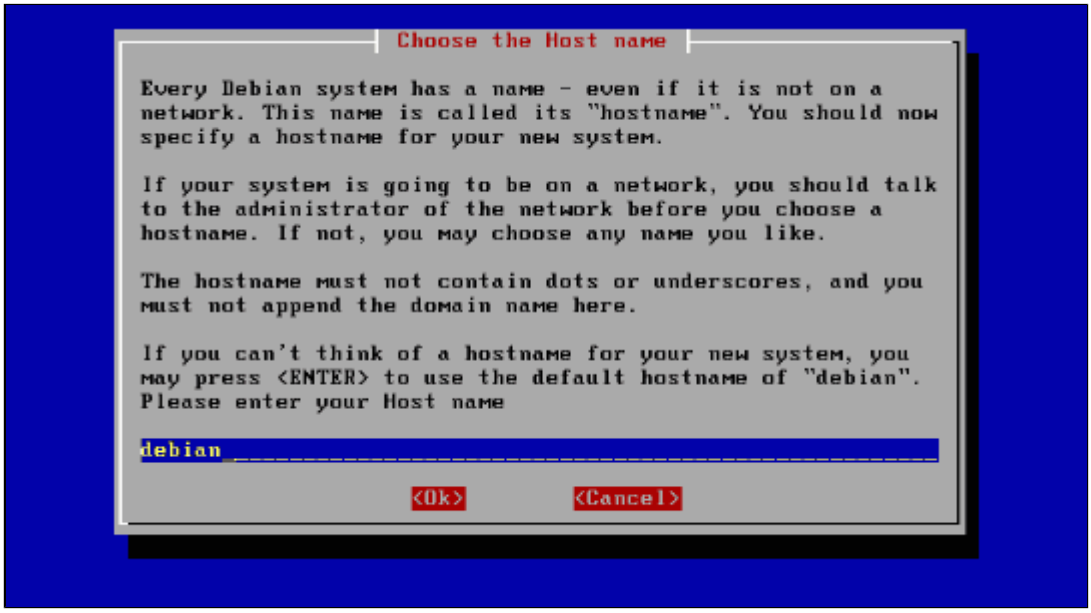


Abbildung 15: Der Rechner benötigt einen Namen

Befindet sich Ihr Rechner in einem lokalen Netzwerk, so sollte der Name innerhalb der Domäne eindeutig sein. Beschränken Sie die Länge des Rechnernamens auf 8 Zeichen (alphanumerisch, der erste muss ein Buchstabe sein), so vermeiden Sie eventuellen Ärger mit einigen älteren Programmen.

Netzwerk-Grundkonfiguration

Nur für den Fall eines zuvor geladenen Netzwerkmoduls erfolgt nun die Konfiguration elementarer Netzwerkparameter.

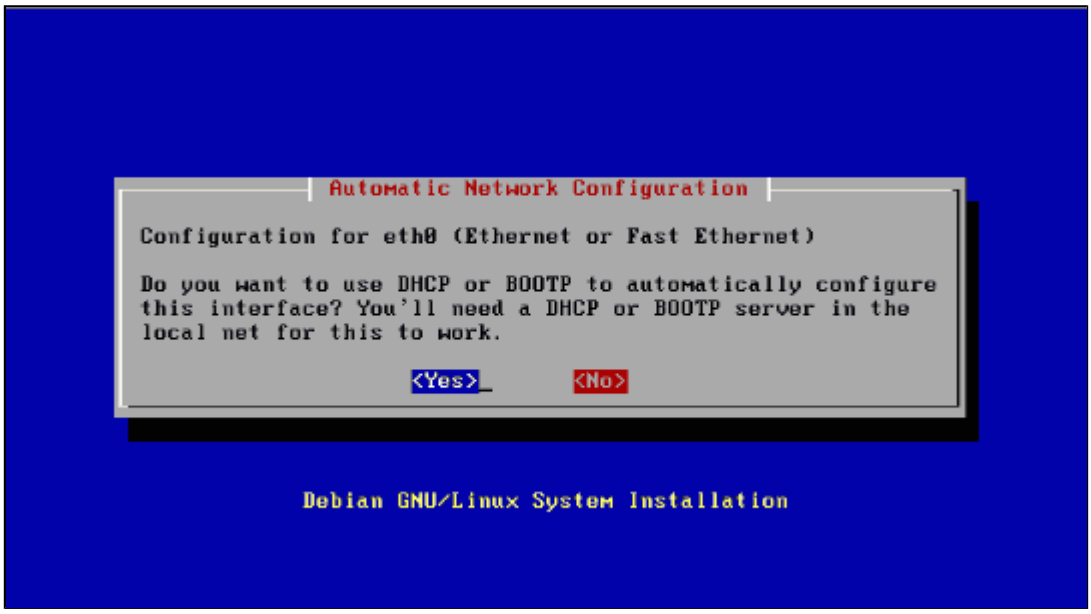


Abbildung 16: Parameter von einem Server beziehen?

Bei DHCP (*Dynamic Host Configuration Protocol*) und BOOTP (*Boot Protocol*) handelt es sich um zwei Netzwerkdienste, die die dynamische Zuweisung von Parametern an einen Rechner unterstützen. Sie sollten diese Maske nur bejahen, wenn sich tatsächlich ein entsprechender Server im Netz für Ihren Rechner zuständig ist. Die nachfolgenden Abfragen entfallen für diesen Fall.

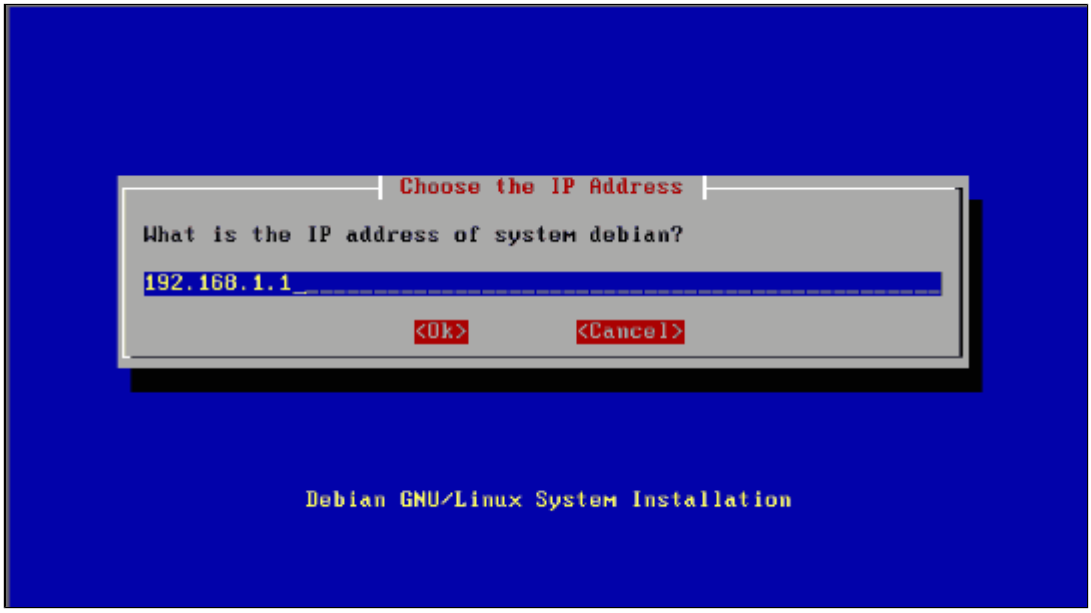


Abbildung 17: Internet Protokoll Adresse

Die IP-Adresse ist die (zumindest im lokalen Netzwerk) eindeutige Anschrift Ihres Rechners. Verfügt Ihr Rechner über keine Verbindung zum Internet, können Sie hier nahezu jede Adresse (auch die voreingestellte) verwenden, insofern sie nicht schon im lokalen Netz vergeben ist. Informieren Sie sich bei weiteren Fragen im Kapitel Netzwerk-Grundlagen.

Auch die folgende Frage nach der Netzmaske (Klasse C) können Sie so übernehmen, falls Sie nicht gerade Subnetting konfigurieren möchten.

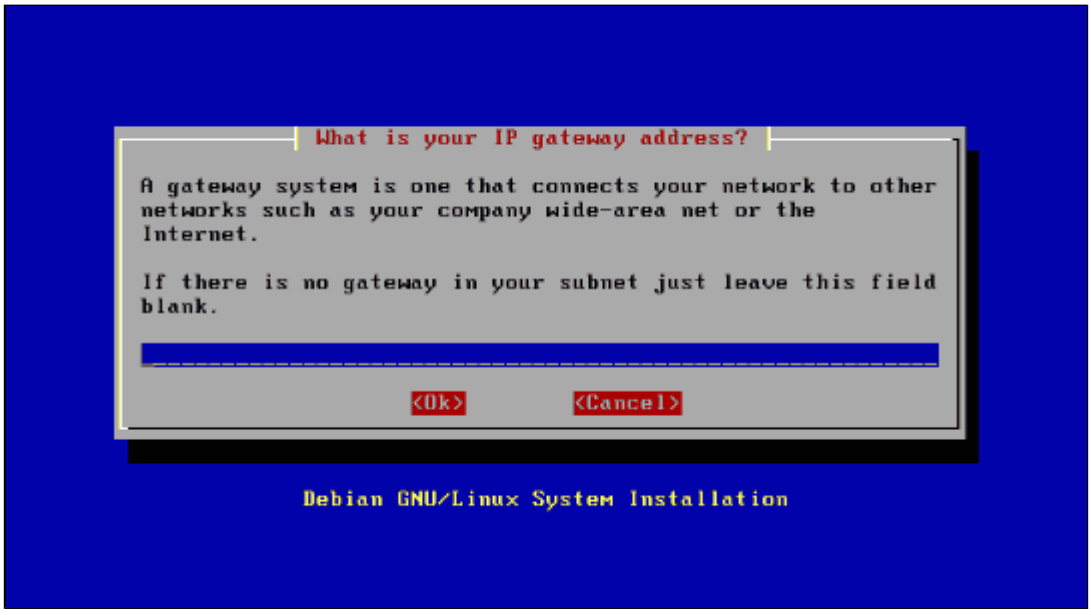


Abbildung 18: Das Gateway

An einen Rechner aus dem lokalen Netzwerk gerichtete Pakete, werden direkt an diesen gesandt. Pakete, für die der eigene Rechner nicht entscheiden kann, wo sich der Zielrechner befindet, werden hingegen an eine zentrale Stelle geleitet - das Gateway -, welche sich um die weitere Vermittlung kümmert. Lassen Sie das Feld frei, wenn in Ihrem Netz kein Gateway existiert.

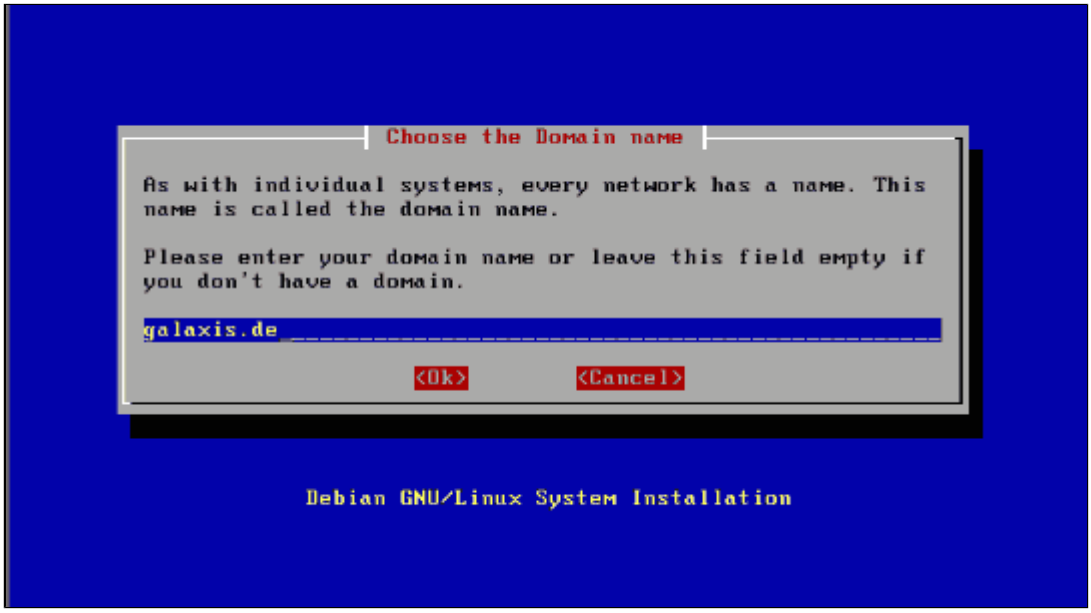


Abbildung 19: Domainname

Ein Domainname beschreibt einen Verwaltungsbereich eines Netzwerks, für einen Einzelplatzrechner oder einen Rechner, der nur per Modem an ein Netz angeschlossen wird, kann die auch Angabe entfallen.

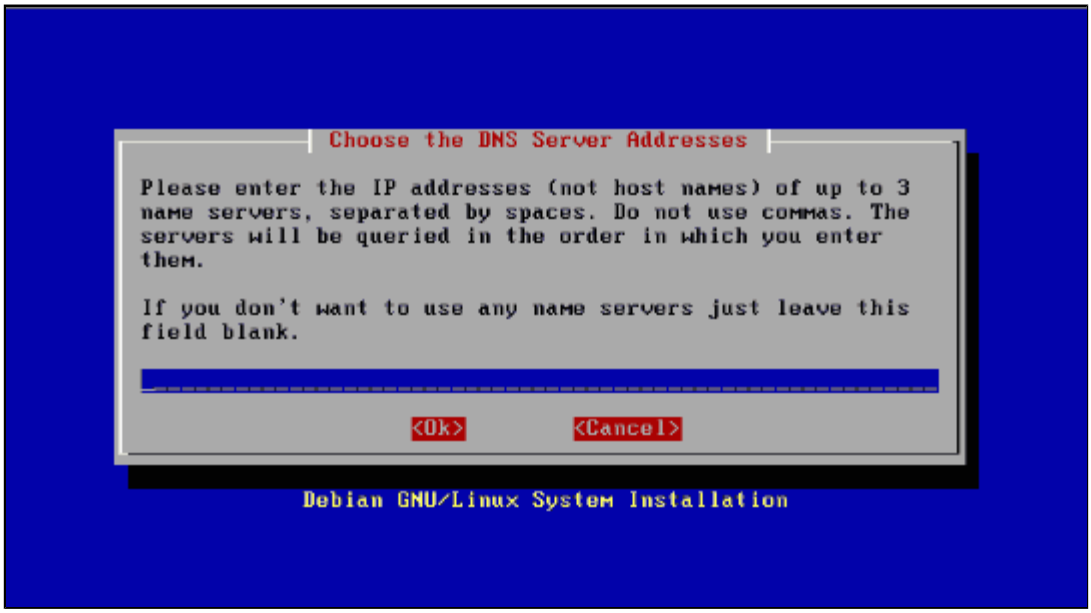


Abbildung 20: Der Nameserver

Wenn Sie auf andere Rechner zugreifen, so werden Sie dies meist über dessen symbolischen Namen tun (bspw. "http://www.linuxfibel.de"). Die Adressierung im Internet läuft aber auf IP-Adressebene ab, d.h. der Name muss in die korrekte IP-Adresse aufgelöst werden. Zuständig dafür ist u.a. (und in großen Netzwerken eigentlich immer) ein so genannter Nameserver, der die gewünschte Information ermittelt. Aus Effizienzgründen sollten Sie maximal drei solcher Server benennen.

Installation des Basissystems

Analog zum Vorgehen beim Kernel werden Sie nun zur Angabe des Installationsmediums, des Verzeichnisses mit den Basispaketen auf diesem und der Archivversion aufgefordert. Für den Fall der CDROM-Installation bestätigen Sie einfach die vorgegebenen Werte.

Einen Einfluss auf den Umfang des Basissystems haben Sie nicht. Hierin unterscheidet sich Debian deutlich von RedHat oder SuSE. Die anschließende Installation kann einige Minuten beanspruchen.

Konfiguration des Basissystems

Abbildung 21: Setzen der Zeitzone

Es wird Zeit, Ihrem System die »richtige« Zeit zu verpassen. Belassen Sie also in der ersten Maske die Zeitzone auf CET (Mitteleuropäische Zeitzone) und selektieren »Europe« aus der linken Maske. Die folgende Auswahl lässt eine detaillierte Konfiguration zu. Für Deutschland ist die durch »Berlin« repräsentierte Zeitzone der empfohlene Wert. Im anschließenden Dialog kann die batteriegepufferte Uhr des Rechners (Hardwareuhr) auf Greenwich Mean Time (GMT) oder lokale Zeit gestellt werden. Für am Internet beteiligte Rechner wird dringend GMT empfohlen, da im Netz sich (fast) alle Rechner anhand der GMT synchronisieren. Ihr System wird Ihnen dennoch die reale Zeit Ihres Standorts anzeigen, da diese anhand der Zeitzone berechnet werden kann.

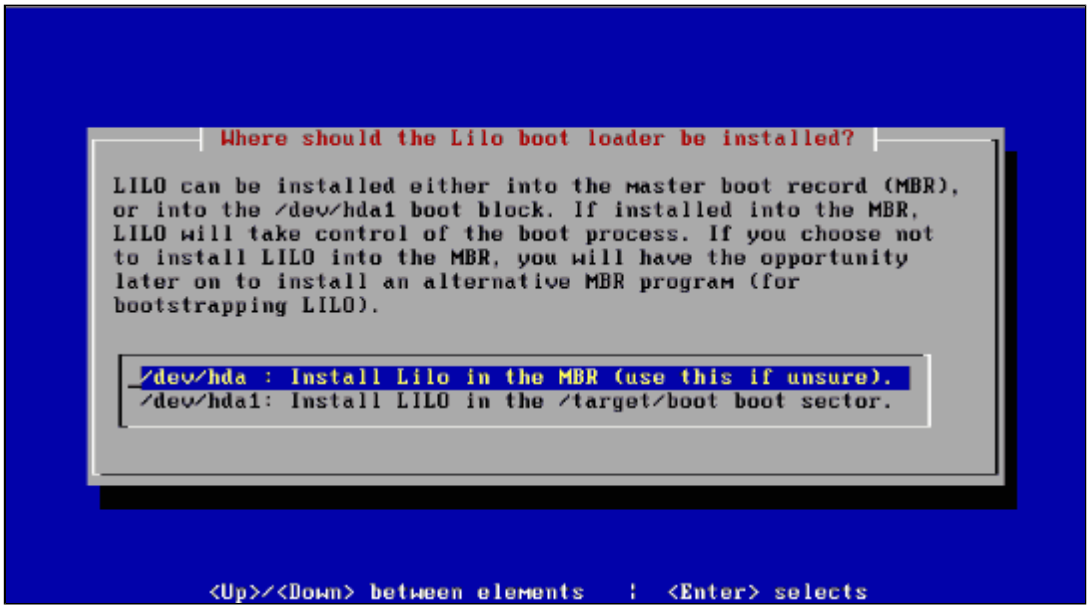


Abbildung 22: Platzierung von Lilo

Der Linux Loader wird zum starten des Systems benötigt. Der übliche Ort dessen Installation ist der Master Boot Record (Mbr), falls Linux das alleinige System auf Ihrem Rechner ist. Residieren mehrere Betriebssystem auf Ihrem Computer, so kann der Bootsektor der /boot-Partition die richtige Wahl sein. Für den Fall müssen Sie einen anderen Bootmanager so konfigurieren, dass er Lilo starten kann. Vergleichen Sie hierzu die Informationen des Abschnitts Bootmanager.

Alternativ lässt sich das System auch über die nachfolgend zu erstellende Rettungsdiskette starten (optional).

Zumindest dem Linux-Neuling kann eine solche Diskette bei fehlgeschlagenen Konfigurationsversuche ein wertvoller Rettungsanker sein.

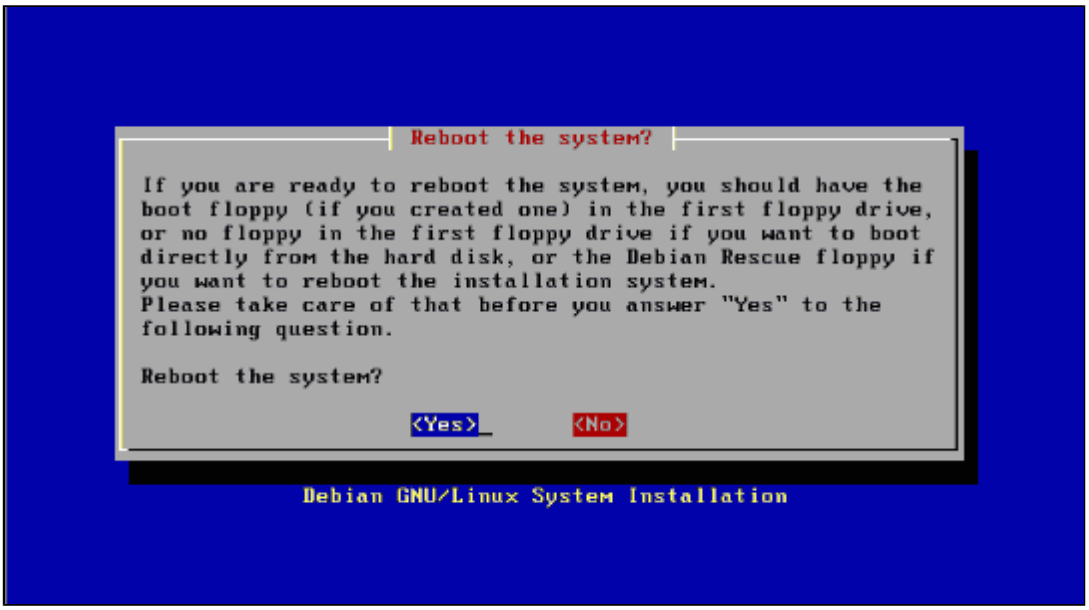


Abbildung 23: System-Neustart

Mit dem Neustart Ihres System schließen Sie die Basisinstallation ab. Vergessen Sie nicht, die Installations-CD zuvor zu entfernen!

Fertigstellen der Installation

Passwort-Sicherheit

Zu Passwörtern und Passwort-Sicherheit finden Sie im Buch noch ausführliche Beiträge, deshalb muss an dieser Stelle eine kurze Anmerkung zu den nächsten Konfigurationsmasken genügen.

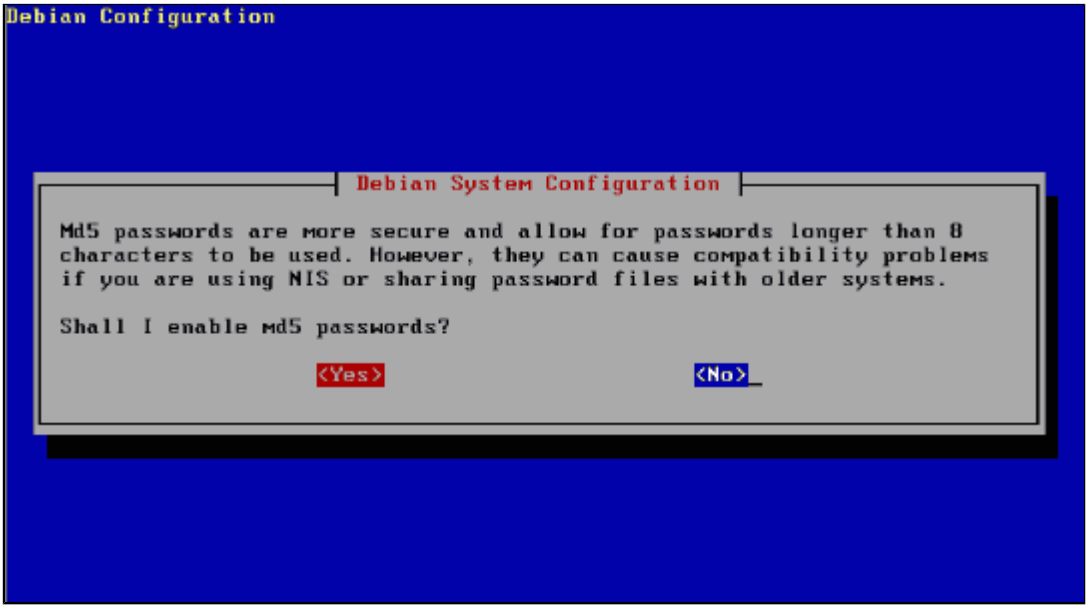


Abbildung 24: Passwort-Sicherheit

MD5 ist ein modernes Fingerprint-Verfahren, das zu jedem Passwort eine zusätzliche Prüfsumme speichert, sodass Manipulationen an einem Passwort erkennbar werden. Allerdings bedarf es einiger zweifelhafter Umstände, dass überhaupt jemand derartigen Zugriff auf die Passwortdatei erlangt. MD5 beschwört auch Probleme mit älteren

Programmen herauf, die eine Authentifizierung verwenden. Verzichten Sie besser auf MD5.

Die »shadow«-Option sollten Sie auf jeden Fall auswählen, den Hintergrund vermittelt der Abschnitt Systemsicherheit - Sichere Passwörter des Kapitels Systemadministration.

Des Weiteren müssen Sie ein Passwort für Root eingeben. Um Tippfehler auszuschließen, werden Sie zur zweimaligen Angabe aufgefordert. Nachfolgend steht die Wahl, einen ersten »normalen« Benutzer im System einzurichten. Sie können dies auch auf später verschieben, aber Sie sollten sich rasch angewöhnen, niemals als Root mit dem System zu arbeiten.

PCMCIA sollten Sie entfernen, wenn Sie es nicht benötigen (Laptop). Es belegt nur unnötig Plattenplatz.

Einen Point-to-Point-Zugang konfigurieren

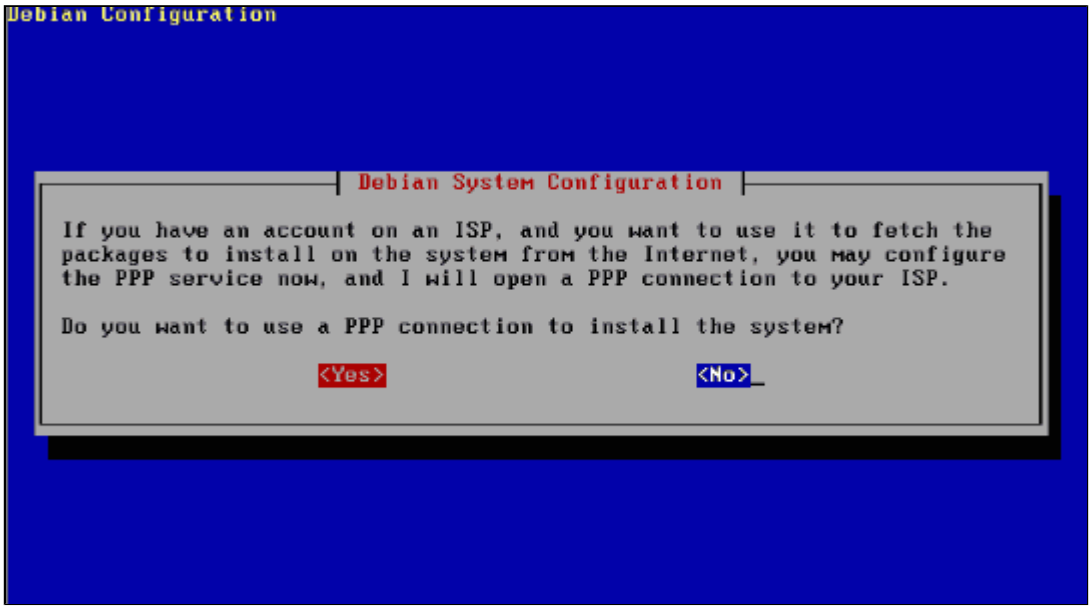


Abbildung 25: Modem-Zugang über PPP

Die Frage nach Konfiguration eines Zugang zu einem Internet Service Provider (ISP) kommt für Sie nur in Frage, wenn Sie über ein Modem verfügen. Da die Konfiguration auch nachträglich mit dem Werkzeug **pppconfig** erfolgen kann, verweisen wir hier auf den entsprechenden Abschnitt im Kapitel Netzwerk-Grundlagen, Initialisierung der Hardware.

Software installieren

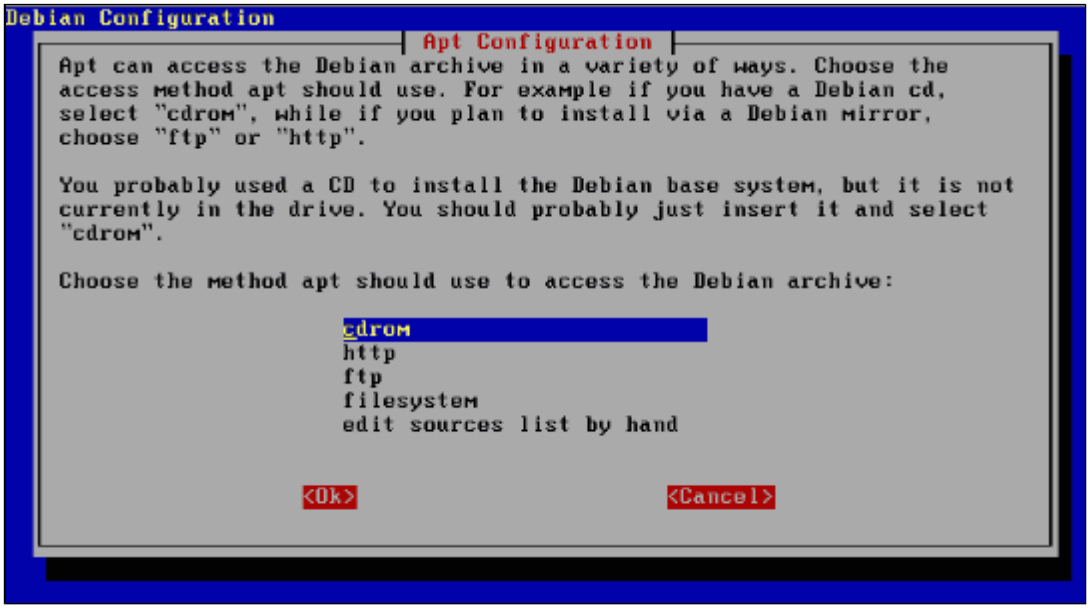


Abbildung 26: Apt-Setup zur Auswahl einer Installationsquelle

Das *Advanced Package Tool* ist ein neuartiges Werkzeug zur Paketverwaltung, das die Installation von Paketen aus unterschiedlichen Quellen komfortabel unterstützt.

Die einfachste - und von uns nachfolgend beschriebene Methode - ist die Installation von CDROM. Im Falle der zuvor erfolgten PPP-Konfiguration besteht ebenso die Wahl, zur Installation weiterer Pakete einen FTP- bzw. HTTP-Server zu kontaktieren (Interessant ist diese Methode sicher nur für den Zugang über eine Standleitung, da sonst die Kosten eines Downloads den Kauf einer CD-Version rechtfertigen).

Abbildung 27: Angabe weiterer Quellen

Apt liest die Paketliste von der angegebenen Quelle ein und erzeugt eine Datei **/ etc/ apt/ sources.list**. In dieser stehen die Quellen, das Paketverzeichnis auf diesen, die Version (bspw. "stable", "unstable",...) und der Typ ("main", "contrib",...), nicht jedoch die Namen der Pakete. Jene werden in einer Datei /var/lib/dpkg/available abgelegt.

```
user@sonne> cat / etc/ apt/ sources.list
# See sources.list(5) for more information, especialy
# Remember that you can only use http, ftp or file URIs
# CDROMs are managed through the apt-cdrom tool.
#deb http://http.us.debian.org/debian stable main contrib non-free
```

```
# deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
# deb http://security.debian.org stable/updates main contrib non-free

#  Uncomment if you want the apt-get source function to work
# deb-src http://http.us.debian.org/debian stable main contrib non-free
# deb-src http://non-us.debian.org/debian-non-US stable non-US

deb cdrom:[Debian GNU/Linux 2.2 r0 _Potato_ - Unofficial i386 Binary-3 (20000821)]/ unstable contrib main non-US/contrib non-US/main non-US/non-free non-free
deb cdrom:[Debian GNU/Linux 2.2 r0 _Potato_ - Unofficial i386 Binary-4 (20000821)]/ unstable contrib main non-US/contrib non-US/main non-US/non-free non-free
deb cdrom:[Debian GNU/Linux 2.2 r0 _Potato_ - Unofficial i386 Binary-2 (20000821)]/ unstable contrib main non-US/contrib non-US/main non-US/non-free non-free
deb cdrom:[Debian GNU/Linux 2.2 r0 _Potato_ - Unofficial i386 Binary-1 (20000821)]/ unstable contrib main non-US/contrib non-US/main non-US/non-free non-free
```

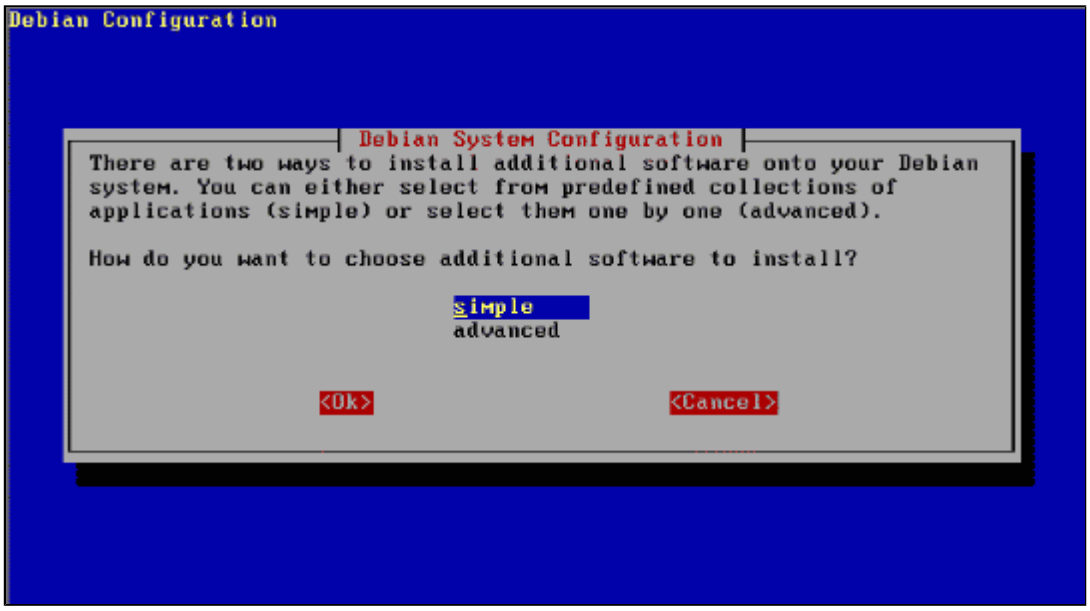


Abbildung 28: Einfache oder detaillierte Paketauswahl?

Endlich geht es an die Auswahl zu installierender Software. Um im Dschungel der mehreren tausend Pakete nicht den Überblick zu verlieren, bietet sich das Werkzeug **tasksel** an, das sich hinter der Alternative **simple** verbirgt. Hier folgt Debian dem Trend der anderen Distributionen, dem Benutzer eine auf den späteren vornehmlichen Verwendungszweck abgestimmte Softwarekonfiguration anzubieten.

Über **advanced** ist dselect direkt erreichbar, womit die Entscheidung zur Installation für jedes Paket einzeln getroffen werden kann. Letzteres Vorgehen eignet sich wohl eher zur nachträglichen (De)Installation von Software oder für Leute, die sich aus reiner langer Weile durch endlose Listen hangeln mögen.

Die Bedienung beider Werkzeuge soll im Anschluss diskutiert werden. Wir fahren mit der Beschreibung der »simplen« Methode fort.

Das sich nun anschließende Einlesen der Paketinformationen kann etwas Zeit in Anspruch nehmen. Nach Abschluss startet tasksel, das die einzelnen Pakete, gruppiert nach ihren Verwendungszweck, auflistet.

Abbildung 29: Paketauswahl

Mit den Cursortasten navigieren Sie zwischen den einzelnen Einträgen. Mit [Leertaste] oder [Enter] markieren Sie einen Eintrag zur Installation. Durch wiederholte Markierung verschwindet diese wieder. Einzelne Pakete einer Gruppe erreichen Sie durch Eingabe von [i]; [q] bringt Sie zurück in die übergeordnete Maske. Nach Selektion der zur Installation erwünschten Pakete, starten Sie den Installationsvorgang durch Eingabe von [f].

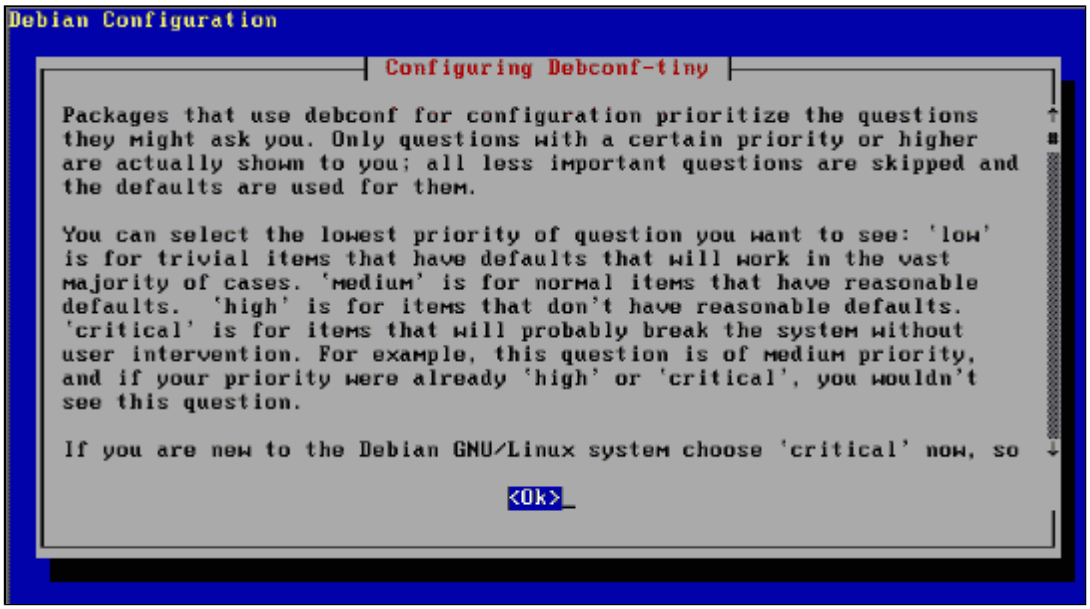


Abbildung 30: Grad der automatisierten Konfiguration

Während der Installation bedürfen zahlreiche Pakete einer Konfiguration. Inwiefern diese Konfiguration automatisch erfolgen soll (mit Standardwerten), können Sie in einem Dialog festlegen. Der Grad des Automatismus reicht von »low«, was sämtliche Entscheidungsgewalt dem Nutzer überträgt bis zu »critical«, wodurch das Installationsprogramm stets nach eigenem Gutdünken handelt.

In Abhängigkeit der zuvor gewählten Option werden Sie nachfolgend zu verschiedenen Eingaben aufgefordert. Bei Installation von mehreren CDs fordert das Installationsprogramm Sie zur rechten Zeit zum Wechsel auf.

Zum Abschluss gibts noch einige informative Ausgaben bevor das Login-Prompt zum Anmelden in Ihr neues Linuxsystem einlädt.

Installation - RedHat-basierende Distributionen

Übersicht

Als weltweiter Marktführer der Linuxdistributoren hat RedHat unterdessen zahlreiche Nachahmer gefunden. Im Unterschied zu bspw. SuSE veröffentlichte RedHat die Quellen zu seinen Eigenentwicklungen. Deshalb findet man Anwendungen wie **linuxconf** oder **sndconfig** auf zahlreichen Distributionen. In Deutschland haben insbesondere Mandrake, Helloween und Go! Linux eine gewisse Bedeutung erlangt. Zumindest die beiden zuletzt genannten unterscheiden sich von ihrem Ursprung nur marginal.

RedHat war ebenso Initiator des grafischen Installationsprozesses. Dennoch steht Ihnen die Wahl, auf den Textmodus zurückzugreifen. Das kann durchaus notwendig werden, wenn Sie bspw. über keine Maus verfügen oder diese vom System nicht erkannt wurde (USB-Mäuse werden erst ab RH 7.0 unterstützt).

Erstes Booten

Nachfolgend beschränkt sich die Beschreibung auf eine typische Installation von einer CDROM. Alternativ unterstützt RedHat das Aufspielen des Systems von Quellen auf der Festplatte, via NFS, per FTP oder HTTP.

In heutigen Zeiten sollte nahezu jeder Rechner das Booten von CDROM unterstützen, deshalb kann die erste Installations-CD als Bootmedium dienen. Wenn dem nicht so ist, können Sie die mitgelieferte Bootdiskette verwenden oder eine solche erzeugen. Das notwendige Image liegt auf der Installations-CD im Verzeichnis /images. Für eine normale Installation kopieren Sie »boot.img« auf eine Diskette, für ein Laptop nutzen Sie »pcmcia.img« und für eine Installation übers Netz ist »netboot.img« die richtige Wahl. Verwenden Sie zum Kopieren unter Linux das Kommando »dd«; unter DOS/Windows müssen Sie auf »rawrite.exe« zurückgreifen (auch dieses Programm liegt auf der CD im Verzeichnis »/dosutils«).

Abbildung 1: RedHat-Startbildschirm

Der Startbildschirm bietet mehrere Optionen an. Durch Eingabe dieser lassen sich alternative Installationspfade beschreiten:

[Enter]

text[Enter]

Die Komplexität der Installation ist analog zur grafischen Methode, sie erfolgt nun textbasiert (Dialogboxen) Dieser Weg kann notwendig werden, wenn bspw. die Grafikkarte oder die Maus nicht erkannt wurden. Die Bedienung des Programms erfolgt nun mittels der Cursortasten der Tastatur (auch die grafische Oberfläche kann per Tastatur bedient werden, was aber etwas gewöhnungsbedürftig ist).

expert[Enter]

Ein merklicher Unterschied ist die abgeschaltete automatische Hardwareerkennung. Der Experte kann hier Module (auch von einer Treiberdiskette) von Hand laden und somit bewusst die Hardwareinstallation und -konfiguration steuern. Wichtig kann ein solches Vorgehen sein, wenn die automatische Erkennung »falsche Schlüsse« zieht, bspw. automatisch einen Treiber lädt, den man besser durch einen neueren ersetzt hätte d durch die Reihenfolge der Initialisierung Ressourcen einer Hardwarekomponente zugesprochen wurden, die zwingend eine andere Komponente benötigt.

text expert[Enter]

Wie **expert**, nur textbasiert.

Wir betrachten nachfolgend die einfache, grafikbasierte Installation. Die Aussagen gelten unverändert für den Textmodus.

Einstellungen

Zunächst geht es an einige die Installation betreffende Einstellungen. Die Sprache der Installation ist im ersten Dialog konfigurierbar. Eine geänderte Auswahl wird jedoch erst im nächsten Bildschirm wirksam.

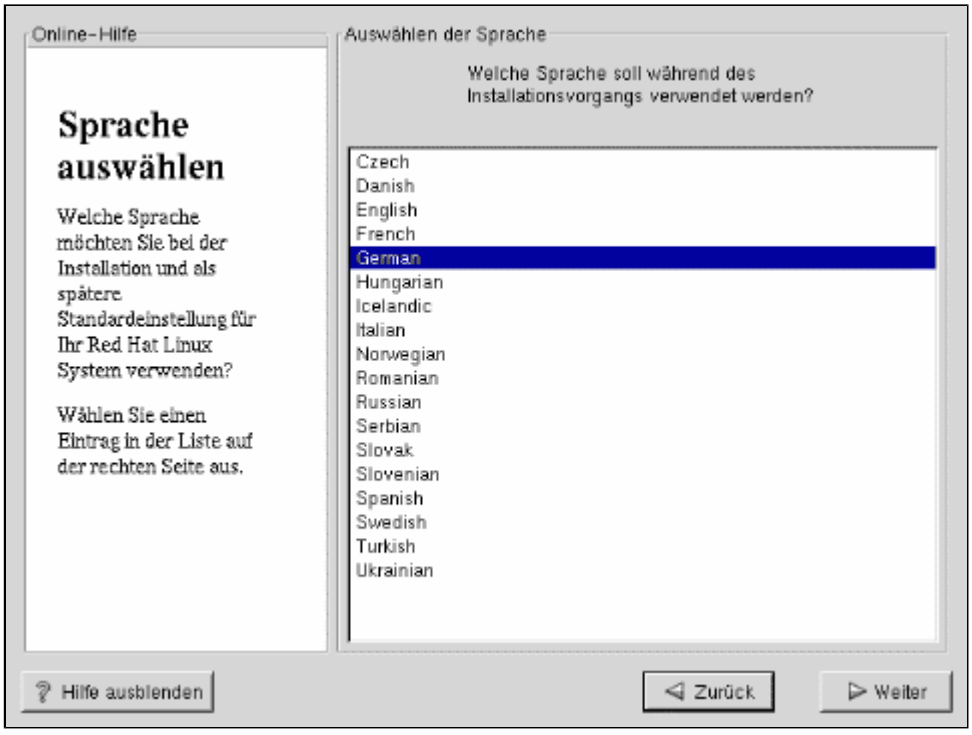


Abbildung 2: Sprachauswahl

Die Modelle in der Tastaturauswahl werden meist anhand der Tastenanzahl unterschieden. Falls Ihr Tastaturmodell nicht namentlich erwähnt ist, so ersparen Sie sich das Zählen. Verfügt Ihre Tastatur über die Windowstasten, so

sind 104-Tasten eine weise Wahl. Sonst sollte es das 101-Tasten-Layout tun.

Für die Belegung ist der landesspezifische Code einzustellen. Zusätzlich vereinfacht die Option *Dead Keys aktivieren* den Umgang mit den toten Tasten.

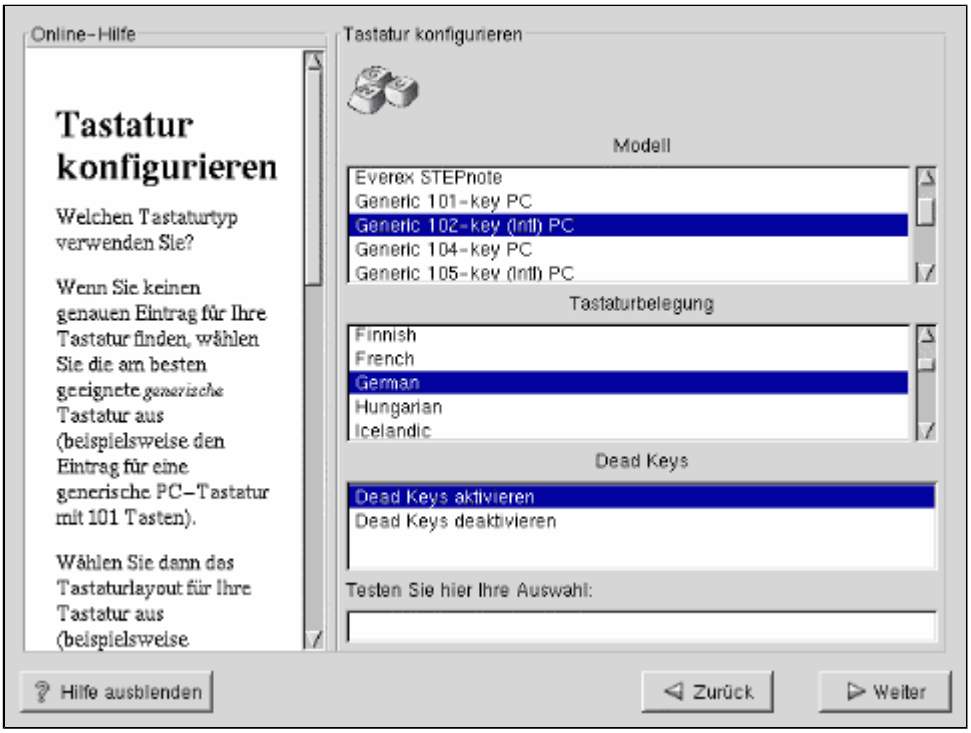


Abbildung 3: Tastaturauswahl

Vermutlich wird Ihre Maus bereits korrekt konfiguriert worden sein. Überprüfen Sie im Falle einer 2-Tasten-Maus die Aktivierung des Kästchens *Drei Tasten emulieren*.

Das Rad einer Wheel-Maus lässt sich hier nicht in Schwung versetzen. Hinweise hierzu finden Sie im Abschnitt Hardware-Installation in der Systemadminsitration.

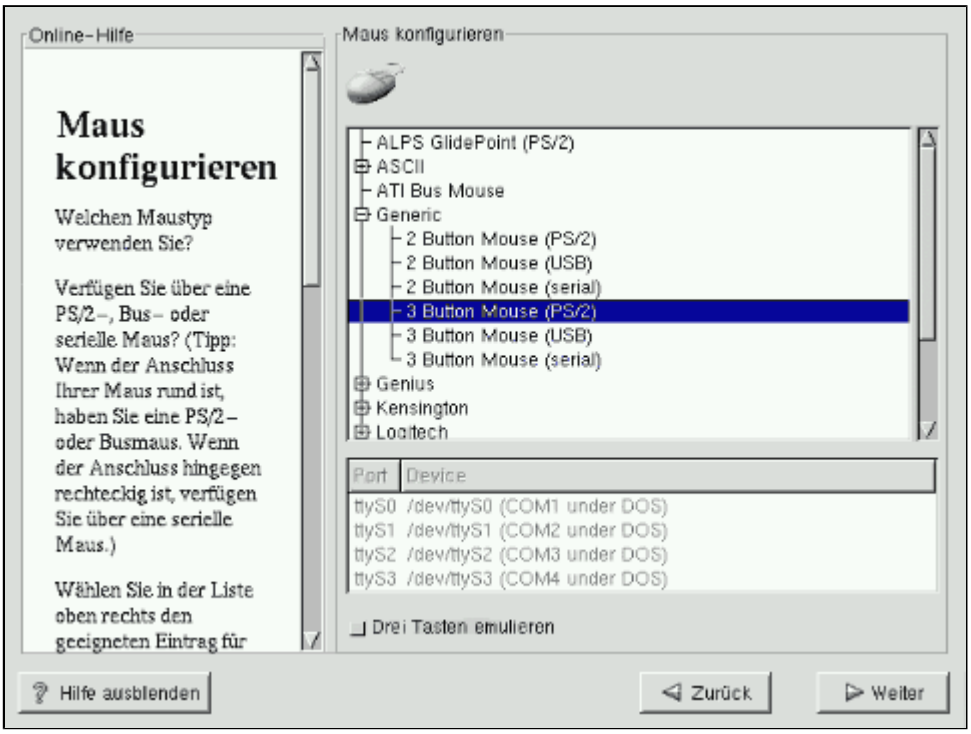


Abbildung 4: Mausauswahl

Nach Bearbeitung dieser drei einleitenden Schritte begrüßt Sie der System-Installer. Wenn Sie etwas Zeit haben, so arbeiten Sie die Hinweise der Hilfe durch. Mit [Weiter] beginnt die eigentliche Installation.

Schritte zum System

Der folgende Dialog ermöglicht zunächst einmal die Wahl zwischen einer **Neuinstallation** und einer **Aktualisierung** (Upgrade) einer bestehenden RedHat-Linux-Installation. Letzteres sollten Sie nicht in Erwägung ziehen, wenn Ihre vorherige RedHat-Installation bereits in die Jahre gekommen ist (eine auf Kernel < 2.0 basierende Version können Sie ohnehin nicht aktualisieren). Prinzipiell gilt, dass eine Aktualisierung immer Probleme herauf beschwört. Wenn Sie die Wahl haben, so sichern Sie alle Daten der alten Installation und bügeln das neue darüber.

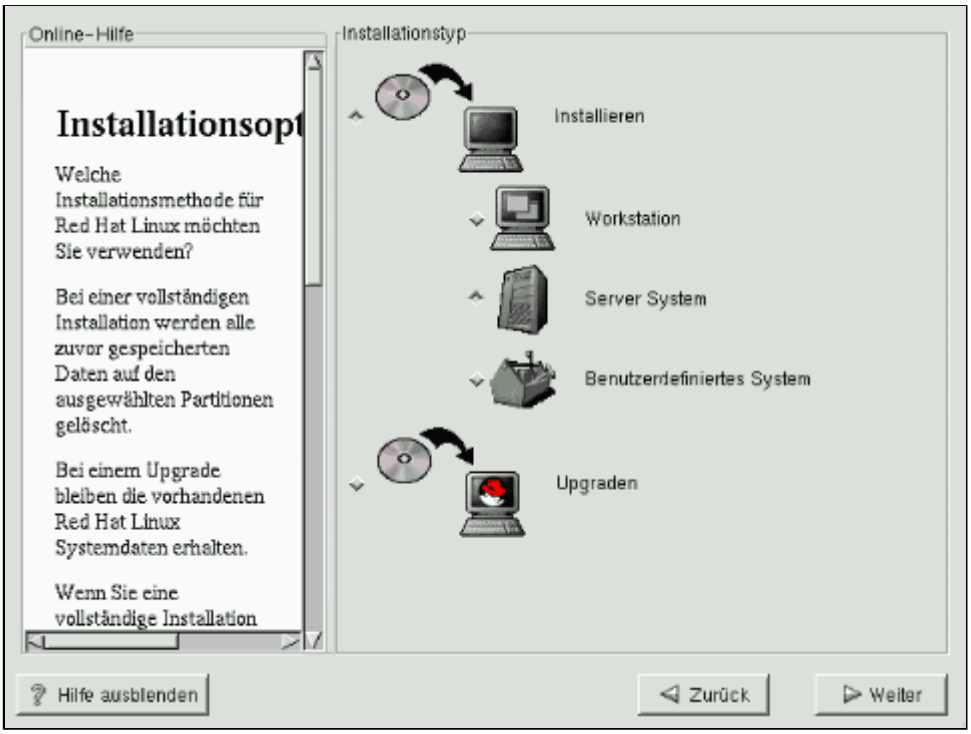


Abbildung 5: Art der Installation

Für eine neue Installation stehen nun 3 Optionen offen. **Workstation** und **Server System** unterscheiden sich hinsichtlich der installierten Software-Pakete. Beide durchlaufen aber dieselben Installationsschritte. Sie sollten sich bewusst sein, dass beide Varianten bei Wahl der automatischen Partitioniertung sämtliche vorhandenen Linux-Partitionen (ext2, swap) des Systems beanspruchen und deren Inhalte überschreiben. Auch wird der Bootmanager Lilo in den Master Boot Record (Mbr) platziert, sodass andere Bootmanager (Windows...), die dort eventuell installiert sind, ungefragt entfernt werden.

Die **Benutzerdefinierte Installation** umgeht obige Hürden, erfordert jedoch einige Linux-spezifische Vorkenntnisse. Die Platten lassen sich beliebig partitionieren, die zu installierenden Pakete bis ins Detail festlegen und der Bootmanager lässt sich an alternative Positionen verbannen.

Die Option **Automatisches Partitionieren...** der folgenden Abbildung steht bei der nutzerdefinierten Installation nicht zur Verfügung.

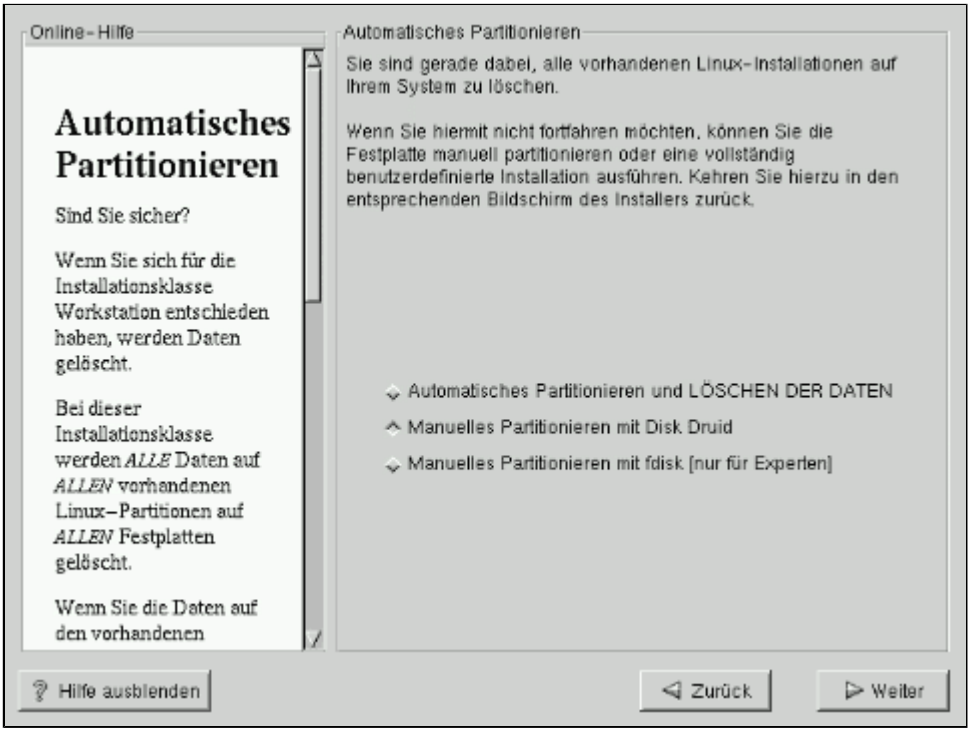


Abbildung 6: Automatische Partitionierung

Bei der Wahl der automatischen Partitionierung nimmt sich das Installationswerkzeug alle Linuxpartitionen und den verfügbaren freien Platz. Es erzeugt eine 64 MByte große Swap-Partition, eine 16MByte umfassende Partition (/boot) und eine Partition für das Rootverzeichnis im Umfang der restlichen Plattenkapazität. Für den ersten Kontakt mag die gewählte Anordnung brauchbar sein, im produktiven Umfeld sind die Swap-Größe drastisch unterdimensioniert und eine weitere Verteilung der Installation auf mehrere Partitionen sinnvoll (vergleiche Abschnitt Vorbereitungen).

Die maximale Flexibilität erlaubt die Partitionierung mit dem Kommando fdisk. Aber die textbasierte Bedienung und die Fülle der Optionen sollten wirklich nur dem Kenner vorbehalten bleiben; für 99% aller Fälle wird auch das Partitionierungswerkzeug **DiskDruid** zu einem optimalen Ergebnis führen.

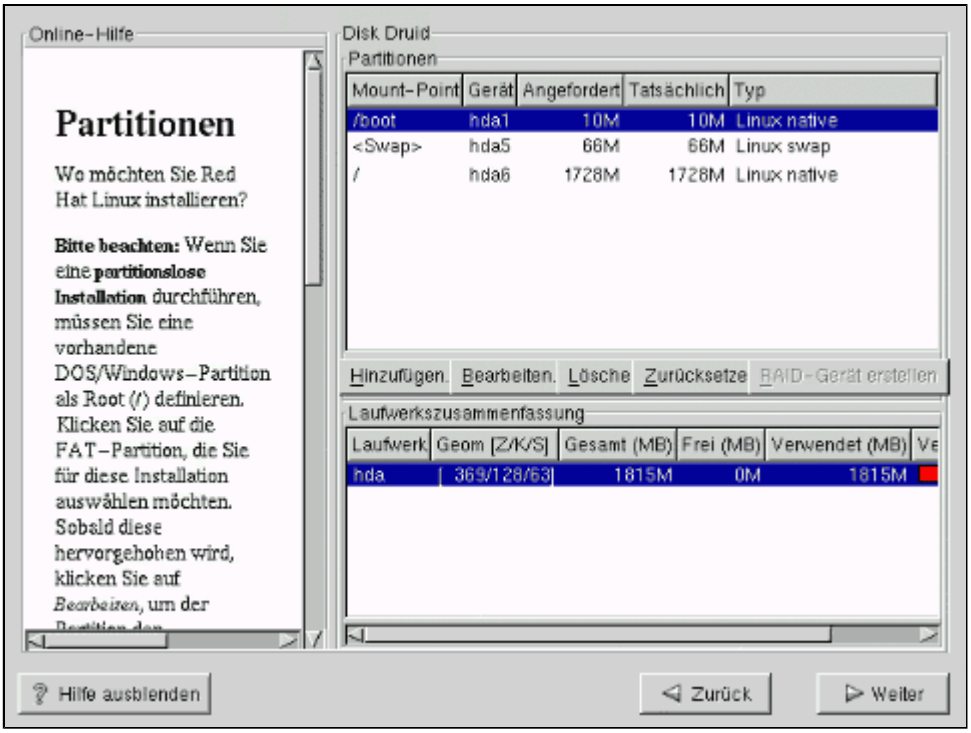


Abbildung 7: Manuelle Partitionierung

Die Oberfläche von DiskDruid unterteilt sich in zwei Listen. Die untere zeigt die in Ihrem System gefundenen Festplatten an. In der oberen Liste ist das momentane Partitionierungsschema des selektierten Eintrags aus der unteren Liste sichtbar.

Solange die Spalte **Frei (MB)** zu einer Festplatte noch freie Kapazitäten verkündet, können Sie weitere Partitionen anlegen. Dies geschieht über den Knopf **Hinzufügen** aus der mittleren Schalterreihe. (DiskDruid legt nur eine primäre Partition an und ordnet alle weitere Partitionen in einer erweiterten an. Somit stehen ausreichend Partitionen zur Verfügung). Im sich öffnenden Dialog geben Sie die gewünschte Partitionsgröße an (die natürlich den freien Platz nicht übersteigen darf). Da intern die Partitionen auf die Zylindergrenzen abgebildet werden, kann sich eine geringe Abweichung zwischen »angefordertem« und »tatsächlichem« Speicherplatz ergeben. Weiterhin sollten Sie den Typ der Partition setzen. Sie benötigen mindestens eine Swappartition, alle anderen sind vom Typ »Linux« (ext2). Den Linuxpartitionen sollten Sie einen Mountpunkt zuordnen, also den Namen des Verzeichnisses, unter dem später auf die Partition zugegriffen wird. Mindestens für das Wurzelverzeichnis »/« muss der Mountpunkt definiert werden, alles Weitere ist optional.

Über den Schalter **Bearbeiten** - oder per Doppelklick auf den entsprechenden Eintrag - kann sowohl der Mountpunkt als auch der Typ einer Partition nachträglich gesetzt werden. Die Partitionsgröße lässt sich immer verkleinern, jedoch nur vergrößern, wenn nicht der sich anschließende Zylinder bereits durch eine weitere Partition belegt ist.

Das Entfernen einer Partition ist über **Löschen** möglich. **Zurücksetzen** stellt die ursprünglichen Werte wieder her.

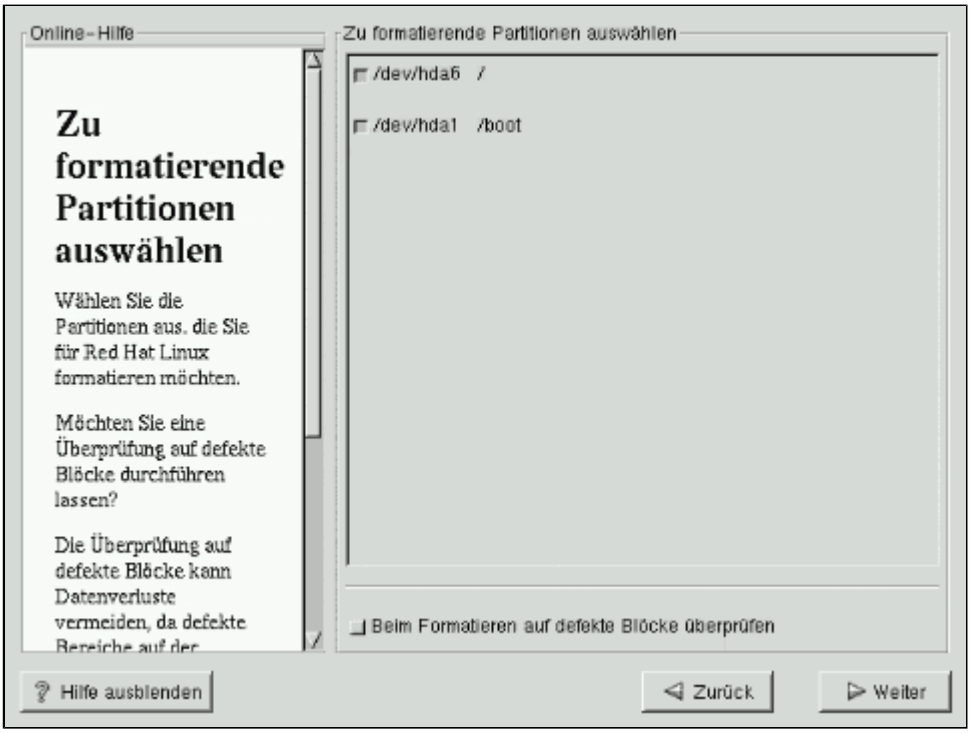


Abbildung 8: Formatierung

Im letzten Dialog zur Einrichtung der Festplatten, müssen Sie sie Partitionen bestimmen, die formatiert werden sollten (d.h. auf ihnen wird das Linux-Dateisystem eingerichtet). Natürlich sollten zumindest alle neu erstellten Partitionen formatiert werden. Das Formatieren bestehender Linux-Dateisysteme bereinigt diese von alten Daten. Eine Überprüfung auf defekte Blöcke ist bei Festplatten, die schon länger ohne Probleme im System arbeiteten, nicht notwendig. Schaden wird es auch nicht, es kostet nur reichlich Zeit.

Erste Konfigurationen

Falls bei der anfänglichen Hardwareerkennung eine Netzwerkkarte in Ihrem System erkannt wurde, erscheint nun ein Dialog, um die Grundfunktionalität des Netzwerks einzurichten. Sie können den Punkt auch überspringen und später ausführen, indem Sie im laufenden System

```
root@sonne> anaconda --reconfig
```

aufrufen. Alle Masken des Installationsprogramms, die die Konfiguration des System (Hardware, Netzwerk...) betreffen, werden Ihnen dann erneut begegnen.

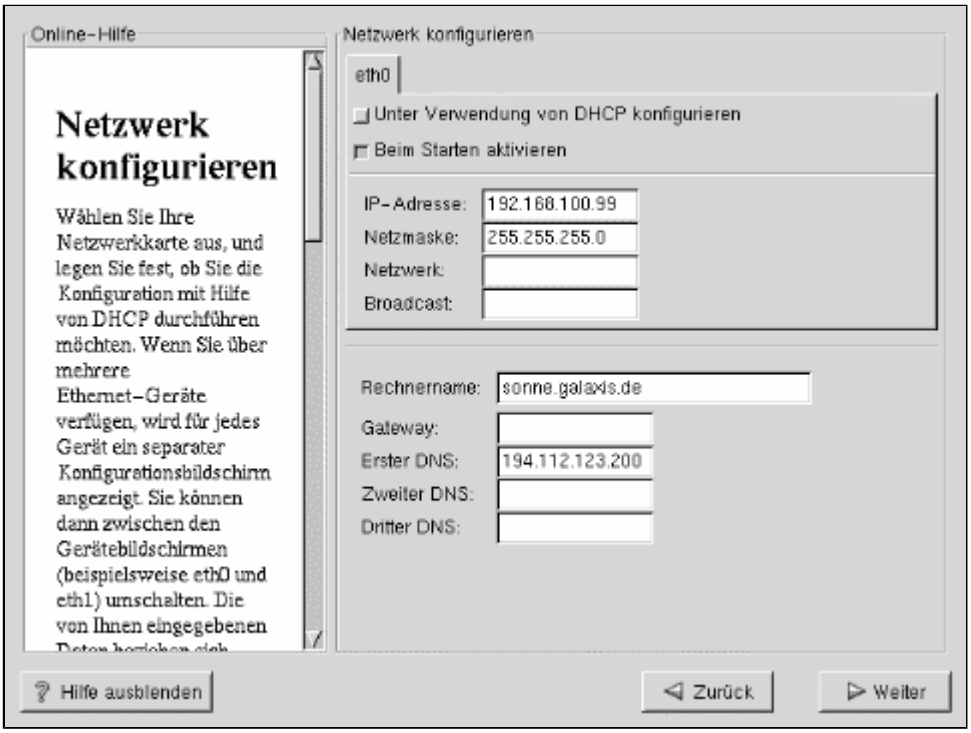


Abbildung 9: Einrichten des Netzwerks

Wenn Sie mit den einzelnen Werten nichts anzufangen wissen, dann informieren Sie sich im Kapitel Netzwerk-Grundlagen. Sobald Sie eine **IP-Adresse** angeben, setzt das Installationsprogramm die **Netzmaske**, das **Netzwerk** und die **Broadcast**-Adresse selbstständig ein. Nur für den Fall, dass Sie das Netzwerk in weitere Subnetze unterteilen möchten, müssen Sie selbst Hand anlegen.

Der **Rechnername** wird inklusive der Domain eingetragen. Domain ist dabei zumeist ein symbolischer Name für das Netzwerk, in dem Ihr Rechner eingebunden ist. Der Name des Rechners selbst muss innerhalb einer solchen Domain eindeutig sein. Im Unterschied zu Dateinamen wird die Groß- und Kleinschreibung bei Rechnernamen nicht beachtet! Falls Sie das Feld nicht ausfüllen, setzt das Installationsprogramm »localhost.localdomain« ein.

Unter **Gateway** ist die Adresse des Rechners einzutragen, der in Ihrem Netz für die Weiterleitung von Paketen zuständig ist, deren Adressat nicht innerhalb des eigenen Netzes liegt. Wenn Sie keinen solchen Rechner kennen (weil Ihr Rechner gar nicht in einem Netzwerk ist...) so lassen Sie das Feld frei. RedHat 7.0 belegt das Feld des **1. DNS-Servers** mit einer gültigen Adresse vor. Verfügt Ihr Rechner nicht über permanten Anschluss ans Internet, sollten Sie das Feld löschen (oder durch die Adresse des DNS-Servers aus Ihrem lokalen Netz ersetzen), da sonst einige Programme umsonst Anfragen an den DNS-Server richten und somit nur verzögert reagieren (bekanntester Kandidat ist der Netscape).

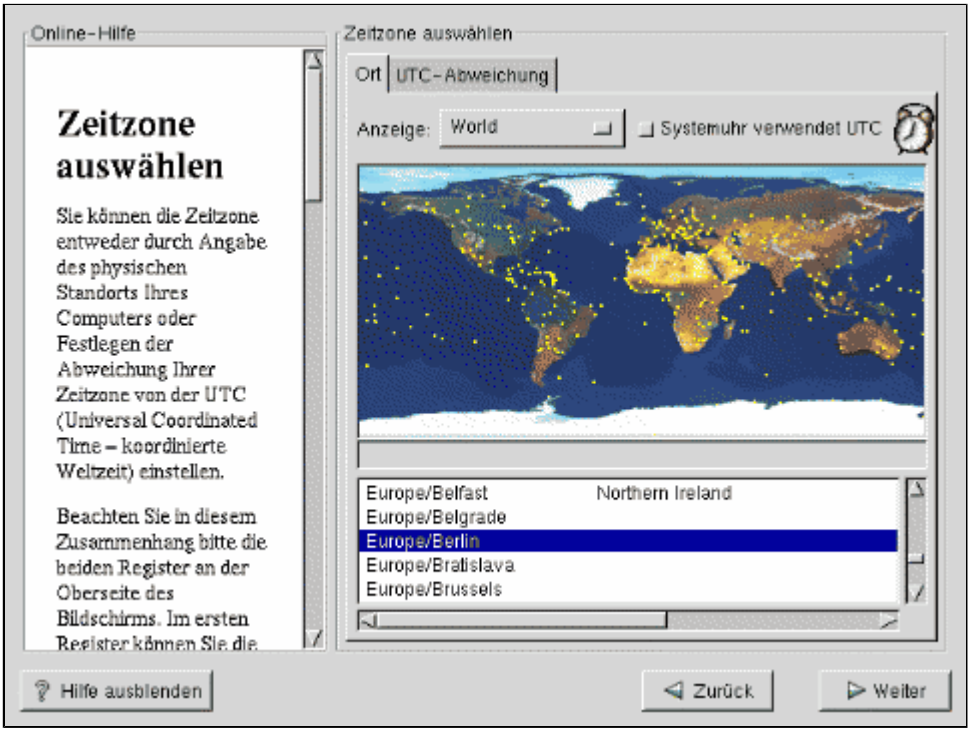


Abbildung 10: Einstellung der Zeitzone

Die Systemzeit des Rechners kann entweder nach der *Universal Coordinated Time* (UCT) oder nach lokaler Zeit laufen. In ersterem Fall konfiguriert man die Abweichung des Rechnerstandorts von UTC, sodass alle Zeitmarken der lokalen Zeit entsprechen.

Am einfachsten Klicken Sie auf den Standort der Weltkarte, der am ehesten dem Rechnerstandort entspricht. Bei fehlender Mausunterstützung wählen Sie aus der Liste unter **World** den Kontinent. Unter der Karte können Sie nun den Standort in einer Liste markieren. Falls die Systemzeit auf UTC steht, muss der entsprechender Schalter noch aktiviert werden.

Alternativ lässt sich die Zeitzone über die **UTC-Abweichung** (andere Registerkarte) konfigurieren. Selektieren Sie hierzu den entsprechenden Eintrag der Liste.

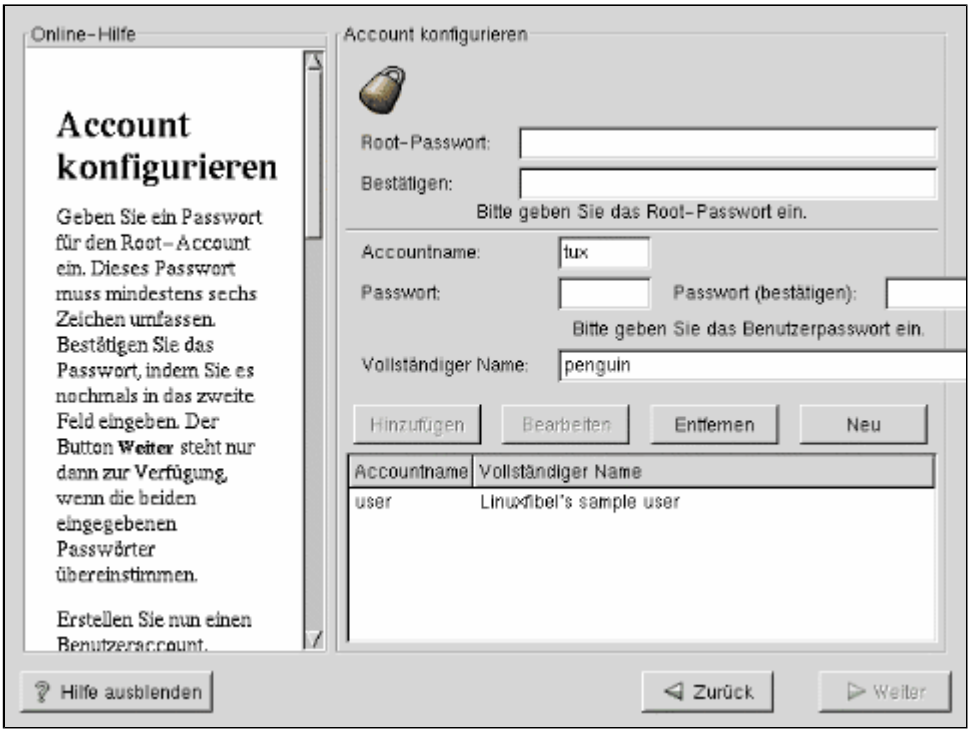


Abbildung 11: Rootpasswort und erste Nutzer

Das Installationsprogramm erwartet im nächsten Dialog zumindest die Festlegung eines Passworts für Root. Dieses muss mindestens 5 Zeichen enthalten. Informationen zu Passwörtern und Sicherheit finden Sie im Kapitel Systemadministration unter den Abschnitten Nutzerverwaltung und Systemsicherheit.

Zusätzlich können Sie einen oder mehrere Benutzer anlegen. Der **Accountname** muss eindeutig sein; er sollte maximal 8 Zeichen umfassen (kein Muss, aber einige ältere Programme akzeptieren nicht mehr und schneiden überschüssige Zeichen ab). Das Passwort ist erforderlich, die Angabe des vollständigen Namens dagegen optional.

Paketauswahl

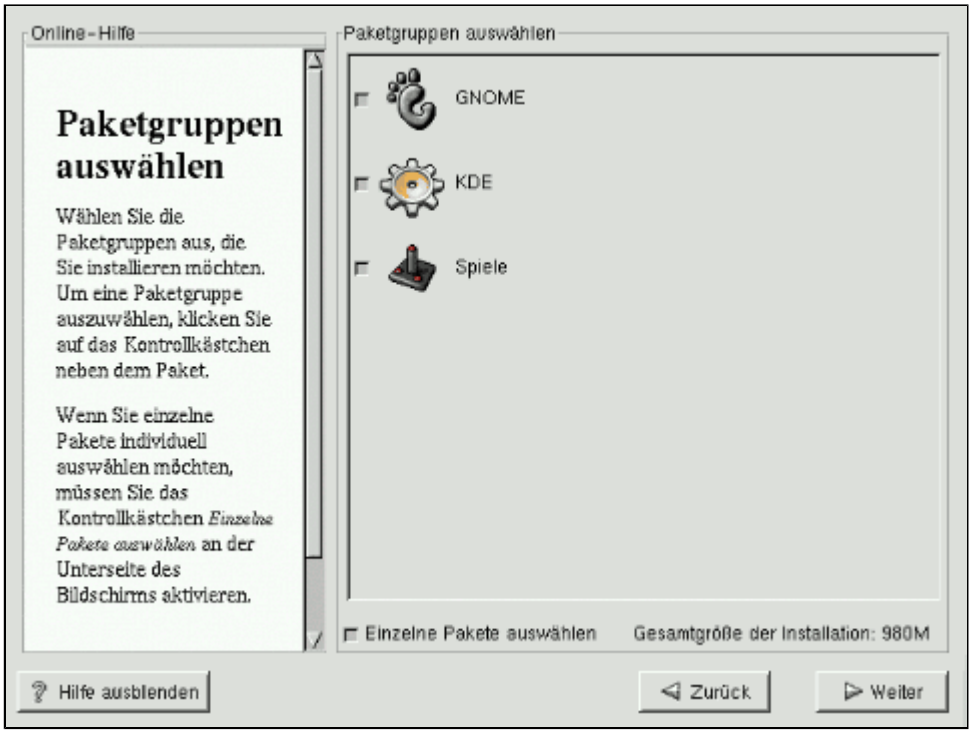


Abbildung 12: Auswahl der Paketgruppen

Viele Alternativen bietet einem das Installationsprogramm bei der Paketgruppenauswahl nicht. Entweder KDE oder Gnome oder beides. Wenn mangelnder Plattenplatz nicht dagegen spricht, sei dem Linux-Neuling die Installation beider Desktop-Umgebungen empfohlen. Pauschal eine als »die Bessere« zu bezeichnen, wäre frevelhaft, denn jede besitzt Vor- und Nachteile.

Die Option **Spiele** sollte selbstverständlich nicht fehlen.

Sollten Sie **Einzelne Pakete auswählen** angekreuzt haben, so gelangen Sie nachfolgend in einen weiteren Dialog, der eine genauere Auswahl der zu installierenden Pakete gestattet. Je nach zuvor aktivierte Paketgruppe sind bestimmte Pakete bereits selektiert. Sie können im linken Teil des Fensters durch einen Baum navigieren, der alle Pakete nach ihrem Verwendungszweck gruppiert. Nach Selektion eines Eintrags werden die Pakete zur Gruppe im rechten Fenster aufgelistet. Ein selektiertes Paket erhält ein rotes Häkchen; durch Doppelklick auf das Symbol ändert sich dessen Status. Zu einem markierten Paket erhalten Sie weitere Informationen über Sinn und Inhalt.

Verlassen Sie die detaillierte Auswahl, so werden die Abhängigkeiten überprüft. D.h. das Installationsprogramm schaut nach, ob zu den selektierten Paketen auch alle Pakete vorhanden sind, die Basisfunktionalitäten für diese Pakete bereitstellen (bspw. Bibliotheken). Werden Pakete vermisst, werden diese in einem Fenster aufgelistet. Wenn Sie nicht genau wissen, was Sie tun, sollten Sie dort die Option **Pakete installieren, um Abhängigkeiten zu erfüllen** aktivieren.

X Window

Prinzipiell kann die Konfiguration des X-Window-Systems auch zu einem späteren Zeitpunkt erfolgen. Allerdings wird der benötigte X-Server bereits automatisch installiert, wenn Sie die Konfiguration jetzt vornehmen. Falls das Installationsprogramm allerdings nur im 16-Farb-Modus startete (damit kommt jede Grafikkarte klar), werden Sie voraussichtlich Probleme haben, dem Server jetzt eine vernünftige Darstellung abzuringen, denn die nachfolgend aufgelisteten Grafikkarten wären der automatischen Erkennungsroutine nicht entgangen. Zu sehr aktuellen Karten müssen Sie sich zumeist erst noch die notwendigen Treiber besorgen (Internet).

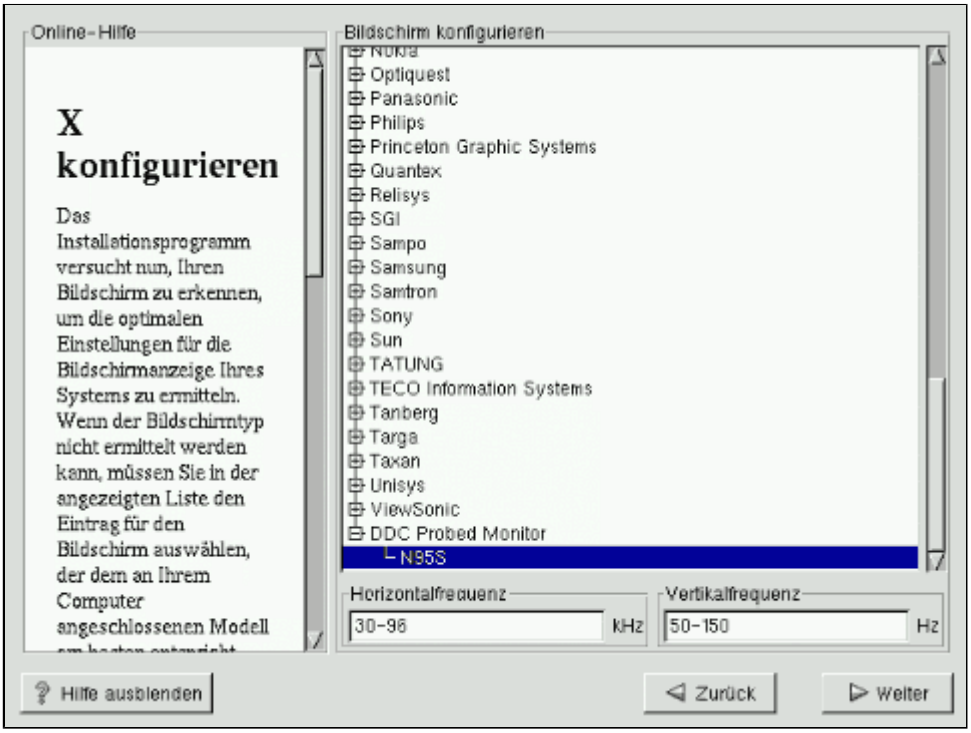


Abbildung 13: Auswahl des Monitors

Falls Ihr Monitortyp aufgelistet ist, so selektierten Sie den Eintrag. Wenn nicht, dann genügt die manuelle Angabe der Frequenzen, die im Handbuch zu finden sein sollten.

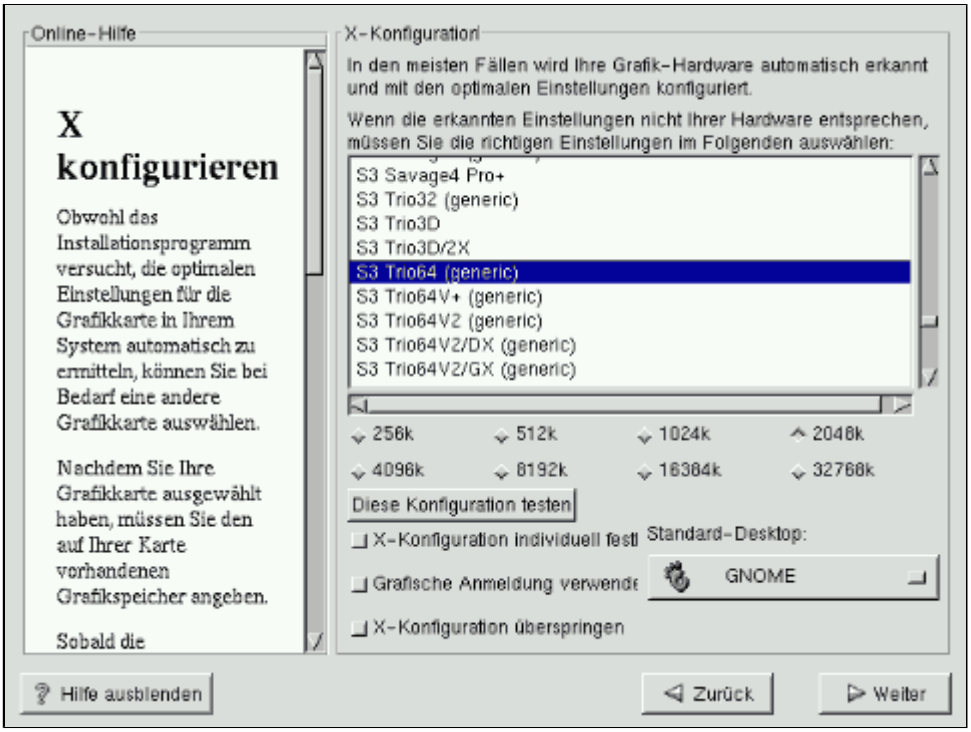


Abbildung 14: Auswahl der Grafikkarte

War die Hardwareerkennung erfolgreich, sollte Ihre Grafikkarte in der Liste ausgewählt worden sein. Falls nicht, so suchen Sie Ihre Karte. Steht diese nicht in der Liste, so sollten Sie den Eintrag **Keine aufgeführte Karte** markieren. Allerdings nützt dies Ihnen nur, wenn Sie wissen, welcher X-Server mit Ihrer Karte umgehen kann.

Wenn der Wert des VideoRAMs passt, können Sie **diese Konfiguration testen**. So gehen Sie sicher, dass die getroffenen Angaben tatsächlich funktionieren.

Selektieren Sie den Punkt **X-Konfiguration individuell festlegen**, so gelangen Sie in einen weiteren Dialog.

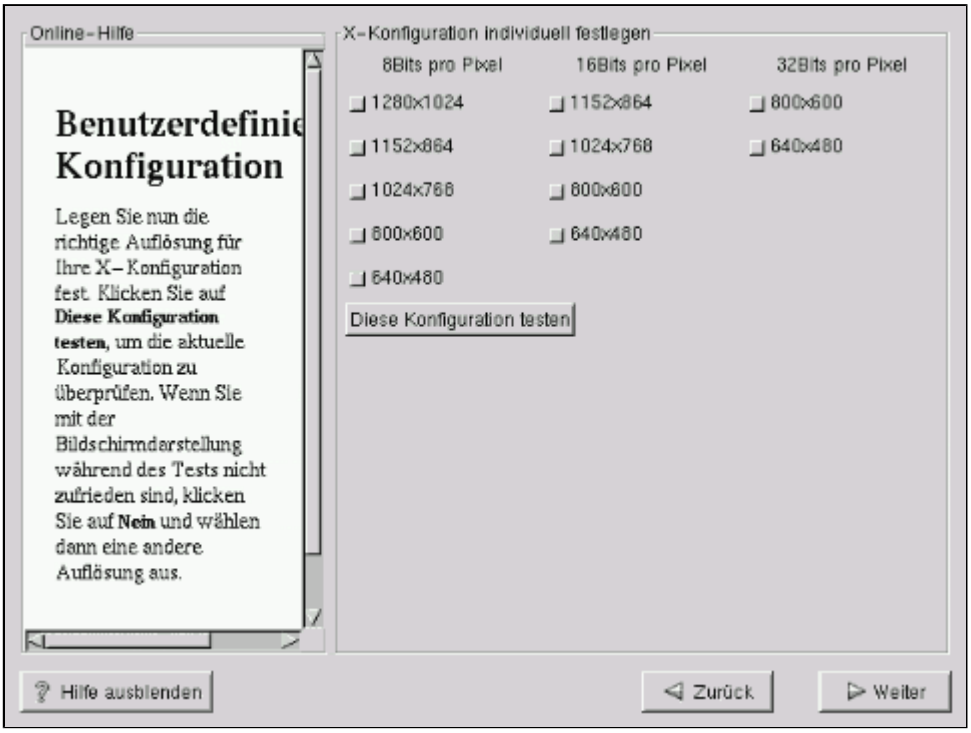


Abbildung 15: Auswahl der Auflösungen und Farbtiefen

Die dargestellten Auflösungen entsprechen den zulassigen Modi für die zuvor konfigurierte Monitor-Grafikkarten-Kombination. Testen Sie die Konfiguration aus gewählter Farbtiefe und Auflösung, bevor Sie sie endgültig übernehmen.

Der Abschluss

Mit dem folgenden Dialog besteht die letzte Möglichkeit, den Installationsprozess abzubrechen. Bislang ist am System selbst nichts verändert worden. Speziell wurden weder die Festplatte(n) partitioniert, noch einzelne Partitionen formatiert. Bestätigen Sie den (nicht dargestellten) Dialog mit [Weiter], können Sie getrost einen Kaffee trinken gehen, denn je nach System und gewähltem Installationsumfang wird das System zwischen 15 (schneller Prozessor, schnelles Installationsmedium) und 120 Minuten werkeln. Mitunter fordert das Programm zum Wechseln der CDROM auf. Den Stand der Installation illustriert eine Fortschrittsanzeige.

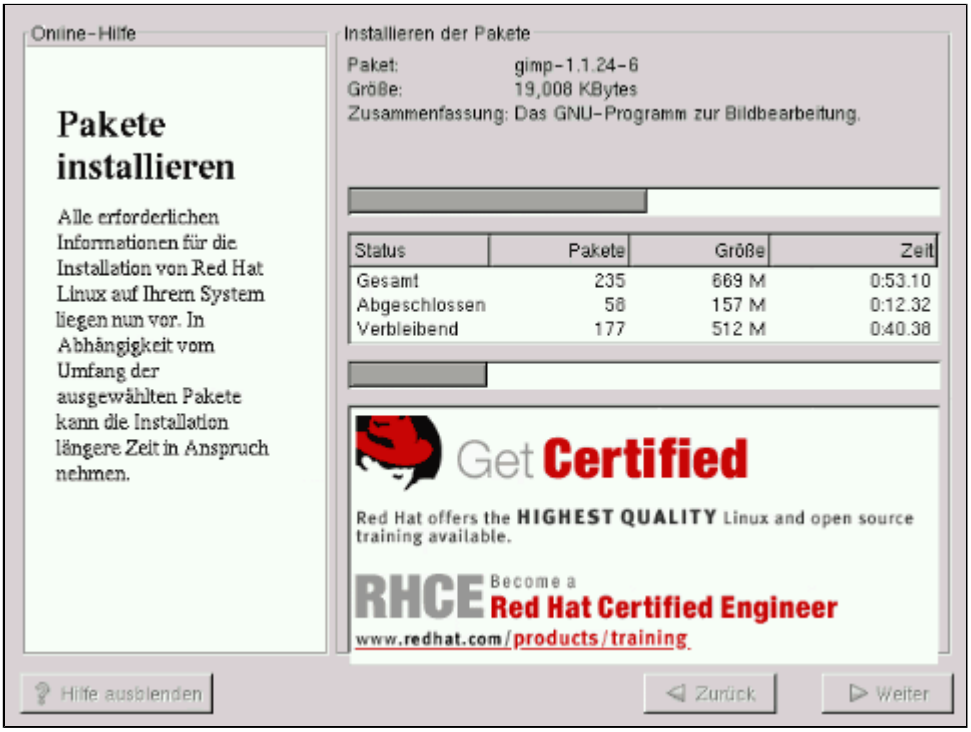


Abbildung 16: Statusausgabe

Eine Bootdiskette sollten Sie erstellen, falls Sie Bedenken wegen der 1024-Zylinder-Grenze hegen oder falls Sie nachfolgend ein anderes Betriebssystem installieren, das eventuell den Bootmanager zu Gunsten seiner eigenen Startroutine (z.B. alle Windows-Typen) überschreibt.

Abbildung 17: Bootdiskette

Zum Abschluss liefert ein letzter Dialog noch einige Hinweise zum weiteren Vorgehen. Beenden Sie das Programm, so startet der Rechner neu und die Aufforderung zum Anmelden sollte alsbald erscheinen.

Abbildung 18: Abschluss der Installation

Installation bei SuSE

Übersicht

Im deutschsprachigen Raum erreicht keine Linuxdistribution eine annähernd große Verbreitung wie SuSE Linux. Den Grund allein in der Vergangenheit zu suchen, als über Jahre hinweg SuSE die einzige ernsthafte Alternative in deutscher Sprache darstellte, würde dem Engagement der Entwickler nicht gerecht werden. SuSE unterstützt intensiv das XFree86-Projekt und das beachtliche Treiberreservoir für Grafikkarten ist nicht zuletzt dieser Zusammenarbeit zu verdanken. Auch das Alsaprojekt und das Reiserfs - um nur zwei zu nennen - gewannen durch SuSE an Dynamik. Die SuSE-Distribution ihrerseits profitiert von dieser Zusammenarbeit, indem aktuellste Neuerungen sehr zügig in den Lieferumfang der Distribution Einzug halten. Wer über neueste Hardware verfügt, hat mit SuSE oft die größten Chancen, diese ohne umfangreiche Eingriffe ins System zum Laufen zu bringen.

Anfänger schätzen SuSE Linux wegen dessen »einfacher« Administration. Die Versionen des **Yast** bilden einen zentralen Anlaufpunkt, um die gebräuchlichen Hausaufgaben zu erledigen. Das dahinter liegende Skript **SuSEConfig** erledigt dann den »Rest«, wobei genau diese wenig transparente Strategie manuelle Eingriffe ins System verbietet. Zwar lässt sich die automatische Konfiguration deaktivieren, doch muss dann jeder Schritt von Hand erledigt werden, womit selbst viele fortgeschrittene Linuxer überfordert sind.

Eine Schattenseite?

Die Gründe für die »eigenwillige« Auslegung des File System Hierarchie Standards aufzuzuählen, würde in Mutmaßungen enden. Fakt ist, dass nicht zuletzt die von den »allgemeinen Gepflogenheiten« abweichende Verzeichnisstruktur zahlreiche Kritiker auf den Plan rief.

Und Fakt ist leider auch, dass bei all der Aktualität der Software, die mit jeder Distributionsversion ausgeliefert wird, die Stabilität des Systems oft leidet. Aber - abgesehen von Debian - befindet SuSE sich da in guter Gesellschaft...

Zumindest der Kritikpunkt der individuellen FHS-Interpretation scheint mit der neuerlichen Orientierung an den Richtlinien der **Linux Standard Base** allmählich seine Basis zu verlieren. SuSE 7.1 zeigt eine deutliche Annäherung an die Praxis der anderen »Großen« der Branche.

Yast1 oder Yast2?

Die nachfolgend skizzierte Installation eines SuSE-Systems basiert auf der Version 7.2. Wir nutzen das Werkzeug **Yast2**, das zwar eine weniger detaillierte Konfiguration als das auf Dialogboxen basierte **Yast1** ermöglicht, für die Bedürfnisse der meisten Anwender jedoch genügen sollte. Auf einige Abweichungen werden wir ggf. eingehen.

Auf **Yast1** sollten Sie zurückgreifen, wenn die Installation über das Network File System oder über FTP erfolgen soll. Auch in Fällen, die das vorherige Laden bestimmter Hardware-Treiber, die die automatischen Erkennungsroutinen nicht selbsttätig vornehmen, erfordern, ist Yast1 die richtige Wahl. Pauschal lässt sich feststellen, dass Yast1 gegenüber Yast2 die manuelle Konfiguration einer breiteren Palette von Diensten und Hardwarekomponenten gestattet. Da dies ebensogut aus einem laufenden System heraus geschehen kann, ist letztere Eigenschaft kaum ein Grund, Yast1 den Vorzug einzuräumen.

Erstes Booten

Der einfachste Weg ist das Booten per 1. CDROM (Pentium optimierter Kernel) oder 2. CDROM (für ältere Prozessoren). Bei Rechnern, die diese Möglichkeit nicht bieten, muss von einer Diskette gestartet werden, die aber zunächst zu erstellen ist. Kopieren Sie hierzu die Datei **bootdisk** von der 1. CDROM (Verzeichnis »disks«) auf eine Diskette. Unter Linux eignet sich hierzu das Kommando **dd**:

```
user@sonne> dd if= bootdisk of= / dev/ fd0
```

Unter DOS/Windows benötigen Sie **rawrite.exe**, das Sie ebenso auf der CDROM im Verzeichnis »dosutils/rawrite« finden. Nach dem Programmstart fordert dieses Sie zunächst zur Angabe der zu kopierenden Datei und nachfolgend zur Eingabe des Ziellaufwerks auf.

Bei einer Framebuffer tauglichen Grafikkarte begrüßt Sie ein grafischer Startbildschirm mit den Menüpunkten »Installation«, »Manual Installation«, »Rescue System« und »Memory Test«.

Hinter »Manual Installation« verbirgt sich das bereits erwähnte Installationswerkzeug **Yast1**.

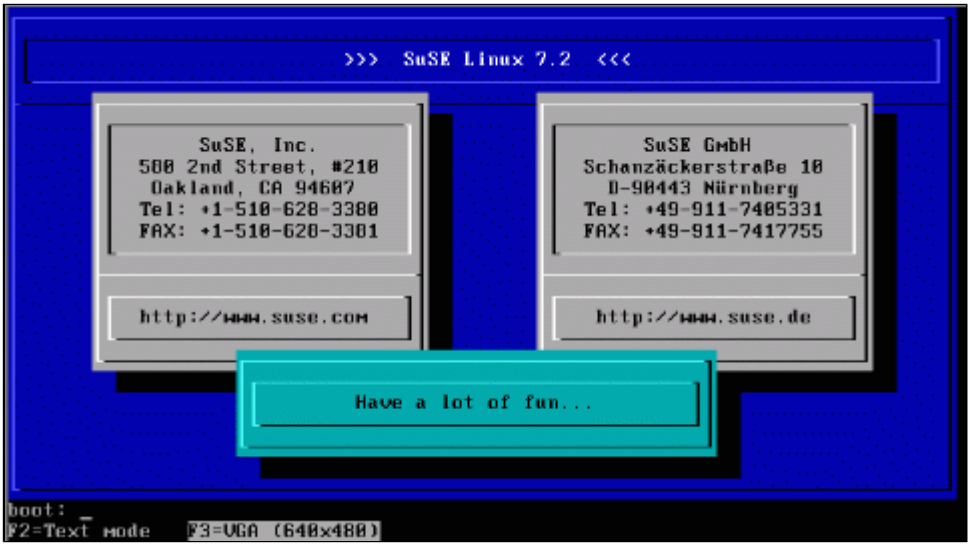


Abbildung 1: Textbasierter Startbildschirm

Vermag Ihre Grafikkarte nicht mit Framebuffer umzugehen, startet eine textbasierte Maske. Geben Sie [Enter] ein, um die Installation mit Hilfe von **Yast2** fortzusetzen. Ebenso mit Yast2 - jedoch im Textmodus - erfolgt die Installation nach Eingabe von [F2]. In beiden Fällen analysiert Yast2 ihr System, was einige Zeit in Anspruch nehmen kann.

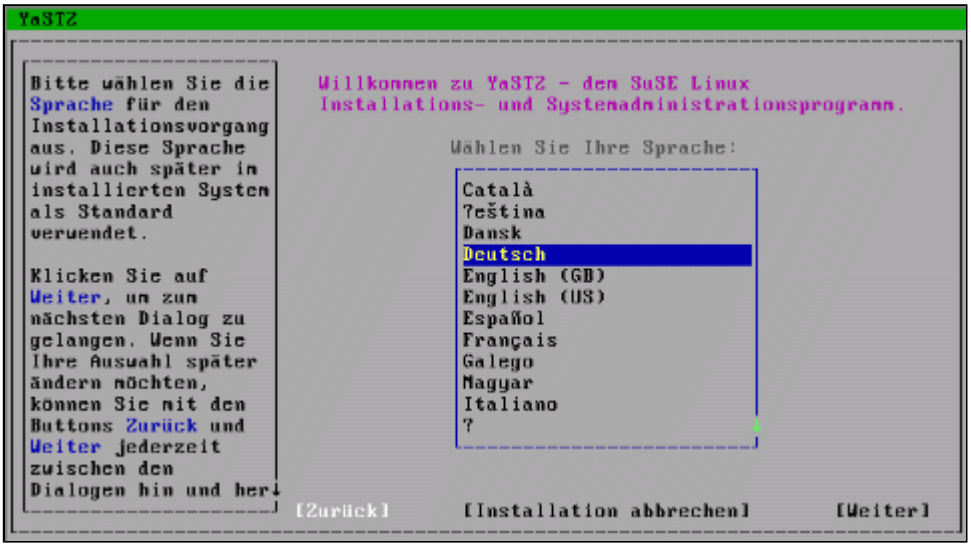


Abbildung 2: Yast2 im Textmodus

Um mit **Yast1** fortzufahren, müssen Sie am Boot-Prompt »manual[Enter]« eingeben.

Einstellungen

Abbildung 3: Sprachauswahl

Die erste Maske betrifft die Einstellung der Sprache während der Installation. Eine Änderung wid unverzüglich wirksam.

Mitunter kann es vorkommen, dass die automatische Hardwareerkennung bei der Maus scheiterte. In diesem (seltenen) Fall wird nachfolgend ein Dialog zur Mauskonfiguration eingeblendet.

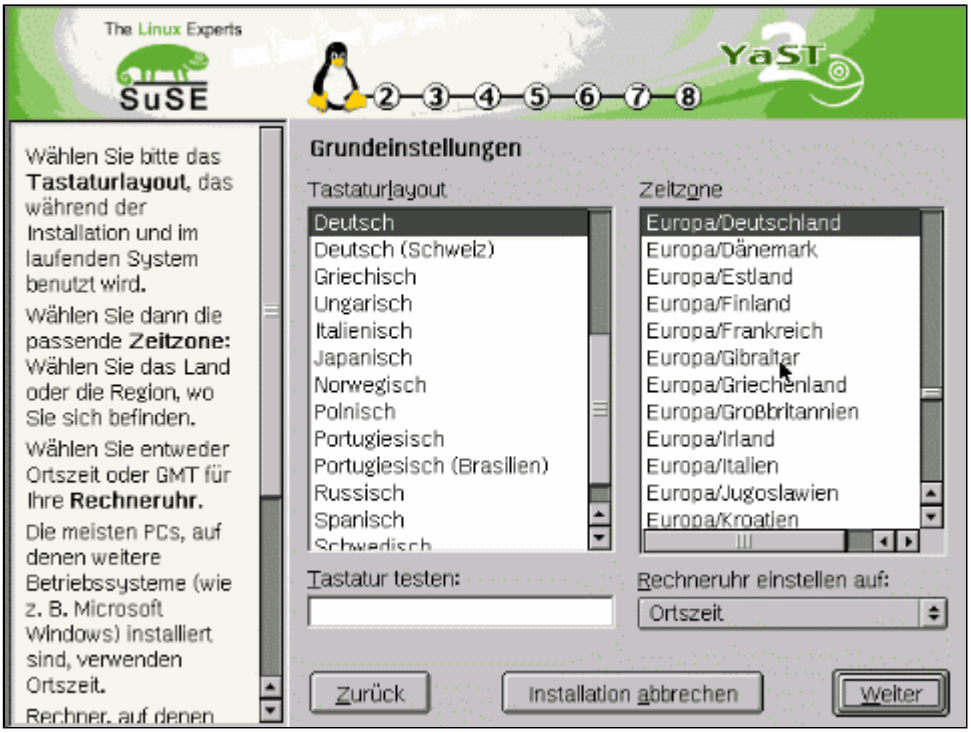


Abbildung 4: Tastatureinstellungen

Die Konfiguration der Tastatur ist für die Belegung der einzelnen Tasten zuständig; bei »Deutsch« werden automatisch die so genannten »Toten Tasten« (^ , ~ , ´) verborgen. Die Zeitzone wird anhand der eingestellten Installationsprache »erraten«. Je nachdem, ob die Hardwareuhr Ihres Rechners nach Ortszeit oder GMT läuft, sollten Sie die Rechneruhr justieren.

Installationsarten und Partitionieren

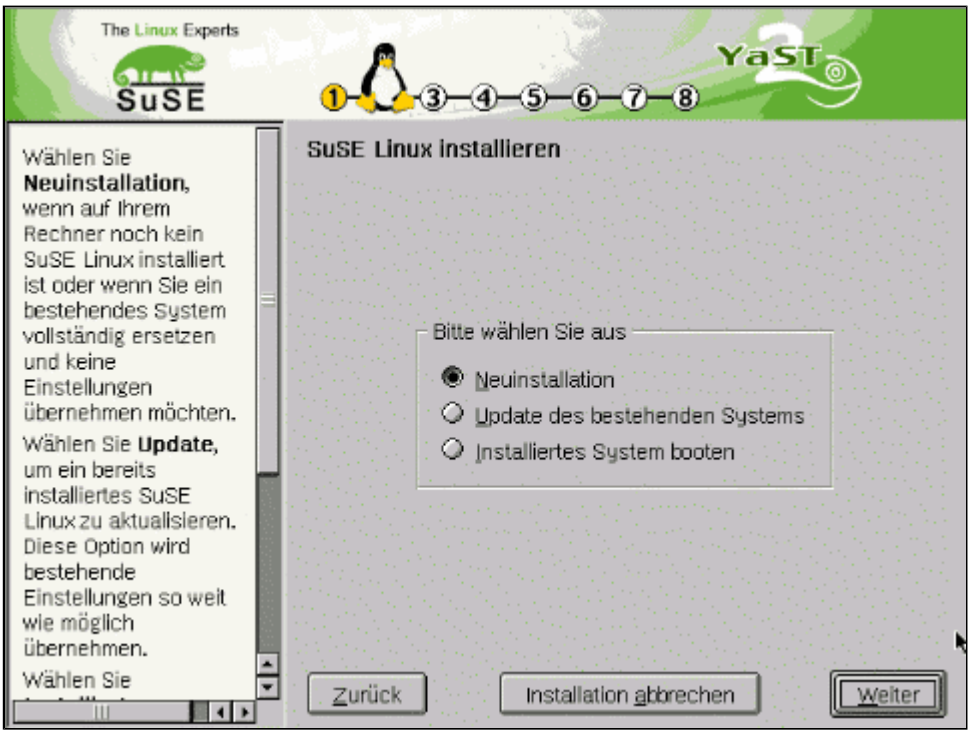


Abbildung 5: Installationsart

Der Unterschied zwischen »Update des bestehenden Systems« und »Neuinstallation« bedarf keiner weiteren Erläuterung. Erfahrungen - und nicht nur die meinigen - haben jedoch gezeigt, dass ein Update eines Systems in vielen Fällen zu unbrauchbaren Konfigurationen führt. Nicht, dass das gesamte System das Zeitliche segnet, aber irgend etwas klappt stets nicht wie erhofft. Ein Backup wichtiger Daten mit nachfolgender Neuinstallation ist deshalb ratsam.

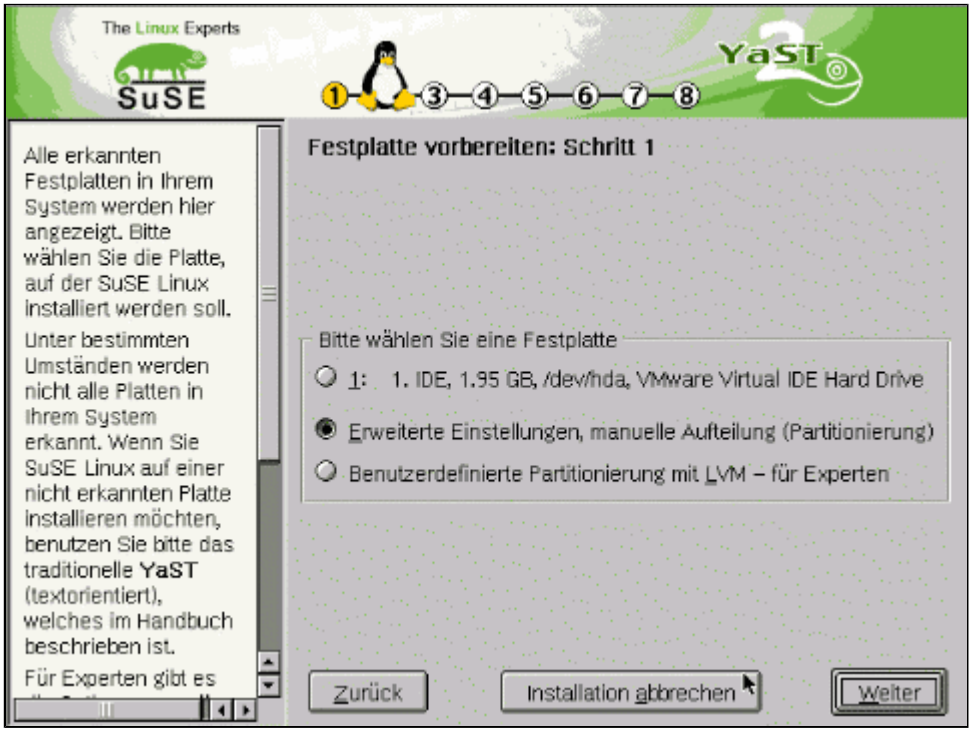


Abbildung 6: Partitionierung (1)

Die Partitionierung der Festplatte ist einer der heikelsten Punkte der gesamten Installation, vor allem wenn bereits andere Betriebssysteme auf Ihrem Rechner existieren. Zu diesem Zeitpunkt besteht allerdings noch keine Gefahr, die Änderungen an Partitionstabelle und Formatierung der Platte(n) geschieht erst zu einem späteren Zeitpunkt. Sie können durchaus die verschiedenen dargebotenen Varianten begutachten, ohne (vorerst) Schaden anzurichten.

Alle von Yast2 erkannten Festplatten werden zuvorderst aufgelistet. Sie können eine davon als Installationsmedium für Linux auswählen. Unter »Erweiterte Einstellungen, manuelle Aufteilung (Partitionierung)« besteht die Möglichkeit, die Installation auch auf mehrere Festplatten zu verteilen. Die Variante, mehrere Partitionen als eine einzige anzusprechen (»Logical Volume Manager«) wollen wir hier nicht besprechen.

Für den Fall, dass Sie sich für die Installation auf einer der gelisteten Festplatten entschieden haben, bietet Yast2 Ihnen nun an, die gesamte Platte zu formatieren bzw. - falls schon Partitionen existierten - einige davon für Linux freizuschaufeln. Gehen Sie sehr umsichtig mit dem Löschen um, die Daten auf solchen Partitionen sind, sobald Sie später die eigentliche Installation angestoßen haben, unwiderruflich verloren!

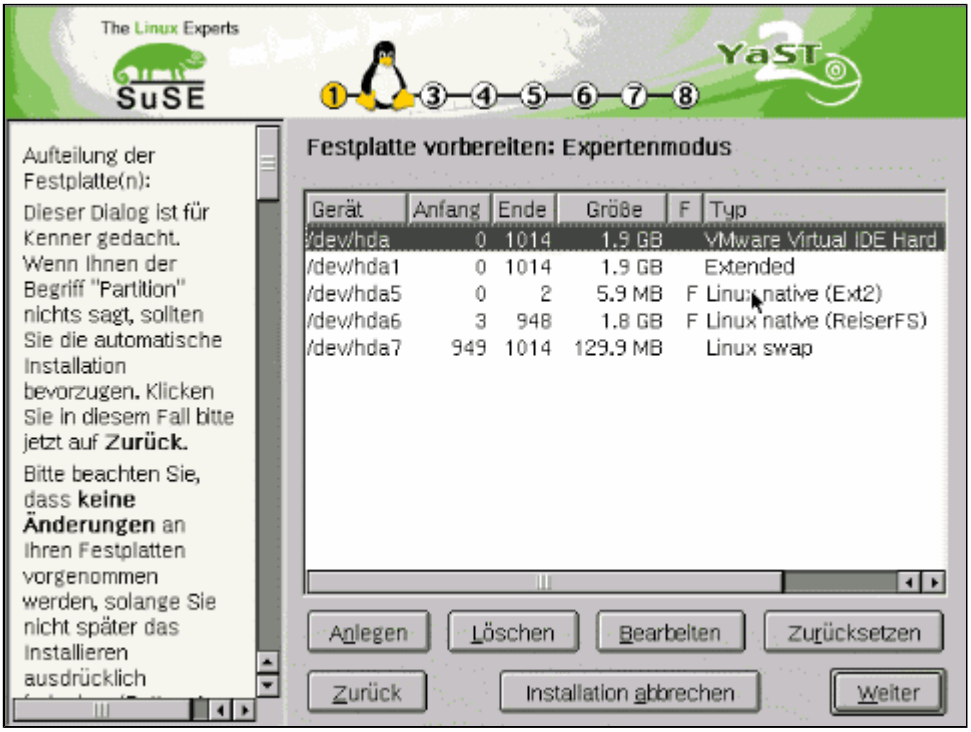


Abbildung 7: Partitionierung (2)

Haben Sie sich für die »Manuelle Aufteilung« entschieden, lassen sich die einzelnen Partitionen aller Festplatten gezielt für Linux zuordnen. Die Bedienung der Maske ist selbsterklärend. Typ (allerdings nur Ext2, Reiserfs und Swap) und Mountpunkt einer Partition können Sie unter »Bearbeiten« anpassen.

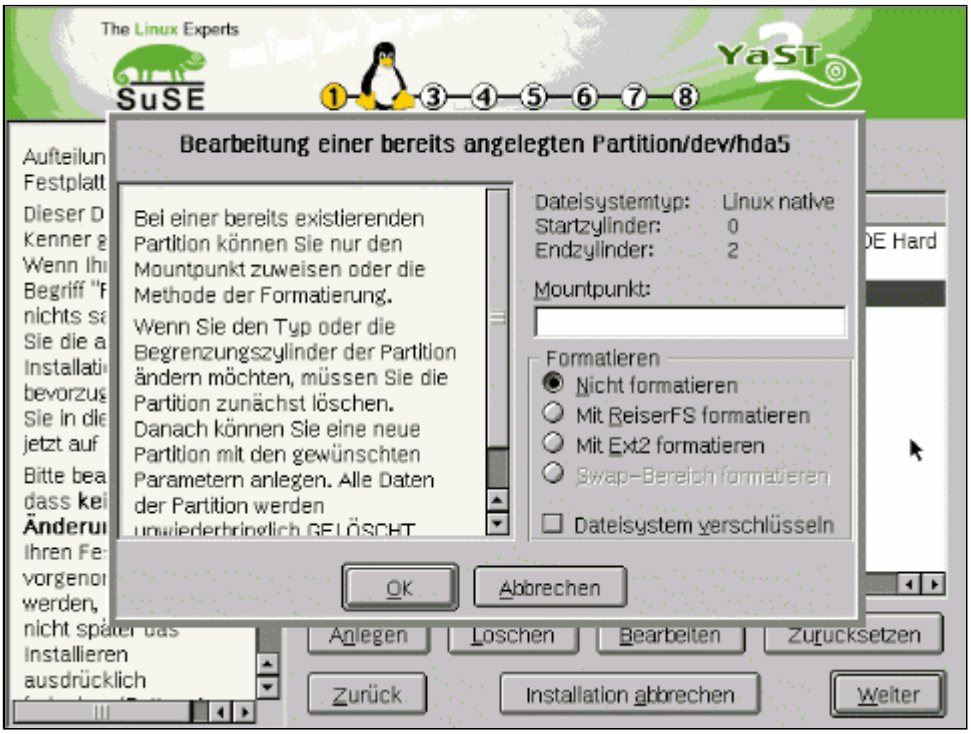


Abbildung 8: Partitionieren (manueller Modus)

Softwareauswahl

Abbildung 9: Software - Basisauswahl

Yast2 ermöglicht die Auswahl zu installierender Pakete in drei Stufen. In einem ersten Auswahlmenü müssen Sie sich für eine Grundkonfiguration entscheiden. »Minimal System« ist zwar der kleinste Softwareumfang, der sich in dieser Maske selektieren lässt, jedoch liegt die tatsächliche Menge an installierten Pakete noch weit über dem, was minimal benötigt wird. Letztlich sind die beiden »Standard«-Typen als Einstieg geeignet, wer konkrete Vorstellungen vom Installationsumfang hegt, wird um die »Erweiterte Auswahl« nicht umhin kommen.

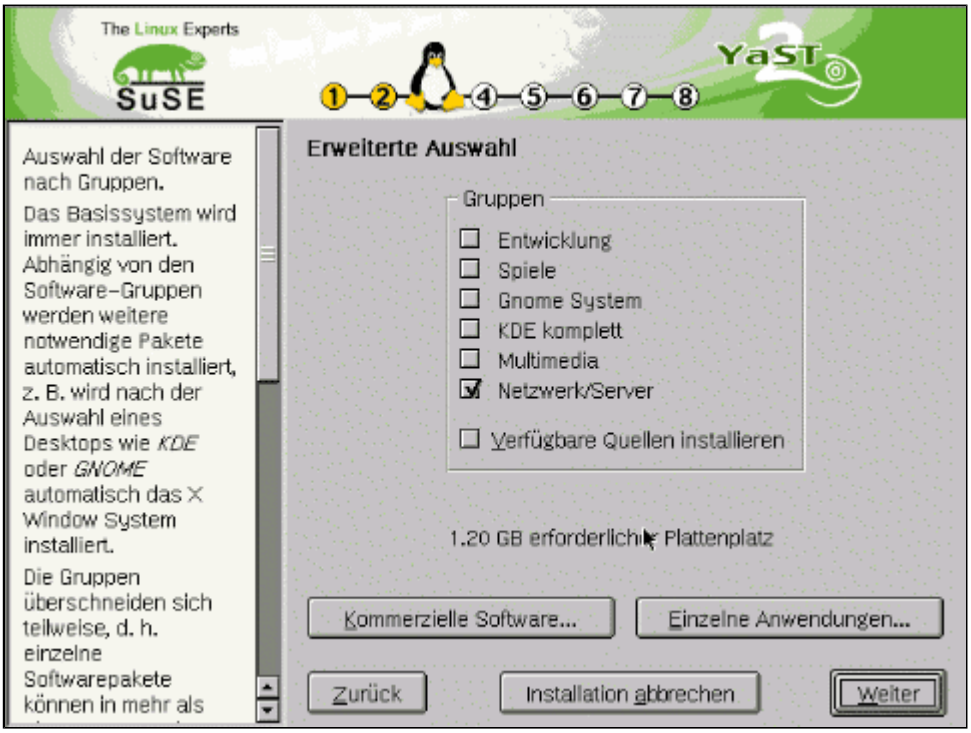


Abbildung 10: Software - Erweiterte Auswahl

Aber selbst die »Erweiterte Auswahl« ist für die gezielte Steuerung der Installation kaum ausreichend. Wer seine Platte nicht unnötig verstopfen will, muss zwangsläufig auf die »Einzelnen Anwendungen...« zurückgreifen.

Abbildung 11: Software - Einzelpaketauswahl

Die einzelnen Pakate sind thematischen Gruppen untergeordnet, sodass die Suche nach einer Anwendung schnell vonstatten geht. Bedingen Pakete das Vorhandensein bestimmter weitere Software, weißt Yast2 Sie im Anschluss an die Auswahl darauf hin und markiert betreffende Pakete selbsttätig zur Installation.

Ein Paket (de)markieren Sie durch Doppeklick auf den Eintrag oder durch Selektion und anschließender Betätigung des Schalters »Übernehmen«. Die »Beschreibung« listet neben einer kurzen Einführung zum Paketinhalt die enthaltenen Dateien auf.

Bootmanager

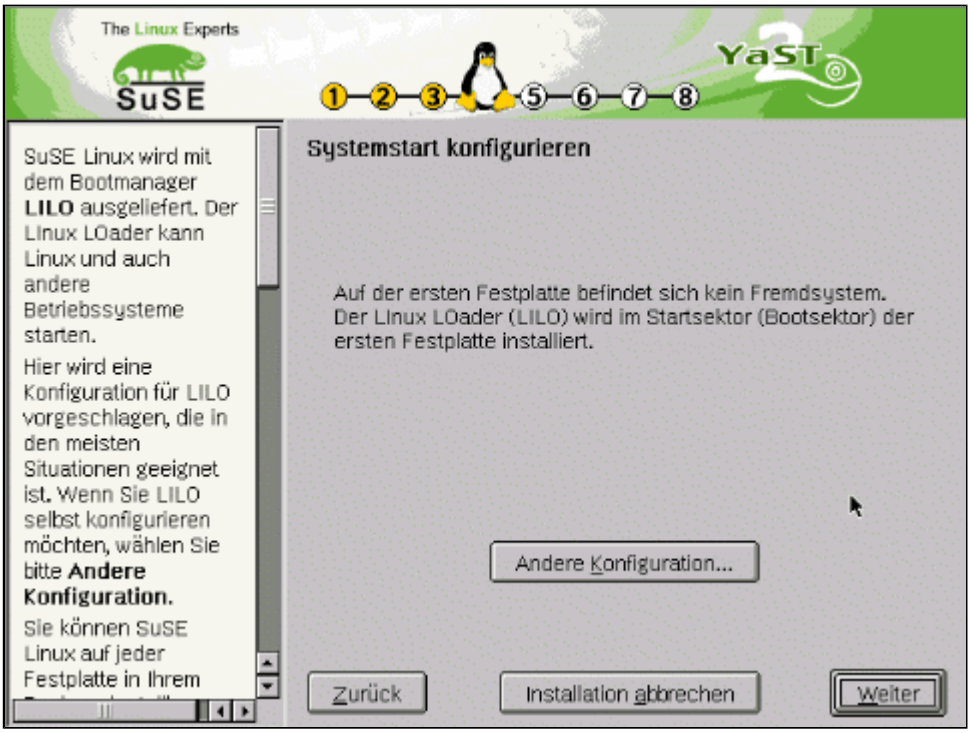


Abbildung 12: Lilo - Automatische Installation

Yast2 unterstützt derzeit nur den Linux Loader. Ist Linux das einzige System auf dem Rechner und es existiert nur eine Festplatte, bietet die nächste Maske die Installation des Bootmanagers in den Master Bootsektor (Mbr) an. In einem solchen Fall können Sie die Vorgabe bedenkenlos übernehmen. Bei mehreren Festplatten empfiehlt Yast2 die Installation in den Mbr der ersten Platte. Mit »Andere Konfiguration« können Sie Ihren eigenen Willen kundtun.

Erkannte Yast2 Fremdsysteme auf Ihrem Rechner, bietet es in der Voreinstellung die Lilo-Installation auf einer Diskette an. Für den Linux-Neuling ist das Verfahren sicher vorerst zu empfehlen; später lässt sich Lilo immer noch auf die Festplatte bannen. Dennoch ist die Diskettenlösung keine komfortable Variante; unter »Andere Konfiguration« kann daher eine alternative Installation erzwungen werden.

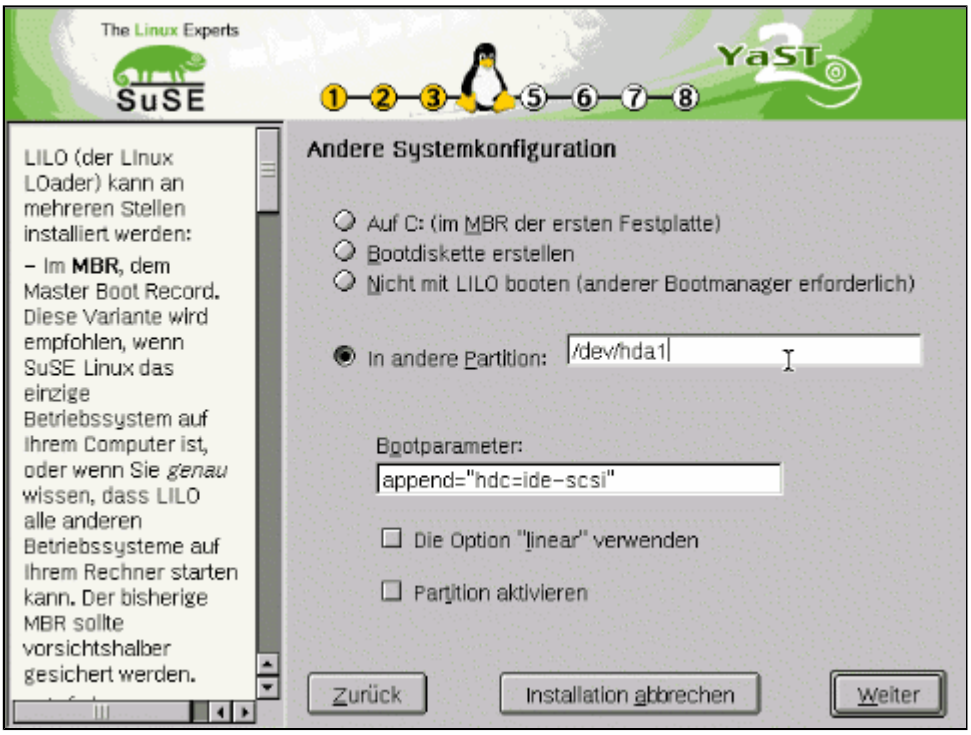


Abbildung 13: Lilo - Manuelle Konfiguration

Yast2 versucht erkannte Fremdsysteme (bspw. Windows) automatisch in die Konfiguration von Lilo einzubinden. Leider bietet der Dialog keine Möglichkeit, das Resultat vorab zu begutachten oder selbst Hand an die Konfiguration zu legen.

So besteht nur die Wahl ob der (Nicht)Installation des Bootloaders und ob dessen Installationsorts. Wenn Sie sich für den Master Bootrecord der ersten Festplatte entscheiden, könnte in einigen Fällen ein existierendes System nicht mehr starten, da eventuell dessen Bootcode somit überschrieben wurde. Für Windows-Systeme sollte Yast2 den Lilo entsprechend vorbereitet haben, aber Fehler sind nicht auszuschließen. Zum Testen kann der Bootmanager daher auf eine andere Festplatte installiert werden (2. Festplatte /dev/hdb (SCSI /dev/sdb), 3. Festplatte /dev/hdc (dev/sdc)...). Durch Änderung der Bootreihenfolge im BIOS des Rechners lässt sich dann der Lilo auf der »hinteren« Platte starten.

Wenn Sie Lilo in eine beliebige Partition packen, müssen Sie durch » Partition aktivieren« sicher stellen, dass diese auch bootbar ist. Allerdings gilt auch hier, dass eine andere bootbare Partition damit ihre Bootfähigkeit verliert. Informieren Sie sich im Abschnitt zum Linux Loader, wie Konfigurationen erfolgen, um mehrere Systeme durch Lilo bedienen zu können.

Benutzer und Rootpasswort

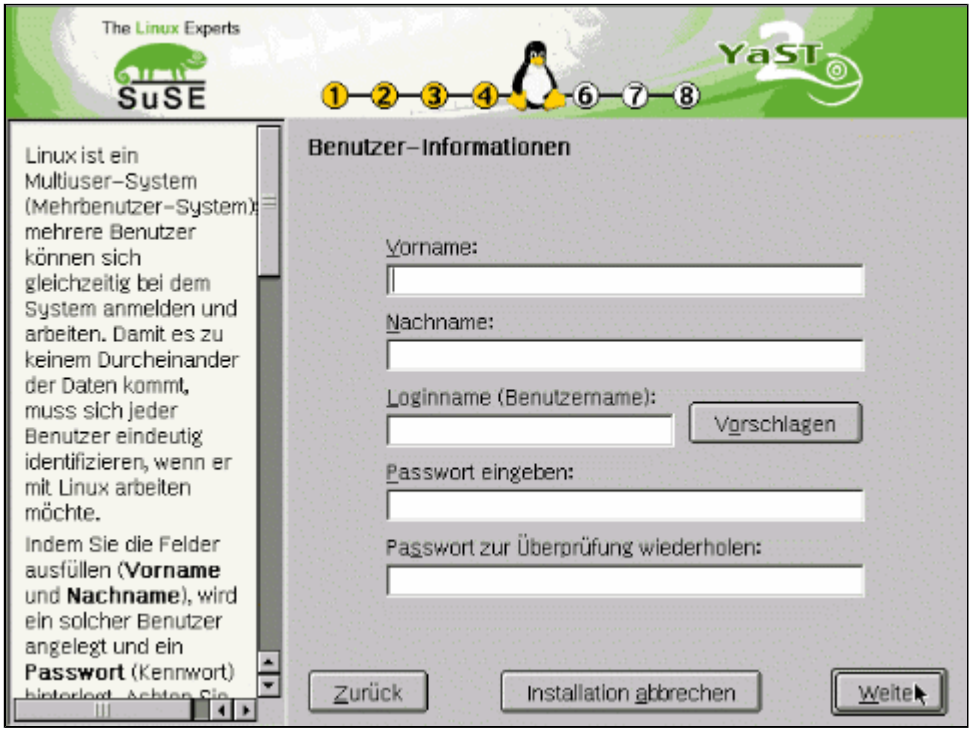


Abbildung 14: Anlegen von Benutzerzugängen

Da sich reichlich Gründe aufzählen ließen - und an mehreren Stellen des Buches werden Sie solche lesen können - die tägliche Arbeit unter Linux mit den Rechten eines »normalen« Benutzers zu verrichten, ist es sinnvoll, nun einen ersten Benutzer im neuen System einzurichten. In der Maske sind zwingend die Felder »Benutzername« und für das Passwort auszufüllen, die Namensangaben sind optional. Aus praktischen Gründen sollte der Benutzername maximal 8 Zeichen lang sein und keine Ziffern enthalten (einige Programme haben damit ihre Probleme). Das Passwort muss mindestens 5 Zeichen lang sein. Bis 128 sind möglich, wobei tatsächlich nur 8 betrachtet werden.

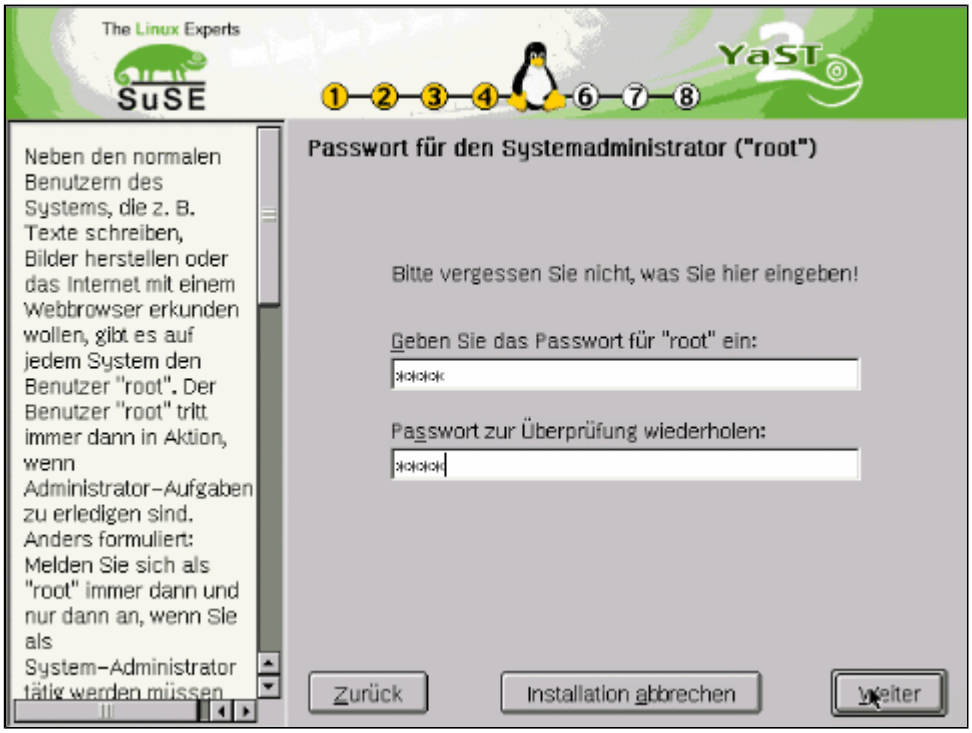


Abbildung 15: Passwort für Root

Für das Passwort des Systemadministrators »root« gelten dieselben Bedingungen der Länge von 5-8 Zeichen.

Bestätigung

Abbildung 16: Letzte Abbruchmöglichkeit

Die folgende Maske fasst noch einmal die getroffenen Einstellungen zusammen. Bis zu diesem Zeitpunkt besteht die Möglichkeit über den »Zurück«-Schalter einer jeden Maske die Konfiguration zu ändern. Erst durch »Weiter« beginnt die eigentliche Installation mit der Partitionierung und Formatierung der Festplatte(n).

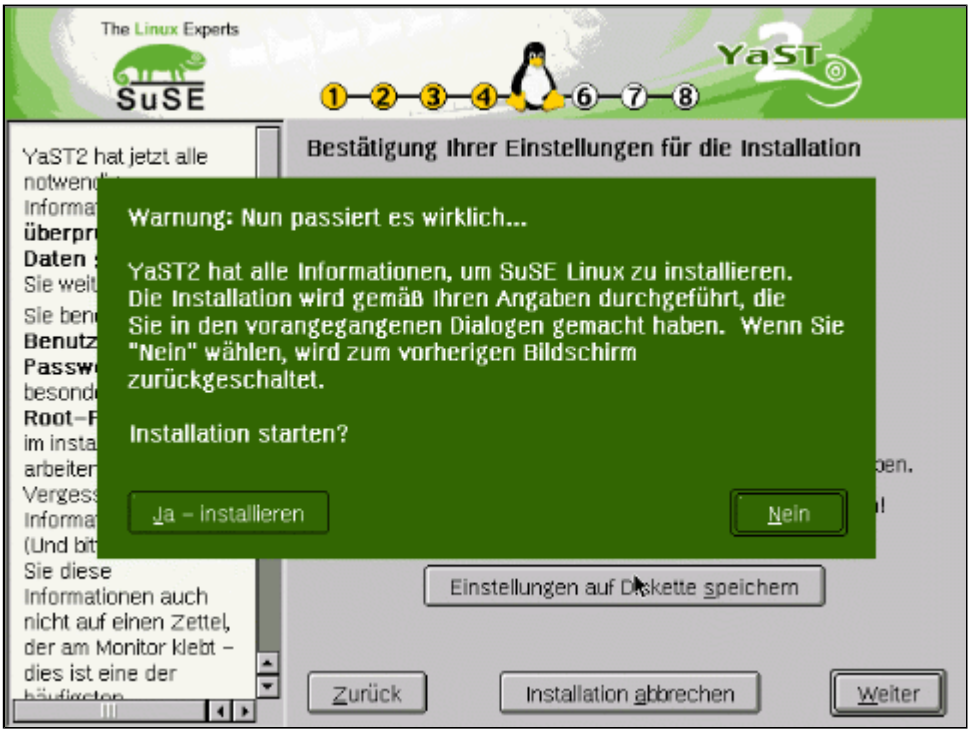


Abbildung 17: Warnung

Kaffeepause

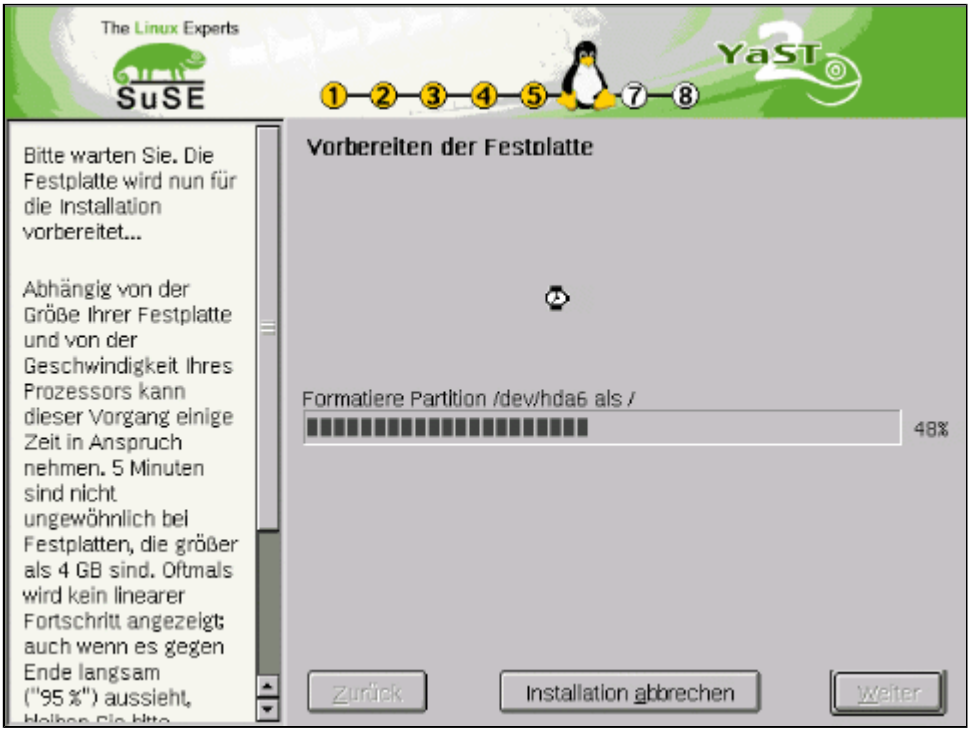


Abbildung 18: Die Festplatte wird formatiert...

Über den Fortschritt während der Formatierung der Festplattenpartitionen informiert eine Statusmaske. Bei großen Partitionen und schwachbrüstigen Prozessoren kann die Aktion mehrere Minuten in Anspruch nehmen.

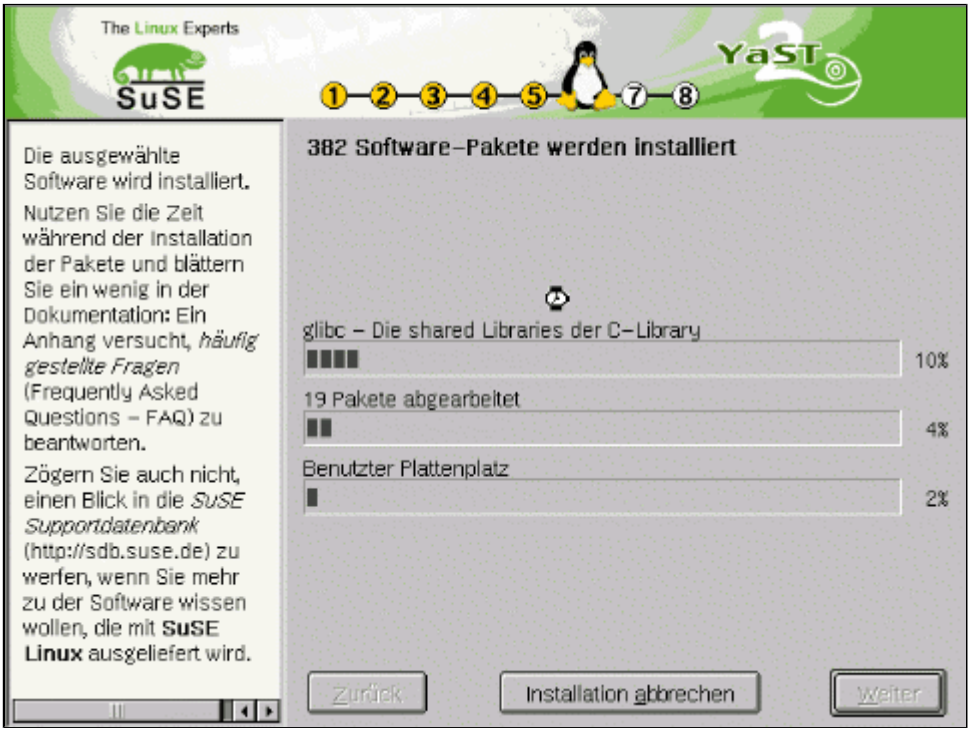


Abbildung 19: Die Software wird installiert...

Das Kopieren der Softwarepakete verkonsumiert in Abhängigkeit von Prozessorleistung und Paketumfang zwischen 5 Minuten (Minimalinstallation, schnelle CPU) bis zu mehreren Stunden bei einer vollen Installation. Ggf. fordert das Programm Sie zum Wechsel der CD auf. Wenn Sie den Vorgang abbrechen, wiederholt Yast2 nach dem Booten des neuen Systems die Forderung.

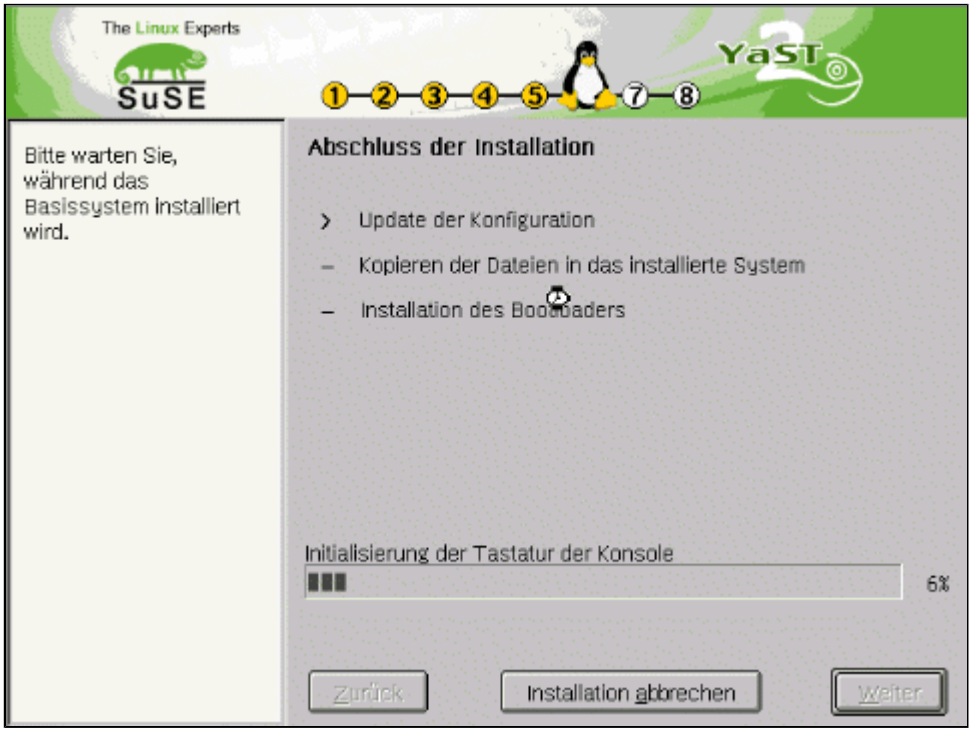


Abbildung 20: Abschluss der Softwareinstallation

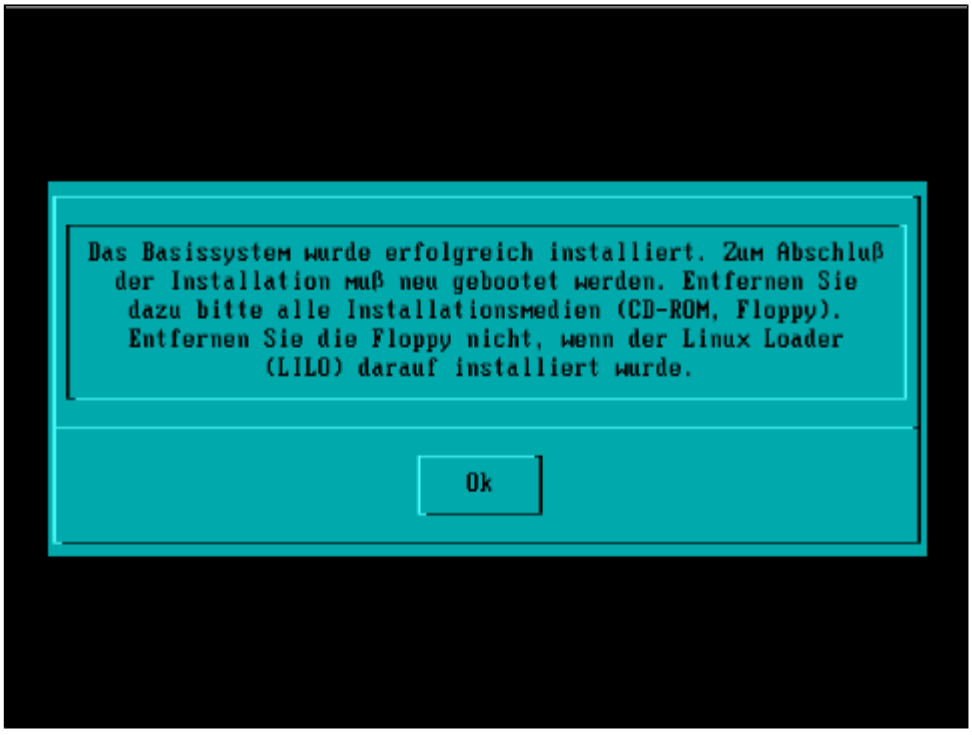


Abbildung 21: System-Neustart

Zum Abschluss werden Sie aufgefordert, die CD und Disketten zu entfernen. Das neue System wird gestartet. Yast2 übernimmt sofort wieder die Regentschaft.

Hatten Sie zuvor den CD-Wechsel verhindert, versucht Yast2 nun, die noch fehlenden Pakete zu installieren. Die anschließenden Schritte betreffen die Konfiguration des X-Servers.

X-Server konfigurieren

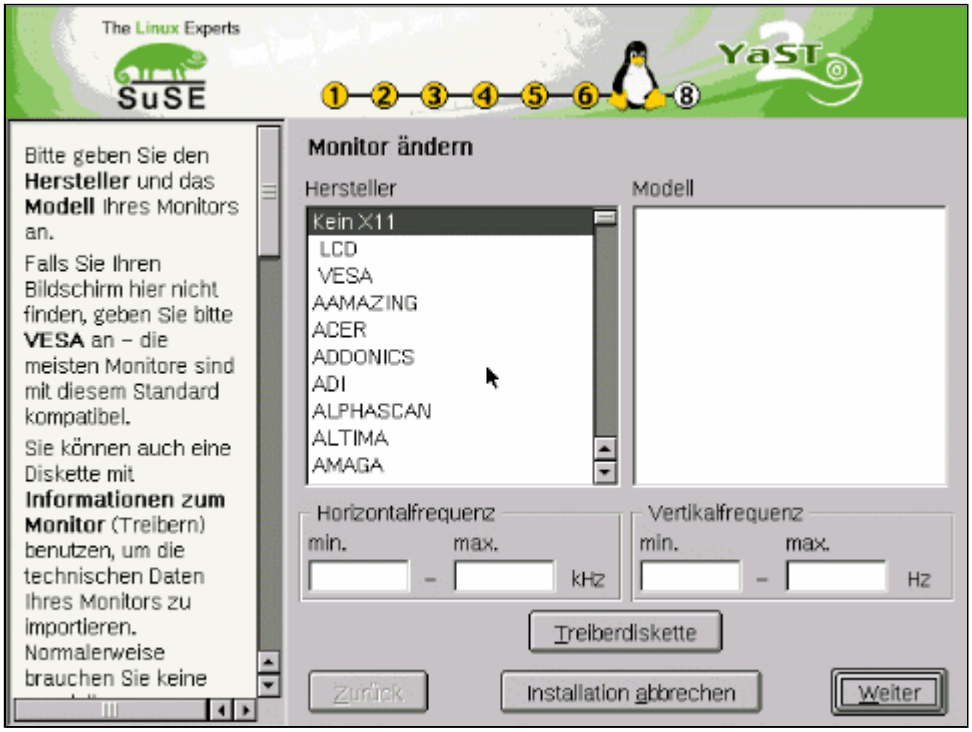


Abbildung 22: Monitorauswahl

Wenn in der Monitorliste Ihr Modell aufgeführt ist, wählen sie dieses. Ansonsten ist die Kenntnis der exakten horizontalen und vertikalen Frequenzen erforderlich, die Sie dem Handbuch zum Monitor entnehmen können. Analog verfahren Sie im Dialog zur Grafikkartenauswahl. Konfigurieren Sie eine Karte aber nur, wenn der aufgelistete Typ exakt Ihrer Hardware entspricht. Eine Grafikkarte ist doch recht nachtragend, wenn sie außerhalb ihrer Spezifikation betrieben wird. Mitunter segnet sie das Zeitliche. Bei sehr aktuellen Grafikkarten, die eine Weiterentwicklung einer Modellserie darstellen, kann meist der Treiber des Vorgängermodells verwendet werden. Eventuell gehen damit einige Vorzüge des neuen Stücks verloren, aber sie sollte sich so zur Arbeit überreden lassen.

Wenn Ihre Grafikkarte in keinster Weise erwähnt ist, dann überspringen Sie den Schritt. Yast2 versucht, die Karte in einem Standardmodus zu aktivieren.

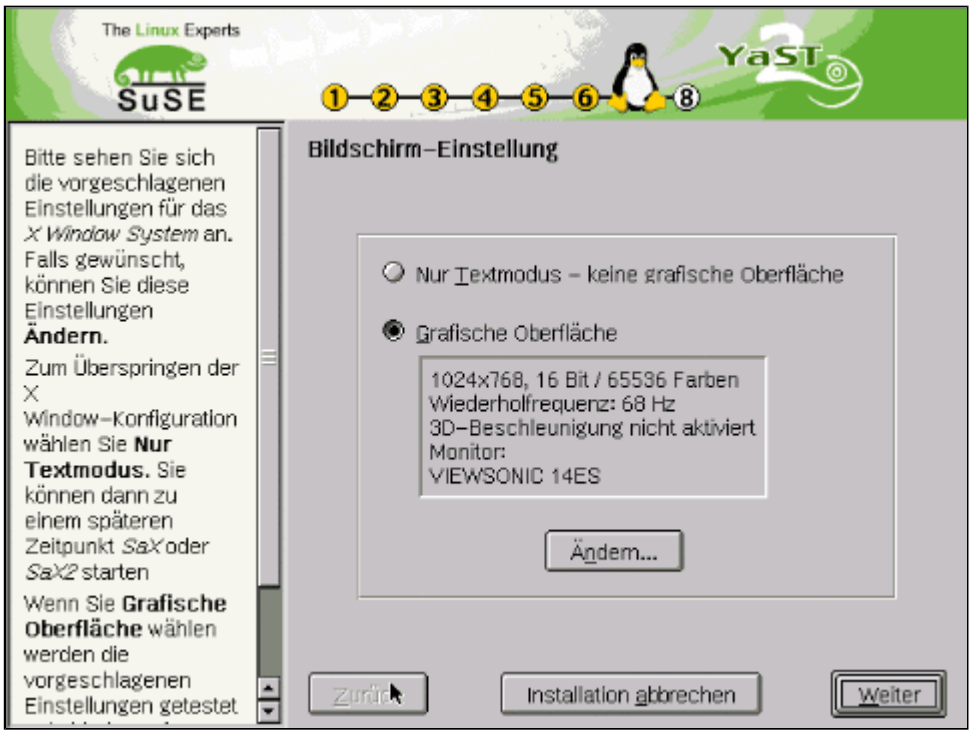


Abbildung 23: Verzicht auf die X-Server-Einrichtung?

Yast2 schlägt eine Farbtiefe und Auflösung für den Server vor. Sie können diese »Ändern...« oder die Konfiguration des X-Servers ganz auslassen.

Abbildung 24: Anpassung der Bildschirmeinstellung

Die Auflösung können Sie soweit erhöhen, bis entweder Ihr Monitor an seine Grenzen stößt (TFT arbeiten eigentlich nur mit einer Auflösung) oder aber Ihre Augen tränen.

Eine Farbtiefe von 16 Bit (65000 Farben) sollte den meisten Ansprüchen genügen. Aktuelle Grafikkarten mit ihrem großzügigen Speicherausbau erlauben auch 24 bzw. 32 Bit bei höchsten Auflösungen. Da steigende Farbtiefen auch eine gesteigerte Rechenleistung von der Karte erfordert, sollten sie den kleinsten Wert bevorzugen, bei dem Sie den optischen Eindruck von perfekter Farbwiedergabe haben. Eventuell spielen einige Grafikkartentreiber mit einigen Farbtiefen nicht zusammen, sodass die Wahlmöglichkeiten eingeschränkt sind.

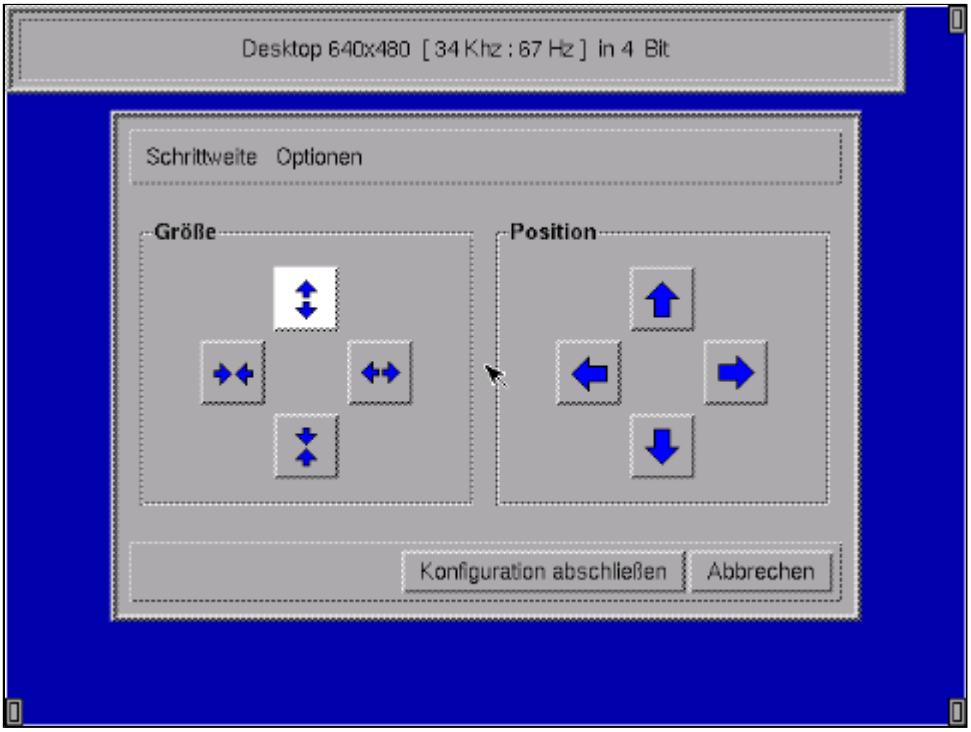


Abbildung 25: Feinabstimmung

Bevor die X-Einstellungen übernommen werden, werden diese getestet. Justieren Sie das Bild so, dass alle Ecken und Ränder vollständig und unverzerrt zu sehen sind. Die endgültige Bildgröße korrigieren Sie im Menü des Bildschirms.

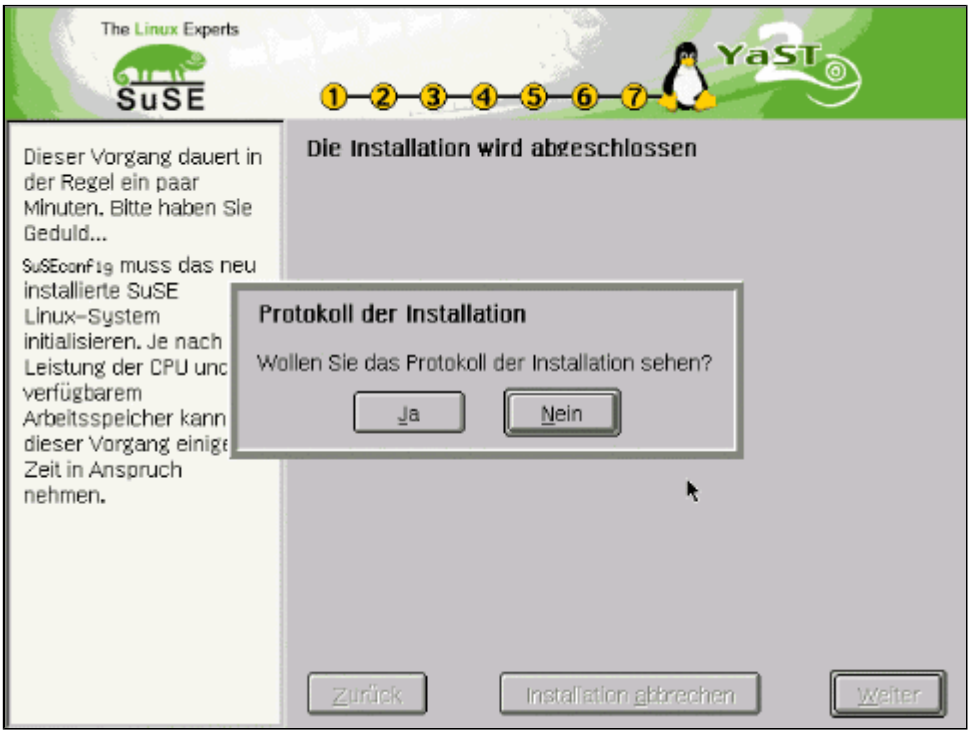


Abbildung 26: Abschluss der Installation

Der Blick ins Protokoll lohnt nur, wenn Sie Fehler während der Installation vermuten.

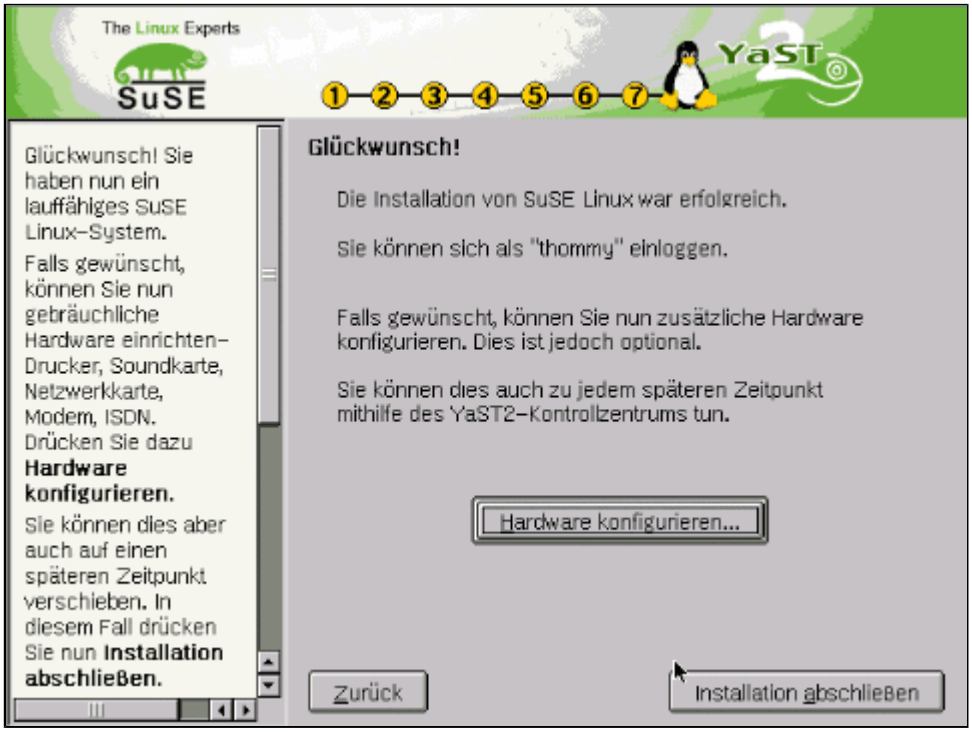


Abbildung 27: Weiter zur Hardware-Konfiguration

Sobald das Resultat Ihren Vorstellungen entspricht und Sie den Vorgang über »Konfiguration abschließen« beenden, bietet Yast2 die Konfiguration der weiteren Hardwarekomponenten des Systems an. Sie können diese Schritte auch später vollziehen, indem Sie als »root« yast2 aufrufen.

Mit Abschluss der Installation startet ein X-Login-Bildschirm, an dem Sie sich entweder als root oder unter dem weiter oben eingerichteten Benutzerzugang am System anmelden können.

Automatische Installation

Übersicht

Standen Sie schon einmal vor der Aufgabe, ein Computerkabinett mit Linux zu bestücken? Gelangen Sie oftmals in diese Situation, weil in der einen Woche Linux und in der anderen ein anderes System benötigt wird?

Spätestens hier bieten sich erste Überlegungen an, wie man den Installationsprozess automatisieren könnte. »**Diskette rein und [Enter]**« sollte doch genügen, um die unbeaufsichtigte Linuxeinrichtung auf einem Rechner in Gang zu setzen.

Und genau diese Möglichkeiten der Installation bringen nahezu alle heutigen Distributionen mit. Mit ein wenig einmaligem Aufwand reduziert sich das Prozedere auf das Einlegen einer Diskette...

Installationsserver?

Die gebräuchlichste - und bei **Fai** einzige - Methode ist wohl der Zugriff auf einen Installationsserver. Dieser Server besitzt die komplette zu installierende Software (und noch einige Skripte mehr); im einfachsten Fall werden die CD-ROMs der Distributionen in einem Verzeichnis des Serverrechner abgelegt.

Für Clients ohne Netzanbindung nützt ein Server herzlich wenig. Hier hilft nur die Beschränkung auf die Software, die die einzelnen Distributionen mit der ersten CDROM ausliefern (zumindest das Basissystem sollte vollständig enthalten sein). Der Client ist nun quasi sein eigener Server und das CDROM-Laufwerk das Quellverzeichnis.

Bei SuSE-Linux zieht diese Beschränkung leider auch eine Einschränkung nach sich - die Partitionierung der Festplatte kann nun nicht automatisiert ablaufen, da die notwendige Beschreibungsdatei auf der CDROM fehlt.

Wem allerdings die dargebotene Software ohnehin nicht genügt, der kommt um eine eigene Kreation einer Installations-CD nicht umhin; ergänzt man diese im Falle von SuSE-Linux um die Datei mit den Partitionierungsdaten, so steht einer Kaffeepause nichts im Wege...

Für die nachfolgenden Anleitungen gehen wir vom Zugriff auf einen Installationsserver aus; der Leser sollte die notwendigen Anpassungen bei rein lokalem Zugriff leicht nachvollziehen können.

Debian

Fai *Fully Automatic Installation* ist durchaus keine Floskel, erlauben die Routinen von Debian selbst die automatische Konfiguration zu installierender Anwendungsprogramme. Darüber hinaus lässt sich **Fai** ebenso als Recovery-Werkzeug nach einem Plattenausfall »missbrauchen«.

Fai automatisiert all jene Schritte, die der Administrator zum Aufsetzen eines kompletten Linuxsystems zu absolvieren hat. Ist die Konfiguration für einen Rechner erst einmal vollzogen, erfordert die Anpassung auf weitere »Clients« kaum Mehraufwand. **Fai** entfaltet seine Wirksamkeit umso eindrucksvoller, je größer der zu administrierende Rechnerpool ist. **Fai** kann prinzipiell auf jeder Distribution eingesetzt werden; die dafür notwendigen Eingriffe in die Shell- und Perlskripte bleiben jedoch dem Experten vorbehalten.

Ein wesentlicher Unterschied zu den von RedHat und SuSE entwickelten Mechanismen ist die Lage der Konfigurationsdateien. **Fai** speichert sie konsequent auf dem Installationsserver, womit das Erstellen einer einzigen Bootdiskette (bzw. mehrerer Kopien) für die Clients genügt.

Installationsserver

Beim Server muss es sich nicht zwingend um ein Linuxsystem handeln, auch wenn die nachfolgende Darstellung in

manchem Punkten davon ausgeht. Weiterhin nehmen wir vereinfachend an, dass alle notwendigen Dienste durch einen einzigen Server bereit gestellt werden. Aber auch das ist kein Muss. Wer sich in der Netzwerkadministration etwas auskennt, wird dem Text entnehmen können, welche Bestandteile auch andere Rechner übernehmen könnten.

Ein via Fai zu konfigurierender Client kennt in der Bootphase weder seinen Rechnernamen noch seine IP-Adresse. Beide Parameter (und ggf. anderes mehr) erfragt er anhand der weltweit eindeutigen MAC-Adresse seiner Netzwerkkarte bei einem DHCP- oder einem BOOTP-Server. Ein solcher muss somit vorhanden sein und seine Datenbasis muss Einträge für jeden Client umfassen. Ein Client mountet während der Installation sein Root-Dateisystem über das Network File System. Der Fai-Server ist als NFS-Server einzurichten. Des Weiteren benötigt der Client Zugang zu einem Archiv mit den Debian-Paketen. Im einfachsten Fall bietet gleich der Installationsserver ein solches via NFS an. Alternativen (denen wir nicht folgen werden) sind der Zugriff zu einem Debian-Spiegel über FTP oder HTTP.

Paketinstallation

Neben dem eigentlichen Fai-Basis-Paket empfiehlt sich die Installation des Pakets mit speziellen Fai-Kerneln, die die Fähigkeiten zum Booten übers Netz bereits enthalten. Es spricht natürlich nichts dagegen, einen solchen Kernel selbst zu erzeugen.

```
root@sonne> dpgk -i fai_1.4.2_i386.deb
Selecting previously deselected package fai.
(Reading database ... 29597 files and directories currently installed.)
Unpacking fai (from fai_1.4.2_i386.deb) ...
...
root@sonne> dpgk -i fai-kernels_1.0_i386.deb
Selecting previously deselected package fai.
(Reading database ... 30163 files and directories currently installed.)
Unpacking fai-kernels (from fai-kernel_1.0_i386.deb) ...
...
```

Ein lokaler Spiegel

Für die weiteren Aussagen nehmen wir an, dass der lokale Spiegel aus den Debian-Installations-CD's erzeugt werden soll. In den meisten Situationen dürfte dies wohl zutreffen.

Wie ist eine Debian-Installationsquelle aufgebaut? Unterhalb eines mit »dist« benannten Verzeichnisses existieren immer (mindestens) 4 Einträge:

```
root@sonne> ls dist/
frozen  potato  stable  unstable
```

»potato« ist der Name der aktuellen Debian-Distribution (ein Synonym für die sonst gebräuchliche Versionsnummer, dieser Name kann sich ändern). »frozen«, »stable« und »unstable« sind auf den CD's symbolische Links auf »potato«; sie sind als Verweise auf Entwicklerversionen einer Distribution gedacht.

Der Aufbau unterhalb eines solchen Verzeichnisses ist wiederum identisch (bei symbolischen Links erübrigt sich die Aussage):

```
root@sonne> ls dist/ potato
Contents-i386.gz  contrib  main  non-free
```

Hiermit realisiert Debian eine strikte Trennung der GPL-Software vom »Rest«. Unter »main« sammelt sich die Software, die der GPL und vergleichbarer freier Lizenzen unterliegt. »contrib« beinhaltet freie Software, die allerdings von »nicht freier« Software abhängt (bspw. KDE von der Freigabe der QT-Bibliotheken). »non-free« enthält schließlich die »nicht freien« Pakete.

Eventuell finden Sie noch ein Verzeichnis »non-US«, das Software enthält, deren Export durch US-Gesetze beschränkt ist (bspw. kryptografische Programme).

»Contents-i368.gz« enthält eine Zuordnung zwischen jeder im Debian-System verfügbaren Datei (mit Pfadangabe) und dem Paket, das diese enthält. Sie dient einzig der Information.

Eine weitere Stufe enthält für eine i386-Distribution folgende Verzeichnisse:

```
root@sonne> ls dist/ potato/ main
binary-all   binary-i386   sources
```

»binary-all« umfasst die Architektur unabhängigen Pakete, wie Schriften, Dokumentationen... und »binary-i368« beinhaltet - analog zum Serienkonzept von SuSE - Verzeichnisse, in denen letztlich die Pakete nach ihrem Verwendungszweck gruppiert sind:

```
root@sonne> ls dist/ potato/ main/ binary-i386
Packages.gz  Release  admin   base  comm
devel        doc      editors ...
```

Das Anlegen eines lokalen Spiegels läuft also darauf hinaus, die Verzeichnisstruktur der Installationsquelle auf dem Server nachzubilden und alle benötigten Pakete in das entsprechende Verzeichnis - laut Debian »Sektion« genannt - zu kopieren.

Einziger Haken ist die Beschreibungsdatei »Packages.gz«, die die Informationen zu den auf *dieser* CD befindlichen Paketen enthält:

```
root@sonne> zless dist/ potato/ main/ binary-i386/ Packages.gz
...
Package: adduser
Version: 3.11.1
Priority: required
Section: base
Maintainer: Guy Maor <maor@debian.org>
Depends: passwd (>=961025)
Architecture: all
Filename: dists/potato/main/binary-i386/base/adduser_3.11.1.deb
Size: 17274
MD5sum: 2f78fdd289c971374bd548c3da9c02a6
Description: Add users and groups to the system.
Adduser can create new users and ...
...
```

Debian benötigt während der Installation die Beschreibung aus der »Packages«-Datei. Für den Fall, dass der Inhalt mehrerer CD's komplett in den lokalen Spiegel übernommen wurde, lassen sich die verschiedenen Beschreibungsdateien einfach verketten (»zcat«).

Eine Möglichkeit, den Inhalt der Datei auf die tatsächlich gespiegelten Debian-Pakete abzustimmen, besteht im Auslesen der Paketinformationen und daraus generierter »Packages«-Datei:

```
root@sonne> cd dist/ potato/ main/ binary-i386/
root@sonne> for i in $(find . -name "*.deb"); do
> dpkg -I $i | sed -n '/^\ *Package.*/ ,$p' >> Packages
> echo "" >> Packages
> done
root@sonne> gzip Packages
```

Die zentrale Konfigurationsdatei /etc/fai.conf

Die Konfiguration von FAI nimmt das Skript **fai-setup** anhand der Angaben in der Datei /etc/fai.conf vor. Vermutlich sind einige der Variablen auf die lokalen Gegebenheiten anzupassen:

FAI_BASETGZ

FAI_SOURCE_LIST

Angabe der Lage der Debianquellen und der Zugriffsmethode. Die Syntax entspricht der der Datei **sources**

```
deb <Lage des Archivs> <Distributionsname> [Komponente(n)]
```

Die *Lage des Archivs* wird mit einem Schlüsselwort eingeleitet. Die drei Wichtigsten sind:

file	Die Quellen liegen lokal auf der Platte (in einem gemounteten Verzeichnis); der lokale Mountpunkt (MNTPOINT) und der Zugriffspfad zum NFS-Server (FAI_PACKAGEDIR) sind anzugeben.
http	Die Quellen werden über einen Webserver bezogen
ftp	Die Quellen liegen auf einem FTP-Server

Dem Schlüsselwort folgt, durch einen Doppelpunkt getrennt, der Zugriffspfad zum Archiv.

Der *Distributionsname* ist das schon weiter oben erwähnte Synonym für die Version, also bspw. »potato«, »woody« oder »stable«. Bei Zugriff auf mehrere Distributionen ist jede auf einer eigenen Zeile zu beschreiben.

Komponenten sind die Namen der zu verwendenden Verzeichnisse unterhalb des Distributionsverzeichnisses (bspw. »contrib«, »non-free«).

NFSROOT

Verzeichnisname, unter dem das NFS-Wurzelverzeichnis des Client eingerichtet werden soll.

KERNELPACKAGE

Name (inklusive Pfad) des Debian-Pakets, das den auf Clientseite zu bootenden Kernel enthält, das Paket w unter NFSROOT installiert.

FAI_ROOTPW

Das verschüsselte Passwort für Root

SSH_IDENTITY

Datei mit dem öffentlichen SSH-Schlüssel (identity.pub), um einem Benutzer (um dessen Schlüssel es sich handelt,) das passwortfreie Anmelden am Client als Root (!) via ssh zu gestatten. Das Setzen ist nur sinnvo wenn der Bootp- bzw. Dhcp-Server dem Client das Flag "sshd" übermittelt (Siehe: »Einen Bootp-Server einrichten«).

NFSROOT_PACKAGES

Liste zusätzlicher Debian-Pakete, die unter NFSROOT zu installieren sind.

Das folgende Beispiel einer Datei »fai.conf« sollte leicht an die eigenen Bedürfnisse anzupassen sein:

```
root@sonne> cat /etc/fai.conf
# /etc/fai.conf -- Konfiguration für FAI (Fully Automatic Installation)
#

# Damit die Pakete für die richtige Architektur verwendet werden
FAI_ARCH=`dpkg --print-installation-architecture`
```

```
# Lage der Datei mit dem Basissystem auf dem Fai-Server
FAI_BASETGZ=/usr/local/src/base2_2.tgz

# Wohin soll der Client den Debian-Spiegels mounten?
MNTPOINT=/mnt

# Wo liegt der Debian-Spiegel (NFS-Server)?
FAI_PACKAGEDIR=192.168.0.1:/opt/debian/export/debian_22

# Was soll von welchem Debian-Spiegel verwendet werden?
# Im Beispiel erfolgt der Zugriff auf ein NFS-Verzeichnis
FAI_SOURCES_LIST="deb file:$MNTPOINT/debian stable main contrib non-US/main"

# Verschlüsseltes Passwort für Root; wird auf den Clients gesetzt
FAI_ROOTPW="56hNVqht51tzc"

# Lage der Datei .identity.pub des Benutzers, der sich als Root am Client anmelden darf
# SSH_IDENTITY=/home/admin/.ssh/identity.pub

# Zusätzlich soll "ssh" installiert werden
NFSROOT_PACKAGES="ssh"

# Setzen der Zeitzone auf GMT; bei "no" wird "lokale Zeitzone" verwendet
UTC=yes

# Basisverzeichnis, sollte nicht verändert werden
LIBFAI=/usr/lib/fai

# Lage des NFS-Wurzelverzeichnisses (mit mind. 100MB freien Speicherplatz!)
NFSROOT=$LIBFAI/nfsroot

# Kernelpaket, das unter NFSROOT zu installieren ist
KERNELPACKAGE=/usr/lib/fai/kernel/kernel-image-2.2.17_BOOTP1_i386.deb

# Zu installierende Kernelversion (Kernel muss im Kernelpaket enthalten sein!)
KERNELVERSION=2.2.17

# Lage der weiteren Fai-Konfigurationsdateien auf dem Server
FAI_CONFIGDIR=/usr/local/share/fai
```

fai-setup

Nachdem die Konfigurationsdatei (fai.conf) für **fai-setup** angepasst wurde, erledigt der Aufruf des Skripts die eigentliche Einrichtung auf Serverseite:

```
root@sonne> /usr/sbin/fai-setup
Adding system user fai...
Stopping Name Service Cache Daemon: nscd.
Adding new user fai (100) with group nogroup.
Starting Name Service Cache Daemon: nscd.
Creating home directory /home/fai.
/home/fai/.rhosts created.
User account fai created.
Creating FAI nfsroot can take a long time and will
need more than 100MB disk space in /usr/lib/fai/nfsroot.
Unpacking /tmp/base2_2.tgz
...
```

Im Einzelnen vollbringt das Skript Folgendes:

1. Erstellen des Benutzers *fai*
2. Einrichten des NFS-Wurzelverzeichnisses mit
    - Installation des Basispakets base*xxx*.tgz
    - Installation der zusätzlichen Pakete
    - Ggf. Einrichten der SSH
    - Einbinden von Fai-Diensten in den Bootprozess (faireboot, faillog, fai_config,...)

4. Ggf. Anpassung der Datei /etc/exports und Neustart des NFS-Servers

fai-setup ist somit immer zu starten, sobald Sie Änderungen an der Datei /etc/fai.conf vorgenommen haben oder einen neuen Kernel installieren möchten.

Den Bootp-Server einrichten

Der Server selbst befindet sich bei Debian im Paket »net-boot.deb«. Details zum Aufsetzen eines Bootp-Servers erfahren Sie im Abschnitt DHCP&Co. unter Netzwerk-Server. Dort finden Sie auch die Erläuterungen zu den im nachfolgenden Beispiel verwendeten Optionen.

```
root@sonne> cat / etc/ bootptab
# »Makro«, das Paketgröße (ms), Verzeichnis der Bootdatei (hd), Bootdateigröße (bs) und NFS-Verzeichnis (rp) festlegt
# Zusätzlich wird dem Client sein Rechnername zugewiesen (hn)
.faiglobal:\
      :ms=1024:\
      :hd=/boot/fai:\
      :hn:\
      :bs=auto:\
      :rp=/usr/lib/fai/nfsroot:

# »Makro«, das die Wert aus dem Makro ».faiglobal« übernimmt
# Weitere Parameter betreffen den TFTP-Server (sa), Zeitserver (ts), Subnetzmaske (sm), Gateway (gw), Domainname (dn) und DNS-Server (ds)
# Erläuterung zu T.. folgt im Anschluss
.failocal:\
      :tc=.faiglobal:\
      :sa=192.168.100.1:\
      :ts=192.168.100.1:\
      :T170="FAISERVER:/usr/local/share/fai":\
      :T171="sysinfo":\
      :sm=255.255.255.0:\
      :gw=192.168.100.1:\
      :dn=galaxis.de:\
      :ds=192.168.100.1:

# Beschreibungen für jeden einzelnen Client
# Weitere Parameter sind Typ des Netzwerks (ht), die Hardwareadresse (ha), Name der Bootdatei (bf) und die IP des Client (ip)
faiclient01:\
   ht=ether:ha=0050569a0001:bf=faiclient01:ip=192.168.100.101:\
   tc=.failocal:T171="sysinfo":T172="sshd verbose":

faiclient02:\
   ht=ether:ha=00e07d79ac3b:bf=faiclient02:ip=192.168.100.102:\
   tc=.failocal:T171="install":T172="sshd":

faiclient03:\
   ht=ether:ha=00E07D79CBE9:bf=faiclient03:ip=192.168.100.103:\
   tc=.failocal:T171="showclasses":T172="sshd":
```

Ein Fai-Client benötigt Informationen, die über den üblichen Informationsgehalt eines Bootp-Servers hinausgehen. Um diese dennoch bereit zu stellen, greift der Server auf »Hersteller spezifische« Erweiterungen zurück, die sich hinter den mit **T..** beginnenden »Tags« verbergen.

Die derzeit von Fai verwendeten Tags sind:

T170

Verzeichnis, das die FAI-Konfiguration enthält

T171

Die so genannte FAI-Aktion, welche im Hauptinstallationsskript des Clients (rcS_fai) ausgewertet wird

T172

debug	Debugmodus; ggf. notwendige Konfiguration von installierten Paketen muss interaktiv vorgenommen werden
reboot	Rechner wird nach Abschluss der Installation neu gestartet
sshd	Der Secure Shell Daemon wird gestartet, um entferntes Anmelden zu ermöglichen
verbose	Aktiviert eine erweiterte Ausgabe während des Installationsvorgangs

Die Bootdiskette

Dem Fai-Paket liegt ein Programm zum Erzeugen der Bootdiskette bei, sodass sich die ganze Arbeit auf das Einlegen einer leeren Diskette und den Aufruf eines Kommandos reduziert:

```
root@sonne> /usr/sbin/make-fai-bootfloppy
```

Kopieren und Bearbeiten der Templates

Die bisherigen Vorarbeiten ermöglichen es einem Client, seine IP-Adresse von einem Server zu beziehen, den Kernel vom Server zu laden, zu booten und sein Root-Dateisystem via NFS zu mounten.

Das Ziel von **Fai** ist jedoch, auf Clientseite ein vollständiges Linuxsystem zu installieren. Der Client startet hierzu im folgenden Schritt das Programm **/sbin/rcS_fai**. Dieses Skript erzeugt zunächst eine Ramdisk und mountet diese nach /tmp - das vorerst einzig beschreibbare Verzeichnis auf dem Client.

Vom Fai-Server wird über NFS nun das Verzeichnis (vergleiche FAI_CONFIGDIR in fai.conf) mit den eigentlichen Konfigurationsdateien nach /fai gemountet. Genau jene Dateien beschreiben die Schritte, die ein Client nachfolgend zu vollziehen hat.

Natürlich steht es jedem frei, die Konfiguration komplett selbst zu erstellen. Einfacher ist jedoch die Verwendung der Templates aus /usr/share/doc/fai/templates und das Anpassen von Kopien an die eigenen Gegebenheiten:

```
root@sonne> cp -pR /usr/share/doc/fai/templates /usr/local/share/fai
```

Ein Blick unter /usr/local/share/fai fördert einige Verzeichnisse zu Tage:

class

Skripte und Dateien zur Definition von Klassen (siehe im Anschluss an diese Tabelle)

disk_config

Beschreibung der Partitionierung der Festplatten auf den Clients mit Dateinamen gleich Klassennamen

fai_config

Dateien mit Variablendefinition (Dateinamen gleich Klassennamen + global.conf) zur Überschreibung der Varaiblen aus /etc/rcS_fai.conf

files

Dateien für cfengine

package_config

Software-Paket-Konfigurationen mit Dateinamen gleich Klassennamen

scripts

cfengine-Skripts mit Dateinamen gleich Klassennamen

Schritt 1: Definition der Klassen

Jeder Konfigurationsschritt, der auf dem Client zu vollziehen ist, wird durch eine so genannte Klasse beschrieben. Da für verschiedene Clients sicherlich verschiedene Konfigurationen sinnvoll sind, ermitteln Skripte die Klassennamen, die nachfolgend gelten sollen. **rcS_fai** arbeitet hierzu - ähnlich dem Ressource Control Script eines »normalen« Linux-Bootvorgangs - die im Verzeichnis **/ fai/ class**(nach dem Mounten liegt es dort!) liegenden und mit "S[0-9][0-9]" beginnenden Skripte in der durch die alphabetische Anordnung gegebenen Reihenfolge ab.

Bei den Skripten kann es sich wohl um Bash- (Endung *.sh) als auch um Perlskripte (Endung. *pl) handeln; jedes Wort, das ein Skript auf die Standardausgabe schreibt, wird als Klassenname interpretiert.

Beispiel: Ein typisches Skript könnte anhand des Rechnernamens gleichnamige Klassen definieren, um für jeden Client oder eine Gruppe von Clients eine dedizierte Konfiguration vorzunehmen:

```
root@sonne> cat / usr/ local/ share/ fai/ class/ S01alias.sh
#! /bin/sh

case $HOSTNAME in
  erde)
    echo ERDE
    ;;
  disk??)
    echo DATALESS
    ;;
esac
```

Etwas vereinfacht ausgedrückt, testen die Skripte unter **class** im Wesentlichen die lokale Hardware und ermitteln daraus die Klassenname zur Steuerung der weiteren Konfiguration. Zusätzlich werden immer die Klassennamen $HOSTNAME und LAST gesetzt.

Schon der nächste Schritt von **rcS_fai** verdeutlich das Konzept der Klassennamen. Das Skript sucht im Verzeichnis **/ fai/ class** nach Dateien mit dem Namen ***Klassenname*.var** und führt sie aus, wodurch bestimmte Shellvariablen mit konkreten Werten belegt werden. Folgendes Beispiel (aus den Quellen des Fai-Pakets) setzt eine Variable **hdparm**, um Festplattenparameter zu konfigurieren:

```
root@sonne> cat / usr/ local/ share/ fai/ class/ ATA33.var
# enable ultra ATA/33 modus for hard disk hda
# create etc/rcS.d/S61hdparm

# if defined, this line is executed and written to /etc/init.d/S61hdparm
hdparm='hdparm -c1 -d1 -m16 -X66 /dev/hda'

# tune harddisk during installation
[ "$hdparm" ] && eval $hdparm
```

Schritt 2: Partitionieren der Festplatte(n), Erzeugen der Dateisysteme

Existiert das Skript **/ usr/ local/ bin/ my-fdisk**, so übernimmt dieses die Partitionierung und das Einrichten der Dateisysteme. Auf diese Weise kann die Ausführung von **setup_harddisks** durch **rcS_fai** verhindert werden. Letzteres Skript durchsucht das Verzeichnis **/ fai/ disk_config**, ob ein Klassenname mit dem Namen einer Datei übereinstimmt. Die erste passende Datei (bei sauberer Konfiguration sollte es nur eine geben) wird eingelesen und anhand der Angaben die Festplatte(n) eingerichtet:

```
root@sonne> cat / usr/ local/ share/ fai/ disk_config/ 4GB
# disk configuration for one disk with up to 4GB disk space

# <type> <mountpoint> <size in mb> [mount options]  [;extra options]
```

```
disk_config hda

primary  /          50         rw,errors=remount-ro ;-c
logical  swap       200        rw
logical  /var       200        rw
logical  /tmp       100-250                         ;-m 0
logical  /usr       700        rw
logical  /scratch   0-         rw,nosuid            ;-m 0 -i 50000
```

Schritt 3: Software-Installation

Zunächst wird das Basissystem (vergleiche FAI_BASETGZ aus /etc/fai.conf) entpackt und installiert. **rcS_fai** liest nachfolgend jede in **/ fai/ package_config** enthaltene Datei ein, deren Name einem definierten Klassennamen entspricht. Auf diese Art und Weise kann einfach eine clientabhängige Software-Ausstattung erzielt werden:

```
root@sonne>  cat / usr/ local/ share/ fai/ package_config/ NFSSERVER
PACKAGES install
nfs-server
```

Schritt 4: Lokale Konfiguration

Zu einem kompletten System gehört noch die Konfiguration verschiedenster Dienste. So ist das Netzwerk einzurichten, der Druckerzugriff zu gewährleisten oder die Uhrzeit zu setzen...

Zuständig für diesen abschließenden Schritt sind die Skripte im Verzeichnis **/ fai/ scripts**. Auch hier werden wiederum genau jene Skripts zur Ausführung gebracht, deren Namen mit eingangs definierten Klassennamen übereinstimmen. Zulässig sind Bash-, Perl-, Expect- und Cfengine-Skripts. Manche dieser Skripts werden vorgefertigte Konfigurationsdateien ins System einspielen. Solche Vorgaben können im Verzeichnis **/ fai/ files** gesammelt werden.

Der letzte Schritt

Nach Abschluss werden die Protokolldateien des Installationsvorgangs nach **/ var/ log/ fai/ $HOSTNAME/ install/** geschrieben und der Rechner neu gebootet.

Der Installationsvorgang beim Client

Genau genommen sind drei Handlungen von Nöten:

1. Bootdiskette einlegen
2. BIOS auf »Booten von Floppy« einstellen
3. Abwarten und Tee trinken

Die im Verborgenen ablaufenden Vorgänge sind dann doch um Einiges komplexer:

1. Booten des Kernels (Rechnername und IP-Adresse wird über BOOTP/DHCP ermittelt)
2. Mounten des Root-Dateisystems via NFS
3. Start von **Fai** (Mounten von /fai)
4. Bestimmen der Klassen
5. Ggf. Laden von Kernelmodulen
6. Partitionierung der Festplatte(n)
7. Erzeugen der Dateisysteme
8. Installation des Grundsystems
9. Installation zusätzlicher Pakete
10. Anpassung der Konfiguration
11. Sicherung der Protokolldateien sowohl lokal als auch auf dem Installationsserver
12. Reboot des neu installierten Systems

RedHat

Für die unbeaufsichtigte Installation hat sich bei RedHat der Name **Kickstart** eingebürgert. Die Tätigkeit des "Installationsservers" beschränkt sich hier einzig auf die Bereitstellung der RPM-Pakete; die gesamte Steuerung der Installation liegt in der Hand des Clienten; er bestimmt über die Aufteilung der Festplatte, über den Umfang der Installation usw.

Echte administrative Arbeit steht somit nur bei der Erzeugung der Bootdiskette und der Anpassung der darauf befindlichen Konfigurationsdateien an.

Installationsserver

Wie schon angedeutet, stellt der Server einzig die Software-Pakete zur Verfügung. In den meisten Fällen wird es sich um einen NFS-Server handeln, so dass die Schritte:

1. Einrichten eines NFS-Servers
2. Anlegen eines Verzeichnisses (bspw. /export/RH_70)
3. Kopieren aller notwendigen Software-Pakete nach /export/RH_70:
   - Die komplette CDROM 1
   - Den Inhalt aus dem Verzeichnis "images" der CDROM 2
   - Weitere Pakete nach Bedarf
4. Exportieren des neuen Verzeichnisses

durchzuführen sind.

Alternative Quellen für die Pakete sind lokale CDROMs, lokale Festplatten oder eine Webseite (URL).

Die Bootdiskette

Auf der zweiten CDROM Ihrer RedHat-Distribution finden Sie im Verzeichnis "images" die beiden Dateien **boot.img** und **bootnet.img**. Erstere Datei sollten Sie auf die Bootdiskette kopieren, wenn sich die zu installierenden Pakete auf einem lokalen Medium (CDROM, Festplatte) befinden; letztere, falls Sie auf ein Verzeichnis im Netz zugreifen:

```
# Installation von CDROM oder Festplatte
root@sonne> dd if=boot.img of=/dev/fd0

# Installation von einem NFS-Server oder URL
root@sonne> dd if=bootnet.img of=/dev/fd0
```

Auf der Diskette finden Sie nun im Wurzelverzeichnis die Datei **syslinux.cfg** (es handelt sich um die Konfigurationsdatei für den syslinux-Bootloader), die Sie ggf. an Ihre Umgebung anpassen müssen;

```
root@sonne> mount /dev/fd0 /mnt/floppy
root@sonne> cat /mnt/floppy/syslinux.cfg
default linux
label linux
  kernel vmlinuz
  append initrd=initrd.img devfs=nomount lang=de ks=floppy
prompt 0
timeout 10
```

Erläuterung: Der Name des Eintrags lautet "linux" (die "default"-Zeile ist genau genommen unsinnig, da nur ein einziger Kernel gebootet werden kann). Der zu startende Kernel ist "vmlinuz". Die "append"-Zeile definiert die Parameter, die dem Kernel beim Start als Argumente zu übergeben sind. Hier wird eine Ramdisk verwendet, Informationen dazu finden Sie im Kapitel Systemadministration unter Der Bootvorgang. Die Anzeige eines Prompts wird abgeschaltet und mit "timeout" wird die Verweildauer (in Milisekunden; 0 bedeutet "Warten auf eine Eingabe") des Eingabeprompts angegeben.

Die wesentliche Administration findet in der Datei **ks.cfg** statt, die Sie ebenfalls im Wurzelverzeichnis auf der Diskette finden. Hier wird sowohl die Partitionierung der Festplatte beschrieben, als auch die Liste zu installierender Pakete spezifiziert. Des Weiteren lassen sich Skripte angeben, die unmittelbar vor bzw. nach der Installation auszuführen sind.

Auf Einhaltung einer bestimmten Reihenfolge der Einträge in der Datei ist dringend zu achten:

1. **Angabe der Schlüsselworte**, wobei die zugehörigen Parameter auf einer Zeile stehen müssen (die Reihenfolge der Anordnung der Schlüsselworte untereinander spielt keine Rolle):

```
# ALLGEMEINE ANGABEN

# Auswahl der Sprache
# lang : en_US, fr_FR, it_IT, ru_RU.KOI8-R, ..
lang de_DE

# Auswahl der Zeitzone (Angaben analog zu 'timeconfig')
# --utc :                       # BIOS Uhr auf GMT
timezone --utc Europe/London

# HARDWAREKONFIGURATION

# Auswahl der Tastatur
# keyboard : azerty, uk, us, fr, ...
keyboard de-latin1-nodeadkeys

# Auswahl der Maus
# mouse
# --device <Mausgerät> :            # ps/2, logitech, microsoft, none ..
# --emulthree :           # Emulation einer 3-Button-Maus
mouse --device ps/2

# KERNELMODULE

# Die Angaben sind optional, da versucht wird, die Hardware automatisch zu erkennen und notwendige Module zu laden (eine Angabe kann dem
"Autoprobing" in manchen Fällen "auf die Sprünge" helfen)
# Netzwerk und SCSI
# device [eth|scsi] <module>          # entsprechende Kernelmodule
device eth pcnet32
# device scsi aic7xxx

# SICHERHEIT

# Authentifizierung
# auth : Default : verschlüsselt, aber kein shadow
# --enablemd5 :                 # md5 Verschlüsselung
# --useshadow :                 # Shadow Passwörter
# NIS (Network Information System)
#
# --enablenis :                 # NIS einschalten
# --nisdomain :                 # NIS Domäne
# --nisserver :                 # NIS Server
auth --useshadow --enablemd5

# Passwort von Root
# rootpw <Passwort>
# --iscrypted <Verschlüsseltes_Passwort>
rootpw Guiness

# NETZWERK

# Diese Angabe ist nur bei Installation übers Netz notwendig; die Parameter bedeuten:
# --device : eth0, ..           # Device-Name
# --ip :                        # IP-Adresse
# --hostname :                  # Name des Rechners
# -- netmask :                  # Netzmaske
# --bootproto : dhcp, bootp, static     # Bootmethode; Default : dhcp
network --device eth0 --ip 172.16.91.100 --hostname pc20 --netmask 255.255.255.0 --gateway 172.16.91.100 --names
172.16.91.1 --bootproto static

# INSTALLATIONSMETHODE

# Entfällt die Angabe, wird die CDROM als Installationsmedium angenommen
# Mögliche Paketquellen sind:
# CDROM :                       # Voreinstellung
# NFS :                         # NFS-Server
# HARDDRIVE :                   # lokal auf der Festplatte
```

```
# URL :                          # aus dem Internet

nfs --server 172.16.91.1 --dir /export/RH_70/
# cdrom
# harddrive --partition <Verzeichnis> dir <Verzeichnis>
# url --url http://<Server>/<Verzeichnis>

# X WINDOW

# Diese Angaben sind optional und machen nur Sinn, wenn die X-Pakete installiert wurden
# Falls nicht angegeben, dann manuelle Konfiguration
# xconfig :
# --card <Karte aus Xconfigurator>      # Grafikkarte
# --monitor <Monitor aus Xconfigurator> # Monitor
# oder
# --hsync <Wert> mit --vsync <Wert>     # Frequenzwerte des Monitors
# --defaultdesktop=[GNOME|KDE]          # Standarddesktop
# --startxonboot                        # Grafisches Login
# xconfig

# Falls X Windows installiert, aber keine Konfiguration gewünscht
skipx

# LILO

# Konfiguration von LILO
# --location : mbr, none, partition (auf Partition mit Kernel)
# --append :                          # Kernelparameter
lilo --location mbr --append mem=128M

# Mit den optionalen Parameter "lilocheck" kann geprüft werden, ob der Linux Loader bereits installiert ist. Falls ja, wird keine Installation vorgenomm
(Verhinderung einer Neuinstallation!)
# lilocheck

# NACH DER INSTALLATION...

# Falls ein automatischer Neustart am Ende der Installation erfolgen soll (ohne die Angabe wird der Benutzer gefragt):
# reboot

# PARTITIONIERUNG

# Die optionale Angabe von "clearpart" ermöglicht das vorherige Löschen von Partionen (Voreinstellung: kein Löschen)
clearpart --linux                     # nur Linux-Partitionen
# clearpart --all                     # alle Partitionen

# Für jeder anzulegende Partition ist deren Mountpunkt, deren Lage (welches Device) und die Größe in MByte anzugegebn:
part /boot  --ondisk hda --size 10
part swap   --ondisk hda --size 128
part /      --ondisk hda --size 1024
```

2. Liste der zu installierenden Pakete oder Serien; eine Beschreibung befindet sich auf der CDROM unter "/RedHat/base/comps":

```
# Eine vorangestellten "@" deklariert den Namen als eine Serie; sonst werden die Angaben als Paketnamen interpretiert
% packages
@ Base
@ X Window System
# @ Printer Support
# @ GNOME
# @ Mail/WWW/News Tools
# @ Multimedia Support
# @ Networked Workstation
# @ Dialup Workstation
tcsh
XFree86-xf86cfg
XFree86-VGA16
XFree86-Mono
XFree86-SVGA
```

3. Optionales Pre- und Post-Install-Skript:

```
# Die beiden Skripte werden direkt in der Datei formuliert
# Das Pre-Install-Skript:
% pre
echo "Pre-Install-Skript!"
```

```
# Das Post-Install-Skript
% post
echo "Post-Install-Skript!"
```

Eine typische Anwendung des Preinstall-Skripts ist das Anlegen einer Sicherungskopie der durch den Installationsvorgang zu überschreibenden Partitionen. In einem Postinstall-Skript ließen sich bswp. erste Benutzer im System einrichten...

Der Installationsvorgang beim Client

Die Schritte, die der Client ab dem Booten von der Diskette durchläuft, sind:

1. Booten des Kernels (mit bzw. ohne Netzwerk)
2. Anlegen einer Ramdisk
3. Start von **linuxrc**
4. Auslesen der Datei ks.cfg
5. Zugriff auf das Installationsverzeichnis (bei Zugriff auf einen NFS-Server wird dieses als Rootverzeichnis gemountet)
6. Start von **Anaconda** (RedHat-spezifisches Konfigurationstool)
7. Partitionieren der Festplatte
8. Skripte, die vor der Installation abzuarbeiten sind, werden gestartet
9. Installation der Pakete unterhalb von /mnt
10. Skripte, die nach der Installation abzuarbeiten sind, werden gestartet
11. Wechsel des Rootverzeichnisse (neue Wurzel ist /mnt)
12. Reboot (automatisch, falls in der ks.cfg "reboot" aufgeführt ist; sonst nach Bestätigung)

SuSE

Installationsserver

Der Server ist zunächst als NFS-Server einzurichten. Für die Software, die später auf den Clients zu installieren ist, benötigen Sie ein eigenes Verzeichnis. Dieses darf **nicht suse** heißen, da das Skript **linuxrc** in diesem die RPM-Pakete erwartet; SuSE selbst diese aber in einem Unterverzeichnis **suse** ablegt. **linuxrc** kann den Pfad "suse/suse" (warum auch immer) nicht auflösen. Die NFS-Freigabe dieses Stammverzeichnisses wird später in den Skripten als **$I:** referenziert; nachfolgend soll dieses Symbol für das von Ihnen angelegte Verzeichnis stehen. Vergessen Sie nicht, das neue Verzeichnis in die Datei "/etc/exports" aufzunehmen und dem NFS-Server ein SIGHUP zu senden.

Kopieren Sie den Inhalt der SuSE-CD's, den Sie benötigen (also zumindest die 1.CD komplett) in dieses neue Verzeichnis. Die verschiedenen CD's enthalten teils identische Dateien (Indexverzeichnisse), diese werden im Zielverzeichnis (logischer weise) nur einmal benötigt und können getrost überschrieben werden.

Zusätzliche RPM-Pakete, die nicht zum Umfang der SuSE-Distribution gehören, lassen sich unter **$I:/ suse/ ADD_RPMS/** ablegen und werden automatisch mit installiert.

Zusätzliche Skripte lassen sich nach dem selben Schema in ein Verzeichnis **$I:/ suse/ ADD_SCRIPTS/** integrieren. Diese können unmittelbar vor oder nach der Installation gestartet werden (dies wird in der Datei **info** fest gelegt).

Ein sinnvolles Skript, das unmittelbar nach der Installation ausgeführt werden sollte, betrifft die Vergabe des Rootpasswortes. Dieses könnte wie folgt ausschauen:

```
user@sonne> cat suse/ ADD_RPMS/ post_install.sh
#!/bin/sh

# Damit niemand die Ausgaben sieht, leiten wir sie in den Mülleimer
exec > /dev/null 2>&1
```

```
ORIG=/mnt/etc/shadow
TEMP=/mnt/etc/shadow.tmp
LOG=/mnt/var/adm/inst-log/post.sh.log

echo "Postinstall Script!"
# Das verschlüsselte Passwort lautet: "DAbzbyj39/Lq"; wir setzen es in der Passwortdatei:
sed 's/root::/root:DAbzbyj39\/Lq\.:/' $ORIG > $TEMP

cat $TEMP > $ORIG
rm $TEMP

# Die Logdatei sollte nur Root lesen dürfen:
chmod 600 $LOG
```

Eine Datei **AutoSuSE.sel** (der Name kann frei gewählt werden; die Endung ".sel" ist bindend) im Verzeichnis **$I:/ suse/ setup/ descr** beinhaltet die Liste der zu installierenden Pakete. Am einfachsten ist es, im **YaST1** unter "Konfiguration erstellen/ändern" eine solche Auswahl zu erzeugen:

```
user@sonne> cat AutoSuSE.sel
# SuSE-Linux Configuration YaST Version 1.03 -- (c) 1994-2000 SuSE GmbH
# generated on Sat Jul 29 20:41:09 GMT 2000

Description.english: AutoSuSE
Description.french: AutoSuSE
Description.german: AutoSuSE
...

Info.english:
Ofni.english:
...

Kind: baseconf

Visible: true

Toinstall:
aaa_base
aaa_dir
aaa_skel
ash
base
bash
bc
...
# Endekennung nicht vergessen (reine SuSE-Syntax)
Llatsinot:
```

Da es sich um eine ASCII-Datei handelt, steht einer manuellen Anpassung nichts im Wege (man muss nur die Paketnamen kennen).

Ein Client kann sich bez. der Softwareauswahl nur auf die Vorgaben einer solchen Datei beziehen; benötigt man mehrere Konfigurationen, ist für jede eine Paketauswahldatei zu generieren.

Auf dem Server fehlen einzig noch die Dateien mit den **Partitionierungsanweisungen**. Sie müssen im Verzeichnis **$I:/ suse/ setup/ descr** liegen und nennen sich **part_*XXXXX***, wobei *XXXXX* eine fünfstellige Zahl > 0 ist, die die Anzahl des für das System anzulegenden Speicherplatzes in Megabyte angibt.

Existieren mehrere part_*XXXXX*-Dateien im Verzeichnis, wählt der Client selbsttätig diejenige, deren Größe bez. der Festplattenkapazität am besten passt (die also kleiner oder gleich dieser ist).

```
user@sonne> cat suse/ setup/ descr/ part_02000
# Partitionstabelle
# Schlüsselwörter :
# size=nnnn          # Größe in MB, 0 entspricht dem Rest
# fsys=reiser        # Reiserfs
# id=xx              # Id der Partition
```

```
# bs=nnnn            # Blockgröße
/boot   size=10 bs=1024
swap    size=64
/       size=0 bs=4096 fsys=reiser
```

Erläuterung: 2 GByte Plattenplatz werden wie folgt verteilt:

- Die erste Partition /boot ist 10 MByte groß; es wird ein Dateisystem **ext2** (Voreinstellung) mit einer Blockgröße von 1024 Bytes (bs=...) erzeugt
- Die zweite Partition wird mit 64 MByte Kapazität angelegt und wird als Swap formatiert
- Der Rest (physische Plattenkapazit&aumlt >=2 GB - 74MB) wird das Rootdateisystem, das mit 4k-Blöcken und ReiserFS zu formatieren ist

Die Bootdiskette

Auf der ersten SuSE-CD finden Sie im Verzeichnis **images/** die Datei **boot.img**, die auf die Diskette zu kopieren ist:

```
root@sonne> dd if=boot.img of=/dev/fd0
```

Mounten Sie anschließend die Diskette und passen die dortige Datei **syslinux.cfg** (es handelt sich um die Konfigurationsdatei für den syslinux-Bootloader) an:

```
root@sonne> mount /dev/fd0/floppy
root@sonne> cat /floppy/syslinux.cfg
label linux
  kernel linux
  append initrd=initrd rw ramdisk_size=65536 linuxrc=auto

timeout 1
```

Erläuterung: Der Name des Eintrags lautet »linux«, ebenso nennt sich der zu startende Kernel »linux«. Die »append«-Zeile definiert die Parameter, die dem Kernel beim Start als Argumente zu übergeben sind. Hier wird eine Ramdisk verwendet, Informationen dazu finden Sie im Kapitel Systemadministration unter Der Bootvorgang. Mit »timeout« wird die Verweildauer (in Millisekunden; 0 bedeutet »Warten auf eine Eingabe«) des Eingabeprompts angegeben.

Als abschließenden Schritt müssen Sie nun die Datei **info** bearbeiten, die Sie im Wurzelverzeichnis auf der Diskette finden. Hier findet die eigentliche Konfiguration statt, d.h. welchen Server Sie kontaktieren, welche Dateien Sie von diesem benutzen usw. Die folgende Beispieldatei wurde bewusst um ausführliche Kommentare ergänzt und kann als Ausgangspunkt für eigene Experimente dienen:

```
user@sonne> cat info
# Konfiguration der Installation (Sprache, Bildschirm, Tastatur)
#
Language:          german       # oder: english
Display:           color
---------------------------------------------------------------
# oder: mono
Keytable:          de-lat1-nd   # oder: us, fr-latin1, it

# Angabe des Bootmodus (hier übers Netz)
#
Bootmode:          Net          # oder: CD

# Konfiguration des Netzwerkes (Laden eines Moduls, um die Netzwerkkarte anzusprechen)
#
insmod pcnet32                  # Laden von Modulen
Netdevice:         eth0         # Netzwerkinterface
IP:                172.16.91.100  # IP-Adresse
Netmask:           255.255.255.0  # Netmaske
# Konfiguration der Netzinstallation (Installationsserver, Name des Installationsverzeichnisses...)
#
```

```
Gateway:          172.16.91.1     # IP-Adresse des Gateway
Nameserver:       172.16.91.1     # IP-Adresse des Nameservers
Server:           194.180.239.232 # IP-Adresse des NFS-Server
Serverdir:        /export/SuSE_70 # exportiertes NFS-Verzeichnis

# Angabe des CD-ROM's für Installation
#
CDROM_DEVICE      /dev/hdc

# Schlüsselwörter fur Installation
#

# Soll Installation automatisch geschehen?
# FAST_INSTALL    0       # Nein
# FAST_INSTALL    1       # Ja, aber nur mit Bestätigung
FAST_INSTALL      2       # Ja ohne Bestätigung

# Was soll bei Problemen geschehen?
NEVER_STOP        0       # Benutzerabfrage
# NEVER_STOP      1       # Defaultwert und weiter

# Soll automatisch partitioniert werden?
# AUTO_FDISK      0       # Nein
# AUTO_FDISK      1       # Ja, aber nur mit Bestätigung
AUTO_FDISK        2       # Ja ohne Bestätigung

# Frage nach Verwendung der SWAP-Partition
# NO_ASK_SWAP     0       # Ja
NO_ASK_SWAP       1       # Nein

# Angabe der Sel-Datei für Paketauswahl
AUTO_INSTALL      $I:/suse/setup/descr/AutoSuSE.sel

# Interaktion, wenn ungelöste Pakeabhängigkeiten auftreten?
CHECK_DEPENDENCY  0       # Nein
# CHECK_DEPENDENCY 1      # Ja

# Interaktion nach Beendigung der Paketinstallation?
INSTALL_WAIT      0       # Nein
# INSTALL_WAIT    1       # Ja
# Angabe des zu installierenden Kernels
AUTO_KERNEL       k_deflt.rpm     # Standardkernel
# AUTO_KERNEL     k_laptop.rpm    # Kernel mit APM Unterstützung
# AUTO_KERNEL     k_ide.rpm       # Kernel für spezielle IDE Chipsätze
# AUTO_KERNEL     k_smp.rpm       # Kernel mit SMP Unterstützung
# AUTO_KERNEL     k_i386.rpm      # Kernel für i386
# AUTO_KERNEL     MeinKernel      # eigener Kernel unter suse/images/MeinKernel.ikr

# Soll LILO automatisch konfiguriert weren?
# AUTO_LILO       0       # Nein
# AUTO_LILO       1       # Ja mit Bestätigung
AUTO_LILO         2       # Ja ohne Bestätigung

# Sollen Netzparametern automatisch übernommen werden?
# Frage: Loopback/Netzwerk bis Sendmail-Abfrage
# AUTO_NET        0       # Nein
AUTO_NET          1       # Ja

# Soll der Nameserver (von bootp, DHCP ..) übernommen werden?
# AUTO_NAMESERVER 0       # Nein
AUTO_NAMESERVER   1       # Ja

# Soll der Rechnernamen (durch bootp, DHCP ..) übernommen werden?
# AUTO_NAME       0       # Nein
AUTO_NAME         1       # Ja, Name entsprechend IP-Adresse

# Sollen die Parameter des Netzwerkservice abgefragt werden?
# AUTO_SERVICES   0       # Ja
AUTO_SERVICES     1       # Nein

# Frage nach Installation ob System gebootet werden soll.
# END_MESSAGE     0       # Nein
END_MESSAGE       1       # Ja

# Setzen der notwendigen rc.config-Parameter.
RC_CONFIG_0       TIMEZONE        MET
RC_CONFIG_0       SENDMAIL_TYPE   no
```

```
# Tastendruck für Skripte (Konsole 9)
# nicht notwendig
END_STARTUP             0
# END_STARTUP           1

RC_CONFIG_0             START_GPM       no
RC_CONFIG_0             MODEM
RC_CONFIG_0             MOUSE           /dev/psaux

# RC_CONFIG_0           START_LOOPBACK yes
# RC_CONFIG_0           FQHOSTNAME      pc20.saxedu.de
# RC_CONFIG_0           SEARCHLIST      saxedu.de
# RC_CONFIG_0           NAMESERVER      192.168.42.100
# RC_CONFIG_0           SMTP            no

# Setzen von rc.config-Parameter zur Konfiguration des Systems.
### some system setup #
### RC_CONFIG_0         GMT -u
### RC_CONFIG_0         START_INEtd no
### RC_CONFIG_0         START_SSHD yes
### RC_CONFIG_0         START_GPM yes
### RC_CONFIG_0         START_ROUTED no
### RC_CONFIG_0         START_NAMED no
### RC_CONFIG_0         ROOT_LOGIN_REMOTE no
### RC_CONFIG_0         MODEM
### RC_CONFIG_0         FH_ADVANCED_CONFIG yes
###
### RC_CONFIG_0         DISPLAYMANAGER    kdm
### RC_CONFIG_0         CONSOLE_SHUtdOWN  halt
### RC_CONFIG_0         KDM_SHUtdOWN      all
```

Die Erzeugung der Bootdiskette ist damit abgeschlossen.

Der Installationsvorgang beim Client

Die Schritte, die der Client ab dem Booten von der Diskette durchläuft, sind:

1. Booten des Kernels (dessen NFS-Unterstützung ist zwingend erforderlich, falls ein Installationsserver Verwendung findet)
2. Anlegen einer Ramdisk
3. Die Info-Datei wird gelesen
4. Mounten des Installationsverzeichnisses vom Server nach /var/adm/mount
5. YaST1 startet
6. Partitionieren der Festplatte
7. Skripte, die vor der Installation abzuarbeiten sind, werden gestartet
8. Installation der Pakete unterhalb von /mnt
9. Skripte, die nach der Installation abzuarbeiten sind, werden gestartet
10. Wechseln des Rootverzeichnis (/mnt wird zur neuen Wurzel) oder Start des neuen Kernels, falls der "alte" nicht der richtige ist (bspw. SMP)

Installation - Bootmanager

Übersicht

Aus der Fülle kommerzieller und freier Bootmanager hat sich im Linuxumfeld nur ein einziger Kandidat tatsächlich etabliert. Dabei handelt es sich um den mit Linux groß gewordenen **Lilo** (*Linux Loader*).

Lange Zeit sah es aus, als könnte der mit ähnlichen Fähigkeiten ausgestattete und um eine gefällige Oberfläche bereicherte **chos** (*Choose Operating System*) eine ernsthafte Alternative für den Lilo darstellen. Jedoch ist seine Entwicklung ins Stocken geraten und aus so mancher Distribution ist er bereits wieder verschwunden...

Und dennoch wildert mit dem **Grub** (*Grand Unified Boot Loader*) ein weiterer Kandidat im Revier des Lilo. Vor allem dessen unerreichter Funktionsumfang ermächtigt ihn, die Herrschaft über das Bootgebahren zu übernehmen.

Ein Bootloader, oft auch als Bootmanager bezeichnet, selbst ist ein kleines Programm, das beim Hochfahren des Rechners vom BIOS desselben aufgerufen wird. Die wesentliche Aufgabe des Bootloaders ist nun, die Datei mit dem Betriebssystem in den Hauptspeicher zu laden und diesem anschließend die Kontrolle über den Rechner zu übertragen.

Dass die Linux-Bootmanager noch eine Reihe weiterer Möglichkeiten implementieren, werden Ihnen die folgenden Abschnitte vor Augen führen. Und da es doch hin und wieder erforderlich sein soll, Windows NT oder 2000 einzusetzen, werden auch dem Zusammenspiel der Systeme einige Worte gewidmet.

Lilo - Linux Loader

Abbildung 1: Begrüßung durch den Lilo

Genau genommen handelt es sich bei **Lilo** um verschiedene Programme und Dateien. Das Kommando **lilo** ist dabei der so genannte Map-Installer, der alle für den Bootmanager benötigten Dateien an die entsprechenden Stellen im System platziert. **lilo** ist stets aufzurufen, sobald sich an irgend einer für den Bootmanager wichtigen Datei (/etc/lilo.conf, Kernel, /boot/map) etwas geändert hat.

Der Bootmanager **lilo** selbst besteht aus zwei Stufen. Die erste wird in einem vom BIOS erreichbaren Sektor (Mbr, Bootsektor einer Partition, Diskette...) abgelegt. Ihre Aufgabe ist einzig das Laden der zweiten Stufe (meist /boot/boot.b) in den Hauptspeicher.

Die zweite Stufe kennt nun anhand der Datei »/boot/map« die Lage der verschiedenen Kernel bzw. der zu ladenden Bootsektoren anderer Betriebssysteme. Zur Auswahl dieser bietet **lilo** ein einfaches Kommandozeileninterface an, das neben der Selektion des Betriebssystems auch die Übergabe von Optionen an dieses erlaubt. Alternativ präsentieren neuere Programmversionen eine grafische Oberfläche. Diese kann ggf. durch Eingabe von [Esc] verlassen werden, um Optionen an einen Kernel zu übergeben.

Lilo lädt nun den Kernel in den Hauptspeicher, teilt diesem die Lage des Root-Verzeichnisses und eventuelle

Optionen mit und übergibt abschließend die Kontrolle an den Kernel.

Das Problem mit einem alten BIOS

Das 1024-Zylinder-Problem wurde bereits im Abschnitt Vorbereitung der Installation besprochen. Knapp dargestellt, vermögen viele - selbst neuere! - BIOS-Implementierungen Festplatten größer 1024 Zylinder (bei heutigen Platten entspricht dies 8.4Gbyte) nicht zu adressieren. Dies erzwingt die Lage des Bootmanagers innerhalb dieser 1024 Zylinder.

Für Lilo selbst *kann* dies einige Einschränkungen bedeuten:

- Die Dateien /boot/map und /boot/boot.b müssen innerhalb der ersten 1024 Zylinder liegen
- Alle Kernel müssen innerhalb der ersten 1024 Zylinder liegen
- Bootsektoren anderer Betriebssysteme, die von **lilo** verwaltet werden sollen, müssen innerhalb der ersten 1024 Zylinder liegen

Die neuesten **lilo**-Implementierungen (Versionen ab 2.0) vermögen Fremdsysteme und Kernel auch von Zylindern >1024 zu booten, allerdings bedarf es bestimmter BIOS-Versionen (post 1998, für die meisten älteren Mainboards bieten die Hersteller entsprechende Updates an).

Die Datei /etc/lilo.conf

Die Datei /etc/lilo.conf ist die Konfigurationsdatei des Bootmanagers **lilo**.

Beginnen wir mit den wichtigsten **globalen Optionen**, die für alle eingetragenen Images und Bootsektoren gelten. Den einzig zwingenden Eintrag leitet das Schlüsselwort **boot** ein und legt den Ort fest, wohin der Lilo installiert werden soll:

```
boot= /dev/hda
```

Im Beispiel ist der Mbr der ersten IDE-Festplatte Ziel der Installation. Während des Bootens zeigt der Lilo mit »lilo:«, sofern die Option **prompt** in die Konfiguration aufgenommen wurde, die Bereitschaft zur Entgegennahme einer Eingabe an. Wie lange er wartet, kann mit der Option **timout= Zehntel Sekunden** eingestellt werden. Nach Ablauf der Zeit startet Lilo automatisch das erste in der Konfiguration aufgeführte Image, außer eine Option **default= Name** benennt ein anderes Image als das Voreingestellte. Eine Option **read-only** bewirkt das schreibgeschützte Mounten des Root-Dateisystems und beschleunigt mitunter dessen Überprüfung. Einen weiterer Versuch zur Beschleunigung des Startvorganges erlaubt die Option **compact**, die Lilo anweist, benachbarte Sektoren »in weiser Voraussicht« in einem Ritt zu lesen.

Die Installation des Lilo überschreibt den bisherigen Inhalt des Zielsektors. Wer sich seiner Sache nicht sicher ist, dem sei ein Backup des alten Inhalts empfohlen. Ein entsprechender Eintrag lautet:

```
backup= /boot/bootsector_save
```

Bevor **lilo** beim nächsten Aufruf den Bootsektor überschreibt, wird es den alten Inhalt in die Datei /boot/bootsector_save sichern. Um diese Datei später wieder als Bootsektor zu platzieren, kann die Option **install= / boot/ boot_sector_save** in die "/etc/lilo.conf" aufgenommen werden. Lilo nimmt den Inhalt der angegebenen Datei und schreibt diesen in den Zielsektor, anstatt die Informationen aus /boot/boot.b zu beziehen.

Lilo kann auch zu etwas Höflichkeit angehalten werden, indem er während des Starts eine Begrüßung auf den Bildschirm schreibt. Den Text schreibt man hierzu in eine ASCII-Datei und teilt die Lage dieser dem Bootmanager mit:

```
message= /boot/greetings.txt
```

Damit **lilo** die Installation bei Zugriff auf Bereiche jenseits der 1024er Grenze nicht verweigert, ist eine Option **lba32** aufzunehmen.

Die Möglichkeiten der globalen Einstellungen sind hiermit noch lange nicht erschöpft, die wichtigsten sollten jedoch genannt worden sein.

Die nun diskutierten Optionen betreffen die von Lilo zu verwaltenden Images und gliedern sich nochmals in **allgemeine Kerneloptionen** und **Linux Kerneloptionen. Werden sie im Rahmen der globalen Optionen angegeben, so gelten sie für alle Kernel, in denen sie zulässig sind.**
Die Definitionen zu einem Linuxkernel werden mit image= *Pfad_zum_Kernel* eingeleitet. Alle nachfolgenden Einträge bis hin zu einer neuen image- oder einer other= *Partitionsname*-Zeile betreffen diesen Kernel. Mit dem other-Eintrag lädt Lilo den Bootsektor der angegebenen Partition und übergibt diesem die Kontrolle. Dieses so genannte »Chain Loading« ist der einzige Weg, Nicht-Linux-Systeme zu starten.

Jedes Betriebssystem muss einen eindeutigen Namen zugewiesen bekommen label = Name, im Falle des Linuxsystems sollte Lilo die Lage des Wurzelverzeichnisses root= *Partitionsname* mitgeteilt werden. Zwar enthält ein Kernel selbst die Information, wo sein Root-Dateisystem liegt (es wird ihm zur Kompilierzeit oder per »rdev« mitgeteilt), aber zumindest die Standardkernel beinhalten selten den richtigen Ort. Damit ergibt sich als minimaler Eintrag für einen Linuxkernel:

```
image = /boot/vmlinuz
root = /dev/hda2
label = Linux
```

Bedient Lilo mehrere Images, erfolgt deren Auswahl am Lilo-Prompt durch Eingabe des vollständigen Label-Namens. Der Schreibaufwand lässt sich minimieren, indem zu einem Image ein Alias-Name definiert wird alias= *Anderer_Name*. Wählt man als Alias ein einzelnes Zeichen, muss sicher gestellt sein, dass kein anderer Name mit diesem Zeichen beginnt. Fügt man nun noch die Option single-key in den globalen Abschnitt ein, genügt das Drücken der einen Taste, um ein Betriebssystem zu laden:

```
...
single-key

image = /boot/vmlinuz
root = /dev/hda2
label = Linux
alias = 1

other = /dev/hdb
label = dos
alias = 2
```

Das obige Beispiel birgt eine Falle: DOS weigert sich, von einem anderen Laufwerk als dem Laufwerk C zu starten. Aber auch hierfür kennt Lilo einige Kniffe und gaukelt DOS eben das erwartete Laufwerk vor. Mit der Option map-drive lassen sich die Zuordnungen gezielt anpassen:

```
...
other = /dev/hdb
label = dos
alias = 2
map-drive= 0x81
  to= 0x80
map-drive= 0x80
  to= 0x81
```

Aus Sicht von DOS ist Laufwerk D nun C und C ist D. An dieser Stelle sei nur erwähnt, dass sich mit der Option change ganze Partitionen vor einem System verbergen oder deren Eigenschaften sich ändern lassen.

Eine hilfreiche Option ist sicherlich der Schutz des unbefugten Startens einzelner Systeme. Jedes Image kann mit password= *Passwort* durch ein Passwort geschützt werden. Lilo fordert zur Eingabe dessen auf, wenn es an den Start des Systems geht. Sollten Sie von dieser Möglichkeit Gebrauch machen, dann stellen Sie sicher, dass "/ etc/ lilo.conf" nur für Root lesbar ist, sonst wäre der Effekt

derselbe, als würden Sie die Geheimnummer auf der Rückseite Ihrer Kreditkarte notieren. Etwas moderater wird die Passwortabfrage mit der zusätzlichen Option restricted. Hier wird dessen Eingabe nur notwendig, wenn der Benutzer Optionen an den Kernel übergeben möchte. Bei einem simplen Start entfällt die Abfrage.

Für Linuxkernel ist häufig die Übergabe von Argumenten an den Kernel erwünscht. Dies kann der mit der Option append= "*Parameterliste*" erfolgen. Typische Anwendung ist, dem Kernel die Parameter zu Hardware mitzuteilen, die nicht automatisch erkannt werden kann. Z.B. erkennt der Kernel keine zweite Ethernetkarte, jedoch kann er zur Suche aufgefordert werden:

```
append = "ether=0,0,eth1"
```

Eine Diskussion aller Parameter findet der interessierte Leser im Boot-Prompt-HowTo.

Wer vor einem großen Monitor sitzt, wird bald die voreingestellten Monsterzeichensätze verdammen und sich etwas mehr Information auf der Mattscheibe wünschen. Mit vga=ask wird im Falle eines Linuxkernels Lilo beim Start des betreffenden Images eine Liste aller verfügbaren VGA-Modi darstellen und den Nutzer auffordern, sich für einen zu entscheiden. Hat man letztlich einen brauchbaren Font gefunden, kann das ask auch durch die konkrete Bezeichnung des Modus ersetzt werden, so dass das System immer mit diesem startet.

Eine vollständige Datei "/ etc/ lilo.conf", die zwei Linuxkernel und ein DOS-System unterstützt, könnte wie folgt aufgebaut sein:

```
boot = /dev/hda
timeout = 50
prompt
compact
vga = ask

# Die folgende Zeile ist nur bei Verwendung einer initialen Ramdisk notwendig; entfernen Sie hierzu das Kommentarzeichen:
#initrd=/boot/initrd

password= NixFuerDich
restricted
single-key

image = /boot/vmlinuz.orig
root = /dev/hdb3
label = Linux_Standardkernel
alias = 1

image = /boot/vmlinuz.opt
root = /dev/hdb3
label = Linux_optimierter_Kernel
alias = 2

other = /dev/hdb
label = dos
alias = 3
map-drive= 0x81
  to= 0x80
map-drive= 0x80
  to= 0x81
```

Kommandozeilenoptionen

Unmittelbar nach dem Laden von Lilo, überprüft dieser, ob eine der Tasten [Shift], [Ctrl] oder [Alt] gedrückt ist. In diesem Fall erscheint das Promptzeichen (dieser erscheint ebenso, wenn die Option "prompt" in der Konfigurationsdatei gesetzt wurde) und Lilo wartet auf die Eingabe durch den Benutzer.

Der Benutzer gibt nun den Namen des zu startenden Systems ein (eine Liste erhält er durch Eingabe von [?] oder [Tab]) und bestätigt die Eingabe mit [Enter]. Soweit nicht Neues.

Zusätzlich zum Namen des Images können dem Kernel Optionen übergeben werden. Alles, was dem Imagenamen folgt, reicht Lilo an den Kernel als Argument weiter, die einzelnen Parameter sind durch Leerzeichen voneinander zu trennen. So weist nachfolgende Eingabezeile den Kernel an, im Single User Modus zu starten und als Rootdateisystem / dev/ hdc1 zu mounten:

```
LILO: Linux_Standardkernel single root= / dev/ hdc1
```

Die wichtigen Argumente sollen knapp erläutert werden: ro bzw. rw veranlassen den Kernel das Rootverzeichnis nur-lesend bzw. lesend-und-schreibend zu mounten. no378 schaltet den Coprozessor ab und no-hlt verhindert die Powermanagement-Funktionen, falls der Rechner mal nichts zu tun hat.

debug lässt die Fülle der Kernelmeldungen in schwindelerregende Dimensionen anschwellen und vga= *Modus* modifiziert die Ausgabe (wie weiter oben schon erwähnt).

Des Weiteren können nahezu zu allen Hardwaregeräten deren Einstellungen, wie IO-Adresse, Interrupts, ... an den Kernel durchgereicht werden. Die Auswahl ist so vielfältig, dass man eigens ein HowTo (BootPrompt) dem Thema widmete.

Die Umgebungsvariablen für den init-Prozess lassen sich auch per Kommandozeile Umbiegen. Um z.B. eine andere Einstellung der PATH-Variablen zu erzwingen, setzt man "PATH= / opt/ kde/ sbin:/ usr/ sbin:/ sbin/ " als Argument ein.

Lilo startet durch

Lilo selbst gibt Auskunft über das Fortschreiten des Bootvorganges. Anhand der Ausgabe "LILO" lassen sich die Vorgänge beobachten. Hängt der Start hier, hilft ein angegebener zweistelliger Code einem bei der Ursachenforschung oft auf die Sprünge.

Eine verstümmelte Ausgabe hat folgende Ursachen:

L

Die erste Stufe des Lilo kann die zweite nicht finden, der angegebene Code beschreibt den Gru

LI

Die zweite Stufe wurde geladen, kann aber nicht ausgeführt werden. Entweder besitzt der Lilo falsche Informationen über die Festplattengeometrie oder die Datei / boot/ boot.b wurde manipuliert, ohne das Kommando "lilo" auszuführen.

LIL

Die zweite Stufe kann die "map"-Datei mit den Informationen über die Lage der Kernel nicht finden. Die Ursache ist wiederum in einer falschen Plattengeometrie zu suchen.

LIL?

Die Daten des Lilo wurden an eine falsche Speicheradresse geladen, vermutlich wurde "lilo" na einer Kernelinstallation nicht neu gestartet.

LIL-

Die "map"-Datei ist fehlerhaft. Wahrscheinlich wurde irgendeine der Startdateien geändert, ol "lilo" zu rufen.

LILO

Alles im Lot, der Ladevorgang ist erfolgreich verlaufen

Die Fehlercodes des ersten Schrittes identifizieren folgenden Fehler:

0x00

Der "Internal error" deutet auf eine beschädigte Datei oder die Überschreitung der 1024-Zylin Grenze hin.

0x01

Das "Illegal command" erscheint, wenn der Zugriff auf eine Festplatte erfolgt, die nicht vom BI unterstützt wird.

0x02

"Address mark not found" deutet an, dass auf die Platte nicht zugegriffen werden kann (Hardwareproblem?).

0x03

"Write protected disk" erscheint bei einer Schreiboperation auf eine schreibgeschützte Diskett

0x04

"Sector not found" lässt auf eine falsche Plattengeometrie schließen. Helfen könnte eventuell Abschalten der "compact"-Option und/ oder Hinzufügen von "linear".

0x06

"Change line active" sollte höchstens temporär auftauchen und nach wiederholten Booten verschwunden sein.

0x07

"Invalid initialization" schiebt dem BIOS die Schuld in die Schuhe, da dieses vermutlich den Festplattencontroller nicht initialisieren kann.

0x08

"DMA overrun" kann bestenfalls durch einen Neustart behoben werden.

0x09

"DMA attempt across 64k boundary" kann bestenfalls durch einen Neustart behoben werden.

0x0C

"Invalid media" entweder klappt es nach einem Neustart oder das Bootmedium ist beschädigt.

0x10

"CRC error" deutet auf eine Fehler des Mediums hin. Sollten weitere Bootversuche ebenso scheitern, wie das erneute Installieren des Lilo, so kann versucht werden, die defekten Sektor auszublenden.

0x11

"ECC correction successful" beschreibt einen erkannten und korrigierten Lesefehler. Lilo bricht dennoch den Ladevorgang ab. Ein Neustart sollte helfen.

0x20

"Controller error" ist ein Hardwarefehler und man sollte hoffen, dass er beim Neustart nicht wieder auftritt...

0x40

"Seek failure" ist ein Zugriffsfehler auf das Bootmedium. Ein erneuter Versuch bringt hoffentlic Abhilfe.

0x80

"Disk timeout" zeigt an, dass das Medium nicht bereit ist. Im Falle einer Diskette ist vermutlich das Laufwerk nicht verriegelt.

0xBB

Zeigt sich "BIOS error" hartnäckig und erscheint auch nach weiteren Bootvorgängen, so sollte die Optionen "compact" und "linear" entfernt oder hinzugefügt werden.

Chos - Choose Operation System

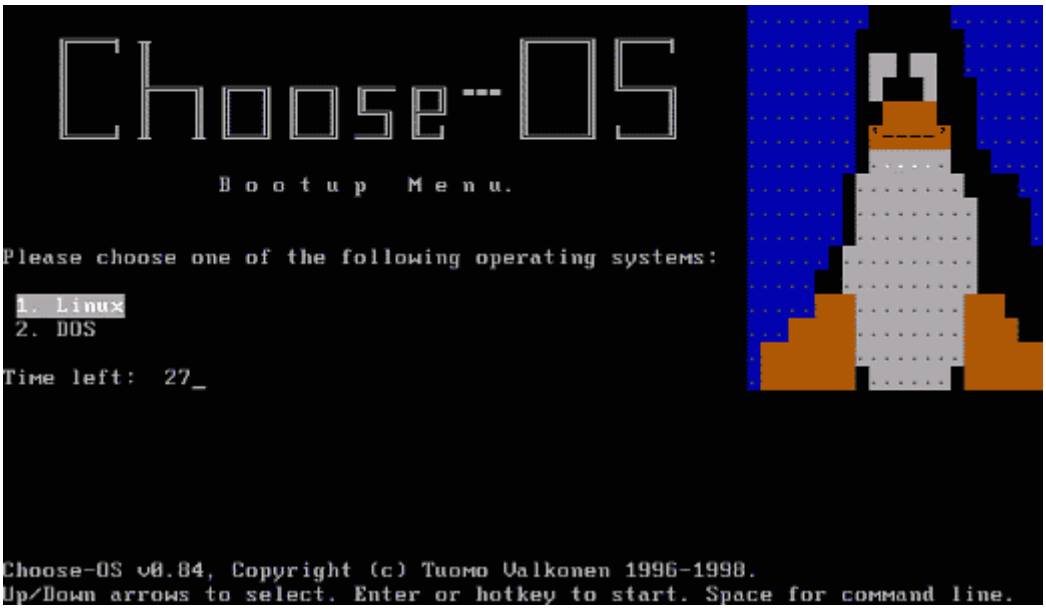


Abbildung 2: Begrüßung durch den Chos

Auch wenn lilo nach wie vor den Standard unter den Linux Bootloadern setzt, so hat sich der chos dank seiner attraktiven Oberfläche und der einfachen Administration aus dem Status eines Geheimtipps erhoben.

Wozu dann immer noch Lilo? Weil der chos sich noch immer im Entwicklungsstadium befindet und noch nicht alle Möglichkeiten des Lilo unterstützt. Und... der Status des Projekts scheint derzeit unklar, da sich der Koordinator des Projekts aus diesem verabschiedet hat.

Sei es wie es sei, selbst wenn noch keine "1" der Versionsnummer voran steht, arbeitet der chos zuverlässig und stellt eine echte Alternative zum Lilo dar.

Der Bootmanager ist in der Lage den Linux-Kernel sowie die Bootsektoren anderer Betriebssysteme zu laden. Für Linux unterstützt er einen mit bzip2 gepackten Kernel, Ramdisks und die Übergabe von Parametern an den Kernel. Des Weiteren ist er so konzipiert, dass ein Programm ihm mitteilen kann, welches Betriebssystem beim nächsten Systemstart zu laden ist. Sollte er einmal seine Daten nicht finden (weil die Festplatte nicht mehr will...) so ist er in der Lage, einen speziellen Notfall-Bootsektor zu starten.

Die Datei /etc/chos.conf

Analog zum Aufbau der "/ etc/ lilo.conf" gruppieren sich die Einträge in allgemein gültige und in kernelspezifische Optionen. Wenden wir uns zunächst dem Eintrag zum Laden eines Betriebssystems zu, jeder Eintrag besitzt folgende Gestalt:

```
Image-Typ "Name" {
    Image-Optionen
}
```

Image-Typ kann dabei einer von 3 Bezeichnern sein:

- **linux Es handelt sich um einen Linux-Kernel (auch bzip2-gepackt)**
- **bootsect Es handelt sich um einen Bootsektor (alle Nicht-Linux-Systeme)**
- **bootfile Der Bootsektor ist in Form einer Datei abgelegt**

Name ist der Eintrag, der im späteren Bootmenü als Bezeichner erscheint. Alle Optionen, die den Betriebssystemeintrag betreffen, sind nun in geschweifte Klammern eingeschlossen:

color= Farbe

Setzt die Farbe des Menüeintrags, als Angaben gelten BLACK, BLUE, GREEN, CYAN, RED, MAGENTA, brOWN, LI GHTGRAY.

hotkey= Taste

Durch diese Taste kann das System sofort gestartet werden

position= Spalte,Zeile

Positioniert den Menüeintrag

loader= loader

Nutzt den angegebenen Lader anstelle des "normalen" Bootloaders für dieses System

image= Kernel

Im Falle von Linux der vollständige Name des Kernel (z.B. / boot/ vmlinuz), sonst der Bootsekt oder der Name der Datei mit dem Bootsektor

password= yes| no| restricted| cmdline

Mit yes fragt der chos und mit no fragt er nicht nach dem Passwort. Mit restricted oder cmdlin fordert er zur Eingabe des Passwortes nur auf, falls man die Kommandozeile editieren möchte (Drücken der Leertaste im Menü)

Nur für Linux-Images relevante Optionen:

cmdline= Parameter

Zur Übergabe von Parametern an den Kernel

initrd= Datei [,Adresse]

Lädt die Datei an eine optional angegebene Adresse in eine Ramdisk. Ohne Adresse wird die D an den Beginn des Hauptspeichers geladen.

Einige Beispiel sollen den praktischen Einsatz der Optionen demonstrieren:

```
linux "Stabiler Kernel 2.2.16" {
  image= /boot/vmlinuz
  cmdline= vga= EXTENDED
  position= 1,11
  hotkey= l,hidden
}

linux "Testkernel 2.4.pre2" {
  image= /boot/kernel-2.4.pre2
  cmdline= vga= EXTENDED
  position= 1,12
  password= cmdline
}

bootsect "Window" {
  image= /dev/hda1
  position= 1,13
  color= blue
  hotkey= w,hidden
}
```

Fehlen uns noch die allgemeinen Optionen. Der Autor des Chos teilte diese in die drei Gruppen "Allgemein" "Aussehen" und "Dateien" ein. Betrachten wir zunächst die Optionen der "allgemeinen" Gruppe:

delay= Sekunden

Verzögerung, bis der erste Menüeintrag automatische gestartet wird

autoboot= no | pressed | not_pressed

Steht hier "no", so wird das erste Betriebssystem gestartet, ohne dass der Benutzer etwas dar ändern kann, bei "pressed" wird es geladen, sobald der Benutzer eine der Tasten [Ctrl], [Alt] o [Shift] drückt und "not_pressed" bewirkt das sofortige Laden des ersten Systems, wenn der Benutzer zum Zeitpunkt der Aktivierung der Bootmenüs keine Taste (nicht alle Tasten bewirke die Auswahl) drückt.

install= device

Wohin soll der Bootloader "chos" geschrieben werden (Angabe des Devices)?

emergency= Datei

Die Notfalldatei, die zu starten "chos" versucht, wenn er seine eigenen Dateien nicht finden ka

password= Passwort

Das Passwort, mit dem der Zugang zu den Einträgen reguliert werden kann. Wenn Sie ein solc verwenden, dann stellen Sie auch sicher, dass niemand außer Root die Datei / etc/ chos.conf le darf.

default= Nummer

Dieser Eintrag ist in der Voreinstellung selektiert, die Zählung beginnt bei 1.

mapdrive= bios_number, device

Hiermit hilft der Bootloader einem BIOS auf die Sprünge, wenn dieses ein Gerät nicht erkenne kann, indem er das Mapping des Gerätes an die BIOS-Adresse vornimmt.

Die Optik beeinflussen folgende Einträge:

banner= Text

infoline= on | off

Mit "on" wird ein Copyrigth- und Hilfe-Hinweis ausgegeben

color= Farbe

Die voreingestellte Farbe, sie kann bei den einzelnen Image-Einträgen überschrieben werden

selection= Farbe

Die Balkenfarbe des selektierten Eintrags

background= type,file

Die Datei mit den Hintergrundbild. Der Typ der in der Datei enthaltenen Daten muss angegebe werden, "dump" steht für binäre Daten, "ascii" für reinen Text und "cp*xxx*" für die Zeichensatzkodierung *xxx*.

menupos= Spalte,Zeile

Position des Menüs relativ zum linken, oberen Bildrand (0,0)

timerpos= Spalte,Zeile

Position der Einblendung der verbliebenen Zeit bis zum automatischen Booten

Die Optionen zur Datei sind:

bindir= Verzeichnis

Verzeichnis mit den Daten des chos

bgfile= Datei

Datei, wohin ein erzeugtes Hintergrundbild gespeichert werden soll

mapfile= Datei

Die Datei wird anstelle von chos.map verwendet

Damit könnte eine vollständige Konfigurationsdatei des chos wie folgt aussehen:

```
#   Generelle Einstellungen

# DELAY
delay= 10

# AUTOBOOT
autoboot= no

# INSTALL
install= /dev/hda

# PASSWORD
password= Tux4Ever

#   Einstellungen zur Optik

# INFOLINE
infoline= on

# BACKGROUND
background= dump:/boot/bg/chos_penguin.bin
```

```
# MENUPOSITION
menupos= 0,12

#   Dateieinstellungen

#   Die eigentliche Verwaltung der Images

linux "Linux" {
   image= /boot/vmlinuz
   position= 1,12
   cmdline= vga= EXTENDED
}

bootsect "Anderes System" {
   image= /dev/hda2
   position= 1,13
}
```

GRUB - Grand Unified Bootloader

Die Anfänge des GRUB reichen ähnlich weit in die Vergangenheit zurück wie die des Lilo, doch hat erst dessen Aufnahme in den Fundus der GNU-Software für eine gewisse Verbreitung gesorgt. Dabei wartet GRUB mit einer Reihe von Fähigkeiten auf, die kaum ein zweiter Bootmanager in nur annäherndem Umfang mit sich bringt. Hierzu zählen:

- **Kein 1024-Zylinder-Problem (solange das BIOS dieses nicht verursacht)**
- **Direkte Unterstützung der Dateisysteme Ext2fs, Reiserfs, Fat16, Fat32, FFS (Fast file system) und Minix, womit Betriebssysteme von diesen direkt geladen werden können**
- **Betriebssysteme, die sich auf »unbekannten« Dateisystemen befinden, können über deren Bootloader (»chain loading«) geladen werden**
- **Booten übers Netz möglich**
- **Die GRUB-Shell ermöglicht den Start von Betriebssystemen, die nicht für GRUB vorkonfiguriert sind**

Die Liste der heraus ragenden Eigenschaften ließe sich noch ein Stück verlängern. Einiges wird der weitere Text berichten und manch andere Fähigkeit ist derzeit kaum dem Status der Planung entwachsen.

Installation

Auch wenn GRUB schon für den produktiven Einsatz geeignet ist (sprich: er läuft stabil), so ist es bis zum ersten »kompletten« Release noch ein weiter Weg. So wird man derzeit auch kaum vorkompilierte Pakete finden, sondern kommt um eine Konfiguration und Installation der Quellpakete nicht umhin. Die üblichen Schritte sind:

```
root@sonne> tar xzf grub-0_5_96.tar.gz
root@sonne> ./ configure
root@sonne> make
root@sonne> make install
```

Es ist sicher nicht verkehrt, zuvor die Dateien INSTALL und README im Quellverzeichnis zu betrachten. So lassen sich Merkmale von GRUB durch Optionen des »configure«-Skripts an- bzw. abschalten. Im Falle des Einsatzes von GRUB zum Booten über das Netz ist das Aktivieren des entsprechenden Netzwerktreibers zwingend notwendig (»./ configure --help« verrät Ihnen mehr darüber). Treten bei obigem Vorgehen Fehler auf, sollten Sie den Abschnitt Software-Installation - Quellpakete im Kapitel Systemadministration durchlesen.

Die Dateien des GRUB liegen nun unterhalb von / usr/ local/ share/ grub/ i386-pc (auf manchen Systemen auch unter / usr/ share/ grub/ i386-pc). Folgt man den Richtlinien des File System Hierachie Standards, gehören die Daten eines Bootmanagers unter / boot, sodass folgende Schritte zu

empfehlen sind (die weiteren Beispiele beziehen sich auf dieses Verzeichnis):

```
root@sonne> mkdir /boot/grub
root@sonne> cp /usr/local/share/grub/i386-pc/stage[12] /boot/grub/
root@sonne> cp /usr/local/share/grub/i386-pc/*stage1_5 /boot/grub/
root@sonne> ls /boot/grub
e2fs_stage1_5      ffs_stage1_5      reiserfs_stage1_5  stage2
fat_stage1_5       minix_stage1_5    stage1
```

Das Programm grub liegt nun vermutlich unter /usr/local/sbin (oder /usr/sbin); zum weiteren Vorgehen ist dieses zu starten:

```
root@sonne> /usr/local/sbin/grub

   GRUB  version 0.5.96  (640K lower / 3072K upper memory)

 [ Minimal BASH-like line editing is supported.  For the first word, TAB
   lists possible command completions.  Anywhere else TAB lists the possible
   completions of a device/filename. ]

grub>
```

Namensgebung

Für die weiteren Schritte zur Installation ist eine Betrachtung der Namenskonventionen notwendig, die GRUB für Festplatten und Floppies verwendet.

Zunächst unterscheidet GRUB nicht zwischen IDE- und SCSI-Festplatten, beide tragen in ihrer Bezeichnung den Suffix »hd«. Abweichend von Linux nummeriert GRUB die Festplatten und Floppies mit Ziffern, wobei die Zählung bei 0 beginnt. Somit entspricht GRUB's »hd0« dem »hda« (bzw. sda) von Linux; »hd1« steht für »hdb« (sdb) usw. Entsprechend der Reihenfolge im BIOS reihen sich bei Systemen, die IDE- und SCSI-Platten vereinen, sämtliche IDE-Platten vor SCSI ein oder umgekehrt. Ein Diskettenlaufwerk benennt GRUB nach dem selben Schema wie Linux.

Eher einer Umgewöhnung bedarf da schon das Ansprechen einzelner Partitionen. Auch hier startet GRUB die Zählung bei 0. Diese Nummer ist dem Plattennamen nachzustellen, wobei beide Angaben durch ein Komma zu trennen sind. Während Linux die erste Partition der ersten IDE-Festplatte mit »hda1« benennt, heißt diese im GRUB-Sprachgebrauch »hd0,0«.

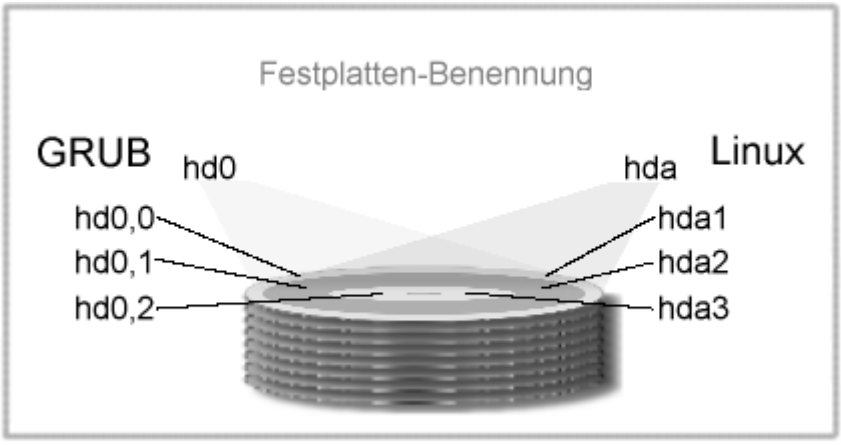


Abbildung 3: Benennung von Partitionen

Der Gebrauch erklärt sich anhand eines Beispiels quasi von selbst. Zur Demonstration nutzen wir das in die Grub-Shell eingebaute Kommando cat, das den Inhalt einer Datei auf den Bildschirm ausgibt. Mit der Annahme, dass sich das Linux-Wurzelverzeichnis auf der ersten logischen Partition befindet (unter Linux /dev/hda5), führt folgende Eingabe zum Ziel:

```
root@sonne> /usr/local/sbin/grub

   GRUB version 0.5.96.1 (640K lower / 3072K upper memory)

...

grub> cat (hd0,4)/etc/lilo.conf
# LILO configuration file
# Start LILO global Section
...
```

Die runden Klammern sind integraler Bestandteil der Device-Benennung von GRUB!

Bootvorbereitung

Wohl jeder Bastler ist im Laufe seiner Experimente mit Linux einmal in die Falle getappt, dass er nach dem Bau eines neuen Kernels diesen zwar korrekt installierte, aber versäumte, anschließend den Bootmanager neu einzurichten. Sowohl lilo als auch chos bestraften die Vergesslichkeit mit einem verkorksten Bootvorgang... Wer kennt nicht Ausschriften der Art: "01010101010"?

Grub's Technologie bei der Suche nach den Kerneln arbeitet jedoch nicht mehr mit den Blocklisten, so wie es lilo und chos handhabten, sondern nutzt die Mechanismen des jeweiligen Dateisystems, um die Datei(en) aufzuspüren. Der reine Bezug auf den Dateinamen macht Grub resistent gegen Änderungen am Kernel, die zumeist mit einer Lageveränderung im Dateisystem (auf Blockebene) behaftet sind. Und dieses Verfahren ermöglicht ebenso, dass ein System bootbar ist, welches Grub nicht vorab bekannt ist.

Doch zunächst muss Grub an den rechten Fleck verschoben werden, um beim nächsten Booten des Rechners zu starten.

Der sichere Weg führt über eine Bootdiskette, auf welche stage1 und stage2 zu kopieren sind:

```
root@sonne> cat /boot/grub/stage[12] > /dev/fd0
```

Wenn Sie das nächste mal per Diskette booten, befinden Sie sich anschließend in der Grub-Shell wieder.

Nicht zuletzt wegen der Fehleranfälligkeit des Mediums Diskette eignet sich der Systemstart von derselben nur als Notmaßnahme oder in der Testphase. Wer den Grub ernsthaft für die Bootmaschinerie in Betracht zieht, wird ihn auch auf die Festplatte bannen wollen, wozu der Start der Grub-Shell von Nöten ist:

```
root@sonne> /usr/local/sbin/grub
...
grub> root (hd0,5)
Filesystem type is ext2fs, partition type 0x83
```

Mit dem root-Befehl teilen Sie Grub mit, auf welcher Partition seine Daten liegen (im Beispiel /dev/hda6 nach Linux-Gerätebezeichnung). Wenn Sie sich der Lage nicht sicher sind, dann lassen Sie Grub diese feststellen, indem Sie in der Grubshell find </Pfad/stage1> eingeben:

```
grub> find /boot/grub/stage1
(hd0,5)
```

Anstatt im Verzeichnis /boot/grub/ liegen die Daten des Bootmanagers auf Ihrem System eventuell unterhalb von /grub. Und zwar genau dann, wenn /boot auf einer eigenen Partition liegt.

```
grub> setup --stage2=/grub/stage2 (hd0)
```

```
Checking if "/boot/grub/stage1" exists... yes
Checking if "/boot/grub/stage2" exists... yes
Checking if "/boot/grub/e2fs_stage1_5" exists... yes
Running "embed /boot/grub/e2fs_stage1_5 (hd0)"... 16 sectors are embedded. succeeded
Running "install --stage2=/boot/grub/stage2 /boot/grub/stage1 d (hd0) (hd0)1+16 p (hd0,5)/boot/grub/stage2"... succeeded
Done.
```

Mit dem setup-Aufruf wird letztlich Grub installiert. Zwingend anzugeben ist der Ort der Installation, im Beispiel der Mbr der ersten Festplatte (hd0). Die Angabe der Lage von stage2 ist eigentlich überflüssig, da sie laut Grub's eigener Statusmeldungen exakt mit dem übereinstimmt, was Grub ermittelte. Allerdings scheiterte bei den von mir getesteten Versionen stets das Schreiben von stage2, sobald ich auf diese Angabe verzichtete.

Neben stage1 und stage2 platziert Grub - falls er im Mbr installiert wird - automatisch den Treiber für das Dateisystem in eine definierte Position der Festplatte. Durch den anfänglichen root-Aufruf erkannte Grub den Typ des Dateisystems, auf dem seine Daten liegen (ext2 im Beispiel). Genau für diesen Typ schreibt Grub einen minimalen Treiber e2fs_stage1_5 in den freien Bereich hinter dem Mbr und vor Beginn der ersten Partition (die erste Partition beginnt auf dem zweiten Zylinder, sodass vom ersten Zylinder ein Sektor durch den Mbr reserviert ist und die nächsten 62 Sektoren frei sind). Die Verwendung dieses Treibers erlaubt es Grub, die Datei stage2 selbst dann noch zu finden, wenn diese verschoben wurde.

Booten mit der Grub-Shell

Nach obigen Vorbreitungen endet ein folgender Bootvorgang mit einem Prompt in der Grub-Shell. Jedes installierte System auf dem Rechner, das Grub direkt unterstützt, kann nun geladen werden. Für nicht unterstützte Systeme, wie die Windows-Familie, kann per »chainloading« deren Bootloader gerufen werden.

Angenommen, ein Linuxkernel »vmlinuz« befindet sich auf /dev/hda6 ebenso wie dessen Root-Dateisystem, so lässt sich diese über folgende Befehlssequenz starten:

```
grub> kernel (hd0,5)/boot/vmlinuz root=/dev/hda6
   [Linux-bzImage, setup=0xe00, size=0x4797]

grub> boot
```

Der kernel-Befehl leitet die Angabe des Kernels ein. Dieser ist mit dem vollständigen Pfad (bzw. der Partition) zu spezifizieren. Anstatt dem im Beispiel gewählten expliziten Bezug auf eine Partition ((hd0,5)/boot/vmlinuz) kann ein vorangehender root-Befehl festlegen, dass weitere Pfadangaben sich auf eine bestimmte Partition beziehen. Folgende Befehlsfolge ist somit äquivalent (auf die Ausgaben von Grub wurde verzichtet):

```
grub> root (hd0,5)
grub> kernel /boot/vmlinuz root=/dev/hda6
grub> boot
```

Möchten Sie einem Kernel Parameter mitteilen, so sind diese auf der Kernel-Zeile anzugeben. Alles dem Namen der Kerneldatei Folgende wird von Grub unverändert an den Kernel übergeben. Einzig erforderlich ist das Root-Device (es sei denn, dieses ist korrekt im Kernel eingetragen - was bei Standardkerneln selten der Fall ist...).

Um ein System wie Windows2000 zu starten, benötigen Sie Kenntnis, in welcher Partition dieses installiert ist. Über den chainloader--Befehl laden Sie hierzu den Bootsektor dieser Partition. Da dieser im ersten Block auf der Partition liegt, wird Grub dies mit der Angabe des Offsets +1 kund getan:

```
grub> chainloader (hd0,0)+1
grub> boot
```

Da obiges Verfahren explizit Blocknummern adressiert, wird es auch als Blocklistenadressierung bezeichnet. Sofern bekannt ist, in welchen Blöcken eine Datei gespeichert ist, vermag Grub hierüber jede Datei unabhängig vom Dateisystem zu lesen. Das Adressierungsschema lautet hierbei *Offset+Länge*, wobei bei fehlendem Offset (wie im Beispiel) implizit 0 angenommen wird. Eine Datei auf der 2.Partition einer ersten Festplatte, die ab Block 100 beginnt und 5 Blöcke umfasst, wäre mit "(hd0,1)100+5 anzugeben. Auch eine über mehrere Bereiche verteilte Datei ließe sich über die Angabe der Blocklisten laden ("(hd0,1)1,100+5,200+28").

Das Bootmenü - etwas Bequemlichkeit

Das Prinzip des Kernelstarts ist schnell verinnerlicht und dennoch wäre die wiederkehrende Eingabe der Befehlsfolgen ein Argument gegen den Einsatz des Grub. Zwar bietet Grub keinen grafisch ansprechend gestalteten Deckmantel, den neue Lilo-Versionen beherrschen, aber eine komfortable textbasierte Menüsteuerung. Zuständig für die Konfiguration ist die Datei menu.lst (Name ist beliebig), die zweckmäßig im selben Verzeichnis wie die weiteren Grub-Dateien liegt. Wir betrachten im weiteren Text ihre wichtigsten Einträge.

Zuvorderst in der Konfigurationsdatei stehen die globalen Optionen. Ein Eintrag zum Start eines Systems beginnt mit dem Schlüsselwort title gefolgt von einem eindeutigen Bezeichner dieses Eintrags. Alle weiteren Befehle bis zur folgenden title-Zeile oder zum Dateiende betreffen den Start dieses Systems. Die grobe Struktur sieht somit wie folgt aus:

```
# Aufbau der Menübeschreibungsdatei
# Allgemeine Befehle
...
title "System x"
# Befehle zum Start von "System x"
...
title "System y"
# Befehle zum Start von "System y"
...
```

Wenn Sie bereits mit Grub gearbeitet haben, so ist Ihnen sicherlich die amerikanische Tastaturbelegung unangenehm aufgestoßen. Glück für denjenigen, der die Lage deren Tasten kennt. Die deutschen Sonderzeichen einmal außen vor gelassen, gestattet der setkey-Befehl der Grubshell die komplette Umgestaltung der Belegung. Sicher artet die Anpassung in reichlich Arbeit aus, aber für die Erstellung der Menüdatei ist diese einmalige Mühe sicher gerechtfertigt. Für jede Taste ist ein eigener setkey-Aufruf erforderlich:

```
setkey <Originalbelegung (amerikanisch)> <Neue Belegung>
```

Der erste Wert beschreibt die Belegung, die sich bei amerikanischem Layout hinter einer Taste verbirgt, die »neue Belegung« soll die zukünftige Wirkung sein. Um bspw. die Tasten [z] und [y] zu vertauschen, sind folgende Aufrufe erforderlich:

```
setkey z y
setkey y z
setkey Z Y
setkey Y Z
```

Versuchen Sie einen solchen Umtausch besser nicht innerhalb der Grubshell, schon mit dem ersten Aufruf im Beispiel stünde die Taste [z] nicht mehr zur Verfügung! Alphanumerische Tasten bezeichnen sich selbst, alle weiteren Sondertasten müssen durch folgende Bezeichner beschrieben werden (in Klammern jeweils die Taste): *exclam* (!), *at* (@), *numbersign* (#), *dollar* ($), *percent* (%), *caret* (^), *ampersand* (&), *asterisk* (*), *parenleft* (), *parenright* ()), *minus* (-), *underscore* (_), *equal* (=), *plus* (+), *bracketleft* ([), *braceleft* ({), *bracketright* (]), *braceright* (}), *semicolon* (;), *colon* (:), *quote* (`), *doublequote* ("), *backquote* (`), *tilde* (~), *backslash* (\), *bar* (|), *comma* (,), *less* (<), *period* (.), *greater* (>), *slash* (/), *question* (?), *space* ( ).

Zu den allgemeinen Befehlen für die Menükonfiguration zählen:

timeout *Sekunden*

Zeitspanne, nach der der erste bzw. der default-Eintrag automatisch gestartet wird

default *Nummer*

Default-Eintrag, ohne die Angabe wird der erste angenommen (Zählung beginnt bei 0)

fallback *Nummer*

Tritt beim Start des default-Eintrags ein Fehler auf, wird dieser Eintrag automatisch gestartet

password *Passwort* [*Menüdatei*]

Ohne Kenntnis des Passworts darf weder der Shellmodus gestartet noch der Menüeintrag eine gesperrten Eintrags (lock, siehe nachfolgend) modifiziert werden. Mit Angabe einer optionalen Menüdatei wird diese neu eingelesen (so könnten bspw. neue Einträge erst nach Eingabe des Passworts sichtbar werden)

Im Unterschied zu den bislang genannten Befehlen können die nachfolgenden mehrfach in der Konfigurationsdatei genannt werden. Bis auf title stehen die weiteren Kommandos auch in der Grub-Shell zur Verfügung (Auswahl):

color *Normalfarbe*[*Hervorgehoben*]

Um etwas Farbe ins Menü zu bringen. Jede Farbe wird als Paar »Vordergrund/ Hintergrund« angegeben. »Hervorgehoben« bestimmt das Aussehen des selektierten Eintrags. Entfällt die Angabe, wird die Komplementärfarbe zur »Normalfarbe« genommen. Zur Bezeichnung der Far sollten Sie "help color" in der Grubshell konsultieren.

help [Befehl]

Gibt eine Kurzhilfe zum Befehl aus. Ohne Angabe des Befehls werden alle möglichen Kommand aufgelistet

initrd

Zum Laden einer Ramdisk in den Speicher

lock

Sichert einen Eintrag durch ein Passwort; nur in Verbindung mit password sinnvoll

makeactive

Setzt das bootable Flag einer Partition

map *Zieldevice Quelldevice*

Mapping des *Quelldevices* auf das *Zieldevice* (sinnvoll, um bspw. einem DOS auf der zweiten Partition vorzugaukeln, es sei auf der ersten...)

root *Partition*

Gibt die Partition an, auf der sich das zu bootende System befindet. Grub versucht die Partitio mounten, um die Partitionsgröße festzustellen

rootnoverify *Partition*

Gibt die Partition an, auf der sich das zu bootende System befindet. Grub versucht nicht die Partition zu mounten (sinnvoll, wenn die Partition außerhalb des addressierbaren Bereichs lieg 1024-Zylinder-Problem)

title *Name*

Leitet einen neuen Menüeintrag ein, der Name erscheint als Balkentext

[un]hide *Partition*

(Zurück)Setzen des »hidden«-Flags einer Partition. Somit kann eine Partition vor einem Syste »versteckt« werden (DOS, Windows)

Mit den genannten Befehlen sollte die nachfolgende Beispielkonfiguration leicht nachzuvollziehen sein. Sie bietet 4 Menüeinträge. Zwei dienen dem Start von Systemen, eines der Installation des Grub in den Mbr der ersten Festplatte und der letzte Eintrag demonstriert die Verwendung von Farben:

```
user@sonne> cat /boot/grub/menu.lst
# Beispieldatei für ein GRUB Bootmenü

# Einige Tastaturumbelegungen
setkey z y
setkey y z
setkey Z Y
einem gesperrten Eintrag (lock, siehe nachfolgend) setkey Y Z

# Automatisch nach 10 Sekunden booten
timeout 10

# Zweiten Eintrag als »default« booten
default 1

# Bei Fehler den ersten Eintrag
fallback 0

# Passwort, um bestimmte Einträge zu schützen
password Only4Tux

title Linux-Standardkernel
lock
root (hd0,4)
initrd = /boot/initrd
kernel /boot/vmlinuz

title Windows 2000
rootnoverify (hd0,0)
makeactive
chainloader +1

# Um GRUB neu zu installieren
title Installiere GRUB in den Mbr
lock
root (hd0,4)
setup --stage2=/boot/grub/stage2 (hd0)

# Alternative Darstellung des Menüs
title Tapetenwechsel
color light-green/brown blink-red/blue
```

Nach dem nächsten Booten begrüßt Sie folgendes Menü (die Farben erscheinen erst nach Auswahl des 4. Eintrags.

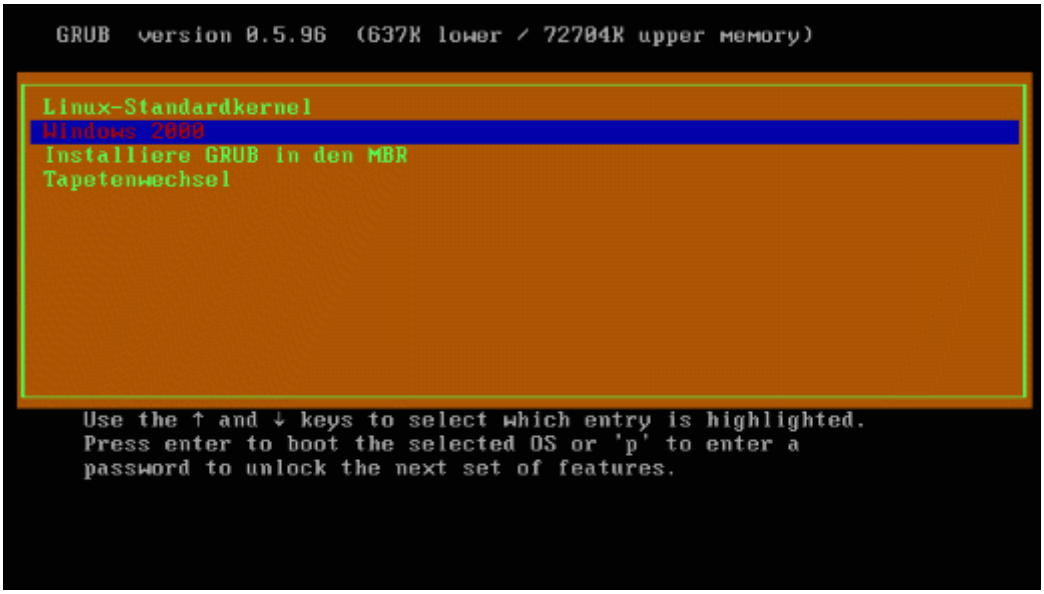


Abbildung 4: Grub-Menü

Die mittels »lock« gesperrten Einträge lassen sich erst nach Eingabe von [p] mit anschließendem Passwort wählen. Ebenso besteht dann die Möglichkeit mittels [e] einen Eintrag zu editieren und mittels [c] die Grubshell zu erreichen.

Linux und der NT-Bootmanager

Geschmäcker unterscheiden sich bekanntlich und so wird auch der eine oder andere den Einsatz seines NT Bootmanagers (bzw. Windows2000) zum Starten anderer Betriebssysteme bevorzugen. Auch ist es in manchen Hardware-Konstellationen nahezu unmöglich, ein Windows-Betriebssystems durch einen »Fremdloader« zur Arbeit zu bewegen, womöglich gelingt es jedoch, Linux durch NT zu starten.

Eigentlich ist es ganz einfach, dem NT Bootmanager das Linux unterzuschieben, auch wenn die notwendigen Kommandozeilenaufrufe einen Hauch von Kryptografie verbreiten. Ein Linux ohne einen »richtigen« Bootloader zu starten, wird wohl misslingen, aber der Start eines linuxfähigen Bootmanagers vom NT-Tool aus, lässt sich stets erzwingen.

Mit der letzten Aussage ist das prinzipielle Vorgehen bereits scharf umrissen:

1. **Installation eines Linux-Bootmanager (lilo, chos, grub) am besten in den Bootsektor der Rootpartition. Dieser Bootmanager darf nicht in den Mbr, da über den dortigen Bootkode der NT Bootmanager geladen wird**
2. **Kopieren des den Linux-Bootloader enthaltenden Bootsektors in eine Datei (siehe nachfolgende Beschreibung)**
3. **Kopieren dieser Datei auf eine von NT lesbare Diskette**
4. **Start von NT und kopieren der Datei in ein Windows-Verzeichnis**
5. **Eintrag der Bootzeile in die boot.ini (siehe nachfolgende Beschreibung)**

***Anmerkung:* Anstatt der Diskette kann im Falle eines FAT-Dateisystems auch direkt von Linux aus auf dieses geschrieben werden. Riskieren Sie aber nicht den schreibenden Zugriff auf ein NTFS-Dateisystem, die unzureichende Implementierung des Treibers (weil Microsoft die Spezifikation nicht veröffentlicht) kann zur Beschädigung der gesamten Windowsinstallation führen!**

Die Installation eines Linux-Bootloaders wurde in den beiden vorangegangenen Abschnitten besprochen. Wir nehmen für die weiteren Ausführungen an, dass sich das Linux-Rootverzeichnis auf der 3. Partition der ersten IDE-Festplatte befindet. Unter Linux lautet der Name dieser also / dev/ hda3.

Um eine Datei mit dem Bootsektor zu erzeugen, kann das Kommando lilo ein zweites Mal mit der Option -s und der Zieldatei aufgerufen werden:

```
root@sonne> lilo -s /tmp/bootsec.lin
Added linux *
Added WINNT
```

Alternativ kann ein Low-Level-Kopierprogramm verwendet werden, das in der Lage ist, auf der Basis von Sektoren und Byteangaben Daten zu vervielfältigen. Unter Linux ist dd das gebräuchliche Kommando:

```
root@sonne> dd if=/dev/hda3 of=/tmp/bootsec.lin bs=512 count=1
1+0 Records ein
1+0 Records aus
```

Erläuterung: Die Quelle »infile« wird mit if=/dev/hda3 spezifiziert. Wohin die Daten kopiert werden sollen, besagt of=/tmp/bootsec.lin (outfile), also in eine Datei mit dem Namen »bootsec.lin« im Verzeichnis /tmp. Als Nächstes muss dd die Größe eines Sektors wissen und da ein jeder Bootsektor 512 Bytes lang ist, teilen wir dies dem Kommando mit bs=512 mit. Letztlich fordert dd die Angabe der Anzahl zu kopierender Sektoren. Wir begnügen uns mit einem count=1.

Das Kopieren der Datei auf Diskette kann auf zwei Wegen geschehen. Zum einen durch Mounten der Diskette und anschließendem Kopieren mit cp. Der schnellere Weg geht über die Verwendung der mtools, dessen komplette DOS-Dateikommando-Sammlung sich ähnlich zum Original bedienen lässt. Die uns interessierenden Aufrufe wären:

```
# Falls die Diskette unter /mnt/floppy gemountet ist
user@sonne> cp /tmp/bootsec.lin /mnt/floppy

# Unter Verwendung der mtools und nicht-gemounteter Diskette in LW A:
user@sonne> mcopy /tmp/bootsec.lin a:
```

NT ist anschließend zu starten und die Datei »bootsec.lin« z.B. in dessen Stammlaufwerk C: zu kopieren. Jetzt ist nur noch eine entsprechende Zeile an die Datei »boot.ini« (Achtung: diese ist eine versteckte Datei!) anzufügen, um Linux im Bootmenü von NT erscheinen zu lassen:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation, Version 4.0"
multi(0)disk(0)rdisk(0)partition(1)\WINNT="Windows NT Workstation, Version 4.0 [[VGA-Modus]" /basevideo /sos
c:\bootsec.lin="Linux"
```

Das Verfahren zur Integration von Linux in das Bootkonzept von Windows 2000 erfolgt analog.

Loadlin

Loadlin.exe ist ein DOS-Programm, das es ermöglicht, einen Linuxkernel direkt von einem DOS-System in den Hauptspeicher zu laden. Sein entscheidender Vorteil ist die einfache Administration; eine Beschädigung der Installation, wie sie durch unsachgemäße Konfiguration von bspw. Lilo denkbar ist, ist nahezu ausgeschlossen. Insbesondere der unerfahrene Windows-Umsteiger wird in »loadlin« den Weg des geringsten Widerstands vorfinden.

Die Nachteile sind neben einer längeren Ladephase (erst DOS, dann Linux) vor allem in der wenig komfortablen Übergabe von Kernelparametern zu sehen. Bei der nachfolgend skizzierten Verwendung eines Bootmenüs ist eine interaktive Eingabe der Parameter nicht möglich. Ohne Menü artet der Aufruf schnell zu einer langen Eingabezeile aus...

Voraussetzungen

Neben einer installierten DOS- oder Win9*x*-Version, benötigen Sie zunächst das Programm loadlin.exe. Den gängigen Distributionen liegt dieses auf den Installationsmedien bei, oft unterhalb eines mit »DOS« oder »DOSUTILS« bezeichneten Verzeichnisses.

Wohin Sie dieses installieren, bleibt Ihnen überlassen. Hauptsache, das DOS-System kann die Datei finden und starten. Im Beispiel gehen wir nachfolgend davon aus, dass »loadlin.exe« in einem Verzeichnis »C:\LINUX\« liegt.

Des Weiteren müssen Sie den Linuxkernel in ein von DOS erreichbares Verzeichnis kopieren; günstig ist wiederum »C:\LINUX\«. Am schnellsten gelangen Sie zum Ziel, wenn Sie Linux booten (Startdiskette oder Rettungssystem) und den Kernel auf eine DOS-formatierte Diskette kopieren (mcopy). Im DOS-System spielen Sie die Datei von der Diskette in das Verzeichnis.

Loadlin verwenden

Loadlin.exe kann nur in einer DOS-Box laufen, daher ist im Falle von Win9*x* zunächt der DOS-Modus zu aktivieren. Der Aufruf besitzt folgendes Format:

```
loadlin Pfad_zum_Kernel root=Root-Dateisystem Boot-Parameter
```

Ohne Argumente gerufen, listet loadlin die vollständige Argumentliste auf.

Als »Root-Dateisystem« ist die Linux-Notation zu verwenden, also »/dev/hda2« falls dieses in der 2. Partition der ersten IDE-Festplatte residiert. »Boot-Parameter« kann all jene Parameter umfassen, die ein Linuxkernel versteht.

Um bspw. den Kernel read-only zu mounten und die VGA-Einstellung »normal« zu wählen, dient folgende Aufrufzeile:

```
c:\Linux\loadlin.exe c:\Linux\vmlinuz root=/dev/hda2 ro vga=normal
```

Verwendet der Kernel eine Initiale Ramdisk, so ist diese ebenfalls von Linux nach DOS zu kopieren. Die Anweisung an »loadlin« zum Laden von Kernel und Ramdisk (nennt sich im Beispiel »diskimage«) lautet:

```
c:\Linux\loadlin.exe c:\Linux\vmlinuz root=/dev/hda2 ro initrd=C:\Linux\diskimage
```

Ein Startmenü

Es wäre schon recht umständlich, zum Zwecke des Linuxstarts erst ein Windowssystem vollständig laden zu müssen. Und den rechten Zeitpunkt, abzupassen, um Windows direkt in den DOS-Modus zu versetzen, kommt auch nicht gerade einer komfortablen Lösung gleich.

Der Weg zum Ziel führt über Einträge in den Dateien »config.sys« und »autoexec.bat«.

```
REM - Die Datei 'config.sys'
[menu]
menuitem=Linux, Linux booten
menuitem=Win98, Win98 booten
menudefault=Linux, 15

[Linux]
shell=c:\linux\loadlin.exe c:\linux\vmlinuz root=/dev/hda2 ro

[Win98]
```

```
REM - Es folgt der 'alte' Inhalt der 'config.sys'
...
```

Durch die Marke [menu] wird eine Menüdefinition eingeleitet. Mit [menuitem] erfolgt die Definition der einzelnen Einträge, wobei der erste Wert nach dem Komma den Namen der Marke enthält, an der die Verarbeitung fortgeführt werden soll, dass der Menüpunkt selektiert wurde. Der »Rest« der Zeile wird als Text im Menü angezeigt.

[menudefault] markiert einen Eintrag als Voreinstellung; die folgende Zahl bestimmt die Wartezeit (in Sekunden), nach der bei fehlender Eingabe mit dem voreingestellten Menüpunkt fortgefahren wird.

Bei Wahl der Marke [Linux] starten wir »loadlin«; bei [Win98] geht es mit dem »normalem« Inhalt der »config.sys« weiter.

```
REM - Die Datei 'autoexec.bat'
goto %config%

:Win98
REM - Es folgt der 'alte' Inhalt der 'autoexec.bat'
...

REM - Ganz am Ende...
:Linux
```

In der Datei »autoexec.bat« wird in einem »goto«-Befehl Bezug auf den gewählten Eintrag genommen. Dieselben Namen der Marken erscheinen wieder, nur mit geänderter Syntax. Die Marke :Linux am Ende der Datei garantiert, dass kein weiterer Befehl abgearbeitet wird, während »loadlin« DOS bzw. Windows durch Linux ersetzt.

Unix Shells - Überblick

Überblick

Ohne Shells geht es nicht!

Die meisten Computeranwender werden sich von der tristen Konsole verabschiedet und zu einer grafischen Benutzeroberfläche hingewendet haben. Und tatsächlich erleichtert ein ergonomisch durchdachter Aufbau des Desktops in vielen Fällen die Orientierung. Und dennoch behaupte ich: Ohne eine Shell kommt kein heutiges Betriebssystem aus.

Stellen Sie sich vor, alle im System installierten Programme fänden ihre Repräsentation innerhalb eines Menüs wieder. Ob Sie zügig durch die 20000 Einträge umfassende Menüstruktur navigieren könnten?

Selbst wenn Sie eine sinnvolle Anordnung finden sollten, wie geben Sie beim Programmstart diesem die Optionen mit? Mittels einer sich öffnenden Registerkarte, in der Sie die zwei benötigten Optionen aus den 150 verschiedenen Möglichkeiten markieren?

Sie merken schon... der sonst gebräuchliche Programmstart in einer Shell kann in zahlreichen Fällen hinter einer grafischen Maske verborgen werden. Aber es ist schier unmöglich, jede Anwendung mit jeder erdenklichen Option auf diese Weise »benutzerfreundlich« zu gestalten. Sie werden immer wieder auf Situationen treffen, in denen Sie auf eine simple Shell zurück greifen, weil Sie zum einen gar keine andere Möglichkeit haben oder zum anderen einfach effizienter arbeiten können.

Shells unter Unix sind Programme. Und Programme sind austauschbar. So sollte die Vielfalt an unter Unix verfügbaren Shells nicht weiter verwundern. Speziell unter Linux hat sich die **Bash** als »quasi« Standard etabliert und die meisten Anwender werden kaum jemals dazu veranlasst, von ihrer gewohnten Umgebung abzurücken. Aber nicht wenige Linuxer haben ihr Handwerk auf den »großen« Unix-Sytemen erlernt und sind von anno dazumal mit anderen Shells vertraut. Warum also sollten Sie sich der Bash zuwenden? Die in diesem Kapitel ebenso behandelten **Tcsh** und **Ksh** (genau genommen der freie Klone pdksh) bringen mindestens gleichwertige Eigenschaften mit, die den Umgang mit dem System so viel komfortabler gestalten.

Damit haben wir die »3 Großen« genannt. Die »restlichen« 10-15 Shells sind wenig verbreitet, wenngleich sie auf Grund ihrer Schnelligkeit oder auch ihrer Bescheidenheit hinsichtlich erforderlicher Ressourcen für bestimmte Anwendungszwecke (Realtime-Linux, embedded Linux) besser geeignet sind.

Ein Kriterium zum Bevorzugen der einen oder anderen Shell könnte auch die Unterstützung der Programmierung sein. In nicht wenigen Situationen ermöglicht die an die Programmiersprache C angelehnte Tcsh wesentlich effizientere Techniken, um ein gegebenes Problem in ein Programm umzusetzen, als es die recht ähnlichen Bash und Ksh vermögen. So soll auch das Erstellen von Shellskripten ein Gegenstand dieses Kapitels sein, um den Leser für die Feinheiten und Möglichkeiten des Einsatzes von Skripten zu sensibilisieren.

Dem Kapitel steht ein allgemeiner Abschnitt voran, der grundsätzliche Konzepte aller Shells behandelt. Dieser sollte immer vor den Ausführungen zu einer konkreten Shell gelesen werden, da auf die dortigen Aussagen im Speziellen nicht erneut eingegangen wird.

Ziel des Kapitels

Effizient arbeitet man erst, wenn man genau weiß, was man tut. Und das Arbeitsfeld in Unix ist die Shell, weshalb das Verständnis ihrer Funktionsweise das A und O des Administrators und System-Programmierers (u.a.m.) ist. Der Leser muss nicht alle Shells kennen, da er sich früher oder später ohnehin auf eine Benutzerschnittstelle festlegen wird. Aber diese eine sollte er beherrschen. Damit lassen sich die Ziele knapp formulieren:

- Verstehen der Aufgaben der Shells
- Beherrschung der fortgeschrittenen Hilfen einer konkreten Shell
- Das Schreiben (einfacher) Skripte mit den Mitteln der vom Leser bevorzugten Shell

Inhalt des Kapitels

Unix Shells - Allgemeines

Geschichte der Shells

1979 lieferte AT&T UNIX Version 7 mit der nach ihrem Erfinder Stephen Bourne benannten Shell bsh aus, die als erste Standard-Shell für Unix-Systeme gilt. Bsh basierte auf der Programmiersprache Algol 68 und sollte die Aufgaben eines Administrators automatisieren. Neue Errungenschaften der Bsh waren vor allem die History, Aliasing und die Fähigkeit der Prozesskontrolle.

So wie die verschiedenen Unix-Derivate aus dem Boden schossen, entstanden auch eine Vielzahl von Kommandozeileninterpretern mit zum einen unterschiedlichen Merkmalen und zum anderen unterschiedlichem Erfolg. Parallel zur Bsh wurde in Berkeley von Bill Joy die C-Shell - der Name lässt die nahe Verwandtschaft zur Programmiersprache C schon erahnen - entwickelt, die bald zum Lieferumfang eines jeden BSD-Unix avancierte. Gerade die Mächtigkeit der Programmierung verhalf dieser Shell zu einer großen Verbreitung. Unter Linux kommt heute oft die erweiterte Tenex-C-Shell (tcsh) zum Einsatz.

Vielleicht war ja der umstrittene Disput zwischen Bsh- und Csh-Verfechtern der Anlass für David Korn, 1986 mit System V Release 4 (AT&T) seine Version einer Shell auszuliefern. Die Ksh beinhaltet alle Fähigkeiten der Bsh, ergänzt um einige Möglichkeiten der Csh.
Die Korn Shell war so ziemlich die erste Shell, die auch für andere Betriebssysteme verfügbar wurde. Heute sind auch Bash - die um die wichtigsten Konstrukte der Csh und Ksh erweiterte Bsh - und Tcsh unter anderen Systemen (auch Windows) vorhanden.

Aufgaben einer Shell

Die wichtigste Aufgabe einer Shell im interaktiven Modus ist die Interpretation der Kommandozeile. Eine Shell nimmt die Eingabe des Nutzers entgegen, teilt diese an den Whitespaces (Leerzeichen, Tabulator, Zeilenumbruch) in so genannte Token auf, ersetzt Aliasse und Funktionen, organisiert die Umleitung von Ein- und Ausgaben, substituiert Variablen und Kommandos und nutzt vorhandene Metazeichen zur Dateinamensgenerierung. Anschließend sucht die Shell nach dem angegebenen Kommando und übergibt diesem die komplette Kommandozeile als Argumente.

Während der Initialisierung ist die Shell für die Umgebung des Nutzers verantwortlich. Dazu liest eine Shell spezielle, ihr zugeordnete Konfigurationsdateien ein und setzt z.B. Aliasse oder die PATH-Variable des Nutzers usw.

Ebenso bietet jede Shell die Möglichkeit der Shellprogrammierung, um bestimmte Abläufe automatisieren zu helfen. Mächtige Shells beinhalten hierzu Programmkonstrukte wie bedingte Ausführung, Schleifen, Variablenzuweisung und Funktionen.

Parsen der Kommandozeile

Das Prinzip des Parsens der Kommandozeile ist allen in diesem Kapitel besprochenen Shells gleich. Auf die konkreten Realisierungen kommen wir in den Abschnitten der speziellen Shells zurück.

1. Lesen der Kommandozeileneingabe
2. Zerlegen der Kommandozeile in Token
3. Aktualisierung der History
4. Auflösung der in (doppelten) Anführungsstriche eingeschlossenen Bereiche ("Quotes")

5. Ersetzung der Aliasse und Funktionen (die (T)csh kennt keine Funktionen)
6. Organisation von EA-Umleitung, Hintergrundprozessen und Pipes
7. Ersetzung der Variablen durch ihren Inhalt
8. Substitution von Kommandos durch ihr Ergebnis
9. Substitution von Dateinamen ("Globbing")
10. Ausführen des Programms

Arten von Kommandos

Es existieren 4 Arten von Kommandos:

1. Aliasse
2. Funktionen (nicht (T)csh)
3. Builtins
4. Ausführbare Programme

Die Nummerierung entspricht dabei der Reihenfolge, nach der eine Shell nach einem Kommando sucht.

Ein **Alias** ist einfach nur ein anderer Name für ein Kommando oder eine Kommandofolge. Würden wir einen Alias mit dem Namen eines vorhandenen Kommandos definieren, so wäre das originale Kommando nur noch durch Angabe seines vollständigen Zugriffspfades erreichbar:

```
user@sonne> alias ls= "echo 'ls ist weg'"
user@sonne> ls
ls ist weg
user@sonne> / bin/ ls
ToDo        entwicklung.htm   khist.htm        shallg.htm~
access.htm
user@sonne> unalias ls
```

Da ein Alias wiederum einen Alias beinhalten kann, wiederholen die Shells die Substitution solange, bis der letzte Alias aufgelöst ist.

Funktionen gruppieren Kommandos als eine separate Routine.

Builtins sind Kommandos, die direkt in der Shell implementiert sind. Einige dieser Kommandos müssen zwingend Bestandteil der Shell sein, da sie z.B. globale Variablen manipulieren (cd setzt u.a. PWD und OLDPWD). Andere Kommandos werden aus Effizienzgründen innerhalb der Shell realisiert (ls).

Ausführbare **Programme** liegen irgendwo auf der Festplatte. Zum Start eines Programms erzeugt die Shell einen neuen Prozess, alle anderen Kommandos werden innerhalb des Prozesses der Shell selbst ausgeführt.

Shell und Prozesse

Wer die ersten Kapitel nicht gerade übersprungen hat, dem sollte der Begriff des Prozesses schon geläufig sein. Hier noch einmal eine Zusammenfassung der Charakteristiken eines Prozesses:

Ein Prozess ist ein Programm, das sich in Ausführung befindet, d.h.

- Der Prozess besitzt eine eindeutige Prozessnummer (PID)
- Der Prozess wird vom Kernel kontrolliert
- Zum Prozess gehören:
    - das auszuführende Programm
    - die Daten des Programms
    - der Stack (Speicherbereich für z.B. Rücksprungadressen von Routinen)
    - der Programm- und Stackzeiger
    - ein Eintrag in die Prozesstabelle des Kernels

    - ...
- Der Prozess besitzt die Rechte eines Nutzers und einer (oder mehrerer) Gruppe(n)
- Jeder Prozess (bis auf eine Ausnahme: init) hat einen eindeutigen Vorfahren

Geben wir nun auf der Kommandozeile etwas ein, ist es Aufgabe der Shell, das Kommando zu finden und, insofern es sich nicht um ein builtin-Kommando handelt, einen neuen Prozess zu erzeugen, welcher dann das Programm ausführt.

Die Shell bedient sich dabei verschiedener **Systemrufe**, d.h. Aufrufe von Funktionen des Kernels. Die uns nachfolgend interessierenden Systemrufe sind:

- **fork()** Zum Erzeugen eines neuen Prozesses
- **exec()** Zum Laden und Ausführen eines neuen Programms
- **exit()** Zum Beenden eines Prozesses
- **wait()** Zum Warten auf das Ende eines Prozesses

Der Systemruf fork()

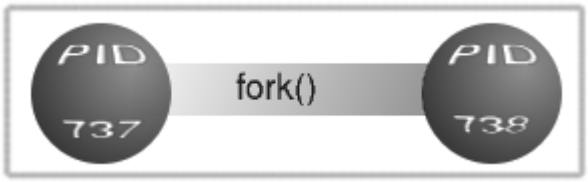


Abbildung 1: fork() - Erzeugen eines neuen Prozesses

Unter Unix ist **fork()** die einzige Möglichkeit, einen neuen Prozess zu erzeugen (Threads - so genannte Leichtgewichtsprozesse - haben nicht viel mit dem klassischen Prozesskonzept gemeinsam; für sie existieren eigene Systemrufe).
Ein existierender Prozess erzeugt also einen neuen **Kindprozess**, der sich zunächst nur durch eine **eigene Prozessnummer** von seinem Elternprozess unterscheidet. Der Kindprozess erhält eine exakte Kopie der Umgebung seines Vorfahrens, einschließlich geöffneter Dateideskriptoren, Rechten (UID, GID), Arbeitsverzeichnis, Signalen...

Beide Prozesse führen zunächst ein und dasselbe Programm aus und teilen sich vorerst diesen Speicherbereich. Erst wenn der Kindprozess ein neues Programm laden sollte, wird ihm ein eigenes Codesegment zugestanden.

Beispiel:

```
/* Programm forktest.c demonstriert das Erzeugen eines neuen Prozesses
und den konkurrierenden Ablauf von Kind- und Elternprozess */
#include <stdio.h>
#include <unistd.h>

int main () {
  int pid;
  int counter = 8;
  printf("...Programm gestartet...\n");

  pid = fork ();

  while ( --counter ) {
    printf("%d ", getpid());
    sleep(1);        /* 1 Sekunde schlafen */
    fflush(stdout);  /* Standardausgabe leeren */
```

```
  }
  printf(" Ende\n");

  return 0;
}
```

Wir übersetzen das Programm und starten es:

```
user@sonne> gcc forktest.c -o forktest
user@sonne> ./ forktest
...Programm gestartet...
681 680 681 680 681 680 681 680 681 680 681 680 681 Ende
680 Ende
```

Erläuterung: Anhand der Ausgabe ist zu erkennen, dass der Kindprozess tatsächlich den ersten Befehl nach dem "fork()"-Aufruf ausführt, d.h. auch der Programmzeiger wurde vererbt. In der nachfolgenden Schleife gibt jeder Prozess seine PID aus. Die Aufrufe von "sleep()" und "fflush()" dienen nur der Demonstration, ohne diese würde vermutlich jeder Prozess innerhalb eines CPU-Zyklus sein Programm abarbeiten können und die Ausgaben wären nicht alternierend.

Insofern der Elternprozess nicht explizit auf die Beendigung seiner Kinder wartet (siehe "wait") , laufen die Prozesse parallel ab.

Der Systemruf exec()

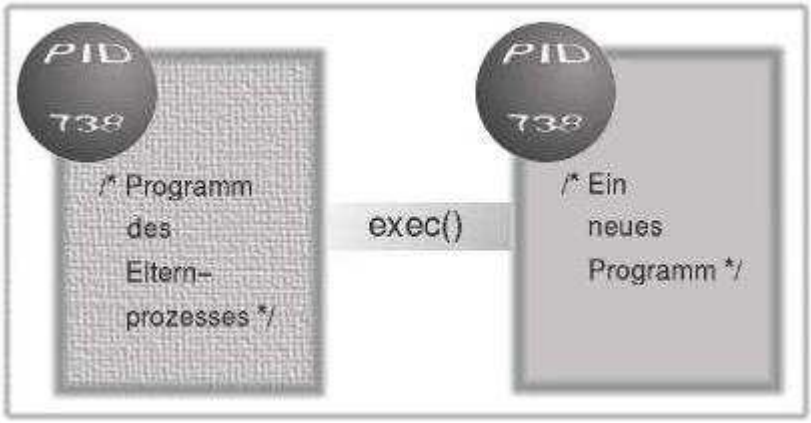


Abbildung 2: exec() - Laden eines neuen Programms

Das Starten von Prozessen, die alle ein und dasselbe Programm ausführen, ist auf die Dauer nicht sehr produktiv... Zum Glück liefert der Aufruf von fork() einen Wert zurück, anhand dessen die beteiligten Prozesse erfahren können, wer denn nun das Kind und wer der Vorfahre ist. fork() gibt, falls kein Fehler auftrat, beim Elternprozess die PID des Kindes zurück (bei Fehler "-1"). Im Kindprozess bewirkt der Systemruf die Rückgabe des Wertes "0". Anhand des Rückgabewertes wird ein Programm nun **alternative Programmpfade** beschreiten. Der Kindprozess wird mittels **exec()** (es existieren verschiedene Systemrufe zu diesem Zweck) ein neues Programm nachladen. Ab diesem Zeitpunkt wird für das Programm des Kindes ein eigener Speicherbereich reserviert.

```
user@sonne> bash
user@sonne> ps -T
  PID TTY      STAT   TIME COMMAND
 9905 tty4     S      0:00 login -- user
 9977 tty4     S      0:00 -sh
 9986 tty4     S      0:00 bash
 9994 tty4     R      0:00 ps -T
user@sonne> exec ps -T
```

```
  PID TTY      STAT   TIME COMMAND
 9905 tty4     S      0:00 login -- user
 9977 tty4     S      0:00 -sh
 9986 tty4     R      0:00 ps -T
user@sonne> ps -T
  PID TTY      STAT   TIME COMMAND
 9905 tty4     S      0:00 login -- user
 9977 tty4     S      0:00 -sh
10018 tty4     R      0:00 ps -T
```

Beispiel: In einem C-Programm wird ein "exec()"-Aufruf wie folgt vollzogen:

```
/* Programm exectest.c demonstriert das Laden eines neuen Programms
durch den Kindprozess */
# include <stdio.h>
# include <unistd.h>

int main () {
  int pid;

  printf("...Programm gestartet...\n");

  pid = fork ();

  if ( pid ) {
    printf("[Eltern (%d)] PID des Kindes: %d\n", getpid(), pid);
  }
  else {
    printf("[Kind (%d)] PID des Elternprozesses: %d\n", getpid(), getppid());
    execl("/bin/date", "-u", NULL);
  }
  printf("[%d ]Programmende\n", getpid());
  return 0;
}
```

Wiederum übersetzen wir das Programm und starten es:

```
user@sonne> gcc exectest.c -o exectest
user@sonne> ./ exectest
...Programm gestartet...
[Eltern (3311)] PID des Kindes: 3312
[Kind (3312)] PID des Elternprozesses: 3311
Sam Jul 8 09:05:07 MEST 2000
[3311] Programmende
```

Erläuterung: Das Programm demonstriert das Beschreiten alternativer Programmpfade. Für den Elternprozess ist "pid" verschieden von "0", so wird der erste Teil des "if"-Konstrukts betreten. Im Falle des Kindprozesses liefert "pid" "0", so dass mit dem "else"-Zweig fortgefahren wird. Nach einer Ausgabe wird das Kind das aktuelle Programm durch das Programm "date" ersetzen. Der Systemruf "exec()" existiert nicht selbst als C-Funktion, sondern ist in verschiedenen Varianten implementiert, eine davon ist "execl()". Der Kindprozess hat nun nichts mehr mit dem alten Programm zu tun, deshalb erscheint von ihm auch keine "Programmende"-Ausgabe.

Der Systemruf exit()

Ein Prozess beendet seine Existenz spätestens, wenn das ausgeführte Programm abgearbeitet ist. Innerhalb eines Programms kann dieses an beliebiger Stelle mit dem Systemruf **exit()** verlassen werden. Ein Prozess sendet vor seinem Ende noch ein Signal "SIGCHILD" an seinen Elternprozess und wartet, dass dieser das Signal behandelt. Gleichzeitig sollte jedes Programm einen Rückgabewert ("Status") liefern, der den Erfolg (Wert "0") oder einen

möglichen Fehlercode (Wert "1-255") meldet.

Der Status des letzten Kommandos kann in einer Shell abgefragt werden:

```
# bash, ksh, tcsh
user@sonne> echo $?
0

# (t)csh
user@sonne> echo $status
0
```

Der Systemruf wait()

Eine Shell wird, während ein Kindprozess seine Arbeit erledigt, auf dessen Rückkehr warten. Dazu ruft die Shell (und die meisten Programme, die weitere Prozesse erzeugen, verfahren so) den Systemruf **wait()** auf und geht in den schlafenden Zustand, aus dem sie erst erwacht, wenn der Kindprozess terminiert.

```
user@sonne> sleep 10 & wait % 1
# 10 Sekunden verstreichen...
user@sonne>
```

Hat der Elternprozess das Signal des Kindes behandelt, gilt dieser Prozess als beendet, d.h. sein gesamter Speicherbereich wurde freigegeben und sein Eintrag aus der Prozesstabelle des Kernels entfernt.

Beispiel:

```
/* Programm waittest.c demonstriert das explizite Warten des
Elternprozesses auf das Ende des Kindes */
#include <stdio.h>
#include <unistd.h>

int main () {
  int pid;
  int counter = 8;

  printf("...Programm gestartet...\n");

  pid = fork ();
  if ( pid ) {
    printf("[Eltern (%d)] PID des Kindes: %d\n", getpid(), pid);
    wait(pid);
  }
  else {
    printf("[Kind (%d)] PID des Elternprozesses: %d\n", getpid(), getppid());
  }

  while ( --counter ) {
    printf("%d ", getpid());
    sleep(1);        /* 1 Sekunde schlafen */
    fflush(stdout); /* Standardausgabe leeren */
  }
  printf(" Ende\n");

  return 0;
}
```

Wir übersetzen das Programm und starten es:

```
user@sonne> gcc waittest.c -o waittest
user@sonne> ./ waittest
...Programm gestartet...
[Eltern (3341)] PID des Kindes: 3342
[Kind (3342)] PID des Elternprozesses: 3341
3342 3342 3342 3342 3342 3342 3342 Ende
3341 3341 3341 3341 3341 3341 3341 Ende
```

Erläuterung: Nach der ersten Ausgabe durch den Elternprozess wartet dieser mittels "wait()", bis sein Kindprozess seine Arbeit beendet hat. Deshalb erscheinen die Ausgaben der PID des Elternprozesses erst nach denen des Kindes.

Ein Elternprozess **muss nicht** auf die Terminierung seiner Kindprozesse **warten**. Sendet in diesem Fall ein Kind das Signal "SIGCHILD", dann befindet es sich aus Sicht des Systems im Zustand **Zombie**. Der Prozess ist zwar beendet, aber sein Signal wurde noch nicht behandelt und der Eintrag in der Prozesstabelle existiert weiterhin.

Es soll auch vorkommen, dass die Eltern das Zeitliche vor ihren Nachfahren segnen (z.B. durch expliziten Abbruch durch den Nutzer oder durch einen Programmfehler). In einem solchen Fall spricht man vom Kind von einem **Waisen**. In dieser Situation übernimmt **init** - der Vorfahre aller Prozesse - die Rolle des Elternprozesses.

Prozesskommunikation

Es gibt eine Reihe von Möglichkeiten, wie Prozesse untereinander Nachrichten austauschen können. Gebräuchliche Mechanismen unter Unix-Systemen sind die Interprozesskommunikationen "**IPC**" nach SystemV und einige BSD-Entwicklungen. Dazu zählen:

- **Gemeinsame Speicherbereiche (Shared Memory):** (SystemV) Mehrere Prozesse teilen sich einen Speicherbereich im Hauptspeicher und kommunizieren über gemeinsame Daten.
- **Semaphore:** (SystemV) Über solche kann der Zugriff auf Ressourcen geregelt werden. Insbesondere im Zusammenhang mit Shared Memory werden Semaphore benutzt, um die gleichzeitige Manipulation von Daten durch mehrere Prozesse auszuschließen.
- **Nachrichtenaustausch (Message Passing):** (SystemV) Prozesse senden und empfangen Nachrichten.
- **Pipes:** (BSD) Prozesse kommunizieren über gemeinsame Puffer oder spezielle Dateien.
- **Signale:** (BSD) Prozesse senden anderen Prozessen Signale.
- **Sockets:** (BSD) Prozesse kommunizieren über einen Kommunikationsendpunkt (Socket) im Netzwerk.

Von den erwähnten Mechanismen stellen die Shells zur Kommunikation der in ihnen gestarteten Prozesse Pipes (benannt/unbenannt) und Signale zur Verfügung.

Pipes

Eine Pipe verbindet die Ausgabe eines Kommandos mit der Eingabe eines anderen Kommando. D.h. das erste Kommando schreibt seine Ausgaben anstatt auf die Standardausgabe in die Pipe, während das zweite Kommando aus dieser Pipe und nicht von der Standardeingabe liest. Beliebig viele Kommandos lassen sich durch Pipes miteinander verknüpfen. Die Shell kümmert sich dabei um die Synchronisation, so dass ein aus der Pipe lesendes Kommando tatsächlich erst zum Zuge kommt, nachdem ein schreibendes Kommando den Puffer der Pipe gefüllt hat.

Nicht alle Kommandos akzeptieren ihre Eingaben aus einer Pipe. Kommandos, die es tun, bezeichnet man deshalb auch als **Filter**.

Alle uns interessierenden Shells stellen als Symbol für die Pipe den Strich "|" zur Verfügung:

```
user@sonne> ls -l | cut -b 0-12,56- | less
```

Eine durch das Zeichen "|" realisierte Pipe bezeichnet man als **unbenannte Pipe**. Die Shell bedient sich des **pipe ()**-Systemrufes und erhält vom Betriebssystem einen Deskriptor auf die Pipe. Gleichzeitig stellt der Kernel einen Speicherbereich bereit, den sich die durch die Pipe verbundenen Prozesse teilen. Sind alle Pipes der Kommandozeile erzeugt, startet die Shell für jedes Kommando einen Kindprozess und setzt deren Dateideskriptoren auf die entsprechenden Pipedeskriptoren.
Eine Kommunikation über einen solchen Pufferbereich im Kernel geht immer schneller, als die Kommunikation über eine Fifo-Datei (First In First Out). Fifo-Dateien implementieren die **benannten Pipes** (named pipes). Das Anlegen einer Fifo-Datei erfolgt mit:

```
user@sonne> mkfifo fifo_datei
user@sonne> ls -l fifo_datei
prw-r--r--   1 user    users        0 Jun  2 11:00 fifo_datei
```

Mit den Mechanismen der Ein- und Ausgabeumleitung kann ein Prozess in eine solche Datei schreiben, während ein anderer aus ihr liest. Mittels "Named Pipes" können auch Prozesse kommunizieren, die nicht unmittelbare Nachfahren der Shell sind.

Signale

Jedes Signal ist mit einer bestimmten Reaktion verbunden. Es liegt in der Verantwortung des Entwicklers, ob ein Programm auf ein Signal mit den "üblichen" Aktionen antwortet oder ob es dieses Signal sogar ignoriert. Das Signal "KILL" (Signalnummer 9) beendet einen Prozess, ohne dass dieses vom ausgeführten Programm abgefangen werden kann.

Wichtige Signale und ihre Wirkung sind (die vollständige Liste der Signale ist in der Datei "/usr/include/signal.h" zu finden):

SIGHUP (1)

Terminal-Hangup, bei Dämonen verwendet, um ein erneutes Einlesen der Konfigurationsdateien zu erzwing

SIGINT (2)

Tastatur-Interrupt

SIGQUIT (3)

Ende von der Tastatur

SIGILL(4)

Illegaler Befehl

SIGAbrT (5)

Abbruch-Signal von *abort(3)*

SIGFPE (8)

Fließkommafehler (z.B. Division durch Null)

SIGKILL (9)

Unbedingte Beendigung eines Prozesses

SIGSEGV (11)

Speicherzugrifffehler

SIGPIPE (13)

SIGTERM (15)

Prozess soll sich beenden (default von kill)

SIGCHLD (17)

Ende eines Kindprozesses

SIGCONT (18)

Gestoppter Prozess wird fortgesetzt

SIGSTOP (19)

Der Prozess wird gestoppt

SIGTSTP (20)

Ausgabe wurde angehalten

SIGUSR1 (30)

Nutzerdefiniertes Signal

Die Shells beinhalten das Kommando **kill**, um Signale "von außen" an Prozesse versenden zu können. Der Aufruf lautet:

```
kill <Signalnummer> <Prozessnummer[n]>
```

Die "Signalnummer" kann dabei als numerischer Wert oder durch den symbolischen Namen ("TERM", "HUP, "KILL", ...) angegeben werden.

```
user@sonne> kill -HUP 'cat / var/ run/ inetd.pid'
user@sonne> kill -15 2057 2059
```

Das Versenden einiger Signale unterstützen die Shells durch Tastenkombinationen. Während die Csh nur das [Ctrl]-[C] akzeptiert, existieren unter der Bsh, Ksh und Tcsh mindestens folgende Shortcuts:

[Ctrl]+[C]

SIGINT - Ausführung des aktiven Prozesses abbrechen

[Ctrl]+[Z]

SIGSTOP, stoppt die Ausführung des aktiven Prozesses

Umgebung und Vererbung

Nach einem erfolgreichen Anmelden ins System steht dem Nutzer i.d.R. eine komplette Arbeitsumgebung zur Verfügung. Unter anderem sind wichtige Shellvariablen (PATH, USER, DISPLAY, ...) vorbelegt, einige Aliasse sind vorhanden, bestimmte Programme wurden bereits gestartet...

Die Startup-Dateien der Bash und Ksh

Für diese initiale **Umgebung** sind verschiedene Konfigurations-Dateien der Shells verantwortlich, die teils bei

jedem Start einer Shell, teils nur beim Start einer Login-Shell abgearbeitet werden.

Die **Bash** und die **Ksh** führen während des ersten Starts die Kommandos der Datei **/ etc/ profile** aus. Ein Ausschnitt dieser, in dem einige Shellvariablen vorbelegt und Aliasse gesetzt werden, sei kurz aufgelistet:

```
# Ausschnitt einer Datei /etc/profile
PROFILEREAD=true

umask 022
ulimit -Sc 0         # don't create core files
ulimit -d unlimited

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 "$UID" = 0 && PATH=/sbin:/usr/sbin:/usr/local/sbin/:$PATH
export PATH
if test "$UID" = 0 ; then
    LS_OPTIONS='-a -N --color=tty -T 0';
else
    LS_OPTIONS='-N --color=tty -T 0';
fi
export LS_OPTIONS

alias dir='ls -l'
alias ll='ls -l'
alias la='ls -la'
alias l='ls -alF'
alias ls-l='ls -l'
```

Zum jetzigen Zeitpunkt sollte den Leser nur die Tatsache interessieren, **dass** die wichtigen Variablen beim Start einer Shell vorbelegt werden. Die Erläuterung der verwendeten Syntax bleibt den Abschnitten zur Programmierung der Shells vorbehalten.

Im Falle der **Bash** folgt die Abarbeitung einer Reihe weiterer Dateien, sofern diese im Heimatverzeichnis des jeweiligen Nutzers existieren. Die Dateien sind: **~ / .bash_profile**, **~ / .bash_login** und **~ / .profile**. Alle drei Dateien können vom Besitzer, also dem Nutzer selbst, bearbeitet werden und ermöglichen eine individuelle Einrichtung der Umgebung.

Die **Ksh** betrachtet neben der schon erwähnten **/ etc/ profile** nur die Datei **~ / .profile**.

Wird die Bash nicht als Loginshell gestartet (z.B. als Subshell zum Ausführen eines Skripts), dann bearbeitet sie nur die Datei **~ / .bashrc**. Die Ksh liest keine Konfigurationsdatei ein.

Die Startup-Dateien der Tcsh

Die **Tcsh** verwendet eine grundlegend andere Programmiersprache. Demzufolge benötigt sie auch andere Konfigurationsdateien. Die Dateien **/ etc/ csh.cshrc** und **/ etc/ csh.login** übernehmen dabei die Rolle der /etc/profile von Bash und Ksh. Die Ausschnitte aus diesen, die im wesentlichen dieselben Initialisierungen veranlassen, die weiter oben im Beispiel zur /etc/profile dargestellt wurden, seien hier angeführt:

```
# Ausschnitt einer Datei /etc/csh.cshrc
setenv MACHTYPE `uname -m`

unalias ls
if ( "$uid" == "0" ) then
    setenv LS_OPTIONS '-a -N --color=tty -T 0';
else
    setenv LS_OPTIONS '-N --color=tty -T 0';
endif
```

```
alias ls 'ls $LS_OPTIONS'
alias la 'ls -AF --color=none'
alias ll 'ls -l --color=none'
alias l  'll'
alias dir  'ls --format=vertical'
alias vdir 'ls --format=long'
alias d dir;
alias v vdir;

# Ausschnitt einer Datei /etc/csh.login
umask 022
setenv SHELL /bin/tcsh
```

Weitere Dateien, die während des Starts einer (T)csh betrachtet werden, liegen im Heimatverzeichnis und sind: **~/.tcshrc** oder, falls erstere Datei nicht existiert, **~/.cshrc**, **~/.history**, **~/.login** und **~/.cshdirs**. Die Reihenfolge der Auswertung der Dateien hängt von der konkreten Implementierung ab.

Vererbung

Vererbung im Sinne der Prozessentstehung wurde im Punkt Shell und Prozesse besprochen. Für den Nutzer ist insbesondere der Geltungsbereich von Variablen von Bedeutung, die alle Shells verwenden, um das Verhalten der in ihnen gestarteten Programme zu steuern.

Die Shells kennen **lokale** und **globale** Variablen. Eine lokale Shellvariable ist dabei nur innerhalb der Shell ihrer Definition sichtbar und wird **nicht** an die in der Shell gestarteten Programme weiter gereicht. **Globale** Variablen hingegen sind ab der Shell ihrer Einführung sichtbar, also auch in allen aus dieser Shell gestarteten Programmen.

Zur Definition von Variablen existiert in allen Shells das builtin-Kommando **set**:

```
user@sonne> set local_x=5
user@sonne> echo $local_x
5

user@sonne> set local_string="Enthält die Zuweisung Leerzeichen, muss sie 'gequotet' werden"
user@sonne> echo $local_string
Enthält die Zuweisung Leerzeichen, muss sie 'gequotet' werden
```

In der Bash und Ksh kann auf das Kommando **set** verzichtet werden, da beide Shells anhand des Gleichheitszeichens eine Zuweisung erkennen. Allerdings dürfen im Unterschied zur (T)csh bei beiden keine Leerzeichen vor und nach dem Gleichheitszeichen stehen.

Dass eine solche Variable tatsächlich nur in der aktuellen Shell sichtbar ist, lässt sich leicht überprüfen:

```
user@sonne> set local_x=5
user@sonne> bash
user@sonne> echo $local_x

user@sonne> exit
user@sonne> echo $local_x
5
```

Im Beispiel wurde eine Subshell (die bash) gestartet, in dieser ist die Variable "local_x" nicht bekannt (leere Zeile).

Soll eine Variable nun auch in den anderen Shells sichtbar sein, muss sie **exportiert** werden:

```
user@sonne> set local_x=5
# Bash und Ksh:
user@sonne> export local_x
# (T)csh
user@sonne> setenv local_x
```

Wir testen die Existenz der Variable in einer Subshell:

```
# Beispiel anhand der Bash:
user@sonne> export local_x=5
user@sonne> bash
user@sonne> echo $local_x
5
user@sonne> exit
user@sonne> echo $local_x
5
```

Umleitung der Ein- und Ausgaben

Alle Ein- und Ausgaben werden vom Kernel über den Mechanismus der File-Deskriptoren behandelt. So ein Deskriptor ist eine kleine nichtnegative Zahl (unsigned Integer), die einen Index auf eine vom Kernel verwaltete Tabelle ist. Jeder offene E/A-Kanal und jede offene Datei (named Pipe, Socket) wird nur über einen solchen Eintrag angesprochen.

Jeder Prozess erbt nun seine eigene Deskriptortabelle von seinem Vorfahren. Üblicherweise sind die drei ersten Einträge der Tabelle (0, 1 und 2) mit dem Terminal verbunden und werden mit **Standardeingabe** (0), **Standardausgabe** (1) und **Standardfehlerausgabe** (2) bezeichnet. Öffnet ein Prozess nun eine Datei (oder eine named Pipe oder einen Socket), so wird in der Tabelle nach dem nächsten freien Index gesucht. Und dieser wird der Deskriptor für die neue Datei. Die Größe der Deskriptor-Tabelle ist beschränkt, so dass nur eine bestimmte Anzahl Dateien gleichzeitig geöffnet werden können (Vergleiche Limits unter Linux).

Man spricht von **E/ A-Umleitung**, sobald ein Dateideskriptor nicht mit einem der Standardkanäle 0, 1 oder 2 verbunden ist. Shells realisieren die Umleitung, indem zunächst der offene Deskriptor geschlossen wird und die anschließend geöffnete Datei diesen Deskriptor zugewiesen bekommt.

Standardein- und Standardausgabe lassen sich in allen drei Shells analog manipulieren:

```
user@sonne> cat < infile > outfile
```

Die Shells lösen solche Umleitungen von links nach rechts auf. Im Beispiel wird also zunächst der Deskriptor der Standardeingabe geschlossen (symbolisiert durch "<"), anschließend wird die Datei "infile" geöffnet. Sie erhält den ersten freien Index zugewiesen und dieser ist nun die "0". Im nächsten Schritt wird die Standardausgabe geschlossen (Symbol ">"). Die nun zu öffnende Datei "outfile" bekommt den Deskriptor "1" zugewiesen. Das Kommando "cat" bezieht also seine Eingaben aus "infile" und schreibt das Ergebnis nach "outfile".

Die Umleitung der Standardfehlerausgabe wird bei der (T)csh etwas abweichend vom Vorgehen bei Bash und Ksh gehandhabt. Der Grund ist, dass man bei Bash und Ksh das Umleitungssymbol mit einem konkreten Deskriptor assoziieren kann (z.B. bezeichnet "2>" die Standardfehlerausgabe), die (T)csh aber keine solche Bezeichnung für die Standardfehlerausgabe kennt.

Das obige Beispiel hätte man in Bash und Ksh auch folgendermaßen ausdrücken können:

```
user@sonne> cat 0< infile 1> outfile
```

Und analog zu diesem Schema lässt sich gezielt die Standardfehlerausgabe umleiten:

```
# Nur Bash und Ksh
user@sonne> find / -name "README" > outfile 2> errfile
```

Die (T)csh kennt, wie auch Bash und Ksh, das Symbol **>&**, womit gleichzeitig Standard- und Standardfehlerausgabe in eine Datei umgeleitet werden können. Der Umweg, den man zur Umleitung der Fehler in der (T)csh beschreiten muss, geht über die Verwendung einer Subshell, in der nun die normalen Ausgaben abgefangen werden, und das, was übrig bleibt (Fehler), wird außerhalb der Subshell behandelt.

```
# Tcsh
user@sonne> (find /  -name "README" > outfile) >& errfile
```

Sie als Leser finden das verwirrend? Dann wollen wir Ihnen einen Vorgeschmack geben, welche Feinheiten der Ein- und Ausgabeumleitung die kommenden Kapitel für Sie bereit halten.

Hier einige Beispiele für die Bash und Ksh:

```
# Standardausgabe dorthin senden, wo die Fehler hingehen
user@sonne> find /  -name "README" 1> &2

# Datei als Deskriptor 3 zum Lesen öffnen
user@sonne> exec 3<  infile

# "infile" sortieren und nach "outfile" schreiben
user@sonne> sort <&3 > outfile

# "infile" schließen
user@sonne> exec 3<&-
```

Skripte

Immer wieder wird man auf Probleme treffen, bei denen man sich fragt, warum es kein Programm gibt, das dieses löst? Entweder weil es zu komplex ist, ein solches zu schreiben und noch niemand solch starken Bedarf verspürte, dass er sich selbst dem Brocken widmete. Oder weil es so einfach ist, dass es sich quasi jeder selbst zusammenbasteln könnte.

Den umfangreichen Problematiken wird man nur schwerlich oder gar nicht mit Mitteln der Shellprogrammierung beikommen können, aber gerade wer reichlich auf der Kommandozeile hantiert, wird Problemstellungen begegnen, die immer und immer wiederkehren. Warum jedes Mal mühsam die Gedanken ordnen, wenn es einzig der Erinnerung eines Programmnamens bedarf, welches das Schema automatisiert?

In den nächsten Kapiteln werden Sie genügend Gelegenheit haben, der einem Neuling verwirrenden Syntax der Shellskripte Herr zu werden. Sie werden die unterschiedlichen Möglichkeiten, die die Shells bieten, bewerten und sich letztlich auf die Anwendung nur einer Shellprogrammiersprache festlegen. Welche das sein wird, ist Geschmackssache und wir hoffen mit der Reihenfolge unserer Darlegung (bash, tcsh, ksh), keine vorschnellen Wertungen beim Leser hervorzurufen. Jede der Sprachen hat Stärken und Schwächen...

Zu diesem Zeitpunkt möchten wir nur die Realisierung eines kleinen Programms mit den drei Shellsprachen gegenüber stellen, ohne irgend welche Wertungen einfließen zu lassen. Voran gestellt sei die Idee, die zum Schreiben des Skripts animierte.

Die Aufgabe. Die vorliegende Linuxfibel entsteht auf Basis von html-Dateien. Wir wirken zu dritt an der Realisierung, teils testen und schreiben wir in der Firma, teils daheim. Um die Dateien zu übertragen, sammeln wir sie in einem Archiv. Nun geschieht es immer wieder, dass ein schon "fertiger" Abschnitt oder eine Grafik doch wieder verworfen wird, so löschen wir nicht gleich die alte Datei, sondern nennen sie um und arbeiten auf einer Kopie weiter. Erzeugten wir nun ein neues Archiv, enthielt es die alten und die neuen Dateien und war damit viel größer, als eigentlich notwendig. In dem Wirrwarr von ca. 300 Dateien den Überblick zu wahren, war vergebene Müh', also kam die Idee zu einem Skript, das Folgendes realisieren sollte:

- Ausgehend von einer als Argument zu übergebenden Startdatei (.htm), findet das Skript alle html-Dateien, die erreichbar sind
- Referenzen über Rechnergrenzen hinweg sollen ausgeschlossen werden ("http://...")
- Alle Bilder (.gif, .jpg), die notwendig sind, sollen gefunden werden
- Quasi als Nebenprodukt sollen fehlerhafte Verweise auf Dateien gefunden werden
- Die Liste der html- und Bild-Dateien ist auszugeben, so dass sie von einem anderen Programm verarbeitet werden kann

Die Realisierung in der Bash und der Ksh

Bash und Ksh unterscheiden sich im Sinne der Programmierung nur unwesentlich, in unserem Beispiel überhaupt nicht. Um das Skript von der Korn Shell ausführen zu lassen, müssen Sie nur die erste Zeile von "#!/bin/sh" auf "#!/bin/ksh" ändern.

```
user@sonne> cat find_in_bash
#!/bin/sh
actual_entry=0
filelist[$actual_entry]=${1:?"Usage: $0 <filename>"}
dir=$(dirname $filelist[actual_entry])

# Liste der *.htm-Dateien, die von der Startdatei aus erreichbar sind
while :; do
# set ""$(...) auf eine Zeile!
    set ""$(sed -n 's/.*[Hh][Rr][Ee][Ff]="\(.*\.htm\).*/\1/p' ${filelist[$actual_entry]} | egrep -v"http|${filelist[$actual_entry]}")

  for i do
    [ "$i" = "" ] && continue
    if ! echo ${filelist[@]} | fgrep -q $i; then
      filelist[${#filelist[*]}]="$dir/$i"
    fi
  done

  actual_entry=$(($actual_entry+1))
  [ "$actual_entry" -ge"${#filelist[*]}" ] && break
done

# Bilderliste
pictures=$(sed -n's/.*[Ss][Rr][Cc]="\(.*\.gif\).*/\1/p' ${filelist[*]})
pictures="$pictures $(sed -n 's/.*[Ss][Rr][Cc]="\(.*\.jpg\).*/\1/p' ${filelist[*]})"

set $pictures
picturelist[0]="$dir/$1"
for i do
  if ! echo ${picturelist[@]} | fgrep -q $i; then
    picturelist[${#picturelist[*]}]="$dir/$i"
  fi
done

echo ${filelist[*]} ${picturelist[*]}
```

Die Realisierung in der Tcsh

Eine gänzlich andere, an die Programmiersprache C angelehnte Syntax verwendet die Tcsh. Beachten Sie bitte den Hinweis im Skript, dass bestimmte Zeilen in der Programmdatei zusammengefügt werden müssen. Die Tcsh kennt kein Zeilenfortsetzungszeichen, wie die Bash und Ksh mit dem Backslash.

```
user@sonne> cat find_in_tcsh
#!/bin/tcsh

set actual_entry = 1
set filelist = $1
set dir = `dirname $filelist`

# Liste der *.htm-Dateien, die von der Startdatei aus erreichbar sind
while (1)
# Eine Zeile!
  set files = `sed -n's/.*[Hh][Rr][Ee][Ff]="\(.*\.htm\).*/\1/p' $filelist[$actual_entry] | egrep -v "http|$filelist[$actual_entry]"`
# Zeile beendet...
  while ( $#files )
    echo $filelist | fgrep -q $files[1]
    if ( $status ) then
      set filelist=($filelist $dir/$files[1])
    endif
    shift files
  end
  @ actual_entry += 1
  echo "Vergleich $actual_entry mit $#filelist"
```

```
  if ( $actual_entry > $#filelist ) then
    break
  endif
end

# Bilderliste
set picturelist=""
set tmp=($filelist)
while ($#tmp)
# Eine Zeile!
  set pictures = `sed -n's/.*[Ss][Rr][Cc]="\(.*\.gif\).*/\1/p' $tmp[1]`
# Zeile beendet...
# Eine Zeile!
  set pictures = ($pictures `sed -n's/.*[Ss][Rr][Cc]="\(.*\.jpg\).*/\1/p' $tmp[1]`)

# Zeile beendet...
  while ($#pictures)
    echo xxx $picturelist | fgrep -q $pictures[1]
    if ($status) then
      set picturelist = ($picturelist $dir/$pictures[1])
    endif
    shift pictures
  end
  shift tmp
end

echo "$filelist $picturelist"
```

Das Skript ausführen

Falls Sie eines der Skripte in einer Datei gespeichert haben, so müssen Sie dieses noch mit den Ausführungsrechten versehen:

```
user@sonne> chmod +x find_in_bash
```

Sollten Sie dieses vergessen, wird eine Shell mit der Ausschrift "command not found" Ihr Anliegen abweisen.

Die Bourne Again Shell

Übersicht

Was ist die Bourne Again Shell?

Die Bash ist die mit Abstand beliebteste Shell unter Linux. Zum Teil mag das an ihrer frühen Verfügbarkeit liegen, während freie Implementierungen der Tcsh und Ksh für Linux erst nach zogen. Entscheidend scheint doch eher der enthaltene Funktionsumfang, den die anderen Shells nicht annähernd mit sich bringen und die Konformität mit dem IEEE POSIX P1003.2/ISO 9945.2 Shell und Tools Standard. Hierin ist auch der Grund für die Unterschiede zur Bourne Shell (sh bzw. bsh) zu sehen.

Auch wenn der Name Bash die enge Verwandtschaft zur Bsh nahe legt, so hat die GNU Bash gleich drei Vorfahren. Die eben genannte Bsh, die Csh und die Ksh. Von jedem erbte die Bash die Konstrukte, die sich aus der Erfahrung heraus als besonders nützlich erwiesen hatten.

Die Bash wird von der Free Software Foundation entwickelt und ist die Standardshell des Gnu-Betriebssystems HURD. Mit der Ende 1996 erschienenen Version 2.0 haben sich grundlegende Dinge gegenüber dem Vorgänger geändert. Wir werden uns ausschließlich auf die Bash > 2.0 beziehen; einige Gaben sind gar dem neuesten Vertreter entliehen. Aktuelle Version ist 2.0.4.

Gliederung dieses Abschnitts

Vorab ein paar Worte zu den Überschriften und was Sie unter den einzelnen Punkten finden.

Die Fähigkeiten der Bash und Definitionen stellen einen Versuch der Erklärung dar, warum so zahlreiche Anwender die Bash als "die tolle Shell" an sich ansehen. Enthalten sind auch einige Begriffe, die wir nachfolgend verwenden. Der Erklärungsbedarf besteht, weil es uns nicht gelang, für manche Bezeichnung einen adäquaten deutschen Ausdruck zu finden.

Jedes nützliche Programm wird einmal gestartet werden. So beginnt die Exkursion mit dem Start der Bash. Hierbei geht es uns um die Optionen, über die die Bash beim Aufruf gesteuert werden kann und um interne Abläufe, die zumeist unbemerkt im Hintergrund geschehen. Mit dem Erscheinen der ersten Eingabeaufforderung hat die Startmaschinerie ihre Aufgabe erfüllt und hier endet auch der Einblick dieses Abschnitts.

Und beim Beenden der Bash geschehen auch noch Dinge, von denen der Anwender selten etwas bemerkt...

Syntax klingt stark nach trockener Theorie. Und mit jener beschäftigt sich auch dieser Abschnitt. Der Schwerpunkt liegt auf »korrekten Eingaben«. So wird bspw. das Erzeugen, Manipulieren und Löschen von Variablen behandelt; mit keinem Wort wird allerdings der Wirkungsbereich von Variablen erwähnt; da dies kein syntaktisches Problem darstellt.

Wesentlich ist das Verstehen der Betrachtung der Kommandozeile durch die Bash. Syntax zeigt auf, nach welchen Regeln die Shell die Eingabe bearbeitet, beschränkt sich jedoch auf die Benennung der Mechanismen. Expansionen schließlich durchleuchtet Regel für Regel und erschließt dem ambitionierten Shellprogrammierer sein künftiges Betätigungsfeld.

Initialisierungen behandelt alle Methoden, die zur Konfiguration der Laufzeitumgebung der Bash beitragen. Hierunter fallen die Konfigurationsdateien, aber auch die Variablen - diesmal im Kontext ihrer Wirkungsweise. Bash-interne Variablen aktivieren bestimmte Verhaltensweisen oder schalten sie ab (sollte Ihre Bash abweichende Reaktionen zeigen, könnte eine Shelloption dafür verantwortlich zeichnen). Des Weiteren gliedern sich Aliasse,

Funktionen und eingebaute Kommandos in diesen Abschnitt ein.

Mit der Interaktiven Bash bewerten wir Methoden zur Navigation und Manipulation der Kommandozeile. Die Eingabehilfen passen ebenso in diese Thematik wie die Handhabung von Prozessen.

History wäre auch bei der *Interaktiven Bash* bestens aufgehoben. Jedoch sind die Möglichkeiten so weit reichend, dass wir dem Kommandozeilenspeicher einen eigenen Abschnitt widmen.

Fähigkeiten der Bash und Definitionen

Wichtige Fähigkeiten

Der Funktionsumfang der Bash ermöglicht sowohl ein komfortables Arbeiten als auch die Verrichtung komplexer Aufgaben. Hierzu zählen:

- Das Editieren der Kommandozeile kann nach beliebigen Schemen erfolgen. Sowohl vi-, emacs, als auch nutzerdefinierte Modi sind möglich.
- Die Bash besitzt einen Kommandozeilenspeicher (»History«). Alte Eingaben können gesucht, bearbeitet und erneut ausgeführt werden. Das konkrete Verhalten kann wiederum konfiguriert werden.
- Prozesse können vom Benutzer gestartet, gestoppt und ihre Eigenschaften verändert werden.
- Unvollständige Eingaben von Kommando- und Dateinamen u.a.m. können automatisch expandiert werden.
- Geringe Fehler in der Eingabe können automatisch korrigiert werden
- Das Eingabeprompt ist konfigurierbar.
- Berechnungen analog zu C werden unterstützt.
- Die Bash kennt Aliasse, Funktionen und Felder...

Einige Definitionen

Was es mit den knappen Aussagen konkret auf sich hat, soll Gegenstand der folgenden Abschnitte sein. Zuvor sollen benötigte Begriffe kurz erläutert werden.

Metazeichen

Ein Zeichen, das einzelne Worte trennt: | & ; ( ) < > Leerzeichen Tabulator

Name

Zeichenkette, bestehend aus alphanumerischen Zeichen und dem Unterstrich, wobei das erste Zeichen kein Ziffer ist.

Kontrolloperator

Ein Token, das eine spezielle Kontrollfunktion auslöst, die Symbole sind: | & && ; ;; ( ) Zeilenumbruch

Token

Eine Zeichenfolge, die von der Shell als eine Einheit betrachtet wird.

Whitespace

Steht für Leerzeichen und Tabulatoren und ggf. dem Zeilenumbruch.

Start der Bash

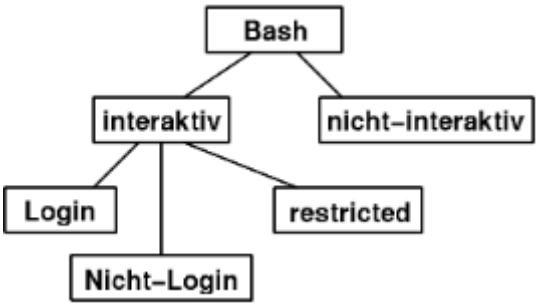
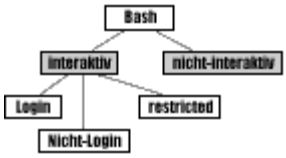


Abbildung 1: Unterteilung der Bash

Interaktive und Nicht-Interaktive Bash



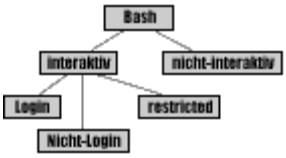
Ist eine Shell mit der Standardein- und Standardausgabe (sprich »mit einer (virtuellen) Konsole«) verbunden, so bezeichnet man sie als interaktive Shell. Die interaktive Bash teilt sich wiederum ein in die Login Bash, die Nicht-Login Bash und die Restricted Bash. Alle drei unterscheiden sich im Einlesen von Initialisierungsdateien; letztere Bash schränkt die Befugnisse des Anwenders ein.

Wenn die Bash startet, entscheidet sie anhand des Ergebnisses von »tty -s«, ob es sich um eine interaktive Shell handelt (dann ist der Rückgabewert des Kommandos "0"). Eine Login-Shell führt die Kommandos aus der Datei /etc/profile und aus der ersten gefundenen Datei ~/.bash_profile, ~/.bash_login oder ~/.profile aus, sofern die Dateien existieren, lesbar sind und die Bash nicht mit der Option »--noprofile« gestartet wurde.

Eine Nicht-Login Bash liest die Datei ~/.bashrc. Mit der Option »--rcfile Diese_Datei« kann eine alternative Ressourcen-Datei benannt und mit »--norc« das Einlesen unterdrückt werden.

Eine Nicht-Interaktive Shell (typisch sind Shellskripte) wertet einzig die Umgebungsvariable BASH_ENV aus. Enthält sie den vollständigen Pfad zu einer Datei, so wird deren Inhalt gelesen und ausgeführt.

Optionen beim Start



Die Bash ist ein Programm und nahezu jedes Programm unter Unix kann durch Optionen auf der Kommandozeile gesteuert werden. So kennt auch die Bash eine Reihe von Optionen, deren wichtigste vorgestellt sein sollen:

-c *Kommandofolge*

Die Bash liest und startet die Kommandos aus "Kommandofolge", welche als eine einzelne Zeichenkette anzugeben ist. Alles, was der Zeichenkette folgt, wird als Argument dem letzten Kommando der Kommandofolge übergeben:

```
user@sonne> bash date -u
date: /bin/date: cannot execute binary file
user@sonne> bash -c date -u
Sam Jul 8 10:16:17 MEST 2000
```

(Wird der Bash ein Argument übergeben, das keine Option ist, so interpretiert sie das Argument als Datei, d die Kommandos enthält. »date« ist natürlich keine Datei, die Kommandos beinhaltet, also beschwert sich di Bash...). Die Bash in Verbindung mit »-c« arbeitet als nicht-interaktive Shell.

-r bzw. --restricted

Die Bash arbeitet als »Restricted« Shell (siehe weiter unten).

-i

Die Bash arbeitet als interaktive Shell, d.h. die Standardein- und -ausgabe sind mit einem Terminal verbunc

--login

Die Bash arbeitet wie eine Login-Shell, d.h. alle Konfigurationsdateien werden eingelesen (Eine Login-Shell nichts mit der Anzeige einer Login-Aufforderung zu tun, sie nennt sich nur so, weil die einzulesenden Dateie zumeist einmalig beim Start einer Sitzung geladen werden.).

--noprofile

Unterdrückt das Einlesen der profile-Dateien.

--norc

Unterdrückt das Einlesen der »~/.bashrc«.

--posix

Die Bash verhält sich wie eine POSIX-Shell. Sie liest hierzu einzig die in der Umgebungsvariablen ENV angegebene Datei ein. Wird die Bash über den Link »sh« gerufen, startet sie ebenso im POSIX-Modus.

--rcfile Datei

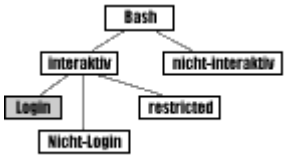
Eine nicht-interaktive Bash liest die angegebene Datei ein (anstatt ~/.bashrc)

-v bzw. --verbose

Schaltet die erweiterte Ausgabe ein. Sinnvoll ist diese Option in der Testphase von Shellskripten, um jede Z die ausgeführt wird, angezeigt zu bekommen.

Als weitere Optionen können zahlreiche Flags bereits beim Start der Bash aktiviert werden. Da dies auch bei laufender Shell geschehen kann - und dies der gebräuchliche Weg ist - behandeln wir die Flags erst im Zusammenhang mit dem builtin-Kommando set.

Bash als Login-Shell



In den meisten Konfigurationen, die die Distributoren ausliefern, dürfte die Bash als Login-Shell voreingestellt sein. Ist dem nicht so, kann der Benutzer mit Hilfe des Kommandos chsh die Bash in die Passwortdatei eintragen.

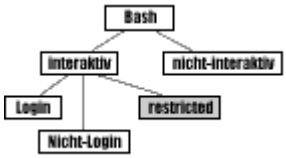
Dies funktioniert jedoch nur, wenn die Bash in der Datei /etc/shells eingetragen ist. Ist dem nicht so, könnte einem entweder der Administrator aus der Patsche helfen, indem er den Eintrag in obiger Datei ergänzt oder man baut in einem der Startup-Skripte seiner bisherigen Login-Shell den Aufruf »exec /usr/bin/bash --login« ein.

Bei dem zuletzt beschriebenem Vorgehen sollten Sie sicher stellen, dass das Skript mit dem Aufruf nicht auch von der Login-Bash ausgeführt wird, sonst hängt die Shell in einer Schleife mit Reinitialisierungen fest. Am besten fragen Sie im Skript ab, ob der Name des aktuell ausgeführten Programms die Bash ist:

```
...
case "$0" in
  *bash )
    ;;
  * )
```

```
    test -x /usr/bin/bash && exec /usr/bin/bash --login
    ;;
esac
...
```

Restricted Bash



Mit dem Argument **-r** gestartet, arbeitet die Bash als **restricted Shell**. Dasselbe Verhalten kann erreicht werden, wenn ein Link **rsh** bzw. **rbash** auf die Bash verweist und diese über den Link gerufen wird.

Das Anliegen einer »eingeschränkten Shell« ist, einen Anwender in seinen Befugnissen weiter einzuschränken. **Unter einer restricted Bash ist es nicht möglich:**

- Das Arbeitsverzeichnis zu wechseln:

```
user@sonne> cd ..
bash: cd: restricted
```

- Die Variablen SHELL, PATH, ENV und BASH_ENV zu verändern:

```
user@sonne> export PATH= ./ :$PATH
bash: PATH: readonly variable
user@sonne> unset SHELL
bash: unset: SHELL: cannot unset: readonly variable
```

- Einen Kommandoaufruf mit einem enthaltenen Schrägstrich (Slash) abzusetzen (es können also nur Kommandos ausgeführt werden, die in den Pfaden der Variable »PATH« enthalten sind):

```
user@sonne> / bin/ ls
bash: /bin/ls: restricted: cannot specify `/' in command names
```

- Mit dem Punkt-Kommando eine Datei auszuführen, deren Angabe einen Slash enthält:

```
user@sonne> . / etc/ profile
bash: .: /etc/profile: restricted
```

- Die Umleitung der Ein- und Ausgaben vorzunehmen:

```
user@sonne> ls > bla
bash: bla: restricted: cannot redirect output
```

- Während des Starts der restricted Bash Funktionen zu importieren und Shelloptionen zu setzen
- Mit dem **exec**-builtin-Kommando die Bash durch ein anderes Programm zu ersetzen
- builtin-Kommandos zu aktivieren bzw. zu deaktivieren (also ihren momentanen Status zu ändern)
- Den restricted-Modus abzuschalten
- Das Bash-builtin-Kommando **command -p** anzuwenden

Bei den Einschränkungen scheint eine Erlaubnis zum Zugang einer »nicht-vertrauenswürdigen« Person unkritisch zu sein. Doch **Vorsicht**! Stellen Sie sicher, dass die exportierte PATH-Variable keine Verzeichnisse enthält, die eine Shell enthalten. Es ist durchaus sinnvoll, zu diesem Zweck ein eigenes Binaryverzeichnis mit »unkritischen« Kommandos (analog zu FTP) anzulegen und einzig dieses in PATH aufzunehmen. Allerdings gibt es eine Reihe Kommandos (bspw. Editoren), die intern das Eröffnen einer Shell unterstützen und somit das Umgehen der Restriktionen ermöglichen. So ist es erlaubt, im vi eine »normale« Bash zu starten, selbst wenn der Editor

innerhalb einer restricted Shell gestartet wurde (neuere Versionen starten wiederum nur eine restricted Bash)!

Beenden der Bash

Handelt es sich um eine Login-Shell, so lässt sich die Bash sogar vor ihrem Ende zu einem letzten Groß-Reine-Machen motivieren. Alle Aktionen, die die Shell vor ihrem Ableben noch erledigen soll, sind hierzu in eine Datei **.bash_logout** zu schreiben.

Ob die Bash nun mittels **exit** oder **logout** verlassen oder sie mit einem Signal (Ausnahme: KILL) zum Ende gebracht wird, so wird sie abschließend noch das ihr auferlegte Pensum erledigen.

Die Bash muss nicht unbedingt verlassen werden. Ein Ersetzen des Programms durch ein anderes mittels **exec** kommt einer Beendigung gleich. **exec** wird im Zusammenhang mit eingebauten Kommandos behandelt.

Syntax der Bash

Es ist schier unmöglich, ein komplexes Thema wie die Bash »in einem Fluss« darzulegen. Nahezu jedes Konzept steht in Wechselwirkung mit einem anderen, nicht unwesentlicheren Aspekt. Mit welchem sollte man beginnen?

Bevor uns die praktischen Dinge der Bash den Kopf zerbrechen, steht deshalb die (trockene) Theorie der allgemeinen Syntax voran, natürlich garniert mit (hoffentlich) sinnvollen Beispielen. Und hier erwächst ein weiteres Problem. Wie soll ein Beispiel praktisch sein, wenn wir uns auf das bisherige Wissen beschränken?

Indem wir die Schranken missachten und den Leser auf den weiteren Text verweisen, der die fehlenden Wissenslücken schließen wird.

Der weitere Text erklärt:

- Die Verwendung von Kommentaren
- Die Syntax zur Definition von Variablen
- Die Regeln der Variablen-Expansionen
- Den Begriff »Einfache Kommandos«
- Die Regeln der Expansion von Worten
- Den Rückgabestatus eines Kommandos
- Das Prinzip des Pipelining
- Den Gebrauch von Listen
- Die Methoden zur Ein-/Ausgabe-Umleitung
- Die bedingte Ausführung von Kommandos
- Die wiederholte Kommandoausführung in Schleifen

Kommentare

Normalerweise leitet das Doppelkreuz # in der Bash einen Kommentar ein. Alles, was diesem Zeichen folgt, wird von der Bash ignoriert bis zum nächsten Zeilenumbruch. Dies gilt sowohl für die interaktive als auch für die nicht-interaktive Bash.

Eine **Ausnahme** sind Shellskripte, in denen die erste Zeile mit **#!...** beginnt. Für die Bash ist es die Anweisung, die nachfolgenden Zeilen mit dem hinter »#!« angegebenen Interpreter (eine Shell, Perl, awk, ...) auszuführen.

Die Bedeutung von # kann entweder durch Quoten (siehe später) oder durch die Shelloption "interactive_comments" aufgehoben werden.

```
user@sonne> cat \ # script
#!bin/sh
echo "sinnlos"
user@sonne> # script
user@sonne>
# Wirkung des Doppelkreuzes aufheben:
```

```
user@sonne> shopt -u interactive_comments
user@sonne> # script
sinnlos
# Wirkung des Doppelkreuzes aktivieren:
user@sonne> shopt -s interactive_comments
```

Variablen

Der **Bezeichner** einer Variable ist ein Name (vergleiche Definitionen). Die Anzahl Zeichen, die für einen Bezeichner signifikant sind, ist von der konkreten Bash-Version abhängig. Sie können aber davon ausgehen, dass Sie beim Eingeben eines langen Bezeichners eher ermüden, als dass die Bash kapituliert.

Im Sinne der Syntax werden **skalare** und **Feldvariablen** unterschieden. Zum Erzeugen von Variablen existieren mehrere Möglichkeiten. Die einfachste zum Anlegen einer skalaren Variable ist:

```
user@sonne> variablen_name=wert
```

Eine Feldvariable (Array) erzeugen Sie mittels

```
user@sonne> feld_var_name[index]=wert
# oder
user@sonne> feld_var_name=(wert1 wert2 wert3)
```

Den **Inhalt** einer Variable betrachten Sie mit Hilfe des Kommandos **echo**:

```
user@sonne> echo $variablen_name
wert
user@sonne> echo $feld_var_name
wert1
user@sonne> echo $feld_var_name[2]
wert1[2]
user@sonne> echo ${feld_var_name[2]}
wert3
```

Bemerkung: Wenn Sie auf den Inhalt einer Feldvariable zugreifen, müssen Sie durch Klammerung klar stellen, dass Sie den Inhalt eines konkreten Elementes meinen. Ohne explizites Setzen der Klammern wird als Index implizit »0« angenommen; daraus resultiert die Ausgabe von »echo $feld_var_name[2]«.

Eine Variable können Sie mit einem A t t r i b u t versehen. Hierzu verwenden Sie **declare** oder **typeset**. Beide Kommandos besitzen dieselben Optionen und Wirkungen - wir beschränken uns daher auf ersteres. **Vorsicht:** Entgegen aller Logik setzt »-« ein Attribut und »+ « löscht es!

-a

Setzt das Feldattribut einer Variable (wird einer Variable ein Feld zugewiesen, wird das Attribut automatisch gesetzt)

[-/ +] f

Zeigt eine Funktionsdefinition an/zeigt sie nicht an:

```
user@sonne> first_func() { echo "erste Funktion";}
user@sonne> first_func
erste Funktion
user@sonne> declare -f first_func
first_func()
{
echo "erste Funktion"
}
```

[-/+] i

Setzt/Löscht das Integer-Attribut. Für so deklarierte Variablen ist eine einfachere Syntax für Arithmetik erlau

```
user@sonne> declare -i a=3
user@sonne> b=3
user@sonne> a=a+b
user@sonne> b=a+b
user@sonne> echo $a "  " $b
6  a+b
```

-p

Zeigt die Attribute und den Inhalt einer Variablen an.

```
user@sonne> declare -p a
declare -i a="6"
user@sonne> declare -p SHELL
declare -x SHELL="/bin/bash"
```

[-/+] r

Setzt das Leseattribut/verwirrt den Leser. Ist es aktiv, kann der Wert einer Variablen nicht verändert werde Die Variable kann weder gelöscht, noch kann ein Lese-Attribut entfernt werden:

```
user@sonne> declare -r a
user@sonne> a=nix
bash: a: readonly variable
user@sonne> declare +r a
bash: declare: a: readonly variable
```

[-/+] x

Setzt/Löscht das export-Attribut. Eine exportierte Variable ist auch in den Nachfahren des die Shell ausführenden Prozesses sichtbar (Siehe unter Shellvariablen).

Eine Variable kann auch gelöscht werden. Hierzu dient das builtin-Kommando **unset**:

```
user@sonne> unset a
bash: unset: a: cannot unset: readonly variable
user@sonne> unset b
user@sonne> declare -p b
bash: declare: b: not found
```

Expansion vor der Wertzuweisung an eine Variable

Bevor ein Wert einer Variablen zugewiesen wird, versucht die Bash diesen Wert nach bestimmten Regeln zu substituieren. Dabei durchläuft die Bash folgende Schritte in beschriebener Reihenfolge:

1. Tildeexpansion
2. Parameter- und Variablenexpansion
3. Kommandosubstitution
4. Arithmetische Substitution
5. Entfernen der »Quoting«-Zeichen

Erst jetzt erfolgt die tatsächliche Zuweisung an die Variable.

Was sich hinter den einzelnen Expansionen verbirgt, soll im Anschluss an diesen Abschnitt betrachtet werden.

Einfache Kommandos

Der Begriff des »einfachen Kommandos« definiert einen wichtigen Aspekt der Semantik der Bash. Ein »einfaches Kommando« ist ein Ausschnitt der Kommandozeile, den die Bash in einem Zusammenhang betrachtet. Dazu parst die Shell zunächst die Eingabe und extrahiert alle enthaltenen »einfachen Kommandos«, für die sie nachfolgend definierte Substitutionen vollzieht.

Ein einfaches Kommando wird stets durch einen Kontrolloperator abgeschlossen (Definitionen). Es besteht aus einer Sequenz optionaler Variablenzuweisungen, von durch Whitespaces getrennten Worten und Ein-/Ausgabe-Umleitungen. Das erste Wort, das keine Variablenzuweisung ist, wertet die Bash als Kommandoname, alle weiteren Worte sind die Argumente des Kommandos.

Anmerkung: Man kann sich als einfaches Kommando den Abschnitt der Kommandozeile vorstellen, der innerhalb ein und desselben Prozesses ausgeführt wird.

Die einzelnen Substitutionsschritte, die die Bash für jedes einfache Kommando vornimmt, sind:

1. Variablenzuweisungen und Ein-/Ausgabe-Umleitungen werden markiert (damit sie nicht im folgenden Schritt betrachtet werden)
2. Alle nicht-markierten Worte werden expandiert (siehe nachfolgend)
3. Ein- und Ausgabe-Umleitungen werden vorbereitet (indem die entsprechenden Dateideskriptoren geschlossen/geöffnet werden)
4. Die Variablenzuweisungen werden vorgenommen (inklusive vorheriger Expansionen)

Expansion von Worten

Jede durch Metazeichen getrennte Zeichenkette bildet ein einzelnes Wort in der Eingabe und wird von der Bash nach bestimmten Regeln expandiert. Die Reihenfolge der Auswertung ist:

1. Klammernexpansion
2. Tildeexpansion
3. Parameter- und Variablenexpansion
4. Kommandosubstitution
5. Arithmetische Substitution
6. Wortzerlegung
7. Pfadnamensexpansion

Was sich hinter den einzelnen Expansionen verbirgt, soll im Anschluss an diesen Abschnitt betrachtet werden.

Rückgabestatus eines Kommandos

Mit Beendigung eines Kommandos liefert dieses einen Statuswert an seinen Elternprozess. Dieser Wert kann eine Zahl zwischen 0 und 255 beinhalten und gibt Auskunft über den Erfolg (Rückgabewert »0«) und Misserfolg (Rückgabewert »> 0«) der Ausführung des Kommandos. Viele Kommandos kodieren im Wert die vermutliche Ursache des Scheiterns, wobei Werte zwischen 0 und 127 empfohlen sind. Wird ein Kommando »von außen« beendet (kill), so ist der Rückgabewert die Signalnummer + 128. Die Shell speichert den Rückgabestatus des letzten Kommandos in der speziellen Variable **$?**.

```
user@sonne> grep root / etc/ passwd
root:x:0:0:root:/root:/bin/bash
user@sonne> echo $?
0
user@sonne> grep bla / etc/ passwd
user@sonne> echo $?
1
user@sonne> grep root / ETC/ passwd
grep: /ETC/passwd: Datei oder Verzeichnis nicht gefunden
user@sonne> echo $?
2
```

Pipelining

In einer Pipeline werden mehrere Kommandos mittels des Pipesymbols **|** miteinander verknüpft. Jedes Kommando wird hierbei in einer eigenen Prozessumgebung ausgeführt. Das Kommando links eines Pipesymbols schreibt seine Ergebnisse nachfolgend in die Pipe (»Röhre«) anstatt auf die Standardausgabe. Ein rechts des Symbols stehendes Kommando bezieht seine Eingabedaten aus der Pipe.

Die Pipeline ist somit eine wichtige Unterstützung des Unix-Werkzeugkasten-Prinzips durch die Bash, womit durch geschickte Kombination der scheinbar simplen Kommandos komplexe Aufgaben lösbar werden.

Die vermutlich verbreitetste Anwendung von Pipelines ist die Umleitung von Ausgaben eines Kommandos in einen Pager, um diese bequem seitenweise betrachten zu können:

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

Ohne den Pager hätte man in obigem Beispiel keine Möglichkeit, die ca. 1500 Ausgabezeilen von ls am Bildschirm zu verfolgen.

Eine komplexere Anwendung ist die nächste Verknüpfung zweier tar-Befehle, womit ein ganzes Verzeichnis bei Beibehaltung der Eigentümer kopiert wird:

```
user@sonne> tar cf - ./ bla | (cd / zielpfad; tar xf -)
```

Listen

In Erweiterung des Pipeline-Konzepts lassen sich mehrere Kommandos auf weitere Arten in einer Kommandozeileneingabe angeben. Im Sprachgebrauch der Bash werden diese Varianten als Listen zusammengefasst:

Kommando_1 ; Kommando_2

Führt die Kommandos nacheinander aus. *Kommandos, die unabhängig voneinander arbeiten*, lassen sich so bequem durch Semikola getrennt angeben. Das jeweils nächste Kommando wird unverzüglich gestartet, sobald das vorherige terminierte.

Kommando_1 && Kommando_2
Kommando_1 | | Kommando_2

Führt Kommando_2 nur aus, wenn Kommando_1 erfolgreich (&&) bzw. nicht erfolgreich (||) war.

Insbesondere in Shellskripten wünscht man oft die **bedingte Ausführung**, d.h. ein Kommando soll in Abhängigkeit vom (Miss)Erfolg eines anderen Kommandos gestartet werden. Als Beispiel soll eine Datei mit Kommando rm gelöscht werden. Damit rm allerdings keine Fehlermeldung erzeugt, soll es nur ausgeführt werden, wenn wir die Berechtigung dafür besitzen und die Datei auch existiert. Eine Erfolgsmeldung kann auch nicht schaden... Der Aufruf könnte wie folgt formuliert werden:

```
user@sonne> test -w bla && rm bla && echo "Datei entfernt"
```

Vielleicht sollte die Datei »bla« zur Demonstration des Beispiels angelegt werden? Allerdings nur, wenn sie auch nicht existiert:

```
user@sonne> test -e bla | | touch bla
user@sonne> test -w bla && rm bla && echo "Datei entfernt"
Datei entfernt
```

Kommando &

{ Kommando;}

Startet das Kommando innerhalb der aktuellen Shell. Sinn dieses Konstrukts ist die Gruppierung mehrerer Kommandos, sodass sie eine gemeinsame Ausgabe erzeugen:

```
user@sonne> ls -l | head -1; date > bla
insgesamt 763
user@sonne> cat bla
Die Okt 17 18:40:40 CEST 2000
user@sonne> { ls -l | head -1; date;} > bla
user@sonne> cat bla
insgesamt 763
Die Okt 17 18:40:40 CEST 2000
```

Erläuterung: Die erste Kommandozeile leitet nur die Ausgabe von date in die Datei »bla« um, deshalb lan die Ausgabe von »ls -l | head -1« auf der Standardausgabe. Die zweite Zeile gruppiert beide Kommandos, sodass die Ausgabe komplett in die Datei umgeleitet wird.

(Kommando)

Startet das Kommandos innerhalb einer neuen Shell.

Die letzte Variante zur Eingabe einer Kommandosequenz betrifft deren **Ausführung in einer Subshell**. D die Bash startet als Kindprozess eine weitere Shell und übergibt dieser die auszuführenden Kommandos. Da Verhalten ähnelt stark der Gruppierung mittels geschweifter Klammern und tatsächlich lassen sich dieselber Wirkungen erzielen:

```
user@sonne> pwd; date > output
/home/user
user@sonne> cat output
Don Jun  8 09:38:23 MEST 2000

user@sonne> (pwd; date) > output
user@sonne> cat output /home/user
Don Jun  8 09:38:27 MEST 2000
```

Aus Effizienzgründen sollte, wann immer es möglich ist, auf die Subshell verzichtet werden (Erzeugen eines Prozesses kostet immer Zeit). Notwendig kann sie allerdings werden, wenn verschiedenen Kommandos auf dieselben lokalen Variablen zugreifen. Durch Start in einer eigenen Shell sind Wechselwirkungen ausgeschlossen.

Der Rückgabewert einer Subshell ist immer »0« und die in ihr bearbeiteten Kommandos haben keinen Einfl auf den Elternprozess. So kann eine Subshell verwendet werden, um ein Kommando in einem anderen Verzeichnis auszuführen, ohne dass das Verzeichnis in der aktuellen Shell gewechselt wird:

```
user@sonne> pwd
/home/user
user@sonne> (cd / usr/ X11R6/ bin; pwd)
/usr/X11R6/bin
user@sonne> pwd
/home/user
```

Ein/Ausgabeumleitung

Mit dem Start der Bash öffnet diese die drei Standarddateien mit den Dateideskriptoren 0 (stdin), 1 (stdout) und 2 (stderr). Die Standardeingabe ist dabei zunächst mit der Tastatur verbunden, Standard- und Standardfehler-Ausgabe schreiben initial auf den Bildschirm.

Mit den Mechanismen der E/A-Umleitung lassen sich alle drei Kanäle (genau genommen jeder beliebige Dateideskriptor) in Dateien umlenken, so dass ein Kommando seine Eingaben aus einer Datei bezieht oder die Ausgaben bzw. Fehler in eine solche gelenkt werden. Die Sonderzeichen zur E/A-Umleitung sind:

[n]< Datei

Umleitung der Standardeingabe. Rein interaktiv arbeitende Kommandos erwarten ihre Eingaben von der Standardeingabe. Bestes Beispiel ist das Kommando mail, das den Text einer Nachricht prinzipiell von der Standardeingabe liest. Mail möchte man allerdings häufig automatisch erzeugen und den Inhalt einer Datei versenden; hier kann die Eingabeumleitung die Lösung bedeuten:

```
user@sonne> mail tux@melmac.all -s Wichtig < Message.txt
```

Im Beispiel liest das Kommando mail tatsächlich von Dateideskriptor 0, wohinter sich normalerweise die Standardeingabe verbirgt. Möglich wird dies, indem die Bash vor der Kommandoausführung die Standardeinga schließt, d.h. den Dateideskriptor 0 frei gibt, und der erste freie Deskriptor (also nun 0) beim anschließenden Öffnen der Datei "Message.txt" an diese vergeben wird.

Links des Umleitungssymbols kann sogar das Ziel der Umleitung fehlen, dann steht der Inhalt der umgeleiteter Datei direkt in der Standardeingabe. Eine »sinnvolle« Anwendung ist mir bislang nicht in die Quere gekommen deswegen soll die obskure Syntax eines Kopiervorgangs die Idee verdeutlichen:

```
user@sonne> < foo cat > bla
```

[n]> Datei

Umleitung der Ausgabe in eine Datei. Existiert die Zieldatei nicht, wird sie automatisch erzeugt. Ist sie bereits vorhanden, wird sie geöffnet und ihr alter Inhalt komplett verworfen (über Shelloptionen kann das Überschreib verhindert werden; allerdings nicht, wenn anstatt »>« »>|« verwendet wird).

Mit der optionalen Angabe von »n« kann der umzuleitende Deskriptor angegeben werden; ohne Angabe wird implizit »1« angenommen, was die Standardausgabe betrifft:

```
# Umleitung der Ausgabe von "ls -lt" in eine Datei
user@sonne> ls -lt > time_sorted

# Provozieren eines Fehlers
```

```
user@sonne> touch /etc/passwd
touch: /etc/passwd: Keine Berechtigung

# Umleitung des Fehlers
user@sonne> touch /etc/passwd 2> /dev/null

# Die Shelloption "noclobber" verhindert das Überschreiben existierender Dateien:
user@sonne> set -o noclobber
user@sonne> ls -lt > time_sorted
bash: time_sorted: cannot overwrite existing file
# "noclobber" zurüsetzen:
user@sonne> set +o noclobber
```

[n]>> Datei

Umleitung der Standardausgabe, wobei die Ausgaben an die Zieldatei angehangen werden. Die Arbeitsweise is analog zur »normalen« Ausgabeumleitung, außer dass im Falle einer existierenden Datei die neuen Daten an d Ende angehangen werden.

&> Datei bzw. >& Datei

Umleitung der Standard- und Standardfehlerausgabe. Beide Varianten der Angabe sind semantisch äquivalent; Manual zur Bash empfiehlt die erstere Form. Eine nützliche Anwendung dieser Umleitung findet man im Zusammenhang mit Shellskripten, wo man oftmals einzig am Rückgabewert eines Kommandos, nicht aber an dessen Ausgabe interessiert ist. Als Beispiel testet die folgende Kommandofolge, ob der Benutzer "tux" Mitglied Gruppe "fibel" ist:

```
user@sonne> { groups tux | grep fibel;} &> /dev/null && echo "Mitglied"
```

"Here-Document"

Diese Form der Umleitung ermöglicht die Zusammenfassung mehrerer Zeilen als Eingabe für ein Kommando. Normalerweise ist mit Eingabe des Zeilenumbruchs die Eingabe für die Bash beendet, beim so genannten »Her Document« wird eine Eingabe-Ende-Kennung vorab vereinbart und alle nachfolgenden Zeilen bis zum Erschein der Kennung auf einer neuen Zeile werden einem Kommando als Eingabe zugeführt. Die allgemeine Syntax lau

```
Kommando << 'Das ist die Endekennung'
  Hier folgt das eigentliche "Here-Document",
  das sich über beliebig viele Zeilen
  erstrecken kann
Das ist die Endekennung
```

Als Endekennung können Sie jede Zeichenkette verwenden, die im folgenden Dokument nicht **allein** auf einer existiert. Die Endekennung wird dabei weder der Parameter-, der Kommando-, der Pfadnamen-, noch der arithmetischen Expansion unterzogen. Indem Sie die Kennung quoten, beugen Sie auch den anderen Expansio vor.

Wurde die Kennung **nicht gequotet**, werden die Zeilen des »Here-Documents« der Parameter-, der Komman und arithmetischen Expansion unterzogen, sonst wird keinerlei Expansion vorgenommen:

```
user@sonne> cat << bla
>$(ls -C)
>bla
a.out   bla   foo   ...

user@sonne> cat << 'bla'
>$(ls -C)
>bla
$(ls -C)
```

Sie werden sich fragen, wo man eine solche Anwendung sinnvoll anwenden könnte? Beispiele gibt es genügen bspw. um aus einem Shellskript heraus eine Datei anzulegen.

Vielleicht haben Sie bereits Software eigenständig kompiliert? Dann sind Ihnen sicher die »configure«-Aufrufe bekannt. Diese testen die Umgebung des Systems. U.a. stellen sie fest, welche Werkzeuge vorhanden sind und generieren daraus die späteren Makefiles. Das nachfolgende Skript testet, ob es sich bei dem installierten Kom »cc« um einen GNU-Compiler handelt. Es legt hierzu eine Datei mit einem »Define« an, das nur ein GNU-Comp kennt. Anschließend wird der Präprozessor des Kompilers gestartet und in dessen Ausgabe gesucht, ob der dur das »Define« geschachtelte Ausdruck übernommen wurde:

```
#/bin/sh

cat > out.c << 'ENDE DER TESTDATEI'
#ifdef __GNUC__
  yes
#endif
ENDE DER TESTDATEI

{ cc -E out.c | grep yes; } &> /dev/null && echo "Gnu-Compiler"

rm out.c
```

Duplizieren von Dateideskriptoren

Mit den bisherigen Mitteln war es nur möglich, zu einem Zeitpunkt aus einer Quelle in eine geöffnete Datei zu schreiben bzw. aus dieser zu lesen. D.h. ein Dateideskriptor war exklusiv nutzbar. Indem ein solcher Deskriptor verdoppelt wird, lässt sich diese Beschränkung aufheben.

Die Eingabe lässt sich mit folgendem Konstrukt duplizieren:

```
[n]<&Deskriptor
```

Ohne Angabe von »n« ist die Standardeingabe gemeint, sonst kann »n« jeder zum Lesen geöffnete Deskriptor sein.

Analog dazu verdoppelt folgende Angabe einen Ausgabedeskriptor:

```
[n]>&Deskriptor
```

Mit dem Mechanismus der Ausgabeverdopplung ließe sich die gleichzeitige Umleitung von Standardausgabe un fehler auch wie folgt formulieren:

```
user@sonne> grep root /etc/* > output 2>&1
```

Erläuterung: Der Ausdruck »2>&1« kann wie folgt interpretiert werden: »Sende die Standardfehlerausgabe (dorthin, wohin die Standardausgabe (1) geht.«. Implizit wird eine Kopie des Ausgabedeskriptors erzeugt. Beach Sie, dass die Bash die Umleitungen in der Kommandozeile von links nach rechts bewertet. Vertauschen Sie in obigem Beispiel die beiden Umleitungen:

```
# FALSCH
user@sonne> grep root /etc/* 2>&1 > output
```

so landen die Fehlermeldungen auf dem Bildschirm und nur die normalen Ausgaben in »outfile«. Zum Zeitpunk der Umleitung der Fehler auf die Standardausgabe zeigt diese noch auf die »normale« Ausgabedatei (Terminal Device). Die Fehler werden demzufolge auf eine Kopie des Deskriptors gelenkt, der das Terminal geöffnet hat. nachfolgend wird der Originaldeskriptor auf die Datei »umgebogen«.

Bleibt die Frage nach dem Nutzen derartiger Konstrukte... aber gerade das letzte Beispiel offenbart eine trickre und dennoch sinnvolle Anwendung. Die "normalen" Ausgaben laufen unter Unix i.d.R. gepuffert ab, einzig Fehl finden ohne Verzögerung ihren Weg auf die Konsole. Eine Variante, die Standardausgabe ungepuffert dem Bildschirm zukommen zu lassen, wäre, diese auf Standardfehler umzuleiten. Doch wäre es nun wünschenswert Fehler auf einen anderen Kanal zu dirigieren. Folgendes Beispiel realisiert die notwendigen Umleitungen:

```
# Standardausgaben auf Fehlerausgabe umleiten, Fehler selbst nach /dev/null
user@sonne> find / -name "* bla* " > &2 2> / dev/ null
```

Und wie leitet man die normalen und die Fehlerausgaben eines Kommandos in die Eingabe eines anderen? Mit folgender Notation:

```
# Standardausgaben und Fehlerausgabe in eine Pipe
user@sonne> find / -name "* bla* " 2> &1 | less
```

Um einzig die Fehlerausgaben in eine Pipe zu speisen, hilft folgendes Vorgehen:

```
# Standardfehlerausgabe in eine Pipe
user@sonne> find / -name "* bla* " 3> &1 1> &2 2> &3 | less
```

Öffnen von Deskriptoren zum Schreiben und Lesen

```
[n]< > Deskriptor
```

Eine Beispiel zur Anwendung wird im Zusammenhang mit dem eingebauten Kommando exec gegeben.

Bedingte Ausführung

Die auch als **Flusskontrolle** bekannten Mechanismen ermöglichen eine kontrollierte Beeinflussung des Programmablaufs. Die Bash stellt die **if...fi** und **case...esac**-Konstrukte zur Verfügung.

Erstere Form wird meist zur Unterscheidung einiger weniger Fälle (meist 2) verwendet. Die Syntax lautet:

```
if Liste von Kommandos; then
   Liste von Kommandos
[elif Liste von Kommandos; then
   Liste von Kommandos]
[else
   Liste von Kommandos]
fi
```

Von den angegebenen Zweigen werden die Kommandos höchstens eines Zweiges ausgeführt. Entweder die des ersten Zweiges, dessen Bedingung erfüllt ist oder der optionale "else"-Zweig, falls keine Bedingung erfüllt wurde. Die Bedingung selbst ist der Rückgabewert der Liste der Kommandos (meist also der Rückgabewert des letzten Kommandos der Liste).

Das "case"-Konstrukt wird bei einer höheren Anzahl an Auswahlkriterien bevorzugt. Prinzipiell kann mittels "case" jede "if"-Bedingung abgebildet werden. Ein wesentlicher Unterschied ist die mögliche Abarbeitung mehrerer Fälle, da **alle** Anweisungen ab der ersten zutreffenden Bedingung bis zu einem expliziten Verlassen des Konstrukts ausgeführt werden (d.h. ist eine Bedingung erfüllt, werden die nachfolgenden ignoriert).

```
case Bedingung in
   Muster [ | Muster ] )
      Liste von Kommandos
      [;;]
  [Muster [ | Muster ] )
      Liste von Kommandos
      [;;]]
esac
```

Die *Bedingung* muss ein Token sein. Die Muster unterliegen denselben Expansionen wie Pfadnamen und dürfen somit Metazeichen enthalten. Stimmt ein Muster mit der Bedingung überein, werden alle nachfolgenden Kommandos bis zum Verlassen des Konstrukts mittels ";;" oder bis zum abschließenden "esac" ausgeführt.

Der typische Anwendungsbereich für "if"- und "case"-Konstrukte ist die Shellprogrammierung und in diesem

Zusammenhang werden Ihnen noch genügend Beispiele zur Benutzung begegnen.

```
user@sonne> if test $( id | awk -F'[=(]' '{print $2}') -eq "0"; then echo Superuser; else echo Normaler User; fi
Normaler User
user@sonne> su -
Password:
root@sonne> if test $( id | awk -F'[=(]' '{print $2}') -eq "0"; then echo Superuser; else echo Normaler User; fi
Superuser
```

Das (zugegeben... etwas konstruierte) Beispiel entscheidet, ob der aufrufende Benutzer als Root oder als "normaler" Nutzer arbeitet. Die Verwendung des builtin-Kommandos **test** ist typisch für Bedingungen, wir werden es später im Kontext der eingebauten Kommandos ausführlich kennen lernen. Details zu awk finden Sie im gleichnamigen Abschnitt.

Schleifen

Die Bash bietet **vier Typen** der wiederholten Kommandoausführung. Die "for"-Schleife wird verwendet, um eine Kommandosequenz n-mal auszuführen, wobei die Anzahl vorab fest steht. Im Unterschied dazu wiederholt die "while"-Schleife die Liste der Kommandos nur so oft, solange die angegebene Bedingung erfüllt ist. "until"-Schleifen sind genau genommen nichts anderes als "while"-Schleifen, wobei die Bedingung negiert wurde, d.h. sie wird solange durchlaufen, bis die Bedingung wahr wird. Schließlich helfen die "select"-Schleifen beim Erzeugen von Auswahlmenüs für Benutzereingaben.

Innerhalb jeder Schleife kann diese durch den Aufruf von **break** verlassen werden. Ein enthaltenes **continue** veranlasst das Überspringen nachfolgender Befehle und Fortfahren mit dem nächsten Schleifendurchlauf.

```
for Variable [ in Wortliste ]; do
   Liste von Kommandos
done
```

Die **for-Schleife** wird genau so oft durchlaufen, wie Einträge in der *Wortliste* stehen. Im ersten Durchlauf wird der erste Eintrag der Wortliste an *Variable* zugewiesen, im zweiten der zweite Eintrag usw.:

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

Die *Wortliste* wird zunächst expandiert (dieselben Mechanismen wie bei einfachen Kommandos). Fehlt die *Wortliste*, so wird die Liste der Positionsparameter verwendet (innerhalb von Shellskripten ist dies die Liste der Kommandozeilenargumente; auf der Kommandozeile selbst ist es eine vom Kommando **set** erzeugte Liste):

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

Mit Bash-Version 2.0.4 wurde die **for--**Schleife um eine an die Programmiersprache C angelehnte Syntaxvariante erweitert:

```
for ((Ausdruck_1; Ausdruck_2; Ausdruck_3)); do ...done
```

Bei den *Ausdrücken* handelt es sich um arithmische Substitutionen; *Ausdruck_1* wird üblicherweise die Zuweisung eines Anfangswertes an eine Schleifenvariable beinhalten; *Ausdruck_2* dient als Abbruchbedingung und *Ausdruck_3* zum Weiterschalten der Schleifenvariablen. *Ausdruck_1* wird nur einmal beim Eintritt in die Schleife ausgewertet; *Ausdruck_2* wird **vor jedem** und *Ausdruck_3* **nach jedem** Schleifendurchlauf neu berechnet.

```
user@sonne> for ((i=0; i<10; i++)); do echo -n "$i "; done
0 1 2 3 4 5 6 7 8 9
```

Den Einsatz dieser Form der for-Schleife sollten Sie nur in Betracht ziehen, wenn das Skript nicht portable sein muss oder sicher gestellt ist, dass auf jeder Zielmaschine die Bash in der aktuellsten Version installiert ist.

Beispiel 1: Beim Übersetzen von Softwarepaketen bricht der Vorgang häufig mit einer Fehlermeldung wie *"_itoa.o (.text+0x50): undefined reference to `__umoddi3`"* ab. Ursache ist die Verwendung von "__umoddi3". Vermutlich wurde vergessen, eine bestimmte Bibliothek, die das Symbol enthält, hinzuzulinken. Nun gilt es herauszufinden, welche Bibliothek notwendig ist. Die folgende Kommandozeile findet diese, sofern sie existiert:

```
user@sonne> for i in $(find / usr -name "*.a" 2>/dev/null); do nm $i 2>/dev/null | grep -sq "T __umoddi3" && echo $i; done
/usr/lib/gcc-lib/i486-linux/egcs-2.91.66/libgcc.a
/usr/lib/gcc-lib/i486-linux/2.7.2.3/libgcc.a
/usr/i486-glibc20-linux/lib/gcc-lib/i486-glibc20- linux/egcs-2.91.66/libgcc.a
```

Erläuterung: Das Beispiel beschränkt die Suche auf statische Bibliotheken unterhalb von "/usr". **nm** liest aus jeder Bibliothek die enthaltenen Symbole ("nm" vermag auch die Symbole von Object-Dateien lesen), mit grep suchen wir nach dem Symbol. Uns interessieren allerdings nur Bibliotheken, wo das Symbol definiert ist (und nicht solche, die es nur verwenden), deshalb die Suche nach "T __umoddi3" (Rufen Sie mal "nm /usr/lib/libc.a" auf und durchforsten die Ausgabe, um den Zweck zu verstehen).

Beispiel 2: Der Umsteiger von DOS mag die Möglichkeit vermissen, mehrere Dateien mit einer bestimmten Dateikennung zu kopieren und die Dateikennung gleichzeitig zu verändern ("copy *.bat *.bak"). Hat der Leser die Mechanismen einer Unix-Shell verstanden, sollte ihm geläufig sein, warum ein solcher Aufruf unter Unix nicht das erwartete Resultat erzielt. Mit Hilfe einer "for-Schleife" lässt sich das COMMAND.COM-Verhalten einfach simulieren:

```
user@sonne> for i in $(ls *.bat); do cp $i ${i%.*}.bak; done
```

Erläuterung: Die Generierung des Ziel-Dateinamens enthält eine Parametersubstitution.

Die **while-Schleife** evaluiert die ihr unmittelbar folgenden Kommandos (zwischen "while" und "; do") und führt die Kommandos des Schleifenkörpers solange aus, wie der Rückgabestatus des letzten Kommandos der Liste gleich Null ist. Die **until-Schleife** quittiert dagegen ihren Dienst, sobald das letzte Kommando der Liste den Wert Null zurück liefert:

```
while Bedingung do
   Liste von Kommandos
done

until Bedingung do
   Liste von Kommandos
done
```

Beispiel 1: Die nachfolgende Schleife berechnet die Werte 2^n für n=1..8:

```
user@sonne> declare -i i=1; z=2; while [ $i -le 8 ]; do echo "2^$i=$z"; i=i+1; z=$((2*$z)); done
2^1=2
2^2=4
2^3=8
2^4=16
2^5=32
2^6=64
2^7=128
2^8=256
```

Beispiel 2: Die folgende while-Schleife zählt, wie viele Benutzer die Gruppe "100" als default-Gruppe besitzen:

```
user@sonne> (exec < /etc/passwd; IFS=":"; declare -i users=0; while read line; do set $line; test $4 -eq "100" && users=users+1 ; done; echo "Gruppe users hat $users Default-Mitglieder")
```

Erläuterung: Da die Standardeingabe mittels "exec" auf die Datei /etc/passwd umgeleitet wurde, ist die letzte Eingabe, die aus dieser Quelle kommt, das EndOfFile (EOF). Die Bash unter Linux ist allerdings (meist) so konfiguriert, dass EOF auch zum Beenden der Shell führt ([Ctrl][D]). Um die aktuelle Shell nicht unbeabsichtigt ins Nirwana zu delegieren, wurde die Ausführung in einer Subshell vorgenommen. Der Internal Field Separator (IFS) beinhaltet die Trennzeichen, anhand derer die Bash die Eingaben in einzelne Worte zerlegt. Normalerweise sind diese die Whitespaces; wir benötigen aber den Doppelpunkt, da jener die Felder der Passwortdatei aufteilt. Innerhalb der "while-Schleife" lesen wir jeweils eine Zeile in "line" ein (mittels read) und erzeugen eine Parameterliste (set $line). In dieser interessiert der vierte Parameter (Gruppeneintrag). Ist dieser "100", wird die Variable "users" hoch gezählt.

Näheres zu "IFS", "read" und "set" finden Sie im Abschnitt Initialisierungen.

Die Komplexität obiger Anwendungen von "while" zeigt auf, dass die Verwendung von Schleifen weniger auf der Kommandozeile verbreitet ist, sondern zu wichtigen Bestandteilen in Shellskripten zählt.

select hat mit dem klassischen Schleifenkonzept nichts gemein; sie schreibt in einer Endlosschleife eine Liste von Auswahlalternativen auf die **Standardfehlerausgabe** und wartet auf eine Interaktion mit dem Benutzer.

```
select Name [ in Liste ] do
   Liste von Kommandos
done
```

Das Prinzip lässt sich am einfachsten anhand eines Beispiels demonstrieren. In einem Menü sollen die Dateien des aktuellen Verzeichnisses präsentiert werden. Zur vom Benutzer ausgewählten Datei werden weitere Informationen ausgegeben; außerdem soll die Beendigung der Schleife angeboten werden:

```
user@sonne> select file in * Ende; do
> if test -e $file; then
> ls -l $file;
> else
> break;
> fi;
>done
1) ./index.html      5) ./help.txt       9) ./symbol9.html  13) ./pfad.gif
2) ./symbol6.html    6) ./symbol5.html  10) ./symbol2.html  14) Ende
3) ./foo.tgz         7) ./symbol3.html  11) ./symbol4.html
4) ./symbol8.html    8) ./symbol7.html  12) ./symbol1.html
#? 3
-rw-r--r--   1   user   users       3102 Aug 17 10:49 foo.tgz
1) ./index.html      5) ./help.txt       9) ./symbol9.html  13) ./pfad.gif
2) ./symbol6.html    6) ./symbol5.html  10) ./symbol2.html  14) Ende
3) ./foo.tgz         7) ./symbol3.html  11) ./symbol4.html
4) ./symbol8.html    8) ./symbol7.html  12) ./symbol1.html
#? 14
user@sonne>
```

Die *Liste* in "select" wird allen Expansionen (wie bei einfachen Kommandos) unterzogen. Diese Liste wird allerdings nur beim ersten Eintritt in die Schleife generiert. Eine Änderung dieser wird also im Menü nicht sichtbar.

Verschachtelte Schleifen: Angenommen, das letzte Beispiel sollte dahin gehend modifiziert werden, dass die erwählte Datei gelöscht werden soll. Das Problem ist nun, dass diese Datei im nachfolgenden Menüaufrufs noch immer gelistet wird. Eine Lösung wäre die Kapselung des select-Aufrufs in einer umgebenden "while-Schleife". Die "select"-Auswahl wird nun nach jedem Durchlauf verlassen, sodass das Menü erneut aufgebaut wird. Um nun beide Schleifen zu verlassen, ist die zu verlassenden **Schleifentiefe** dem "break"-Aufruf mitzugeben:

```
user@sonne> while :; do
> select file in * Ende; do
> if test -e $file; then
```

```
> rm $file;
> break;
> else
> break 2;
> fi;
> done
> done
1) ./index.html      5) ./help.txt       9) ./symbol9.html  13) ./pfad.gif
2) ./symbol6.html    6) ./symbol5.html  10) ./symbol2.html  14) Ende
3) ./foo.tgz         7) ./symbol3.html  11) ./symbol4.html
4) ./symbol8.html    8) ./symbol7.html  12) ./symbol1.html
#? 3
1) ./index.html      5) ./symbol5.html   9) ./symbol2.html  13) Ende
2) ./symbol6.html    6) ./symbol3.html  10) ./symbol4.html
3) ./symbol8.html    7) ./symbol7.html  11) ./symbol1.html
4) ./symbol8.html    8) ./symbol9.html  12) ./pfad.gif
#? 13
user@sonne>
```

Expansionen der Bash

Im Abschnitt zur Syntax der Bash tauchte mehrfach der Begriff der Expansion auf. Im Sinne der Bash umfasst Expansion eine Menge von Regeln, nach denen die Eingabe in der Kommandozeile zunächst bearbeitet wird. Bestimmte Zeichenmuster werden hierbei durch andere substituiert - sie "expandieren".

Welche Regeln wann und in welcher Reihenfolge zum Einsatz gelangen, hängt vom konkreten Kontext ab und wurde im Zusammenhang mit einfachen Kommandos (Expansion der Worte) und Variablenzuweisungen genannt. An dieser Stelle soll nun die fehlende Beschreibung der einzelnen Expansionsmechanismen nachgeholt werden, wobei die Reihenfolge der Darstellung einzig der alphabetischen Anordnung entspricht!

- Arithmetische Expansion
- Klammernexpansion
- Kommandosubstitution
- Parameter- und Variablenexpansion
- Pfadnamensexpansion
- Prozesssubstitution
- Quoting
- Tildeexpansion
- Wortzerlegung

Arithmetik

Die Bash ist kein Taschenrechner. Dennoch besitzt sie ein erstaunliches Potenzial an eingebauten Rechenoperationen, die -- nach Prioritäten geordnet -- nachfolgende Tabelle zusammenfasst:

+ -

Einstelliger Operator (Vorzeichen)

! ~

Logische und bitweise Negation

*** ***

Exponentialfunktion

*** / %**

Multiplikation, Division und Modulo-Operator

+ -

Addition, Subtraktion

< < > >

Bitweise Links-/Rechtsverschiebung

< = > = < >

Vergleiche

= = !=

Gleichheit und Ungleichheit

&

Bitweises UND

^

Bitweises Exclusive ODER

|

Bitweises ODER

&&

Logisches UND

| |

Logisches ODER

expr ? expr : expr

Bedingte Zuweisung

= , * = , / = , % = , + = , -= < < = , > > = , &= , ^ = , | =

Zuweisungen

Als Operanden sind Konstanten und Shellvariablen (deren Inhalt als long integer betrachtet wird) erlaubt. Beginnt eine Konstante mit "0", dann wird sie als oktale Zahl verstanden; steht "0x" am Anfang, handelt es sich um eine hexadezimale Konstante.

Konstanten können zu jeder Basis zwischen 2 und 64 angegeben werden, so kann die Zahl 63 u.a. wie folgt dargestellt werden:

- Zur Basis 10: 10#63
- Zur Basis 8: 8#77
- Zur Basis 16: 16#3f

Die so genannte **arithmetische Substitution** ist der gebräuchliche Weg, um Berechnungen durchzuführen:

- **Bash Versionen < 2:** Der zu berechnende Ausdruck wird in eckigen Klammern geschrieben: **$[...]**
- **Bash ab Version 2:** Der zu berechnende Ausdruck wird in doppelte runde Klammern geschrieben: **$((...))**

(die alte Syntax wird weiterhin unterstützt)

Einige Beispiele sollen die Anwendung verdeutlichen:

```
user@sonne> b=5; b=$((b+1)); echo $b
6
user@sonne> a=$((b+=10)); echo $a
16
user@sonne> echo $((a>b?1:0))
1
user@sonne> echo $((8#17**2))
225
user@sonne> echo $((017**2))
225
user@sonne> echo $((-0x64*3#11%6))
-4
user@sonne> echo $((4<<1))
8
```

Wird als Operand eine Variable benutzt, so wird versucht, deren Inhalt in eine Ganzzahl zu konvertieren. Enthält die Variable keine Zahl, wird der Inhalt zu "0" konvertiert:

```
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
```

Klammernexpansion

Mit Hilfe der **Klammererweiterung** lassen sich beliebige Zeichenketten generieren. Im einfachsten Fall verwendet man eine Präfix-Zeichenkette, gefolgt von beliebig vielen, mit geschweiften Klammern umschlossenen und durch Kommas getrennten Zeichen(ketten), wiederum gefolgt von einer optionalen Postfix-Zeichenkette. Das Ergebnis sind nun Zeichenketten der Art "PräfixZeichenkette_1_Postfix", "PräfixZeichenkette_2_Postfix", ..., "PräfixZeichenkette_n_Postfix".

An einem Beispiel lässt sich das Prinzip leicht verdeutlichen:

```
user@sonne> echo Beispiel{_1_,_2_,_3_}
Beispiel_1_ Beispiel_2_ Beispiel_3_
```

Präfix und Postfix können ihrerseits wiederum Klammererweiterungen sein und Klammererweiterungen lassen sich verschachteln, so dass sich z.B. mit nur einem Befehl eine ganze Verzeichnishierarchie erzeugen lässt:

```
user@sonne> mkdir -p bsp/{ucb/{ex,edit},lib/{bla,foo}}
user@sonne> du bsp | cut -b 3-
bsp/ucb/ex
bsp/ucb/edit
bsp/ucb
bsp/lib/bla
bsp/lib/foo
bsp/lib
bsp
```

Kommandosubstitution

Die Kommandosubstitution erlaubt das Ersetzen ihres Aufrufes durch ihre Ausgabe. Es existieren zwei Syntaxvarianten des Aufrufs:

```
$(Kommando)
`Kommando`
```

Die Bash führt das Kommando aus und ersetzt seinen Aufruf auf der Kommandozeile durch dessen Ausgabe, wobei abschließende Zeilenendezeichen entfernt wurden.

```
# ohne Kommandosubstitution user@sonne> find / -name "whatis" 2>/dev/null | ls -l | head -5
insgesamt 15888
-rw-r--r--   1 user   users   787067 Apr  1 09:02 Buch.tar.gz
drwxr-xr-x   4 user   users     4096 Jan 16 19:49 Dhtml
drwx------   5 user   users     4096 Apr 26 09:48 Desktop
drwxr-xr-x   4 user   users     4096 Apr 21 08:43 IGLinux

# mit Kommandosubstitution
user@sonne> ls -l $(find / -name "whatis" 2>/dev/null)
ls -l $(find / -name "whatis"  2>/dev/null)
-rw-r--r--   1 root     root     94414 Jun 13 18:34 /usr/X11R6/man/whatis
-rw-r--r--   1 root     root    792270 Jun 13 18:34 /usr/man/allman/whatis
-rw-r--r--   1 root     root    220874 Jun 13 18:34 /usr/man/whatis
-rw-r--r--   1 root     root         0 Jun 13 18:34 /usr/openwin/man/whatis
```

Eine solche Kommandosubstitution kann auch bei der Zuweisung an eine Variable angewendet werden:

```
user@sonne> AktuellPfad=$(pwd)
user@sonne> echo $AktuellerPfad
/home/user
```

Parameter- und Variablenexpansion

Folgt einem Dollarzeichen **$** ein Variablenname oder eine öffnende geschweifte Klammer **${ ...}** , so spricht man von einer Variablen- bzw. Parameterexpansion. Die geschweiften Klammern dienen zur Gruppierung und sind bei skalaren Variablen, die nicht per Parameterexpansion behandelt werden sollen, nicht notwendig.

Beginnen wir mit einem Beispiel der Expansion einer skalaren Variable ohne Parameterexpansion:

```
user@sonne> var=user
user@sonne> var2=~$var
user@sonne> echo $var2
~user
user@sonne> eval echo $var2
/home/user
```

Bemerkung: Das Beispiel verdeutlicht die Reihenfolge der Auflösung bei Zuweisung eines Wertes an "var2". Im ersten Schritt ist die Tilde nicht auflösbar, deshalb geht sie unverändert in "var2" ein. In einem zweiten Schritt expandiert dann der Inhalt von "var", so dass "var2" nun "~user" beinhaltet. Um den Expansionsmechnismus zu demonstrieren, wurde eine erneute Bewertung von "var2" erzwungen (eval); nun expandiert "~user" zum Heimatverzeichnis "/home/user".

Ist das erste Zeichen eines Parameters das Ausrufezeichen, so handelt es sich um eine **indirekte Expansion**. Die Bash ersetzt den Ausdruck nun nicht mehr durch den Inhalt der Variablen, sondern betrachtet den Inhalt als den Namen einer Variablen, zu deren Inhalt nun expandiert wird. Ein Beispiel erklärt den Sachverhalt wohl deutlicher, als es Worte vermögen:

```
user@sonne> var=user
user@sonne> var2=var
user@sonne> echo $var2
var
user@sonne> echo ${!var2}
user
```

Die weiteren Mechanismen zur Parameterexpansion manipulieren den Inhalt von Variablen. Die Beispiele werden zeigen, dass diese Form der Substitution vor allem für die Shellprogrammierung von immensem Nutzen ist und genau dort werden sie uns wieder begegnen. »parameter« bezeichnet nachfolgend den Variablennamen und »word« steht entweder für eine Zeichenkette oder für eine Variable, die selbst wieder eine Parameter-, Kommando, Tildeexpansion oder eine arithmetische Berechnung beinhalten kann.

${parameter:-word}

Default-Wert setzen: falls »parameter« nicht gesetzt ist, liefert die Expansion »word« zurück, ansonsten den Inhalt von »parameter«:

```
user@sonne> var=5
user@sonne> echo ${var:-test}
5
user@sonne> unset var; echo ${var:-test}
test
user@sonne> unset var; echo ${var:-${test} leer}
leer
```

${parameter:=word}

Default-Wert zuweisen: Das Konstrukt liefert immer den Wert »word« und weist diesen »parameter« zu, falls diese Variable leer war:

```
user@sonne> var=5
user@sonne> echo ${var:=test}
5
user@sonne> echo $var
5
user@sonne> unset var; echo ${var:=test}
test
user@sonne> echo $var
test
```

${parameter:?word}

Inhalt oder Fehlermeldung: Wenn »parameter« gesetzt ist, wird der Inhalt der Variablen geliefert, ansonsten »word« (das meist eine Fehlermitteilung enthalten wird), wobei der »parameter« **nicht** erzeugt und das Skript abgebrochen wird. Ist »word« leer, wird eine Standard-Fehlermeldung generiert:

```
user@sonne> var=5
user@sonne> echo ${var:?}
5
user@sonne> unset var; echo ${var:?}
bash: var: parameter null or not set
user@sonne> echo ${var:?Das Programm $0 kennt diese Variable nicht\ !}
bash: var: Das Programm -bash kennt diese Variable nicht!
```

${parameter:+word}

Alternativer Wert: Falls "parameter" gesetzt ist, wird die Variable gnadenlos mit "word" überschrieben. Ist sie nicht gesetzt, bleibt sie auch weiterhin nicht gesetzt. Rückgabewert ist der Inhalt von "parameter" (also entweder "word" oder nichts):

```
user@sonne> var=5
user@sonne> echo ${var:+Die PID des aktuellen Prozesses ist: $$}
Die PID des aktuellen Prozesses ist: 438
user@sonne> unset var; echo ${var:+$$}
```

${parameter:offset}
${parameter:offset:length}

Teilzeichenkette extrahieren: Beginnend an der durch »offset« angegebenen Position werden maximal »length« Zeichen der Variable »parameter« entnommen. Fehlt die Längenangabe, so werden alle Zeichen ab »offset« bis zum Ende von »parameter« geliefert. Fehlt »offset«, beginnt die Substitution am Anfang der Variable. »offset« und »length« können ganze Zahlen oder arithmetische Berechnungen sein; »offset« kann zu einer negativen Zahl expandieren, dann beginnt die Substitution ausgehend vom Ende der Variable:

```
user@sonne> bsp= "Offset= Anfang und Length= Anzahl"
user@sonne> echo ${ bsp:18}
Length= Anzahl
user@sonne> echo ${ bsp::13}
Offset= Anfang
user@sonne> echo ${ bsp:18:6}
Length
user@sonne> echo ${ bsp:$((-13)):$[12-6]}
Length
```

${ # parameter}

Anzahl Zeichen: Das Konstrukt liefert die Anzahl Zeichen, die in"parameter" enthalten sind:

```
user@sonne> bsp= "Nicht erwähnt wurde die Verwendung von '* ' und '@' als parameter"
user@sonne> echo ${ # bsp}
64
```

${ parameter# word}
${ parameter# # word}

Muster am Anfang: Der Anfang von "parameter" wird mit dem Muster "word" verglichen. Stimmen beide überein, wird **alles nach dem Muster** geliefert, sonst der Inhalt von "parameter". Wird nur ein "#" verwendet, wird die kürzest mögliche Expansion des Musters betrachtet, mit "##" die längst mögliche. Ist "parameter" ein Feld (***** oder @), so wird die Substitution für jedes einzelne Element vorgenommen. "parameter" bleibt immer unverändert:

```
user@sonne> manpath= "/ usr/ man/ man1/ bash.1.gz"
user@sonne> echo ${ manpath# * / }
usr/man/man1/bash.1.gz
user@sonne> echo ${ manpath# # * / }
bash.1.gz
```

${ parameter% word}
${ parameter% % word}

Muster am Ende: Die Substitution arbeitet analog zu der im vorigen Punkt beschriebenen, nur dass nun der Mustervergleich am Ende des Inhalts von "parameter" beginnt. Geliefert wird jeweils die Teilzeichenkette **vor dem Muster**:

```
user@sonne> manpath= "/ usr/ man/ man1/ bash.1.gz"
user@sonne> echo ${ manpath% / * }
usr/man/man1
user@sonne> echo ${ manpath% % .* }
/usr/man/man1/bash
```

${ parameter/ pattern/ string}
${ parameter/ / pattern/ string}

Ersetzen eines Musters: Mit den Substitutionen lassen sich Muster "pattern" im Inhalt einer Variablen "parameter" durch "string" ersetzen. In der ersten Form wird nur ein Auftreten (von vorn), in der zweiten werden alle Auftreten des Musters ersetzt. Beginnt "pattern" mit einem # , so muss das Muster mit dem Anfang von "parameter" übereinstimmen, beginnt es mit "%", muss es mit dem Ende übereinstimmen. Ist "string" leer, wird "pattern" aus "parameter" gelöscht:

```
user@sonne> manpath= "/ usr/ man/ man1/ bash.1.gz"
user@sonne> echo ${ manpath/ man/ }
/usr//man1/bash.1.gz
user@sonne> echo ${ manpath/ / man/ }
/usr//1/bash.1.gz
user@sonne> echo ${ manpath/ # * \ / / }
bash.1.gz
user@sonne> echo ${ manpath/ man/ local/ man}
/usr/local/man/man1/bash.1.gz
```

Pfadnamensexpansion

Bei dem als Pfadnamensubstitution bekannten Mechanismus durchsucht die Bash die Kommandozeile nach den **Metazeichen** *****, **?**, **(** und **[**. Jedes, eines dieser Zeichen enthaltende Token wird als Muster eines Dateinamens interpretiert und durch die alphabetisch sortierte Liste der übereinstimmenden Dateinamen ersetzt. Expandiert ein solches Token nicht, erscheint es unverändert auf der Kommandozeile, falls die Shelloption **nullglob** nicht gesetzt ist. Anderenfalls wird das Token gelöscht.

Die Metazeichen bedeuten im Einzelnen:

*****

Beliebig viele beliebige Zeichen (auch 0)

?

Genau ein beliebiges Zeichen

[...]

Im einfachsten Fall steht es für genau ein Zeichen aus der Menge (bspw. "[aeiou]" für einen Vokal).
Diese Angabe kann negiert werden ("alle außer diese Zeichen"), indem das erste Zeichen nach **[** ein **!** oder ist ("[!abc]" bzw. "[^ abc]").
Anstatt einzelne Zeichen aufzuzählen, lassen sich Bereiche angeben ("[a-z]" meint alle Kleinbuchstaben).
Und letztlich können die Zeichenklassen **alnum alpha ascii blank cntrl digit graph lower print punct space upper xdigit** verwendet werden ("[:digit:]").

***Zeichen*(...)**

Die in den Klammern eingeschlossenen Muster werden nur betrachtet, wenn die Shelloption **extglob** geset ist. Die Muster sind eine Liste von Zeichenketten, die durch **|** getrennt sind. Das Zeichen **vor** der öffnender Klammer reguliert die Auswertung des Musters:

?(Muster-Liste)	Kein oder ein Auftreten eines Musters
*(Muster-Liste)	Kein oder mehrere Auftreten eines Musters
+(Muster-Liste)	Ein oder mehrere Auftreten eines Musters
@(Muster-Liste)	Genau ein Auftreten eines Musters
!(Muster-Liste)	Alle außer den angegebenen Mustern

Prozesssubstitution

Die Ein- bzw. Ausgabe von Prozessen kann mittels der Prozesssubstitution mit einer FIFO-Datei verbunden werden.

Taucht ein Konstrukt der Art **< (Liste)** bzw. **> (Liste)** auf, werden die durch **Liste** benannten Kommandos in einer Subshell gestartet. Gleichzeitig wird die Ausgabe (**> (...)**) bzw. Eingabe (**< (...)**) der Kommandos mit einer automatisch erzeugten FIFO-Datei verbunden. Auf der Kommandozeile erscheint nach erfolgter Substitution der Name der erzeugten FIFO-Datei.

```
user@sonne> ls <(echo "hello")
/dev/fd/63
```

Mit Hilfe der Prozesssubstitution könnte man den vi dazu bewegen, die Ausgaben eines Kommandos zu lesen:

```
user@sonne> vi <(ls /boot/vm*)
/boot/vmlinuz
/boot/vmlinuz.old
~

~
"/dev/fd/63" [fifo/socket] 2L, 32C                    1,1           All
```

Ein weiteres Beispiel dient zur Bestandsaufnahme laufender Prozesse:

```
user@sonne> diff <(ps ax) <(sleep 10; ps ax)
64d63
<  2129 pts/0    S      0:00 /bin/bash
67,68c66
<  2132 pts/0    R      0:00 ps ax
<  2133 pts/0    S      0:00 sleep 10
---
>  2134 pts/1    S      0:00 top
>  2135 pts/0    R      0:00 ps ax
```

Im Beispiel ist der Prozess **top** neu hinzugekommen, dass die Aufrufe der Kommandos **ps** und **sleep** erscheinen, war zu erwarten.

Und abschließend vergleichen wir die Inhalte zweier Archive:

```
user@sonne> diff <(tar tzf Buch1.tar.gz) <(tar tzf Buch.tar.gz)
325a326,328
> Images/tkinfo.gif
> Images/GlobaleVariable.gif
> Images/LokaleVariable.gif
```

Innerhalb der Klammern **>(...)**, **<(...)** können Parameter- Kommando- sowie arithmetische Substitutionen benutzt werden.

Quoting

Das **Quoting** wird verwendet, um die Interpretation von Sonderzeichen durch die Bash zu verhindern. Die Sonderzeichen sind **;**, **&**, **( )**, **{ }**, **[ ]**, **|** , **>** , **<** , **Zeilenumbruch, Tabulator, Leerzeichen, $, ? und * .**

Ein einzelnes Sonderzeichen wird am einfachsten durch einen vorangestellten Backslash "gequotet". Die Bash betrachtet die Kombination aus Backslash und dem nachfolgenden Zeichen als ein einzelnes Zeichen, versucht aber nicht, dieses zu expandieren.

Um mehrere Zeichen vor der Interpretation zu schützen, können diese zwischen zwei Anführungsstrichen (doppelt/ einfach) eingeschlossen werden. Während die einfachen Anführungsstriche die Interpretation aller eingeschlossenen Sonderzeichen verhindern, erlauben doppelte Anführungsstriche die Substitution von Variablen und Kommandos.

Ein kurzes Beispiel demonstriert den Sachverhalt:

```
user@sonne> echo "Das Arbeitsverzeichnis der $SHELL ist $(pwd)"
Das Arbeitsverzeichnis der /bin/bash ist /home/user

user@sonne> echo 'Das Arbeitsverzeichnis der $SHELL ist $(pwd)'
Das Arbeitsverzeichnis der $SHELL ist $(pwd)

user@sonne> echo Das Arbeitsverzeichnis der \ $SHELL ist $(pwd)
Das Arbeitsverzeichnis der $SHELL ist /home/user
```

Tildeexpansion

Beginnt der Wert mit einer ungequoteten Tilde (~), wird versucht, diese zu substituieren. Betrachtet werden alle der Tilde folgenden Zeichen bis zum ersten Schrägstrich (Slash). Ergibt dies eine gültige Benutzerkennung, so expandiert der Ausdruck zum Heimatverzeichnis dieses Benutzers. Folgt der Tilde unmittelbar der Schrägstrich, wird der Ausdruck durch den Inhalt der Variablen HOME ersetzt; ist diese nicht gesetzt, wird das Heimatverzeichnis des aktuellen Benutzers angenommen:

```
user@sonne> var= ~
user@sonne> echo $var
/home/user
user@sonne> var= ~ root/
user@sonne> echo $var
/root/
```

Wortzerlegung

In diesem Schritt wird die Kommandozeile in so genannte Token unterteilt. Welche Zeichen als Separatoren verwendet werden, verrät die Variable $IFS (Internal Field Separator). Ist diese nicht gesetzt, gelten die schon erwähnten Whitespaces als Begrenzer, sofern sie nicht innerhalb von (Doppel) Anführungsstrichen stehen oder durch den Backslash "gequotet" wurden.

Initialisierung

Die Initialisierungsdateien

Handelt es sich bei einer Login-Shell um die Bash oder die Korn Shell, sucht diese als erstes nach der Datei / etc/ profile. Diese wird vom Systemadministrator meist dazu genutzt, allen Nutzern eine auf das System zugeschnittene Umgebung zu gewähren. Häufig enthalten sind:

- Prüfen der Mailbox auf neue Nachrichten
- Ausgabe der "Nachricht des Tages" (Inhalt der Datei / etc/ motd)
- Setzen wichtiger Aliasse
- Vorbelegen wichtiger Variablen

Gerade in Umgebungen, wo die Benutzer sowohl auf die Bash als auch auf die Ksh zurück greifen, sollte die / etc/ profile sorgfältig erstellt werden, da gewisse Unterschiede zu ungewollten Effekten führen können.

Zwei weitere Konfigurationsdateien sollen rein Bash-spezifische Einstellungen ermöglichen. Die Dateien .bash_profile und .bash_login im Heimatverzeichnis eines Nutzers werden, falls sie existieren, in beschriebener Reihenfolge behandelt. Um kompatibel zur kommerziellen Unix-Shell sh zu sein, liest die Bash die beiden Dateien allerdings nicht aus, wenn auf sie unter dem Namen sh (ein Link auf / bin/ bash) zugegriffen wird.

Anschließend wertet die Bash noch eine eventuell vorhandene Datei .profile aus, die, wiederum im Heimatverzeichnis eines Nutzers gelegen, erster Ansatzpunkt für den Nutzer darstellt, persönliche Einstellungen zur Umgebung vorzunehmen.

Im Falle einer interaktiven, Nicht-Login Shell versucht die Bash Befehle aus einer Datei ~ / .bashrc abzuarbeiten. Schließlich hällt eine nicht-interaktive Bash (Shellskripte) nach einer in der Umgebungsvariablen BASH_ENV gespeicherten Datei Ausschau.

In allen erwähnten Dateien können dieselben Befehle stehen (z.B. Kommandos, Alias- und Variablendefinitionen, ...). Für gewöhnlich werden die Benutzer bestenfalls eine existierende Datei nach ihren Bedürfnissen anpassen, anstatt sich das notwendige Wissen anzueignen, um eine Konfigurationsdatei vollständig zu erstellen. Die nachfolgend, reichlich kommentierte Datei kann als Ausgangspunkt für eigene Experimente dienen:

```
user@sonne> cat / etc/ profile
# /etc/profile
#
umask 022

# Abstürzende Programme sollen keine Speicherdumps anlegen
ulimit -Sc 0
# Ein einzelner Benutzer darf maximal 128 Prozesse gleichzeitig starten
```

```
ulimit -u 128
# Kein Limit für das Datensegment eines Prozesses
ulimit -d unlimited
# Die Kornshell kennt die folgende Option nicht (Stackgröße)
test -z "$KSH_VERSION" && ulimit -s unlimited

# Die PATH-Variable wird mit Standardpfaden belegt
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

# PATH für root wird um sbin-Verzeichnisse erweitert
test "$UID" = 0 && PATH=/sbin:/usr/sbin:/usr/local/sbin:$PATH

# PATH wird um Binary-Pfade ergänzt:
for DIR in /usr/lib/java/bin \
           /usr/games/bin \
           /usr/games \
           /opt/bin \
           /opt/gnome/bin \
           /opt/kde/bin; do
    test -d $DIR && PATH=$PATH:$DIR
done
export PATH

# Umgebungsvariablen, die verbreitete Programme verwenden, werden belegt:
if test -n "$TEXINPUTS" ; then
    TEXINPUTS=":$TEXINPUTS:~/.TeX:/usr/doc/.TeX"
else
    TEXINPUTS=":~/.TeX:/usr/doc/.TeX"
fi
export TEXINPUTS

PRINTER='lp'
export PRINTER

LESSCHARSET=latin1
export LESSCHARSET
LESS="-M -S -I"
export LESS
LESSKEY=/etc/lesskey.bin
export LESSKEY
LESSOPEN="|lesspipe.sh %s"
export LESSOPEN

# Suchpfade nach Manuals werden in MANPATH aufgenommen:
MANPATH=/usr/local/man:/usr/share/man:/usr/man:/usr/X11R6/man
for DIR in /usr/openwin/man \
           /usr/share/man/allman \
           /usr/man/allman \
           /usr/man/de ; do
    test -d $DIR && MANPATH=$MANPATH:$DIR
done
export MANPATH

# Suchpfade für die Info-Seiten:
INFOPATH=/usr/local/info:/usr/share/info:/usr/info
export INFOPATH

# Rechnername und Newsserver
HOSTNAME="`hostname`"
export HOSTNAME
NNTPSERVER=`cat /etc/nntpserver 2> /dev/null`
export NNTPSERVER

# Damit die Ausgabe von "ls" farbig ist...
if [ -x /usr/bin/dircolors ] ; then
    if test -f ~/.dir_colors ; then
        eval `dircolors -b ~/.dir_colors`
```

```
    elif test -f /etc/DIR_COLORS ; then
        eval `dircolors -b /etc/DIR_COLORS`
        fi
fi

unalias ls 2>/dev/null

if test "$UID" = 0 ; then
    LS_OPTIONS='-a -N --color=tty -T 0';
else
    LS_OPTIONS='-N --color=tty -T 0';
fi
export LS_OPTIONS

# Einige Aliasdefinitionen (gekürzt)
alias ls='ls $LS_OPTIONS'
alias dir='ls -l'
alias ll='ls -l'
alias o='less'
alias ..='cd ..'
alias ...='cd ../..'
alias rd=rmdir
alias md='mkdir -p'
alias unix2dos='recode lat1..ibmpc'
alias dos2unix='recode ibmpc..lat1'
alias which='type -p'

# Einige Funktionen
function startx { /usr/X11R6/bin/startx $* 2>&1 | tee ~/.X.err ; }
function remount { /bin/mount -o remount,$* ; }
# Das Prompt wird für Root und normale Benutzer unterschiedlich gesetzt:
if [ "$SHELL" = "/usr/bin/pdksh" -o "$SHELL" = "/usr/bin/ksh" -o "$SHELL" = "/bin/ksh" -o "$0" = "ksh" ]; then
    PS1="! $ "
elif [ -n "$ZSH_VERSION" ]; then
    if test "$UID" = 0; then
    PS1='%n@%m:%~ # '
    else
    PS1='%n@%m:%~ > '
    fi
elif [ -n "$BASH_VERSION" ] ; then
    set -p
 if test "$UID" = 0 ; then
        PS1="\h:\w # "
    else
        PS1="\u@\h:\w > "
    fi
else
    PS1='\h:\w \$ '
fi
PS2='> '
export PS1 PS2

unset DIR
```

Abhängig von der konkreten Linuxdistribution werden ggf. aus den beschriebenen Standarddateien heraus weitere Startup-Dateien eingelesen.

Die Shellvariablen

Variablen lassen sich auf verschiedene Art und Weise in konkrete Kategorien einordnen. Im Sinne der Shell beeinflussen sie deren Verhalten. Diese, in dem Zusammenhang als Umgebungsvariablen bezeichneten Variablen unterteilen sich wiederum in Variablen, die die Bash setzt und solche, die sie nur benutzt.

Wichtige Variablen, die die Bash setzt, sind (Auswahl):

PPID

Prozessnummer des Bash-Vorgänger-Prozesses

PWD

Aktuelles Arbeitsverzeichnis.

OLDPWD

Vorhergehendes Arbeitsverzeichnis; bspw. könnten Sie mit "cd -" in dieses wechseln.

REPLY

Variable, die die zuletzt vom Kommando read gelesene Zeile enthält.

BASH

Vollständiger Pfad der aktuellen Bash. In dem Zusammenhang gibt BASH_VERSION die Programmversion preis und BASH_VERSINFO ist ein Feld (5 Einträge) mit genaueren Angaben

RANDOM

Erzeugt eine Zufallszahl bei jedem Zugriff.

OPTARG und OPTIND

Wert und Index des letzten, von getopts bearbeiteten Argument.

HOSTNAME

Rechnername ohne Domain.

HOSTTYPE

Architekturtyp des Rechners

OSTYPE

Typ des Betriebssystems.

SHELLOPTS

Liste der Shelloptionen.

Wichtige Variablen, die die Bash benutzt, sind (Auswahl):

IFS

Der *Internal Field Separator* enthält die Trennzeichen, anhand derer die Bash nach erfolgter Expansion die Eingabe in Worte zerlegt.

PATH

Enthält die Suchpfade, in denen die Bash nach Kommandos sucht. Beachten Sie die Reihenfolg der Suche - bevor die Bash PATH betrachtet, durchsucht sie zunächst die Aliasse, die Funktion und die eingebauten Kommandos nach dem Namen.

HOME

Enthält das Heimverzeichnis des Benutzers.

BASH_ENV

Enthält eine Initialisierungsdatei, die beim Start von Subshells für Skripte aufgerufen wird.

MAIL

Enthält den Namen einer Mailbox; wenn dort neue Nachrichten eintreffen, wird der Benutzer benachrichtigt. Allerdings nur, falls MAILPATH nicht gesetzt ist. Letztere Variable enthält die Pfade zu mehreren Dateien, die überwacht werden sollen.

PS1...PS2

Siehe Prompts

HISTSIZE

Anzahl maximal gespeicherter Kommandos im History-Speicher

HISTFILE

Datei, wo die History bei Ende einer interaktiven Bash gespeichert wird. Wie viele Zeilen diese enthalten kann, steht in HISTFILESIZE.

TMOUT

Ist dieser Wert gesetzt, beendet sich die Bash, wenn innerhalb der angegebenen Zeitspanne (i Sekunden) keine Eingabe erfolgte. Hiermit kann auf einfache Art und Weise ein Auto-Logout realisiert werden.

Variablen besitzen einen Sichtbarkeitsbereich und lassen sich in lokale und globale Variablen unterteilen. Lokale Variablen sind nur in der Shell ihrer Definition sichtbar, d.h. in keinem innerhalb der Shell gestarteten Kommando (Ausnahme: builtin-Kommando, aber dieses "ist" ja die Shell selbst). Plausibler vorzustellen ist der Sachverhalt innerhalb der Prozesshierarchie. Eine lokale Variable ist somit nur im Prozess ihrer Definition sichtbar, nicht jedoch in den daraus abgeleiteten Prozessen. Im Gegensatz hierzu ist eine globale Variable ab dem Prozess ihrer Definition sichtbar; also in allen abgeleiteten, nicht jedoch in den übergeordneten Prozessen.

```
# Definition einer lokalen Variable:
user@sonne> localvar= "eine lokale Variable"
# Lokale Variablen sind nur in der Shell ihrer Definition sichtbar:
user@sonne> bash
user@sonne> echo $localvar

user@sonne> exit
user@sonne> echo $localvar
eine lokale Variable
# Jede Variable kann zu einer globalen Variable erklärt werden:
user@sonne> export $localvar
# Globale Variablen sind ab (aber nicht in einer Vorgängershell) der Shell ihrer Definition sichtbar:
user@sonne> bash
user@sonne> echo $localvar

eine lokale Variable
```

Eine letzte Herangehensweise in die Thematik der Variablen betrachtet deren Eigenschaften, ob sie bspw. änderbar sind oder einen bestimmten Typ besitzen (vergleiche Variablen).

Spezielle Shellvariablen

Neben den soeben vorgestellten Variablen verfügt die Bash über zwei weitere Arten. Im Sprachgebrauch der Bash benennt man diese Positions- und spezielle Parameter. Beiden ist eigen, dass man ihre Werte zwar auslesen kann, aber die unmittelbare Zuweisung derselben nicht möglich

ist.

Die meisten Programme sind über Kommandozeilenoptionen steuerbar, so auch die Bash. Die Positionsparameter dienen nun dazu, auf diese Optionen zuzugreifen. Die Nummerierung erfolgt gemäß der Reihenfolge der Angabe der Parameter (die Token der Kommandozeile); die Zählung beginnt bei 0. Und da i.d.R. als erstes der Kommandoname in der Eingabe erscheint, verbirgt sich dieser hinter der Variable $0:

```
user@sonne> echo $0
/bin/bash
```

Wer sich so seine Gedanken über den Sinn der Positionsparameter macht, wird schnell auf deren Verwendung in Shellskripten schließen. Und genau dort werden wir intensiven Gebrauch davon machen. Wer weiter den Hirnschmalz strapaziert, wird sich fragen, wie man die Anzahl der Positionsparameter ermittelt. Auch diese verrät eine spezielle Variable $# ; der Parameter $0 wird hierbei nicht mit gezählt. Die eigentlichen Positionsparameter verbergen sich in den Variablen $1,..., $9.

Ein kurzes, nicht gerade geistreiches Beispiel soll die Verwendung andeuten; zum Erzeugen einer Parameterliste bemühen wir das eingebaute Kommando set:

```
# Parameterliste mit den Werten 1, 2, ...,9 erzeugen
user@sonne> set `seq 1 9`
user@sonne> echo $#
9
# Summe über die Positionsparameter berechnen
user@sonne> i=1;summe=0
user@sonne> while [ "$i" -le "$# " ]; do
>summe=$(($summe+$i));
>i=$(($i+1))
>done
user@sonne> echo $summe
45
```

Sie verstehen das Beispiel nicht? Dann schauen Sie sich bitte nochmals die Abschnitte zur Kommandosubstitution, zu Schleifen und Berechnungen an.

Der Denker erkennt ein Problem: "Kann ich nur 9 Parameter an ein Shellskript übergeben?" Nein! Aber der Zugriff über die Positionsparameter ist nur auf die ersten 9 Einträge möglich. Benötige ich mehr, hilft das eingebaute Kommando shift weiter, das bei jedem Aufruf die Positionsparameterliste um eine (oder die angegebene Anzahl) Position(en) nach rechts verschiebt, d.h. nach dem Aufruf von "shift" steht in "$1" der Wert, der vor dem Aufruf in "$2" stand; in "$2" steht der aus "$3" usw. Der ursprüngliche Inhalt von "$1" ist ein für allemal verloren, es sei denn, man hat ihn sich in einer Variablen gemerkt. Wir kommen nochmals auf das letzte Beispiel zurück und berechnen nun die Summe der Zahlen 1, 2, ..., 100:

```
user@sonne> set `seq 1 100`
user@sonne> echo $#
100
# Summe über die Positionsparameter berechnen
user@sonne> i=1;summe=0
user@sonne> while [ "$# " -ne "0" ]; do
>summe=$(($summe+$i));
>i=$(($i+1))
>shift
>done
user@sonne> echo $summe
5050
```

Im Zusammenhang mit den Positionsparametern stehen $* und $@, die auf den ersten Blick genau das gleiche Ergebnis erzielen, nämlich die gesamte Parameterliste zu expandieren, wobei die einzelnen Einträge jeweils durch den ersten Feldtrenner aus der Variablen IFS abgegrenzt sind. Der

erste Parameter "$* " liefert dabei genau ein Wort zurück, während der zweite jeden Eintrag als eigenes Wort erzeugt.

Das Ganze klingt mächtig verworren, aber zwei Anwendungsgebiete offenbaren den eigentlichen Nutzen. Zum einen wünscht man, eine Kommandozeile nach einem bestimmten Muster zu scannen. Jeden Parameter in einer Schleife einzeln zu durchleuchten, ist der Pfad des Sisyphus; hingegen den Inhalt von "$* " bzw. "$@" zu durchmustern, das elegante Vorgehen des aufmerksamen Lesers:

```
user@sonne> set -- -o bla -x foo -z mir fällt nix Gescheites ein
# Ist die Option "-o" gesetzt?
# Beachten Sie das Quoten des Suchmusters!
user@sonne> echo $* | grep -q \\\-o && echo "ja!"
ja!
```

Die zweite (mir bekannte) Anwendung ist bei der Speicherung der Positionsparameter in einer Feldvariablen. Somit kann ohne Verwendung von shift auf alle Positionen zugegriffen werden.

```
user@sonne> set `seq 1 20`
user@sonne> feld=($*)
user@sonne> echo ${feld[0]} ${feld[19]}
1 20
```

Da IFS als erstes das Leerzeichen enthielt, »zerfällt« »$* « in einzelne Token und verhält sich analog zu »$@«. Quotet man beide Parameter in doppelte Anführungsstriche, offenbart sich der kleine aber feine Unterschied:

```
user@sonne> set `seq 1 20`
user@sonne> feld=($@)
user@sonne> echo ${feld[0]} ${feld[19]}
1 20
user@sonne> feld=("$@")
user@sonne> echo ${feld[0]}
1
user@sonne> feld=("$* ")
user@sonne> echo ${feld[0]}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

Die speziellen Parameter lassen sich nicht so säuberlich in eine Schublade pressen - zu unterschiedlich sind die ihnen angedachten Aufgaben:

$?

Diese Variable beinhaltet den Rückgabewert des zuletzt beendeten Vordergrundprozesses:

```
user@sonne> touch /ect/passwd > /dev/null
user@sonne> echo $?
1
```

$!

Enthält die Prozessnummer PID des zuletzt im Hintergrund gestarteten (und noch immer aktiv Prozesses:

```
user@sonne> sleep 100&
[1] 1324
user@sonne> echo $!
1324
```

$-

Beinhaltet alle durch das eingebaute Kommando gesetzten Shellvariablen (den symbolischen

Buchstaben, nicht den Variablennamen selbst!):

```
user@sonne> echo $-
himBH
```

Aktiv sind im Beispiel: hashall, interactive, monitor, braceexpand, histexpand.

$_

In Abhängigkeit vom letzten Vorgang steht entweder das letzte Argument des zuletzt ausgeführten Shellskripts (oder der zuletzt gerufenen Shell) drin oder, wenn die Mailboxen au neue Nachrichten überprüft werden, der Name der aktuell betrachteten Datei.

```
user@sonne> bash -c echo a b c
user@sonne> echo $_
c
```

Die Prompts

Arbeitet man interaktiv mit der Bash, zeigt die Shell ihre Bereitschaft zur Entgegennahme von Eingaben anhand eines Prompts ("Eingabeaufforderung") an. Vier Prompts werden unterschieden, wobei zwei die Eingabeaufforderung symbolisieren:

1. **Primäres Prompt: Dieses zeigt an, dass die Bash eine neue Eingabe erwartet.**
2. **Sekundäres Prompt: Wurde die Eingabe eines Kommandos nicht vollendet und dennoch versucht, die Zeile durch "ENTER" abzuschließen, zeigt die Bash dieses "Fortsetzungsprompt". Typische Situationen, die das Prompt hervorzaubern, sind: der Abschluss einer Zeile mit dem Backslash \ , eine vergessene schließende Klammer oder Anführungszeichen:**

```
user@sonne> (echo "Das Zeilenende kam eindeutig [ENTER]
> zu zei\ [Enter]
> tig"[Enter]
> )[Enter]
das Zeilenende kam eindeutig
zu zeitig
user@sonne>
```

3. **Drittes Prompt: (nur bash) Dieses Prompt erscheint bei der interaktiven Arbeit mit dem builtin-Kommando select und ist über die Shellvariable "$PS3" konfigurierbar.**
4. **Viertes Prompt: (nur bash) Im Debug-Modus (set -x) wird dieses Prompt ("$PS4") vor jeder expandierten Kommandozeile ausgegeben. Der Modus ist hilfreich, um zu sehen, wie die Bash eine Eingabe expandiert, bevor sie diese an das Kommando zur Ausführung übergibt:**

```
user@sonne> testvar=text
user@sonne> echo $testvar
text
user@sonne> set -x
user@sonne> echo $testvar
+ echo text
text
```

Das Aussehen der beiden ersten Prompts wird durch den Inhalt der beiden Variablen "$PS1" (primäres Prompt) und "$PS2" (sekundäres Prompt) bestimmt und kann in begrenztem Umfang geändert werden. Zunächst schauen wir uns den üblichen Inhalt der Variablen an:

```
user@sonne> echo $PS1
\u@\h:>
user@sonne> echo $PS2
>
```

"$PS1" enthält offensichtlich Sonderzeichen, die zum Nutzerkennzeichen \u bzw. zum Rechnernamen \h expandieren. Wichtige mögliche Zeichen sind:

\e

Leitet eine Escape-Sequenz ein

\d

Datum

\h

Rechnername bis zum ersten enthaltenen Punkt

\s

Shellname

\t

Aktuelle Zeit

\u

Nutzername

\w

Aktueller Pfadname

\$

"# ", falls UID=0 und "$" sonst

\nnn

Die Zahl *nnn* wird als ASCII-Code interpretiert. So lassen sich auch nichtdruckbare Zeichen und Escape-Sequenzen ("\033" entspricht "\e") verwenden.

Das folgende Beispiel ändert den primären Prompt, so dass Nutzername, aktueller Pfad und das Datum angezeigt werden:

```
user@sonne> PS1="\u:\w:\d> "
user:~:Wed Jun 7>
```

Vielleicht wäre etwas Farbe für das Prompt noch besser? Mit Escape-Sequenzen lässt sich auch dieses realisieren:

```
user@sonne> PS1="\[\033[01;31m\]$PS1\[\033[m\]"
user@sonne>
```

Die Erklärung und weitere Informationen zum Umgang mit Farben für die Konsole findet der interessierte Leser im Anhang Skriptsammlung.

Konfiguration der Bash mit set und shopts

set und shopts sind zwei wichtige eingebaute Kommandos der Bash, mit Hilfe derer sich das Verhalten der Shell auf vielfache Art und Weise konfigurieren lässt. Eigentlich eher in den

Initialisierungsdateien angewendet, lassen sich beide Kommandos auch interaktiv verwenden.

Beginnen wir mit set und betrachten zunächst den Status der von einer typischen Bash gesetzten Variablen:

```
user@sonne> set -o
allexport       off
braceexpand     on
errexit         off
hashall         on
histexpand      on
keyword         off
monitor         on
noclobber       off
noexec          off
noglob          off
notify          off
nounset         off
onecmd          off
physical        off
privileged      on
verbose         off
xtrace          off
history         on
ignoreeof       on
interactive-comments    on
posix           off
emacs           on
vi              off
```

Für jede der in obigem Beispiel aufgeführten Variablen kennt set eine "kurze" Option, um diese zu aktivieren (on) bzw. zu deaktivieren. Aus der Fülle der Optionen genügt die Kenntnis der schon erwähnten Option -o. Ohne Angabe weiterer Argumente werden alle Variablen (die mit set manipuliert werden können) angezeigt. Folgt dieser der (die) Namen einer (mehrerer) Variablen, so wird (werden) diese aktiviert. Zum Deaktivieren dient die Option +o in Verbindung mit einem Variablennamen.

```
# Aktivieren einer Variable mit set -o ...
user@sonne> set -o allexport noclobber
user@sonne> set -o |  egrep 'allexport| noclobber'
allexport       on
noglob          on

# Deaktivieren einer Variable mit set +o ...
user@sonne> set +o allexport
user@sonne> set -o |  egrep 'allexport| noclobber'
allexport       off
noglob          off
```

Viele Variablen deuten ihren Zweck bereits durch ihren Namen an. Die nützlichsten Vertreter möchten wir Ihnen nicht vorenthalten:

allexport

Alle Variablen der Shell werden exportiert sobald sie erstellt oder verändert werden.

braceexpand

Schaltet die Klammernexpansion an bzw. ab.

hashall

Ist diese Option gesetzt, werden Kommandos, die die Bash bereits einmal gesucht hat (PATH), vollständigem Pfad in einer Hashtabelle gespeichert. Dadurch wird ein folgender Aufruf dessel

Kommandos erheblich beschleunigt.

histexpand

Die Möglichkeit des Zugriffs auf zuvor eingegebene Kommandozeilen mittels eines vorangestellten ! wird ein- bzw. ausgeschalten.

monitor

Durch Deaktivieren wird die Statusmeldung eines im Hintergrund gestarteten Prozesses (» Job unterdrückt. In ähnlichem Zusammenhang steht notify, womit der Status eines beendeten Hintergrundprozesses ohne Verzögerung auf dem Bildschirm landet.

noclobber

Wird diese Variable aktiviert, lassen sich existierende Dateien nicht mittels Ein- und Ausgabeumleitung überschreiben:

```
user@sonne> touch testdatei; ls > testdatei
# Alles im grünen Bereich...
user@sonne> set +o noclobber'
user@sonne> ls > testdatei
bash: testdatei: cannot overwrite existing file
```

Wird die Ausgabeumleitung mittels ">| " realisiert, schützt auch "noclobber" nicht vorm Überschreiben.

noexec

Kommandos werden zwar gelesen, aber nicht ausgeführt. Diese Variable wird gern in der Testphase während der Erstellung von Shellskripten gesetzt.

noglob

Die Pfadnamensexpansion kann ein- und ausgeschalten werden.

onecmd

Die Shell wird nach Ausführung des ersten Kommandos beendet. Eine denkbare Anwendung w in Verbindung mit Postfächern, wo ein Benutzer sich "von außen" anmeldet, durch ein Skript e Kommando, das die Mail überträgt, gestartet wird und anschließend der Benutzer automatisch ausgeloggt wird. Damit erhält ein Benutzer niemals vollwertigen Zugang zum Rechner.

posix

Die Bash verhält sich nach IEEE POSIX P1003.2/ ISO 9945.2 Shell und Tools Standard.

emacs, vi

Schaltet das Verhalten des Kommandozeileneditors auf den jeweiligen Modus.

shopt ist ab der Bash-Version 2.0 neu hinzugekommen. Es behandelt weitere Shellvariablen und kennt (leider) abweichende Optionen. So verhilft "-p" zu einem Blick auf den Status der von "shopt" verwalteten Variablen:

```
user@sonne> shopt -p
shopt -u cdable_vars
shopt -u cdspell
shopt -u checkhash
shopt -s checkwinsize
shopt -s cmdhist
```

```
shopt -u dotglob
shopt -u execfail
shopt -s expand_aliases
shopt -u extglob
shopt -u histreedit
shopt -u histappend
shopt -u histverify
shopt -s hostcomplete
shopt -u huponexit
shopt -s interactive_comments
shopt -u lithist
shopt -u mailwarn
shopt -u no_empty_cmd_completion
shopt -u nocaseglob
shopt -u nullglob
shopt -s progcomp
shopt -s promptvars
shopt -u restricted_shell
shopt -u shift_verbose
shopt -s sourcepath
shopt -u xpg_echo
```

Um eine Shellvariable zu aktivieren, ist die Option -s zu verwenden. Dem entgegen schaltet -u die Wirkung der Variablen ab. Wiederum betrachten wir nur eine Auswahl der Variablen im Detail:

cdspell

Ist die Option gesetzt, vermag die Shell geringe Tippfehler in Pfadangaben zu korrigieren:

```
user@sonne> cd / us/ x11R6
bash: cd: /us/x11r6/: Datei oder Verzeichnis nicht gefunden
user@sonne> shopt -s cdspell
user@sonne> cd / us/ x11R6
/usr/X11R6/
```

lithist

Bei der Eingabe von Kommandos, die über mehrere Zeilen gehen, speichert die Bash diese normalerweise ohne die enthaltenen Zeilenumbrüche in der History. Wenn Sie diese Variable a "on" setzen, bleiben diese Zeilenumbrüche auch in der History enthalten.

dotglob

Bei der Pfadnamensexpansion werden Dateien, deren Namen mit einem Punkt beginnen, nur berücksichtigt, wenn diese Variable gesetzt ist:

```
# Im Beispiel existiert im aktuellen Verzeichnis einzig eine Datei ".foo"#
user@sonne> ls *
ls: *: Datei oder Verzeichnis nicht gefunden

user@sonne> shopt -s dotglob
user@sonne> ls *
.foo
```

mailwarn

Ist diese Variable gesetzt, überprüft die Bash die Mailboxen auf neue Nachrichten und gibt ggf eine Mitteilung aus. Die Option korrespondiert mit den Variablen MAIL und MAILPATH.

interactive_comments

Ist die Option gesetzt, gilt alles, was auf der Kommandozeile einem Doppelkreuz # folgt, als Kommentar

nocaseglob

Ist diese Option gesetzt, spielt Groß- und Kleinschreibung von Dateinamen für die Bash keine Rolle:

```
user@sonne> ls .F*
ls: .F*: Datei oder Verzeichnis nicht gefunden

user@sonne> shopt -s nocaseglob
user@sonne> ls -F*
.foo
```

restricted_shell

Diese Option ist gesetzt, wenn die Bash im restricted Modus gestartet wurde; sie kann nicht geändert werden

Die weiteren Variablen sind wohl selten von Nutzen, es sei denn, jemand wünscht gezielt bestimmte Expansionsmechanismen zu deaktivieren.

Aliasse

Zur Abkürzung immer wiederkehrender Kommandofolgen lassen sich für diese so genannte Aliasse definieren. Ein Alias wird mit dem Kommando alias erzeugt:

```
user@sonne> alias dir='ls -l'
user@sonne> alias md=mkdir
user@sonne> alias rd=rmdir
user@sonne> alias rename=mv
user@sonne> alias cdlin='cd / usr/ src/ linux'
```

Ein Alias wird genauso benutzt wie das entsprechende Kommando:

```
user@sonne> dir /
insgesamt 65
drwxr-xr-x   2  root     root         2048 Dec 14 13:23 bin
drwxr-xr-x   3  root     root         1024 Dec 21 10:59 boot
drwxr-xr-x   2  root     root         1024 Dec 14 13:05 cdrom
drwxr-xr-x   6  root     root        30720 Dec 29 08:50 dev
...
user@sonne> md directory
user@sonne> rd directory
user@sonne> rename nt refuse
user@sonne> cdlin; pwd
/usr/src/linux
```

Ein Alias existiert bis zum Ende der Shell, in der er definiert wurde oder bis zum expliziten Löschen mittels unalias:

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

# ein Alias ist nur in der Shell seiner Definition bekannt:
user@sonne> bash
user@sonne> md tmp
bash: md: command not found
```

Ein Aufruf von alias ohne Argumente bewirkt eine Auflistung aller definierten Abkürzungen.

Funktionen

Eine Funktion ist ein Name für ein Kommando oder für eine Gruppe von Kommandos. Funktionen werden vorrangig in Shellskripten verwendet, um wiederkehrende Kommandosequenzen nicht ständig neu schreiben zu müssen. Ein großer Vorteil von Funktionen ist, dass sich diese in einer Datei speichern lassen und diese Datei von anderen Skripten geladen werden kann.

Eine Funktion wird wie folgt definiert:

Format: [function] Funktionsname() { Kommando; [Kommando;] }

Bei der Verwendung von Funktionen sind einige Regeln zu befolgen:

1. **Deckt sich der Name der Funktion mit einem builtin-Kommando, wird immer die Funktion ausgeführt und niemals das Kommando. Ebenso verdeckt ein Funktionsname ein gleichnamiges Kommando:**

```
user@sonne> type test
test is a shell builtin
user@sonne> test(){ echo "ich bin eine Funktion"; }
user@sonne> type test
test is a function
test ()
{
echo "ich bin eine Funktion"
}
user@sonne> unset test
```

2. **Die Funktion muss vor ihrer Verwendung definiert sein.**
3. **Eine Funktion läuft in der aktuellen Umgebung, d.h. alle Variablen der Umgebung sind sichtbar und alle Variablen, die in der Funktion definiert wurden, sind auch außerhalb sichtbar:**

```
user@sonne> func(){ var_in_func=xxx; }
user@sonne> func
user@sonne> echo $var_in_func
xxx
```

4. **Wird eine Funktion mittels »exit« verlassen, wird auch der rufende Prozess beendet:**

```
user@sonne> func(){ exit; }
user@sonne> func
login:
```

5. **Der Rückgabewert einer Funktion ist der Rückgabewert des letzten in ihr gestarteten Kommandos:**

```
user@sonne> func(){ grep -q foo /etc/passwd; echo $?; }
user@sonne> func
1
user@sonne> echo $?
0
user@sonne> func(){ return 255; }
user@sonne> func
user@sonne> echo $?
255
```

6. **Funktionen sind nur in der Shell ihrer Definition bekannt:**

```
user@sonne> func(){ echo "lokal"; }
user@sonne> bash
user@sonne> func
bash: func: command not found
user@sonne> exit
user@sonne> func
lokal
```

Einer Funktion können Parameter als Argumente übergeben werden:

Aufruf: Funktionsname [Arg1] [Arg2] ...

Innerhalb einer Funktion kann auf die Parameter mittels der Positionsparameter $1..$9 zugegriffen werden. Als Beispiel dient eine Funktion "square", die die als Argument übergebene Zahl quadriert:

```
user@sonne> square() { test -z $1 && return 1; expr $1 \* $1; }
user@sonne> square
user@sonne> square 18
324
```

Erklärung: Das builtin-Kommando test liefert den Status "0", falls die Variable "$1" leer ist (kein Argument). In diesem Fall wird "return 1" ausgeführt und der Funktionsaufruf beendet. expr berechnet den Ausdruck und schreibt das Ergebnis auf die Standardausgabe.

Eingebaute Kommandos

Es liegt wohl in der Natur des Linux-Neulings, seine Testprogramme und -skripten «"test« zu benennen (eigentlich kann ich mich an keinen Linuxkurs erinnern, indem nicht mindestens einer der Teilnehmer auf diese Weise mit den builtin Kommandos der Bash konfrontiert wurde). Nach verrichteter Arbeit zeigte der Testlauf:

```
user@sonne> ls -l test
-rwxr-xr-x 1 user users 12177 Sep 23 10:52 test
user@sonne> test
```

...nichts...? Mit Kenntnis des Vorgehens der Bash bei der Suche nach einem Kommando, gelangt man bald zum Schluss, dass sich ein builtin Kommando vorgedrängelt hat.

Es gibt eine Fülle solcher eingebauter Kommandos und mindestens 4 Gründe, warum solche in der Bash überhaupt existieren:

1. **Weil es ein solches Kommando in Unix nicht gibt (Beispiel "source")**
2. **Weil ein builtin Kommando effizienter arbeitet, als ein externes Kommando (keine Prozesserzeugung notwendig; Beispiel »echo«)**
3. **Weil nur ein eingebautes Kommando Bash-interne Variablen ändern kann (Beispiel »export«)**
4. **Weil ein Kommando wie »exec« nur innerhalb der Bash realisierbar ist**

Betrachten wir die einzelnen builtin-Kommandos:

:

Dieses »Kommando« tut nichts, außer einen Rückgabewert »0« zu erzeugen (»0« ist der üblic Rückgabewert eines Kommandos unter Unix, wenn seine Ausführung erfolgreich war). Nützlic es in Shellskripten, falls Sie in Bedingungen einen wahren Wert (»true«) benötigen oder an Positionen, wo syntaktisch ein Kommando erwartet wird, Sie aber keines benötigen:

```
user@sonne> while : ;do echo "Eine Endlosschleife"; done
Eine Endlosschleife
Eine Endlosschleife
Eine Endlosschleife
...
Eine Endlosschleife[Ctrl]+[C]
user@sonne> if test -a foo ; then :; else echo "Datei nicht existent"; fi
Datei nicht existent
```

. <Datei> bzw. source <Datei>

Die angegebene Datei wird gelesen und innerhalb der Shellumgebung ausgeführt. Ist die Date ohne Pfad angegeben, wird dabei in Verzeichnissen der PATH-Variable gesucht. Wird sie dort r gefunden, wird das aktuelle Verzeichnis betrachtet. Das Kommando kann sich der C-Programmierer als "include" der Bash vorstellen.

Manche Linuxdistributionen (bspw. RedHat) verwenden dieses Kommando in ihren Runlevel-Skripten, um eine Funktionsbibliothek einzubinden.

Auf der Kommandozeile bietet sich "source" an, um die initiale Umgebung zu rekonstruieren (v man u.U. die Umgebungsvariablen "verbogen" hat):

```
user@sonne> source /etc/profile
# bzw.
user@sonne> ./etc/profile
```

alias

Dient der Definition einer Abkürzung für ein(e) Kommando(folge). Mit der Option -p werden al vorhandenen Aliasse aufgelistet; Beispiele wurden bereits im einleitenden Abschnitt zu Bash (Eingabehilfen) und in Allgemeines zu Shells genannt.

bg

Listet die Jobnummern aller Hintergrundprozesse auf.

bind

Das Kommando besitzt in erster Linie im Zusammenspiel mit der Datei /etc/inputrc (bzw. ~/.inputrc) eine Rolle. Besagte Dateien enthalten die »Keybindings«, also die Zuordnungen vo Tastenkombinationen zu bestimmten Funktionalitäten. Wenn bspw. die unter Interaktive Bash aufgeführten Eingabehilfen bei Ihnen nicht funktionieren sollten, dann fehlt in der Konfigurati Ihrer Bash die notwendige Zuordnung (oder Sie verwenden eine ältere Bashversion).

bind kann auch interaktiv benutzt werden, um bestimmte Keybindings zu löschen, um sie sich anzeigen zu lassen oder neue zu definieren. Letzteres ermöglicht gar die Ausführung von Kommandos bei Betätigung einer zuvor definierten Tastensequenz. Benötigen Sie solche Sequenzen häufig, so nehmen Sie sie in ihre persönliche ~/.inputrc auf.

Wenn Sie mit der Bash einigermaßen per Du sind, so kennen Sie die eine oder andere Tastenkombination, um geschwind auf der Kommandozeile zu navigieren oder diese zu manipulieren. Hinter den Mechanismen verbergen sich readline-Funktionen. Welche es gibt, ve der Aufruf »bind -l« und über deren aktuelle Belegung weiß »bind -p« bescheid. Erscheint in letzter Ausgabe »not bound«, so ist diese Funktion nicht belegt.

```
user@sonne> bind -p | head -5

"\C-g": abort
"\C-x\C-g": abort
"\e\C-g": abort
"\C-j": accept-line
```

Angenommen Sie verfassen im vi ein deutschsprachiges html-Dokument. Um die Zeichen »ä«, »ö«,... in Browsern, die deutsche Zeichensätze nicht unterstützen, korrekt darzustellen, sollte alle länderspezifischen Zeichen im Unicode dargestellt werden, also bspw. »ü« anstatt » Die komfortable Lösung ist das Verändern der Tastaturbelegung, so dass automatisch der Unic im Text erscheint. In der Bash erreichen Sie dies wie folgt:

```
user@sonne> bind '"ü":"&uuml;"'
```

Sobald Sie »ü« tippen, erscheint in der Eingabe »ü«. Um die Bindung aufzuheben, geber »bind -r <Tastensequenz>« an; aber für gewöhnlich besteht der Wunsch, die alte Belegung wieder herzustellen. Eine Taste, die »als sie selbst« definiert wird, wird in folgender Manier belegt:

```
user@sonne> bind 'ü:self-insert'
```

Obige Form der Tastenbindung nennt man auch Makro; daneben können sie auch Funktionen u gar Kommando(s) an Tasten(sequenzen) binden. Das nachfolgende Beispiel bindet [Ctrl]+[b] die Ausgabe des Kalenders des aktuellen Monats:

```
user@sonne> bind -x '"\C-b":cal'
user@sonne> [Ctrl]+[B]
    November 2000
So Mo Di Mi Do Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

Ihre gesammelten Werke der Tastenbindungen können Sie auch in eine beliebige Datei schreib und diese mittels "bind -f <Datei>" laden.

break [n]

Dient zum expliziten Verlassen einer Schleife. Ohne Angabe eines Arguments wird die unmitte umgebende Schleife verlassen; möchte man tiefere Verschachtelungen verlassen, muss die Tie angegeben werden:

```
...
while [ Bedingung ]; do
  for i in Liste; do
    case "$i" in
      foo* ) break;
      bla* ) tue etwas ;;
      * ) Fehler; break 2;
    esac
  done
done
...
```

builtin Kommando

Bei der Suche nach Kommandos betrachtet die Shell Aliasse und Funktionen noch vor den eingebauten Kommandos. Überdeckt nun ein solcher Name ein builtin-Kommando, so wird bei einfachen Aufruf immer der Alias bzw. die Funktion ausgeführt werden. Mit vorangestelltem builtin weist man nun die Bash an, auf jeden Fall ihr eingebautes Kommando aufzurufen. Der Rückgabestatus ist gleich dem Rückgabewert des Builtin's oder "falsch", falls das Kommando k builtin ist.

cd

command Kommando

Ein weiterer Weg, um die Verwendung von Aliassen oder Funktionen bei der Suche nach einem Kommando temporär auszuschalten, ist ein dem zu startenden Kommandonamen voranzustellendes command. Die Shell sucht nun einzig in der Liste der eingebauten Kommand und den Pfaden aus PATH. Hilfreich ist die Option -p, falls die PATH-Variable einmal "völlig daneben" belegt ist; die Bash sucht in default-Pfaden und findet so zumindest die wichtigsten Programme.

compgen

Mit dem Kommando lassen sich die möglichen Expansionen anzeigen. Um bspw. gezielt die denkbaren Erweiterungen aller mit "l" beginnenden Aliasse zu erhalten, ist folgende Kommandozeile notwendig:

```
user@sonne> compgen -A alias l
l
la
ll
ls
ls-l
```

Anstatt von "alias" können u.a. Funktionen ("function"), Schlüsselworte der Bash ("keywords" Dateinamen ("file"), Verzeichnisse ("directory"), Variablen ("variable") expandiert werden.

Noch weit reichender sind die Möglichkeiten in Bezug auf Dateinamen, da hier mit Suchmuster gearbeitet werden kann:

```
user@sonne> compgen -G '* c* '
Packages
bla.xcf
countdir
linuxbuch
Documents
```

Abgesehen von -r und -r verwendet compgen dieselben Optionen wie das nachfolgend beschriebene complete.

complete

Mit diesem Kommando kann das ganze Verhalten der Bash bei der Vervollständigung von Argumenten verändert werden. Bevor wir uns mit den Optionen auseinander setzen, soll ein Beispiel die Mächtigkeit des Konzepts andeuten.

Beispiel: Als Argumente für das Programm xv (dient der Anzeige von Dateien diverser Grafikformate) sollen bei der automatischen Dateinamensergänzung nur Dateien mit einer typischen Dateiendung (* .jpg, * .gif, ...) berücksichtigt werden. Mit complete müsste die Bash folgt eingerichtet werden:

```
# Die Option "extglob" (vergleiche shopt) muss gesetzt sein:
user@sonne> shopt -q extglob | | shopt -s extglob

# Test des bisherigen Verhaltens:
```

Programmierung der Bourne Again Shell

Wege zum Skript

Hello world!

Wohl jede Einführung in eine Programmiersprache beginnt mit einem besonders einfachen Beispiel. Und die wohl beliebteste Anwendung ist das Erstlingswerk, das ein "Hello world!" auf den Bildschirm zaubert. In der Bash sähe das Ergebnis wie folgt aus:

```
#!/bin/sh

echo "Hello world!"
```

Speichert man die Zeilen in eine Datei namens »hello« und führt diese zur Ausführung einer Bash zu, so erscheint - welch ein Wunder - die vermutete Antwort:

```
user@sonne> bash hello
Hello world!
```

Für die Zukunft werden wir ein solches Shellskript gleich mit den Ausführungsrechten versehen, so dass sich der Aufruf verkürzt:

```
user@sonne> chmod +x hello
user@sonne> ./ hello
Hello world!
```

Wer sich jetzt fragt, was der »Kommentar« auf der ersten Zeile in unserem Beispiel zu suchen hat, der sei in seiner Ansicht bestätigt, dass jede Zeile, die mit einem Doppelkreuz beginnt, ein Kommentar ist. Außer... dem Kommentarzeichen folgt ein Ausrufezeichen. In dem Fall versucht die Shell, den Inhalt der Datei mit dem in dieser Zeile angegebenen Interpreter auszuführen. In diesem Abschnitt wird es sich bei dem Interpreter durchweg um die Bash handeln. Geben Sie den Interpreter immer mit vollständigem Zugriffspfad an, sonst ernten Sie nur verwirrende Fehlerhinweise:

```
user@sonne> cat hello
#!bash

echo "Hello world!"
user@sonne> ./ hello
bash: ./hello: Datei oder Verzeichnis nicht gefunden
```

Vermögen Sie es, aus der Fehlermitteilung auf den wahren Grund zu schließen (Hinweis: Der Inhalt von $PATH wird hier nicht betrachtet.)?

Vielleicht haben Sie schon bemerkt, dass das Shellskript auch ohne diese erste Zeile brav seinen Dienst verrichtet? Dem sollte auch so sein, denn jede Unix-Shell interpretiert diese einfache »echo«-Ausgabe gleich. Gewöhnen Sie sich dennoch an, die Shell, für die Ihr Skript verfasst wurde, auf diese Art zu spezifizieren. So können Sie das Shellskript auch innerhalb anderer Shells (z.B. csh) aufrufen, die womöglich mit der Syntax ihre Probleme hätten.

Von der Idee zum Skript

Jede Aneinanderreihung von Befehlen, die Sie in eine Datei schreiben, formt ein Shellskript. Der Aufwand lässt sich für wiederkehrende Arbeiten auf diese Art und Weise drastisch verringern und dennoch ist der Nutzen von Skripten oft nicht unmittelbar ersichtlich.

Wer sich intensiv mit Unix beschäftigt, wird vielfach simple Werkzeuge vermissen. Warum hat wohl noch niemand jenes programmiert? Vielleicht ja, weil Unix von Haus aus die Mittel mit sich bringt, dass ein jeder - ein geringes Grundwissen vorausgesetzt - durch geschickte Kombination existenter Programme ein solches Hilfsmittel modellieren könnte.

Im vorangegangenen Abschnitt zur Bash haben Sie alles Notwendige kennen gelernt, um munter drauf los »skripten« zu können. Die Beherrschung der Unix-Werkzeuge grep, sed und awk ist für viele elegante Lösungen erforderlich. Was Ihnen nun vermutlich noch fehlt, ist ein konkreter Plan, wie Sie Ihr Problem elegant und effizient in ein Shellprogramm fassen können.

Was Ihr Problem ist (ich meine das jetzt hinsichtlich der Computerfragen), vermögen wir nicht vorherzusehen, aber wozu die Bash fähig ist - im positiven Sinne - sollen die nachfolgend vorgestellten Lösungen zu (oft) praxisnahen Anwendungen verdeutlichen.

Was die Bash nicht kann

Wozu benötigt man »höhere« Programmiersprachen, wenn die Bash das Problem auch behandeln könnte? Weil das Programm der Bash i n t e r p r e t i e r t wird und schon allein aus diesem Grund wesentlich langsamer abläuft, als es ein kompiliertes Programm vermag. Aber auch bei komplexeren Problemen erschöpfen sich rasch die Mittel der Bash.

Aufgaben, für die Sie ein Shellskript keinesfalls in Betracht ziehen sollten, sind:

- Zeitkritische Abläufe (tiefe Rekursionen)
- Hardwarenahe Programmierung (geht definitiv nicht)
- Wirklich große Anwendungen
- Erweiterte Dateizugriffe, die über das serielle Lesen und Schreiben der Daten hinaus gehen
- Grafische Oberflächenprogrammierung (nur »Pseudografik« in Verbindung mit dem Kommando »dialog«)
- Zugriff auf Sockets

Fehlersuche

Reale Skripte sind um Einiges komplexer, als es das einführende "Hello World" vermittelte. Und Fehler bleiben leider nicht aus. Syntaktischen Ungereimtheiten kommt die Bash schnell auf die Schliche. Aber nicht immer sind die Mitteilungen über die Fehlerursache aussagekräftig:

```
user@sonne> cat script_with_an_error
#!/bin/sh

if ["$1" -eq "" ]
then
   echo "Missing an argument."
   exit 1
fi
user@sonne> ./ script_with_an_error foo
./script_with_an_error: [foo: command not found
```

Command not found ist nun wirklich keine hilfreiche Mitteilung...

Wie expandiert die Bash eine Anweisung?

In der Testphase erweist sich die Shelloption **xtrace** als äußerst nützlich, da jede Kommandozeile eines Skripts nach ihrer Expansion ausgegeben und erst anschließend ausgeführt wird. Setzen Sie **set -x** an den Anfang des obigen Fehlerskripts, so entlarvt sich der Fehler von selbst:

```
user@sonne> cat script_with_an_error
#!/bin/sh
set -x

...
user@sonne> ./ script_with_an_error foo
+ '[foo' -eq '' ']'
./script_with_an_error: [foo: command not found
```

Welche Anweisung wurde gerade bearbeitet?

Obige Ausgabe führt bei kürzeren Skripts sofort zur Fehlerquelle; dennoch wäre es wünschenswert, wenn die gerade betrachtete Kommandoanweisung nochmals ausgegeben werden würde. Mit der Shelloption **verbose** (set -v) entlockt man der Bash etwas konkretere Informationen.

```
user@sonne> cat fakultaet.sh
#!/bin/sh
set -xv   # xtrace und verbose aktivieren
declare -i zahl=$1
declare -i fakultaet

while [ $zahl -gt 1 ]; do
        fakultaet=$fakultaet*$zahl
        zahl=$zahl-1
done

echo "Fakultät = " $fakultaet
```

Erkennen Sie den Fehler im Skript? Ein Lauf offenbart ihn schnell:

```
user@sonne> ./ fakultaet.sh 5

declare -i zahl=$1
+ declare -i zahl=5
declare -i fakultaet
+ declare -i fakultaet

while [ $zahl -gt 1 ]; do
        fakultaet=$fakultaet*$zahl
        zahl=$zahl-1
done
+ '[' 5 -gt 1 ']'
+ fakultaet=*5
./schnelltest.sh: *5: syntax error: operand expected (error token is "*5")

echo "Fakultät = " $fakultaet
+ echo 'Fakultät = '
Fakultät =
```

Die letzte Anweisung vor dem Fehler betraf die **while**-Schleife. Der Fehler selbst resultiert aus der Verwendung der nicht initialisierten Variable "$fakultaet". Ändert man die Zeile "declare -i fakultaet" in "declare -i fakultaet=1", liefert das Skript das erwartete Resultat.

Fehler und deren Folgen

Letztes Beispiel weist schon auf ein mögliches Problem hin, das in bestimmten Skripten zu durchaus drastischen Folgen führen kann. Obiges Skript läuft selbst im Fehlerfall weiter. Mitunter wird durch eine fehlgeschlagene Anweisung ein gänzlich anderer Kontrollfluss durchlaufen. Was, wenn nun zum Beispiel Daten einer Dateiliste entfernt werden, wobei die Berechnung der Liste den Fehler verantwortete?

Kritische Skripte - die Änderungen im Dateisystem vornehmen - sollten zunächst ausgiebig getestet und erst bei erwiesener syntaktischer Korrektheit eingesetzt werden. Setzen Sie während der Entstehungsphase eines Skripts

die Shelloption **noexec** (set -n), so wird die Bash die meisten syntaktischen Fallen erkennen, ohne die Anweisungen tatsächlich auszuführen.

Ein fehl geschlagenes einfaches Kommando führt in der Standardeinstellung der Shell ebenso wenig zum Ende eines Skripts. Erst **errexit** (set -e) stoppt die weitere Bearbeitung. Allerdings nützt die Option nichts, wenn das Kommando innerhalb einer Schleife (while bzw. until) oder als Bestandteil logischer Konstrukte (if, ||, &&) scheitert.

Logische Fehler

Denkfehlern ist wesentlich aufwändiger auf die Schliche zu kommen. Einzig auf nicht-gesetzten Variablen basierende Fallstricke kann die Shelloption **nounset** (set -u) durchtrennen.

Das Allheilmittel lautet **echo**. Bauen Sie an allen kritischen Stellen Ihrer Skripte Ausgaben ein und verifizieren Sie die Belegung der Variablen. Werden Ergebnisse in Pipelines verarbeitet, kann das Kommando **tee** dazu dienen, die Daten der Zwischenschritte zu inspizieren.

Entwickeln Sie die Skripte schrittweise. Testen Sie Funktionen zunächst mit statischen Eingabedaten, bevor Sie sie in komplexeren Anwendungen verwenden. Stellen Sie sicher, dass keine unerlaubten Argumente Ihr Skript auf Abwege bringt.

Endreinigung

Ein unfreiwilliger Abgang

In manchen Skripten wird es erforderlich, gewisse Systemressourcen zu reservieren. Das einfachste Beispiel ist die Ablage von Daten in temporären Dateien. Vor Beendigung des Skripts wird dieses sicherlich dafür Sorge tragen, den nicht mehr benötigten Datenmüll zu entsorgen.

Was aber, wenn Ihr Skript gar nicht das Ende erreichte? Vielleicht bootet Ihr Administrator soeben mal den Rechner neu und "schießt" Ihr Skript mir nichts dir nichts ab? Oder Sie selbst vereiteln die Vollendung, weil die Dauer der Berechnung nun doch Ihre Geduld strapaziert. Was ist mit den temporären Daten?

Tatsächlich erachten es die wenigsten Skripte für sinnvoll, im Falle des expliziten Abbruchs diesen abzufangen und ein abschließendes Groß-Reine-Machen vorzunehmen.

Erinnern wir uns, dass die asynchrone Interprozesskommunikation unter Unix im Wesentlichen über Signale realisiert wird. Einem Prozess, dessen Ende man bewirken will, wird man zuvorkommend das Terminierungssignal (SIGTERM) oder - im Falle eines Vordergrundprozesses - den Tastaturinterrupt ([Ctrl]-[C], SIGINT) senden. Erst wenn dieser nicht reagiert, sollte per SIGKILL das Ableben erzwungen werden. Tatsächlich wird beim Herunterfahren des Systems zunächst SIGTERM an einen jeden Prozess gesendet und erst anschließend ein SIGKILL.

Ein Programm kann mit Ausnahme von SIGKILL jedes Signal abfangen und die übliche Bedeutung dessen verbiegen. In der Bash werden Signale mit dem eingebauten Kommando **trap** maskiert.

Fallstudie: Ein simples Backup-Skript

Ob Hardwareausfall oder Bedienfehler... früher oder später wird jeder einmal mit einem unbeabsichtigten Datenverlust konfrontiert. Eine Sicherungskopie wichtiger Daten sollte daher in keiner Schreibtischschublade fehlen. Jedoch verfügt nicht jeder Rechner über Streamer, CD-Brenner oder andere geeignete Backupmedium.

Aber ein Diskettenlaufwerk werden wohl die wenigsten Boliden missen. Nur passen selten alle interessanten Daten auf eine einzige Magnetscheibe, sodass ein so genanntes Multi-Volume-Backup von Nöten wird. Das Kommando **tar** unterstützt zwar von Haus aus die Verteilung großer Archive auf mehrere Medien, allerdings muss in jenem Fall auf eine Komprimierung des Archivs verzichtet werden.

Nach diesen Vorbetrachtungen notieren wir uns die Fähigkeiten, die unser einfaches Backup-Skript mit sich bringen soll:

- Es soll alle Konfigurationsdateien aus **/ etc** sichern (eine Erweiterung bleibt dem Leser überlassen)
- Die Dateien sollen mit **bzip2** gepackt werden
- Ein Aufruf "*Scriptname* -c" erzeugt das Archiv
- Ein Aufruf "*Scriptname* -r" spielt die Dateien zurück

Kopfzerbrechen sollte dem geübten Bash-Anwender einzig das Packen bereiten. Die Lösung, die unser Skript verwenden wird, ist das Packen jeder einzelnen Datei, wobei diese zunächst in einem temporären Verzeichnis abgelegt wird. Abschließend wird **tar** das Archiv anhand dieses Verzeichnisses erzeugen. Die Wiederherstellung der Daten verläuft genau umgekehrt. Das Skript entpackt das Archiv in ein temporäres Verzeichnis und entpackt erst anschließend die Daten an ihren angestammten Ort.

Um endlich den Bezug zur Überschrift des Abschnitts herzustellen, soll unser Skript auch dafür Sorge tragen, dass keine temporären Dateien im Dateisystem verbleiben, falls das Skript abgebrochen werden sollte.

Das Rahmenprogramm

Die Auswertung der Kommandozeilenoption (-r bzw. -c sollen zulässig sein) überlassen wir dem eingebauten Kommando **getopts** (wird später behandelt). Der Rahmen des Skripts sieht wie folgt aus:

```
#!/bin/sh

function benutze()
{
   echo "Unbekannte oder fehlende Option!"
   echo "Anwendung: $0 -c|-r"
   exit 1
}

function archiviere()
{
   !:
   # noch zu schreiben
}

function restauriere()
{
   !:
   # noch zu schreiben
}

while getopts rc Optionen; do
   case $Optionen in
      c) # Aufruf der Archivierungsfunktion
         archiviere
         ;;
      r) # Aufruf der Wiederherstellungsfunktion
         restauriere
         ;;
      *) # Unbekannte Option
         benutze
         ;;
   esac
done

test "$OPTIND" == "1" && benutze # kein Argument auf der Kommandozeile
```

Die Archivierungs-Funktion

Für die Zwischenablage der gepackten Dateien benötigen wir ein Verzeichnis. Um die temporäre Natur dessen zu unterstreichen, wird es in /tmp erzeugt. Als (hoffentlich) eindeutigen Namen wählen wir "etc*Prozessnummer*". Die

entsprechende Anweisung in der Funktion lautet somit:

```
mkdir /tmp/etc$$
```

Die Suche nach den regulären Dateien wird mittels **find** realisiert. Das Packen erfolgt in einer Schleife über die einzelnen Dateien. Zunächst wird die Datei - sofern wir Leserechte besitzen - kopiert und anschließend komprimiert:

```
for i in `find /etc -type f -maxdepth 1`; do
  test -r $i || continue
  cp $i /tmp/etc$$
  bzip2 /tmp/etc$$/`basename $i`
done
```

Die Vorbereitungen sind abgeschlossen, fehlt nur noch das Archivieren auf Diskette (/dev/fd0):

```
tar -Mcf /dev/fd0 /tmp/etc$$
```

Mit dem abschließenden Löschen von /tmp/etc$$ nimmt die komplette Funktion folgende Gestalt an:

```
function archiviere()
{
  test -e /tmp/etc$$ || mkdir /tmp/etc$$

  for i in `find /etc -type f -maxdepth 1`; do
    test -r $i || continue
    cp $i /tmp/etc$$
    bzip2 /tmp/etc$$/`basename $i`
  done

  tar -Mcf /dev/fd0 /tmp/etc$$

  rm -r /tmp/etc$$
}
```

Die Wiederherstellungs-Funktion

Wir überlassen die Realisierung dem Leser. Vergessen Sie nicht das abschließende Entfernen des Verzeichnisses mit den komprimierten Dateien!

Behandlung von Signalen

Um unser Skript vor vorschnellem Abbruch durch Signale zu schützen, müssen wir diese abfangen und behandeln. Unsere Reaktion wird sein, dass wir rasch alle temporären Dateien/Verzeichnisse löschen und anschließend das Skript verlassen. Bei den Signalen beschränken wir uns auf SIGINT (Nummer 2; Tastaturinterrupt) und SIGTERM (15). Ein **trap**-Aufruf zu Beginn unseres Skripts ist alles, was wir tun müssen:

```
#!/bin/sh

trap 'test -e /tmp/etc$$ && rm -r /tmp/etc$$; exit 0' 2 15
```

Wann immer eines der beiden Signale auf den Prozess einherfällt, wird **trap** das temporäre Verzeichnis samt Inhalt entfernen - falls es existiert - und das Skript beenden.

Sie können **trap** ebenso verwenden, um bestimmte Signal zu ignorieren. Wählen Sie hierfür eine der beiden Syntaxvarianten:

```
trap "" 2 3 15 19
```

```
# Alternative Angabe:

trap : 2 3 15 19
```

Um in einem Skript die Behandlung von Signalen wieder zuzulassen, ist ein Aufruf von **trap ohne Argumente** erforderlich.

Korrekte Eingaben

Eine Binsenweisheit der Programmierer besagt, dass die ergiebigste Fehlerquelle vor dem Bildschirm sitzt. Benutzer zeigen oft Verhaltensmuster, die jeglicher Logik entbehren. Ein mit Nutzereingaben arbeitendes Programm sollte deshalb fehlerbehaftete Eingaben erkennen und darauf vernünftig reagieren.

Prüfung der Eingabe

Von einer Eingabe erwarten wir i.A., dass sie von einem bestimmten Typ ist. Wir möchten wissen, ob die eingegebene Zeichenkette ein Datum im geforderten Format ist, ob sie eine Zahl ist, ob eine Zahl im geforderten Bereich liegt...

Sicher stellen lässt sich dies jedoch nur durch eine Prüfung im Anschluss an die Eingabe. Für zwei verbreitete Fälle möchten wir Lösungen oder Lösungsansätze präsentieren:

Ganze Zahl

Eine gültige ganze Zahl darf mit oder ohne Vorzeichen angegeben werden. Führende oder nachfolgende Leerzeichen sollten ebenso akzeptiert werden, wie Zwischenräume zwischen Vorzeichen und Zahl. Folgende Angaben stellen demnach korrekte Eingaben dar (zur Veranschaulichung von Leerzeichen werden die Zahlen in Anführungsstriche eingeschlossen):

```
"1928888" "-12" "+ 332  " " - 0"
```

Lösung 1: Indem eine solche Zahl einer mit "declare -i" typisierten Variable zugewiesen wird, lässt sich anhand des Inhalts der Variablen prüfen, ob eine korrekte Eingabe vorliegt:

```
user@sonne> declare -i var="  - 1234"
user@sonne> echo $var
-1234
user@sonne> var=" + 1bla"
bash: + 1bla: value too great for base (error token is "1bla")
user@sonne> var=" + bla1"
user@sonne> echo $var
0
```

Aus den Beispielen sind zwei Probleme ersichtlich:

1. Die Zuweisung eines fehlerhaften Wertes bewirkt **eine Fehlermeldung** der Shell, wenn der Wert mit einer (vorzeichenbehafteten) Ziffer beginnt.
2. Die Zuweisung eines fehlerhaften Wertes bewirkt **keine Fehlermeldung**, wenn der Wert mit keiner Ziffer beginnt.

Die Konsequenz aus (1) ist, dass wir in einem Shellskript Fehlerausgaben abfangen sollten. Ob ein Fehler auftrat, lässt sich anhand des Rückgabewertes verifizieren:

```
# Die Meldung kommt von der Shell! Deshalb Wertzuweisung in einer Subshell:
user@sonne> (var=" + 1bla") 2> /dev/null || echo "Fehler in Zuweisung"
```

Die Ursache für (2) ist, dass die Bash Zeichenketten bei Bedarf intern zu "0" konvertiert. Da "0" gleichzeitig ein

akzeptierter Eingabewert ist, sollte man sich zusätzlich versichern, dass tatsächlich "0" in der Eingabe stand (Zeichenkettenvergleich)!

Lösung 2: Mit Hilfe der Parametersubstitution scannen wir die Eingabe:

```
user@sonne> Eingabe="+ 1992"
user@sonne> var=${Eingabe##*[^0-9,' ',-]*}
user@sonne> echo $var
1992
```

Viele Leser werden den obigen Ausdruck wohl kaum interpretieren können, außerdem "übersieht" er eine Art Eingabefehler (Finden Sie ihn selbst heraus!). Analog zur "Lösung 1" muss auch hier eine zweite Überprüfung herhalten, um sämtliche Fälle abzudecken.

Lösung 3: Wir greifen auf das Kommando (e)grep zurück und suchen nach "einem Muster" (alternativ lässt sich auch der Stream Editor verwenden):

```
user@sonne> Eingabe="+ 1992 12"
user@sonne> echo $Eingabe | egrep -q '^[[:space:]]*[+-]?[[:space:]]*[[:digit:]]+[[:space:]]*$' || echo "Fehler"
Fehler
```

Zugegeben, die Definition des Suchmusters ist nicht leicht zu verdauen. Dafür deckt sie tatsächlich alle Eventualitäten ab.

(Ganze) Zahl aus einem Bereich

Die Prüfung erfolgt zweckmäßig in zwei Schritten:

1. Handelt es sich um eine gültige (ganze) Zahl?
2. Liegt sie im Intervall?

Bei solchen komplexen Tests bietet es sich an, diese in eine Funktion auszulagern:

```
function CheckRange()
{
  test -z "$1" && return 1
  echo "$1" | egrep -q '^[[:space:]]*[+-]?[[:space:]]*[[:digit:]]+[[:space:]]*$' || return 2
  [ "$1" -lt "-10" -o "$1" -gt "+10" ] && return 3;
  return 0
}
```

Die Funktion testet in drei Schritten. Zuerst wird der korrekte Funktionsaufruf geprüft (mit Argument). Fehlt dieses, wird die Funktion mit dem Fehlercode "1" beendet. Schritt 2 umfasst die Prüfung auf eine ganze Zahl. Bei Fehler wird mit dem Status 2 zurück gekehrt. In Schritt 3 findet schließlich die Bereichsüberprüfung statt (hier von -10 bis +10). Auch hier wird im Fehlerfall ein eigener Rückgabewert (3) verwendet. Wurden alle Tests erfolgreich absolviert, wird "0" geliefert.

Die Anwendung der Funktion kann nach folgendem Schema erfolgen:

```
user@sonne> Eingabe="0815"
user@sonne> CheckRange $Eingabe || echo "Fehlercode $?"
```

Reelle Zahl

Betrachten wir zuvor die Darstellungsmöglichkeiten für reelle Zahlen:

```
+.12993   120.02   11   -0E2   1.44e-02   2.
```

Eine solche Fülle von Varianten lässt sich nur schwerlich in einen einzigen Regulären Ausdruck pressen. Unsere Funktion wird deshalb die Eingabe der Reihe nach mit Mustern für die einzelnen Syntaxarten vergleichen. Wird ein Test bestanden, wird mit dem Status "0" zurück gekehrt. Wurden alle Tests negativ durchlaufen, signalisiert eine "2" einen Syntaxfehler:

```
function CheckRealNumber()
{
  test -z "$1" && return 1

# Gleitkommadarstellung mit Nachkommastellen ".01", " - 10.1 " ...
  echo "$1" | egrep '^ [[:space:]]*[+-]?[[:space:]]*[[:digit:]]*\.?\
[[:digit:]]+[[:space:]]*$' && return 0

# Gleitkommadarstellung ohne Nachkommastellen "1.", " - 10 " ...
  echo "$1" | egrep '^ [[:space:]]*[+-]?[[:space:]]*[[:digit:]]+\.?\
[[:space:]]*$' && return 0

# Exponentialdarstellung mit Nachkommastellen " + 0.01e01"...
  echo "$1" | egrep '^ [[:space:]]*[+-]?[[:space:]]*[[:digit:]]*\.\
[[:digit:]]+[Ee][+-]?[[:digit:]][[:digit:]]?[[:space:]]*$' && return 0

# Exponentialdarstellung ohne Nachkommastellen " + 1e-1"...
  echo "$1" | egrep '^ [[:space:]]*[+-]?[[:space:]]*[[:digit:]]+[Ee]\
[+-]?[[:digit:]][[:digit:]]?[[:space:]]*$' && return 0

  return 2
}
```

Wie Sie erkennen, erfordern scheinbar einfache Aufgaben bereits komplexe Lösungen. Der Aufwand rechtfertigt sich auf jeden Fall, wenn Sie in zahlreichen Skripten derartige Funktionen benötigen. Sammeln Sie solche Funktionen in einer eigenen Datei, so kann jedes Skript darauf zugreifen, indem es mittels **. Datei_Name** bzw. **source Datei_Name** diese "Bibliotheksdatei" einbindet.

Die präsentierten Lösungen sollten Ihnen die Fähigkeit verleihen, Funktionen zu schreiben, um jede beliebige Eingabe - ob per Kommandozeilenoption oder mittels Interaktion mit dem Benutzer - hinsichtlich ihrer Korrektheit zu testen. Nicht nur in der Shellprogrammierung sind unerwartete Eingaben die häufigste Ursache für einen unkontrollierten Programmablauf.

Parsen der Kommandozeilenargumente

Wie ein konkreter Wert auf Herz und Nieren geprüpft werden kann, sollte nun bekannt sein. Wenden wir uns nun den Möglichkeiten zu, die Argumente der Kommandozeile auszuwerten.

Nachfolgend bezeichnen wir als **Argument** jedes dem Kommando folgende Token auf der Kommandozeile. Von einer **Option** sprechen wir, wenn das Token durch ein führendes Minus eingeleitet wird; alle anderen Token sind **Parameter**.

```
# Kommandozeile
user@sonne> ./ mein_skript -c 10 40 -f eingabe -rt
# Kommando
user@sonne> ./ mein_skript -c 10 40 -f eingabe -rt
# Argumente
user@sonne> ./mein_skript -c 10 40 -f eingabe -rt
# Optionen
user@sonne> ./mein_skript -c 10 40 -f eingabe -rt
# Parameter
user@sonne> ./mein_skript -c 10 40 -f eingabe -rt
```

Uns interessiert, ob

- die korrekte Anzahl Argumente angegeben wurde
- die Anordnung der Argumente richtig ist (manche Option erwartet noch einen oder mehrere Parameter, bspw. einen Dateinamen)
- keine unbekannten Optionen angegeben wurden

- die Parameter den erwarteten Typ besitzen (Siehe: "Prüfen der Eingabe")

Anzahl der Argumente

Im Sprachgebrauch der Bash ist oft von Positionsparametern die Rede. Und die Anzahl jener ist in der Varia **$#** gespeichert.

In einem Skript könnte eine Überprüfung folgendermaßen realisiert werden:

```
if [ "$#" -lt "3" ]; then
   echo "Das Skript erwartet mindestens 3 Argumente"
   exit 1
fi

# Andere Schreibweise:
[ "$#" -lt "3" ] && { echo "Das Skript erwartet mindestens 3 Argumente"; exit 1; }
```

Ist eine konkrete Anzahl Argumente erforderlich, bietet sich eine andere Lösung an. Wir testen, ob das letzt erwartete Argument angegeben wurde (die Methode schützt nicht vor zu vielen Angaben):

```
Variable=${2:?"Anwendung: $0 Argument_1 Argument2"}
```

Die Anwendung ist zu lesen als: "Ist $2 (2. Argument) gesetzt, so weise den Wert an "Variable" zu. Sonst g die Meldung aus und beende das Skript". Die verwendete Substitution erklärt der Abschnitt Parameter- und Variablenexpansion der Bash.

Reihenfolge der Argumente und unbekannte Optionen

Beide Prüfungen lassen sich einfach in einem "Aufwasch" erledigen.

Im Zusammenhang mit den eingebauten Kommandos der Bash wurde die Anwendung von getopts anhand eines Beispiels erläutert. Ein Programmausschnitt, der die Optionen "-a", "-l", "-f Dateiname" und "-F" akzeptiert, sieht wie folgt aus:

```
while getopts alf:F Optionen; do
   case $Optionen in
      a) echo "Option a";;
      l) echo "Option l";;
      f) echo "Option f Argument ist $OPTARG";;
      F) echo "Option F";;
   esac
done
```

getopts kümmert sich selbst um unzulässige Optionen (keine Parameter!), indem es mit einer Fehlermeldu abbricht. Die akzeptierten Optionen folgen dem Kommando; ist einer Option ein Parameter zugeordnet, wir dies durch einen der Option nachgestellten Doppelpunkt ausgedrückt.

Unbefriedigend sind allerdings zwei Tatsachen. Zum einen ist das die von **getopts** generierte Fehlermeldur die wir vielleicht in Skripten vermeiden und durch eigene Ausgaben ersetzen möchten. Zum Zweiten ist das automatische Fortfahren mit der der getopts-Anweisung folgenden Zeile (im Beispiel gehts hinter "done" we meist unerwünscht.

Problem Nummer 1 verschieben wir schlicht und einfach nach /dev/null (Umleitung der Fehlerausgabe). Da Makel beseitigen wir, indem wir den Fall innerhalb der **case**-Anweisung berücksichtigen. Denn im Fehlerfall liefert **getopts** ein Fragezeichen ? als Resultat. Somit könnte ein Kommandozeilentest so ausschauen:

```
while getopts alf:F Optionen 2>/dev/null; do
   case $Optionen in
      a) echo "Option a";;
      l) echo "Option l";;
```

```
    f) echo "Option f Argument ist $OPTARG";;
    F) echo "Option F";;
    ?) echo "Unerwartete Option"
       # tue etwas...
       ;;
  esac
done
```

Schwachpunkte besitzt das Konzept noch immer. **getopts** vermag keine Parameter zu testen, sondern "nur Optionen in Verbindung mit einem folgenden Parameter. Dieser Parameter steht zum Zeitpunkt der Bearbeitung der betreffenen Option in **OPTARG**. **OPTIND** enthält allerdings schon den Index des nächsten Kommandozeilenarguments. Um eine Option mit zwei (oder mehr) Parametern mit **getopts** zu realisieren, man mit $OPTARG auf den ersten Parameter und mit "eval echo `echo $OPTIND`" auf den folgenden zugreifen. Allerdings sollte bei dem resultierenden Wert getestet werden, ob es sich tatsächlich um einen Parameter (und nicht etwa um eine Option) handelt. Auch lassen sich lange Optionen "--version" keinesfalls durch getopts behandeln.

Ohne Verwendung von getopts hangeln wir uns durch die Liste der Positionsparameter. Ein Beispielcode der dieselbe Funktionalität wie obiges getopts-Beispiel besitzt, lässt sich so notieren:

```
while [ "$#" -gt "0" ]; do
  case $1 in
    -a) echo "Option a"
        shift;;
    -l) echo "Option l"
        shift;;
    -f) echo "Option f Argument ist $2"
        shift 2;;
    -F) echo "Option F"
        shift;;
    *)  echo "Unerwartete Option"
        # tue etwas...
        ;;
  esac
done
```

Der Trick hierbei ist der Zugriff auf das jeweils erste Element der Positionsliste. Wurde dieses erfolgreich verarbeitet, so verschieben wir die Liste nach links (shift), sodass der ehemalige zweite Eintrag nun der erste ist. Eintrag 1 fällt einfach raus. Im Falle eines Parameters lesen wir die folgenden Listeneinträge aus und verschieben die Liste um die jeweilige Anzahl Elemente.

Bleibt noch anzuraten, Parameter stets auf den korrekten Typ hin zu überprüfen. Auch sollten Sie bei Skripten die Argumente erfordern, eingangs verifizieren, dass tatsächlich welche angegeben wurden, sonst wird die "while"-Schleife gar nicht erst betreten.

Interaktive Eingaben

In Bashskripten fordern Sie mit Hilfe des Kommandos read den Benutzer zu Eingaben auf. Im Abschnitt zur Bash wurde die Anwendung des Kommandos durch mehrere Beispiele untermauert, sodass der Einsatz in Skripten klar sein sollte.

read wartet, bis der Benutzer seine Eingabe mit einem Zeilenumbruch ([Enter]) abgeschlossen hat. Wird **read** kein Argument (Variablenname) mitgegeben, so wird die gelesene Zeile der Shellvariablen **REPLY** zugewiesen. Folgen dem Kommando ein oder mehrere Variablennamen, so wird der Reihe nach das erste Token der Eingabe der ersten Variable zugewiesen, das zweite Token der zweiten Variable usw. Stehen mehr Token als Variablen zur Verfügung, erhält die letzte Variable alle noch nicht zugewiesenen Token. Sind es weniger Token, bleiben die überschüssigen Variablen unbelegt.

Ein **read**-Aufruf blockiert, es sei denn mit der Option **-t Sekunden** (erst ab Bash 2.04!) wurde eine Zeitspanne vereinbart, nach der das Kommando spätestens zurück kehrt, selbst wenn noch keine Eingabe durch den Benutzer erfolgte. Anhand des Rückgabestatus kann entschieden werden, ob der Timeout die Rückkehr bedingte.

Der Schwerpunkt dieses Abschnitt liegt auf "korrekte Eingaben", d.h. uns interessiert, ob die mittels **read** eingelesenen Werte im "grünen" Bereich liegen. Wie dies geht, sollte nach Studium des weiter oben stehenden Textes bereits bekannt sein.

Was wir hier demonstrieren möchten, ist eine Variante zur Positionierung des Cursors, sodass nach einer fehlerhaften Eingabe diese vom Bildschirm verschwindet.

Zum Umgang mit den Eigenschaften es Terminals kann das Kommando tput recht nützlich sein. Es erlaubt u.a. das Löschen des gesamten Bildschirms, die Platzierung des Cursors oder die Ausgabe in fetter Schrift. Für unsere Zwecke sind folgende Aufrufe interessant:

tput bel	Alarmton ausgeben
tput clear	Löscht den Bildschirm
tput cols	Gibt die Anzahl Spalten des Terminals zurück
tput cup *x y*	Setzt den Cursor auf Zeile *x*, Spalte *y*
tput home	Entspricht "tput cup 0 0"
tput el	Zeile ab Cursorposition löschen
tput ed	Bildschirm ab Cursorposition löschen

Im folgenden Skriptfragment wird zur Eingabe einer Zahl aufgefordert. Solange die Eingabe nicht dem erwarteten Typ entspricht, springt der Cursor an die Eingabeposition zurück:

```
function writeln()
{
  tput cup $1 $2  # Cursor positionieren
  tput el         # Rest der Zeile löschen
  shift 2         # Parameter für die Koordinaten entfernen
  echo -n $*" "   # Rest der Parameterliste ausgeben
}

tput clear
writeln 3 5 "Beliebige Zahl eingeben:"

while :; do
  writeln 3 30
  read wert
  echo $wert | egrep -q '[[:digit:]]+' && break
  tput bel
done

echo "Eingabe war: $wert"
```

Möchten Sie die Eingaben am Bildschirm verbergen (Passworteingabe etc.), so hilft die Terminalsteuerung über das Kommando **stty** weiter. Der folgende Programmausschnitt schaltet die Anzeige der eingegebenen Zeichen ab:

```
echo -n "Passwort :"
stty -echo
read $password
stty echo
```

Bei der Anwendung von Passwortabfragen in Shellskripten sollten Sie niemals das unverschlüsselte Passwort im Skript speichern, da das Skript von jedem, der es ausführen darf, auch gelesen werden kann.

Rekursion

Vorab möchte ich deutlich heraus stellen, dass **Rekursion** kein Konzept der Bash, sondern eine allgemeine Programmiermethode darstellt.

Man bezeichnet eine Funktion als rekursiv, wenn sie sich selbst aufruft. Das mag verwirrend klingen, führt aber in zahlreichen Fällen zu besser strukturiertem - weil kürzerem - Kode. Aber Beispiele verdeutlichen ein Prinzip oft besser, als es die Sprache vermag - aus jenem Grund lassen wir Beispiele sprechen.

Arbeit im Verzeichnisbaum

Stünden Sie vor der Aufgabe, in jedem Verzeichnis unterhalb eines gegebenen Startverzeichnisses irgendwelche Maßnahmen vorzunehmen, so würden Sie vermutlich mittels **find** alle Verzeichnisse aufspüren und das Ergebnis einer Liste zuführen. In einer auf der Liste arbeitetenden Schleife würden Sie dann die Arbeiten für jedes einzelne Verzeichnis verrichten. Ihr Lösungsansatz wäre durchaus praktikabel und könnte wie folgt aussschauen:

```
user@sonne> cat doit_in_dir
#!/bin/sh

verzeichnis=${1:-./}

for i in `find $verzeichnis -type d`; do
  cd $i
  echo "Prozess $$ ist in `pwd`"
  cd -
done
```

Nun versuchen Sie einmal, das Skript einzig mit **ls** und den eingebauten Kommandos der Bash zu realisieren. Mit den so genannten iterativen Programmiermethoden wird die Größe Ihres Skripts um Einiges zunehmen. Per Rekursion hingegen schrumpft der Aufwand:

```
user@sonne> cat doit_in_dir_recursiv
#!/bin/sh

cd ${1:-./}

for i in `ls`; do
  test -d $i && $0 $i
done

echo "Prozess $$ ist in `pwd`"
```

Obiges Skript funktioniert nur, wenn es mit vollständigem Pfadnamen gestartet wird. An einer Erweiterung auf relative Pfadnamen kann sich der Leser versuchen; wir werden später im Text eine Lösung präsentieren.

Die Ausgabe wurde in den Beispielen bewusst um die Prozessnummer erweitert, um den Unterschied zwischen beiden Skripten zu kennzeichen. Die rekursive Variante generiert für jedes Verzeichnis einen eigenen Prozess, der die Bearbeitung für diesen Teil des Verzeichnisbaumes übernimmt. Da Systemressourcen allerdings nur begrenzt verfügbar sind, sollte die Rekursion über Programme nur bei "geringen" Rekursionstiefen angewandt werden. In der Hinsicht ist obiges Beispiel sicher schlecht gewählt. Aber bekanntlich heiligt der Zweck die Mittel.

Berechnung der Fakultät (rekursiv)

Eingebettet in den Hinweisen zur Fehlersuche in Skripten, wurden Sie bereits mit einer iterativen Lösung zur Berechnung der Fakultät konfrontiert. Hier zeigen wir Ihnen nun die rekursive Variante, die - sieht man einmal von den Variablendeklarationen ab - mit ganzen zwei Zeilen auskommt. Zum einen die Überprüfung, ob die Abbruchbedingung erreicht wurde und zum anderen ein Rechenschritt:

```
user@sonne> cat fac_rec_bad
#!/bin/sh

declare -i Resultat=${2:-1}
declare -i Fakultaet=$1

test "$Fakultaet" -le "1" && { echo "Ergebnis = $Resultat"; exit 0; }
```

```
$0 $Fakultaet-1 $Resultat*$Fakultaet
```

Lässt man sowohl die iterative als auch die rekursive Variante der Berechnung gegeneinander antreten, wird man schnell bemerken, dass letztere deutlich mehr Laufzeit beansprucht. Überdeutlich tritt der durch die Prozesserzeugungen notwendige Wasserkopf zu Tage. Deshalb gilt die Maxime: Vermeiden Sie, wann immer es möglich ist, die Rekursion auf Programmebene. Realisieren Sie Rekursion besser mit Funktionen:

```
user@sonne> cat fac_rec_good
#!/bin/sh

function factorial()
{
  local value=${1:=1}
  [ "$value" -le "1" ] && { echo 1; return; }
  echo $(($value * `factorial $value-1` ))
}

factorial $1
```

Weitere Anwendungen rekursiver Funktionen werden uns im Verlauf des Abschnitts noch begegnen.

Erwärmung

Beginnend mit kleinen Aufgabenstellungen versucht dieser Abschnitt, Ihnen einen effizienten Stil der Programmierung nahezulegen. Viele der Beispiele eignen sich durchaus zur Aufnahme in Bibliotheken, sodass zumeist die Realisierung als Funktion bevorzugt vorgestellt werden soll. Zum Ende hin schrauben wir den Anspruch allmählich nach oben, um den Grundstein für aufwändige Skriptprojekte zu legen.

Absolute Pfadnamen

Skripte, die Dateinamen übergeben bekommen, benötigen diesen häufig mit vollständiger Pfadangabe. Man könnte die Forderung auch an den Nutzer weiter leiten, aber mit den Mitteln der Bash ließe sich ebenso der absolute Name zu einer relativen Angabe ermitteln:

```
function GetAbsolutePathName()
{
  test -z "$1" && exit
  Pfad=${1%/*}
  Datei=${1##*/}

  echo `cd $Pfad 2>/dev/null && pwd || echo $Pfad`/$Datei
}
```

Eine Funktion in der Bash kann im eigentlichen Sinne keinen Funktionswert zurück liefern, deshalb schreibt sie das Ergebnis auf die Standardausgabe. Die Umleitung der Fehlerausgabe von **cd** ist notwendig, falls der Parameter einen nicht existierenden Pfad enthält. Somit gibt die Funktion im Fehlerfall den Eingangswert unverändert zurück.

Eine andere Realisierung ermöglicht die Verwendung der Kommandos **dirname** und **basename** zum Extrahieren von Datei- bzw Pfadnamen. Allerdings arbeitet die bashinterne Parametersubstitution durch Vermeidung neuer Prozesse schneller.

```
user@sonne> GetAbsolutePathName ../ user
/home/user
```

Zeichenkettenfunktionen

Obige Funktion zur Generierung des absoluten Dateinamens enthält eine Parametersubstitution. So kompliziert deren Anwendung auch ist, so wichtig ist sie für die Shellprogrammierung. Wann immer es etwas an Zeichenketten

zu manipulieren gibt, wird man auf diesen Mechanismus zurück greifen.

Einem C-Programmierer werden Funktionen wie **strcpy**, **strncpy**, **strcat**... bekannt vorkommen. Wir betrachten nachfolgend, wie solche Funktionen mit Mitteln der Bash simuliert werden können.

strcpy dient zum Kopieren von Zeichenketten. Als Parameter soll die Funktion zwei Variablennamen übergeben bekommen. Die erste Variable enthält später eine exakte Kopie des Inhalts der zweiten Variable. Eine Variablenzuweisung in eine Funktion zu packen, erscheint nicht sonderlich schwierig. Jedoch erweist sich die Aufgabe als erstaunlich verzwickt:

```
function strcpy()
{
  eval "$1"=\'"${!2}"\'
}
```

Vermutlich verstehen Sie jetzt gar nichts. Und genau das ist der Sinn der Übung. Zunächst sollte klar sein, dass auf die beiden Parameter in der Funktion über $1 und $2 zugegriffen werden kann. $1 enthält dabei *den Namen* der Variablen, der die Kopie zugewiesen werden soll und $2 ist *der Name* der Variablen mit der Quellzeichenkette.

Würde **eval** die Zeile nicht zieren, so expandiert der Ausdruck links des Gleichheitszeichens ($1) zum Namen der Variablen. Alles rechts expandiert zum Inhalt der durch $2 benannten Variablen. Diese Indirektion wird durch das Ausrufezeichen hervorgerufen. Nach erfolgter Expansion betrachtet die Bash das Resultat als den Namen eines Kommandos. Würde sie es nun starten, wäre ein Fehler die Konsequenz:

```
user@sonne> strcpy_buggy() { "$1"=\'"${!2}"\'}
user@sonne> var2=foo; strcpy_buggy var1 var2
bash: var1='foo': command not found
```

Erst **eval** sorgt für eine nochmalige Expansion durch die Bash. Diese führt nun die Variablenzuweisung aus und startet das resultierende Kommando. Da aus einer Zuweisung aber nichts resultiert, wird die Bash nichts tun...

strncpy kopiert nun maximal *n* Zeichen aus der Quelle ins Ziel. Die Funktion benötigt also als dritten Parameter die Anzahl maximal zu kopierender Zeichen.

```
function strncpy()
{
  local -i n=$3
  local str2=${!2}
  test "${#str2}" -gt "$n" && str2=${str2:0:$n};
  eval "$1"=\'"${str2}"\'
}
```

Die zwei Parameterexpansionen, die im obigen Beispiel hinzu kamen, sind ${#Variable} zum Auslesen der Anzahl in einer Variablen gespeicherten Zeichen und ${Variable:ab:Länge} zum Extrahieren einer Teilzeichenkette einer bestimmten Länge ab einer gegebenen Position.

Die letzte in diesem Zusammenhang präsentierte Zeichenkettenfunktion soll **strcat** sein, die zwei gegebene Zeichenketten miteinander verkettet und das Ergebnis in der ersten Variablen speichert:

```
function strcat()
{
  eval "$1"=\'"${!1}${!2}"\'
}
```

Ich hoffe mit den Beispielen ihr Verständis für die Zeichenkettenbearbeitung durch bashinterne Mittel ein wenig geschult zu haben. Versuchen Sie sich selbst an einer Funktion **strncat**, die maximal **n** Zeichen der Quelle an die Zielzeichenkette anfügt. Schreiben Sie Funktionen **strcmp** und **strncmp**, die "0" liefern, falls zwei Zeichenketten (in den ersten "n" Zeichen) übereinstimmen und "1" sonst. Einen Zacken anspruchsvoller sind Funktionen **strcasecmp** und **strncasecmp** zu schreiben, die die Groß- und Kleinschreibung nicht berücksichtigen (Tipp: Die Shelloption **nocaseglob** kann auch innerhalb einer Funktion (de)aktiviert werden.).

Mathematische Funktionen

Sie sollten niemals versuchen, in umfangreichen Skripten die arithmetischen Substitutionen der Bash als Taschenrechner zu missbrauchen. Dass wir es hier dennoch tun, dient einzig der Lehre, da mathematische Algorithmen häufig mittels Schleifen oder durch rekursive Funktionen implementiert werden können.

Beginnen wir mit der **Potenzfunktion** zur Berechnung von x^y. Ohne ein externes Kommando zu bemühen, bietet sich in der Bash eine Schleife an:

```
function power()
{
  local -i x=${!1}
  local -i result=$x

  for ((i=${!2}; i>1; i--)); do
    result=$result*$x
  done
  echo $result;
}
```

Die Anwendung obiger Funktion geschieht wie folgt:

```
user@sonne> x=2; y=10
user@sonne> power x y
1024
```

Beachten Sie, dass obige Realisierung nur in der Bash ab Version 2.04 korrekt arbeitet; passen Sie sie ggf. auf Ihr System an!

Dieselbe Funktion in der rekursiven Variante könnte wie folgt geschrieben werden:

```
function power_rec()
{
  local -i x=${!1}
  local -i y=${!y}-1

  test $y = 0 && { echo $x; return; }
  echo $(($x * `power_rec x y`))
}
```

Andere Algorithmen wie die Berechnung von Fibonacci-Zahlen oder der binomischen Koeffizienten (Pascal'sches Dreieck) lassen sich nach ähnlichem Schema modellieren, solange sie auf ganzen Zahlen operieren.

Was eher eine Bereicherung des Funktionsumfangs der Shell wäre, sind Funktionen, die bestimmte Zahlenformate in andere überführen. Konkret wollen wir eine Funktion entwickeln, die zu einem gegebenen dezimalen Wert die zugehörige Binärdarstellung liefert.

Der Algorithmus klingt in der Theorie recht einfach: "Dividiere die Zahl ganzzahlig sooft durch 2, bis sie kleiner als 1 wird. Ergibt die Division durch 2 in einem Schritt einen Rest, so schreibe eine '1', sonst schreibe eine '0'. Die sich ergebende Zeichenkette ist - rückwärts gelesen - die gesuchte Binärdarstellung."

Um die Funktion übersichtlich zu halten, lagern wir die Entscheidung, ob eine Zahl (un)gerade ist in eine Funktion **is_even()** aus:

```
function is_even()
{
  local -i var=${!1}/2
  if [ "${!1}" -eq "$(($var*2))" ]; then
    echo "0"
  else
```

```
    echo "1"
  fi
}
```

In einer "echten" Programmiersprache hätte eine einfache Bitoperation als Test genügt; in der Bash ist der Aufwand ungleich höher. Die rekursive Lösung der gesuchten Funktion kann damit wie folgt geschrieben werden:

```
function dec2bin_recursiv()
{
  local -i var=${!1}
  test "$var1" -le "1" && { echo $var1; return; }
  local -i var2=$(($var1/2))
  echo "`dec2bin var2``is_even var1`"
}
```

Die Zeichenkette der Binärdarstellung baut sich bei obiger Anordnung quasi "von hinten" auf.

Die iterative Lösung derselben Aufgabe erfordert den Zugriff auf eine weitere Hilfsfunktion, die die Zeichenkette zunächst in "gedrehter" Reihenfolge generiert:

```
function dec2bin_reverse()
{
  local -i var=${!1}
  for ((i=${!1}; i>0; i=i/2)); do
    echo -n `is_even i`    done
}
```

Die "Hauptfunktion" realisiert einzig das Drehen der Zeichenkette:

```
function dec2bin_iterative()
{
  local -i var=`dec2bin_reverse ${!1}`
  for ((i=${#var}; i>0; i--)); do
    echo -n ${var:$(($i-1)):1}
  done
  echo ""
}
```

Der abschließende **echo**-Aufruf dient einzig dem Einfügen des Zeilenumbruchs.

Die vorgestellten Beispiele demonstrieren zum einen die erstaunlichen Möglichkeiten des Rechnens mit reinen Bash-Werkzeugen; aber auch die gewöhnungsbedürftige Syntax. Versuchen Sie die Lösungen nachzuvollziehen. Eine gesetzte Shelloption xtrace (set -x) kann in vielen Fällen den internen Lauf der Skripte veranschaulichen.

Dateien bearbeiten

Mit Werkzeugen wie awk oder sed stehen mächtige Hilfsmittel zur Verfügung, um Textdateien zu durchsuchen und zu bearbeiten. Beide Programme manipulieren allerdings nicht die Datei selbst, sondern schreiben das Ergebnis auf die Standardausgabe.

Der naive Ansatz, dem mit der Ausgabeumleitung zuvorzukommen, scheitert:

```
sed 's/    / /g' datei > datei
```

Warum? Wenn Sie keine schlüssige Erklärung parat haben, dann lesen Sie nochmals nach, wie und wann die Bash die Ein- und Ausgabeumleitungen bei einfachen Kommandos organisiert.

Im Skript lässt sich schnell Abhilfe finden:

```
...
sed 's/  / /g' datei > /tmp/tmp.$$
mv /tmp/tmp.$$ datei
...
```

Damit gelingt aber immer noch kein »wahlfreier« Zugriff auf den Inhalt der Datei. Es gibt genügend Situationen, in denen man »auf die Zeile davor« zugreifen muss, weil gerade die folgende Zeile bestimmten Bedingungen genügte.

Aber auch hierfür lassen sich Lösungen finden, indem eine Datei »in den Speicher« gelesen wird und die eigentliche Arbeit dann auf diesem Speicher erfolgt.

```
function readfile()
{
  local File=$1
  local SaveTo=$2
  local -i counter=0

  while read line; do
    eval "$2[$counter]"=\'$line\'
    counter=$(($counter+1))
  done < $File

  eval "$3"=$counter
}
```

Die Funktion **readfile()** wird mit dem Dateinamen als ersten und der Variable, die die Datei aufnehmen soll als zweitem Parameter gerufen. Als »Rückgabewert« liefert sie im dritten Argument die Anzahl gelesener Zeilen. Dies ist wichtig, da zu einer Feldvariablen nicht entschieden werden kann, wie viele Elemente sie enthält.

Auf die einzelnen Zeilen der Datei können Sie nun mittels Variable[Zeilenummer] zugreifen und irgendwann, wenn die Bearbeitung abgeschlossen ist, möchten Sie die Daten auch wieder zurück schreiben. Hierzu verhilft Ihnen die Funktion **writefile()**, die als Argumente die Variable mit den Zeilen sowie die Anzahl der Zeilen übergeben bekommt:

```
function writefile()
{
  local Array=$1

  for ((i=0; i<${!2}; i++)); do
    eval echo "\${$Array[$i]}"
  done
}
```

Angenommen, obige Funktionen stünden in einer Datei mit dem Namen "filehandling.sh", dann könnte in einem Programm so vorgegangen werden:

```
#!/bin/sh
# Laden der "Bibliotheksdatei"
. filehandling.sh

# Datei einlesen
readfile zu_bearbeitende_Datei FeldVariable Anzahl

# Inhalt bearbeiten
for ((i=0; i<$Anzahl; i++)); do
  # Bearbeiten von FeldVariable[$i]...
done

# Datei schreiben
writefile FeldVariable Anzahl > zu_bearbeitende_Datei
```

Der Begriff des Embedded Linux ist Ihnen sicher schon einmal zu Ohren gekommen. Es geht darum, die Steuerung technischer Geräte, wie Radios, Uhren, Waschmaschinen, Handys... einem Prozessor zu überlassen. Auf so einem Prozessor werkelt natürlich ein Betriebssystem und neuerdings vermehrt auch Linux. Aus Kostengründen werden sowohl für Prozessor als auch für die Peripherie nur Bauelemente mit arg begrenzten Ressourcen eingesetzt. Oft müssen Betriebssystem und "Betriebssoftware" sich in magere 2 MB FlashROM teilen. Wenn Sie jetzt an Ihre Linux-Installation denken, so fragen Sie sich vielleicht, wie Linux soweit abgespeckt werden kann?

Selbst bei Verzicht auf "unnütze" Programme summieren sich die Dienstprogramme auf mehr als 1 Megabyte. Der Kernel lässt sich auf unter 300 kByte drücken, aber damit erschöpft sich das Sparpotenzial fürs erste.

Auf den Einsatz in embedded Systemen zugeschnittene Linux-Versionen erfahren zumeist eine grundlegende Überarbeitung der Quellen. Der Kernel und jedes für das Zielsystem vorgesehene Programm werden somit um nicht benötigte Funktionalität entlastet; Kernel in Größenordnung von 200 kByte inklusive Netzwerkunterstützung werden so erzielt.

Aber uns beschäftigt hier das Thema der Shellprogrammierung und wir möchten anhand einiger Beispiele demonstrieren, wie sich zwingend notwendige Dienstprogramme mit Mitteln der Bash simulieren lassen. Natürlich erfordert dies eine Beschränkung der Optionen, über die das originale Programm verfügt. Aber die meisten davon wendet der Durchschnitts-Benutzer ohnehin niemals an.

Beginnen wir mit **head**, das in der dynamisch gelinkten Programmvariante immerhin mit 12 kByte zu Buche schlägt. Von den Optionen wird wohl ohnhin nur "-n" in Frage kommen, um die Standardvorgabe von 10 darzustellenden Zeilen abzuändern.

```
#!/bin/sh
# Erweiterte Expansion einschalten
shopt -s extglob
declare -i ANZAHL=10
declare -i DATEIZAEHLER=0
declare -i ZEILENNUMMER=0
Datei=

while [ "$#" -gt "0" ]; do
  case "$1" in
    -n ) # Option: "-n Zeilen"
      ANZAHL=$2
      [ $ANZAHL = 0 ] && exit 0
      shift 2
      ;;
    -+([0-9]) ) # Option: "-Anzahl"
      ANZAHL=${1:1}
      [ $ANZAHL = 0 ] && exit 0
      shift
      ;;
    * ) # Andere Angaben sind Dateinamen (Test auf Existenz!)
      [ -e "$1" ] || { echo "$0: $1: Datei oder Verzeichnis nicht gefunden"; exit 1; }
      Datei[$DATEIZAEHLER]="$1"
      DATEIZAEHLER=$DATEIZAEHLER+1
      shift
      ;;
  esac
done

if [ $DATEIZAEHLER = 0 ]; then
  # Lesen von der Standardeingabe
  while read line; do
  echo "$line"
    ZEILENNUMMER=$(($ZEILENNUMMER+1))
    test $ZEILENNUMMER = $ANZAHL && break
  done
else
  for ((i = 0; i < $DATEIZAEHLER; i++)); do
    # Für jede Datei beginnt die Zählung von vorn
    ZEILENNUMMER=0
    # Formatierung der Ausgabe gemäß des Originals
```

```
    test "$i" -gt "0" && echo ""
    test "$DATEI ZAEHLER" -gt "1" && echo "==> ${Datei[$i]} <=="
    while read line; do
    echo "$line"
      ZEI LENNUMMER= $((ZEI LENNUMMER+ 1))
      test $ZEI LENNUMMER = $ANZAHL && break
    done < ${Datei[$i]}
    i= $((i+ 1))
  done
fi
```

head in Bashfassung arbeitet damit fast exakt wie das originale Kommando, belegt aber gerade mal etwas mehr als 1 KByte Speicherplatz (und der ließe sich durch kürzere Variablennamen weiter reduzieren).

Zur Realisierung des Kommandos **tail** beschränken wir uns auf eine Diskussion der Vorgehensweisen. Was die Auswertung der Kommandozeilenoptionen betrifft, kann das obige Skript "head" als Vorlage dienen. Einzig um die Option "-f" ist die while-Schleife zu erweitern. Um nun die letzten "n"-Zeilen anzusprechen, bieten sich an:

- Die Umkehr der Eingabedateien mittels **tac** und weiteres Vorgehen analog zum head-Skript.
- Das Auswerten der Zeilenanzahl in den Dateien mittels **wc -l**. Anschließend werden alle Zeilen eingelesen, aber nur die letzten "n" ausgegeben.
- Das Einlesen aller Zeilen, wobei die Zeilen der Reihe nach in ein Feld mit "n" Elementen gespeichert werden. Wurde das "n"-te Element geschrieben, wird der Index auf 0 zurück gesetzt und somit das "n-1"-te Element überschrieben (Ringpuffer). Nach Einlesen der gesamten Datei wird die Ausgabe mit Feldelement nach dem zuletzt geschriebenen begonnen:

```
...
# Einlesen in ein Feld
while read line; do
  FELD[ ZEI LENNUMMER] = $line
  ZEI LENNUMMER= $((ZEI LENNUMMER+ 1))
  test $ZEI LENNUMMER = $ANZAHL && ZEI LENNUMMER= 0
done < $Datei

# Ausgabe
declare -i ZAEHLER= $ANZAHL
while [ $ZAEHLER -gt 0 ]; do
  echo ${FELD[$ZEI LENNUMMER]}
  ZEI LENNUMMER= $((ZEI LENNUMMER+ 1))
  test $ZEI LENNUMMER = $ANZAHL && ZEI LENNUMMER= 0
  ZAEHLER= $((ZAEHLER-1))
done
```

Von den skizzierten Lösungen bevorzuge ich die 3., da sie vollends auf Mitteln der Bash basiert.

Ein Problem wird sich dem Leser nun stellen? Wie kann man das Verhalten von **tail -f** (also die permanente Ausgabe neu hinzukommender Zeilen) realisieren?

Kein Problem! Wird eine Datei fest mit einem Dateideskriptor verbunden, so führt jeder Lesevorgang auf dem Deskriptor (read) zum Verschieben des Zeigers um eine Zeile. Wurde das Dateiende erreicht, kehrt der Leseversuch unverzüglich zurück. Ein erneuter **read**-Aufruf liest die nächste Zeile aus - sofern sie existiert. Gibt es keine neue Zeile, wird einfach nichts gelesen...

Das nächste Programmfragment demonstriert die Anwendung des Verfahrens:

```
# Abfangen der Signale 2 und 15, Schließen des Deskriptors 3
trap 'exec 3<&-' 2 15

# Die Datei wird mit dem Eingabedeskriptor 3 verbunden:
exec 3< Beispiel_Datei

# Endlosschleife
while:; do
```

```
  while read line; do
    # Ausgabe der Zeile
    echo $line
  done
  # 1 Sekunde schlafen
  sleep 1
done
```

Obige endlos-while-Schleife kann nur durch ein entsprechendes Signal (meist [Ctrl]-[C]) beendet werden. In dem Fall sollte unbedingt der Dateideskriptor geschlossen werden, was durch Fangen des Signals realisiert wird.

Versuchen Sie sich selbst an der Vollendung des tail-Skripts!

Neben **echo** erweist sich das Kommando **dd** als nahezu unentbehrlich. **dd** kopiert wahlweise die Standardein- auf die Standardausgabe, die Standardeingabe in Dateien, den Dateiinhalt auf die Standardausgabe usw. In Shellskripten kann es damit als Basis für die Simulation zahlreicher anderer Kommandos herhalten.

Da wäre zunächst **cat**, das sich durch einige wenige Skriptzeilen (wir lassen die Optionen von **cat** einmal außen vor) ersetzen lässt:

```
#!/bin/sh
exec 2>&-

if [ $# = 0 ]; then
  dd
else
  for i do
    dd < $i
  done
fi
```

Das Kommando **dd** fabriziert auf der Standardfehlerausgabe eine abschließende Statusmeldung über die Anzahl kopierter Daten. Um diese zu unterdrücken, wird zu Beginn des Skripts die Standardfehlerausgabe geschlossen.

Ein vereinfachtes **cp** ist mittels Bash und dd so zu ersetzen:

```
#!/bin/sh
test -e "$1" || { echo "Quelldatei nicht gefunden"; exit 1; }
test -z "$2" && { echo "Aufruf: $0 <Quelldatei> <Zieldatei>"; exit 2; }

dd if="$1" of="$2"
```

Einen entscheidenden Mangel weist obiges Skript dennoch auf: Die Rechte bleiben nicht erhalten. Um diese zu setzen, bleibt wohl nur der Griff zu chmod übrig. Steht noch die Frage des Auslesens der Rechte der Originaldatei... Unter der Annahme, dass in "embedded Linux"-Varianten ohnehin keine komplexe Benutzerverwaltung sinnvoll ist, können wir uns auf den Erhalt der Schreib- und Ausführungsrechte für den Eigentümer beschränken. Und diese ist per **test** ermittelbar:

```
#!/bin/sh
function calc_rights()
{
  mode='u=r'    # Ohne Leserecht wird schon das Kopieren scheitern
  test -w $1 && mode=${mode}w
  test -x $1 && mode=${mode}x
  echo $mode
}

test -e "$1" || { echo "Quelldatei nicht gefunden"; exit 1; }
test -z "$2" && { echo "Aufruf: $0 <Quelldatei> <Zieldatei>"; exit 2; }

dd if="$1" of="$2"

chmod `calc_rights $1` $2
```

Zuletzt noch eine Bash-Implementierung des Kommandos **ls**. Bei Verzicht auf sämtliche Optionen genügen wenige Kodezeilen:

```
function ls()
{
  set -- *

  for i do
    echo $i
  done
}
```

Ordnung im Dateisystem

Wer sich der Programmierung widmet oder selbst Hand an die Kompilierung von Quellpaketen oder des Kernels legt, der hat bereits Bekanntschaft geschlossen mit Objekt-Dateien. Diese Dateien werden während der Übersetzung von Programmen generiert und sind nach Erzeugung des Binaries oder einer Bibliothek nicht mehr notwendig.

Andere Dateien, denen Sie hin und wieder im Dateisystem begegnen, nennen sich **core**. Dabei handelt es sich um Speicherauszüge (RAM) von abgestürzten Programmen. Ein Experte könnte anhand solcher Dateien der Fehlerursache auf den Grund gehen ("debuggen"); die meisten Anwender werden aber damit nichts anfangen können.

Die dritte Dateiart, von der wir mit dem nachfolgenden Skript unser Dateisystem bereinigen wollen, endet mit der Tilde (*.~). Zahlreiche Programme kennzeichen so ihre Sicherungskopien; nicht selten wird ihr Dasein bei Programmende schlicht ignoriert.

Wünschenswert wäre ein Skript, das, per Kommandozeilenoptionen ("-c" für Core-, "-o" für Objekt und "-t" für "+~"-Dateien) gesteuert, die entsprechenden Dateien im System ausfindig macht und entfernt. Da Löschen unter Linux gleichbedeutend mit dem unwiderruflichen Datenverlust ist, wäre eine Nachfrage vor dem Löschvorgang nützlich. Auch hierzu soll das Skript eine Option ("-i") kennen.

Zunächst das Skript:

```
user@sonne> cat delete_trash
#!/bin/sh
object=
tilde=
core=
inter=

for i do
  case "$i" in
    -c ) core=core;;
    -o ) object=\*.o;;
    -t ) tilde=\*\~;;
    -i ) inter=-i;;
    *) echo "Unbekannte Option"
      exit 1;;
  esac
done

for i in $core $object $tilde; do
  for x in $(find $(pwd) -name "$i"); do
    rm $inter $x
  done
done
```

Das Skript verwendet einige Kodezeilen, die typisch für einen sauberen Stil sind:

```
object=
```

```
...
```

Die Initialisierung mit "Nichts" garantiert, dass eventuell in der Shell vorhandene globale Variablen gleichen Namens keinen Eingang in das Skript finden.

```
...
-o ) object=\*.o;;
...
```

Das anschließende Parsen der Kommandozeile sollte verständlich sein. Allerdings belegen wir die Variablen sofort mit den notwendigen Suchmustern. Der Sinn wird in den folgenden Anweisungen klar:

```
for i in $core $object $tilde; do
  for x in $(find $(pwd) -name "$i"); do
    rm $inter $x
  done
done
```

Überlegen Sie, zu welchen Werten die Liste der äußeren **for**-Schleife expandiert. Angenommen, alle drei Optionen waren gesetzt, so ergibt sich:

```
for i in "core" "*.o" "*~"; do
```

Die Schleifenvariable **i** wird demnach der Reihe nach mit den Werten "core", "*.o" und "*~" belegt. Das **find** der inneren **for**-Schleife sucht demnach zuerst nach Dateien mit dem Namen "core", im zweiten Duchlauf werden auf .o endende Dateien gesucht...

Schließlich expandiert die Zeile mit dem Kommando **rm** zu

```
rm -i Dateiname
```

oder zu

```
rm Dateiname
```

je nachdem, ob "-i" als Kommandozeilenoption übergeben wurde oder nicht.

Komplexe Anwendungen

Symbolsuche

Im Allgemeinen werden Computerprogramme nicht in all ihrer Funktionalität neu geschrieben, sondern sie greifen zu nicht unbedeutenden Teilen auf bestehende Bausteine zurück. Solche Bausteine (»Objekt-Kode«) wiederum liegen als Sammlungen in Bibliotheken vor.

Ausführbare Programme lassen sich in zwei Kategorien gliedern. Zum einen die **statischen Programme**, denen bereits zur Übersetzungszeit alle notwendigen Bausteine aus den Bibliotheken »einverleibt« werden. Die zweite Ausprägung sind die **dynamischen** Programme (genauer: »dynamisch gelinkte Programme«), die nur die Schnittstellen zum Aufruf solcher Bausteine enthalten; nicht aber deren Implementierung.

Die Dateigröße dynamisch gelinkter Programme fällt somit um ein Vielfaches geringer aus, als die statischer. Allerdings erfordert ihre Ausführung das Vorhandensein der Bibliothek, die die **Symbole** - ein Baustein kann beliebig viele Symbole definieren oder verwenden - enthält. Und hier fangen die Probleme oftmals an, nämlich dann, falls ein benötigtes Symbol nicht gefunden werden kann:

```
# Ursache: Bibliothek wurde nicht gefunden
```

```
user@sonne> / opt/ kde2/ bin/ ksplash
/opt/kde2/bin/ksplash: error in loading shared libraries: libjpeg.so.62: cannot open shared object file: No such file or directory

# Ursache: Symbol wurde nicht gefunden
user@sonne> / opt/ kde2/ bin/ ksplash
/opt/kde2/bin/ksplash: error in loading shared libraries: /opt/kde2/lib/libkdeui.so.3: undefined symbol: __ti10KDNDWidget
```

Die erste Ursache, dass eine solche Bibliothek nicht geöffnet werden konnte, muss nicht zwangsläufig bedeuten, dass diese im System nicht doch existiert. Sie konnte eventuell nur nicht durch den »Dynamischen Lader«, den **ldd**, gefunden werden. Hieraus ergibt sich eine erste Anforderung an unser Skript: Es soll nach einer Bibliothek suchen und - falls die Suche positiv verlief - Maßnahmen treffen, um den Fehler zu beheben. Diese sind:

- Konfiguration des **ldd**, sodass er die Bibliothek automatisch findet (erfordert Root-Rechte)
- Setzen der Variablen **LD_LIBRARY_PATH** (darf jeder) in der Datei ».profile«

Komplizierter ist das Vorgehen bei einem vermissten Symbol. Hier gilt es, die Bibliothek zu finden, in der das Symbol **definiert** wird. Das Kommando, mit dem Informationen aus einer Bibliothek (oder auch einer Objekt-Datei) ausgelesen werden können, ist **nm**. Um einen Eindruck vom Aufbau einer Bibliothek zu gewinnen, betrachten wir einen Ausschnitt einer typischen Ausgabe von **nm**:

```
# stark gekürzt...
user@sonne> nm / usr/ lib/ libc.a
...
printf.o:
00000020 t Letext
00000000 T _IO_printf
00000000 T printf
         U stdout
         U vfprintf
...
```

Die für unsere Zwecke interessanten Einträge sind die mit **T** bzw. **U** markierten Zeilen. Erste bezeichnen Symbole, die der Baustein (»printf.o«) definiert, letztere sind verwendete Symbole, die ein anderer Baustein bereit stellen muss. Sie sind an dieser Stelle **undefiniert**. Es gilt nun, zu einem gegebenem Symbol die Bibliothek zu suchen, die dieses definiert. Wird eine solche gefunden, treffen wir dieselben Maßnahmen wie oben beschrieben. Bei negativem Suchergebnis hilft wohl nur die Konsultation einer Suchmaschine im Internet...

Als Anforderungen an unser Skript notieren wir:

- Es soll nach Bibliotheksnamen **-n Name** suchen (mit Metazeichen)
- Es soll nach der Bibliothek suchen, die ein Symbol **-s Symbol** definiert
- Ein Startverzeichnis **-d Verzeichnis** für die Suche kann angegeben werden

Als Aktionen sind möglich:

- Bei erfolgreicher Suche kann der **ldd** konfiguriert werden **-l**
- Bei erfolgreicher Suche kann die Variable **LD_LIBRARY_PATH** exportiert werden **-x** (der Eintrag erfolgt in .profile)

Bevor Sie sich das Skript zu Gemüte führen noch eine Anmerkung. Der dynamische Lader **ldd** durchsucht in der Voreinstellung immer die Verzeichnisse /lib und /usr/lib, sodass diese weder in /etc/ld.so.conf noch in LD_LIBRARY_PATH angegeben werden sollten. Nachfolgende Skriptlösung verhindert allerdings den Versuch, solche Verzeichnisse einzubinden, nicht, weshalb Sie die Optionen -l und -x erst nach einem erfolgreichen Durchlauf des Skripts verwenden sollten.

```
#!/bin/sh
# Einige Funktionen...

function GetAbsolutePathName()
{
  test -z "$1" && exit
  Pfad=${1%/*}
```

```
   Datei=${1##*/}
   echo `cd $Pfad 2>/dev/null && pwd || echo $Pfad`/$Datei
}

function find_symbol_in_lib() {
   file $1 | fgrep -q 'shared object' || return 1
   file $1 | fgrep -q 'not stripped' || return 1
   nm $1 | fgrep -qw "T $2"
}

function Config_LD_LIBRARY_PATH() {
   . ~/.profile
   temp=`echo $LD_LIBRARY_PATH | awk 'BEGIN {RS=":"} $0 ~ suche {print $0}' suche="^$1/?[[:space:]]\*$"`
   test -z $temp || return 0   # schon enthalten => nichts zu tun
   temp="export LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:$1"
   sed '/.*LD_LIBRARY_PATH.*/d' ~/.profile > ~/.profile.$$
   echo $temp >> ~/.profile.$$
   mv ~/.profile.$$ ~/.profile
}

function Config_Ldd() {
   test "$UID" != "0" && { echo "Option '-l' erfordert Rootrechte"; }
   cat /etc/ld.so.conf | grep -q ^[[:space:]]*$1/*[[:space:]]*$
   test "$?" == "0" && return 0;   # Pfad bereits enthalten
   cat $1 >> /etc/ld.so.conf
   /sbin/ldconfig
}

# Das »Hauptprogramm«...

Libraries='lib*'
Symbol=
StartDir=./
ConfigLdd=
SetLibraryPath=
Path=

while getopts n:s:d:lx Optionen; do
   case $Optionen in
      n) Libraries=$OPTARG;;
      s) Symbol=$OPTARG;;
      d) StartDir=$OPTARG;;
      l) ConfigLdd=1;;
      x) SetLibraryPath=1;;
      *) { echo "$Optionen: Unbekanntes Argument"; exit 1; }
   esac
done

test -z $Symbol && { echo "Option: '-s Symbol' erforderlich"; exit 1; }

test -d $StartDir || { echo "Startverzeichnis '$StartDir' nicht gefunden"; exit 1; }

for i in `find $StartDir -name "$Libraries" 2>/dev/null`; do
   find_symbol_in_lib $i $Symbol || continue
   echo "Symbol definiert in $i"
   Path=$(GetAbsolutePathName `dirname $i`/)
done

test -z $Path && { echo "Symbol wurde nicht gefunden"; exit 0; }

test -z "$SetLibraryPath" || Config_LD_LIBRARY_PATH $Path
test -z "$ConfigLdd" || Config_Ldd $Path
```

Das Skript ist sicherlich nicht leicht zu verstehen, aber auf die bereits im obigen Text ausgiebig erwähnten Konstrukte möchte ich hier nicht erneut eingehen. Ich beschränke mich daher auf die Arbeitsweise der enthaltenen neuen Funktionen.

find_symbol_in_lib

Config_LD_LIBRARY_PATH

Als einziges Argument erhält die Funktion den Pfad, indem die Bibliothek gefunden wurde. Nun ist es denkb dass dieser Pfad bereits in der Variablen »LD_LIBRARY_PATH« enthalten ist. Um sicher zu stellen, dass auc etwaige Konfigurationen aus ».profile« in den nachfolgenden Schritten berücksichtigt werden, wird diese Da nochmals eingelesen.

Den Inhalt von »LD_LIBRARY_PATH« durchsuchen wir nach dem Pfad. Problematisch ist, dass das Suchmu (Pfad) in einer Variable steht und dieses auch noch Sonderzeichen der Bash enthält (der Slash). Noch dazu kann dieser Pfad Bestandteil eines anderen Pfades aus »LD_LIBRARY_PATH« sein. Erst der Zeichenkettenvergleich und die Möglichkeit der Angabe eines Zeilentrennzeichens von Awk gestatten die Interpretation des Musters, ohne dass die Substitutionsmechanismen der Bash in die Quere kommen.

Ist der Pfad vorhanden, ist nichts zu tun (Er muss nicht in .profile gesetzt worden sein!).

Im anderen Fall bemühen wir den Stream Editor zum Entfernen des alten LD_LIBRARY_PATH-Eintrag aus d Datei ».profile« und ergänzen die neue Version.

Config_Ldd

In der ersten Zeile testen wir, ob das Skript als Root gestartet wurde. Falls nicht, fehlen für die weiteren Schritte die notwendigen Berechtigungen und das Skript endet.

Da die einzelnen Pfadangaben in der Datei /etc/ld.so.conf auf separaten Zeilen stehen, eignet sich grep zur Suche nach dem Muster. Ist der Pfad bereits enthalten, beenden wir die Funktion.

Im anderen Fall wird der Pfad ans Ende der Datei angefügt und **ldconfig** gestartet, damit die von **ldd** verwendete Cache-Datei (ls.so.cache) neu erzeugt wird.

Linuxfibel-Druckversion

Immer wieder werde ich gefragt, warum ich keine Druckversion der Linuxfibel anbiete? Mit Bedauern gebe ich dann zu verstehen, dass ich derzeit eher Wert auf Inhalt als auf Präsentation lege und meine Zeit eh schon zu knapp bemessen ist, um ernsthaft die Vollendung der Linuxfibel voran zu treiben. Und außerdem könnte ein kleines Skript den störenden Navigationsrahmen entfernen und somit eine temporäre brauchbare Lösung schaffen.

Wer sich die Mühe macht und einige html-Quellen der Fibel betrachtet, wird bald die identische Struktur erkennen, die das Skript sich zu nutze macht, um den druckverhindernden Ballast zu entsorgen. Um gleich noch einen Lehreffekt zu erzielen, verwendet das Skript Awk für die wesentlichen Schritte. Die Awk-Programmdatei wird wiederum temporär erzeugt. Hier zunächst das Skript:

```
user@sonne> cat printversion.sh
#!/bin/sh

linuxfibel_base=${1:-./}
awk_script=/tmp/`basename $0`.$$

trap 'test -e $awk_script && rm $awk_script' 2 15
```

```
test -d $linuxfibel_base || { echo "Verzeichnis $linuxfibel_base existiert nicht"; exit 1; }

 cd $linuxfibel_base
 test -e vorwort.htm           || { echo "Falsches Linuxfibel-Verzeichnis?"; exit 1; }
 test -d printversion          || mkdir printversion
 test -L printversion/images || (cd printversion && ln -s ../images)

cat > $awk_script << EOF
# --------- AWK-SCRIPT BEGINN ----------
#!/usr/bin/awk -f

BEGIN {
    DoPrint="true"
    IGNORECASE=1
}

-->
 /<script language="JavaScript">/        { DoPrint = "false" }
 /<\/head>/                              { DoPrint = "true" }
 /<body bgcolor/                         { print \$0; DoPrint = "false" }
 { getline; print \$0; DoPrint = "false" }
```

Die C-Shell und Tcsh

Übersicht

Was ist die Tenex-C-Shell?

Verbreitung erfuhr die C-Shell vor allem durch ihre enge Bündelung mit den BSD-Unix-Systemen. In ihre Entwicklung flossen all die nützlichen Dinge der Bourne Shell ein, garniert mit einigen neuen Eigenschaften. Herausragend war sicher die Verwendung einer History. Die C-Shell erbte von der Bourne Shell zwar das Konzept, die Realisierung orientierte sich jedoch an der Programmiersprache C. Daher auch ihr Name.

Im interaktiven Umgang ist von der Nähe zur Programmiersprache C nichts zu merken. Erst der Shellprogrammierer wird mit der Syntax konfrontiert, die vor allem dem Programmierer entgegen kommen wird.

Im Vergleich zur Bourne Shell agiert die C-Shell recht schwerfällig. Auf heutigen Megahertz-Boliden mag das kein Kriterium Für oder Wider einer Shell zu sein, aber in den Anfängen von Linux verfügten die Rechner über weitaus weniger Leistung. Vielleicht ist hier ein Grund für die geringe Akzeptanz der (Tenex-)C-Shell unter Linux zu suchen.

Während die C-Shell einem kommerziellen Lizenzmodell unterliegt, ist die Weiterentwicklung in Form der Tenex-C-Shell (Tcsh) frei und damit unter Linux verfügbar. Im Wesentlichen unterscheidet sich die Tcsh von ihrem Ahnen durch die verschiedenen Mechanismen zur Vervollständigung nach so genanntem Tenex-Stil.

Aber nicht nur dem C-Programmierer kommt die C-Shell entgegen. Auch wer in der Solaris-Ecke heimisch ist und nur hin und wieder mit Linux in Berührung kommt, muss die vertraute Shell nicht missen.

Csh vs. Tcsh

Nur kurz sollen die wesentlichen Unterschiede zwischen den beiden Shells aufgezählt werden. Die Tenex-C-Shell erweitert die Fähigkeiten der C-Shell um:

- Editieren von Kommandozeilen (auch nach VI- oder Emacs-Stil)
- Login-/Logout-Überwachung
- History-Vervollständigung
- Periodische Ausführung von Kommandos
- Rechtschreibkorrektur (funktioniert nur bei »geringen« Fehlern)
- Umgang mit langen Argumentlisten

Nur das Unterbinden des Einlesens von Benutzer definierten Startdateien wurde in der Tcsh im Unterschied zur C-Shell nicht implementiert.

Und zahlreiche Fehler, die die Implementierungen der C-Shell in sich bergen, sind in der Tcsh ausgemerzt. Leider noch nicht alle, wobei Kenner das für unmöglich halten, da Designfehler manche Ungereimtheiten bedingen.

Gliederung dieses Abschnitts

Fähigkeiten der Tcsh und Definitionen

Wichtige Fähigkeiten

In obigem Vergleich zwischen Csh und Tcsh finden Sie die wichtigsten Argumente für die Verwendung der Tcsh. Etwas präziser soll die folgende Aufzählung ihre Stärken untermauern:

- Vollständig abwärtskompatibel zur C-Shell (nur einige Fehler des »Vorbilds« wurden behoben;-).
- Im Kommandozeilenspeicher (»History«) der Tcsh können Eingaben gesucht, bearbeitet und erneut ausgeführt werden. Das konkrete Verhalten ist konfigurierbar.
- Die Tcsh verwendet eine C-ähnliche Syntax.
- Prozesse können vom Benutzer gestartet, gestoppt und ihre Eigenschaften verändert werden.
- Unvollständige Eingaben von Kommando- und Dateinamen u.a.m. können automatisch expandiert werden. Wie die Expansion erfolgen soll, ist konfigurierbar.
- Geringe Fehler in der Eingabe können automatisch korrigiert werden
- Das Eingabeprompt ist konfigurierbar.
- Berechnungen analog zu C werden unterstützt.
- Die Tcsh verwendet eine C-ähnliche Syntax.
- Die Tcsh unterstützt *keine* Nutzer definierten Funktionen.

Einige Definitionen

Die folgenden Begriffe werden im Laufe des Textes wiederholt benötigt. Ihre Definitionen sind sicher nicht vollständig, sollten aber zum Verstehen ihrer Bedeutung genügen.

Kommentare

Siehe unter Syntax der Tcsh, Kommentare.

Kommando-Trennzeichen

Das Semikolon oder der Zeilenumbruch begrenzt ein Kommando.

Operatoren

Zusammenfassung von Zuweisungs-, arithmetische, logische und Vergleichsoperatoren. Ein Operator bestel aus ein oder zwei Zeichen mit Sonderbedeutungen (bspw. +, ++, -, ||, +=, <). Wir werden im Laufe des Textes auf die verschiedenen Operatoren genauer eingehen.

Whitespace

Steht für Leerzeichen und Tabulatoren und ggf. dem Zeilenumbruch.

Wort

Ein Wort ist eine Folge von Zeichen, die entweder von Whitespaces, von Kommando-Trennzeichen oder vor Operatoren umgeben ist. Beachten Sie, dass obige Worttrenner auch Bestandteil eines Wortes sein können, wenn diese gequotet sind.

Start der Tcsh

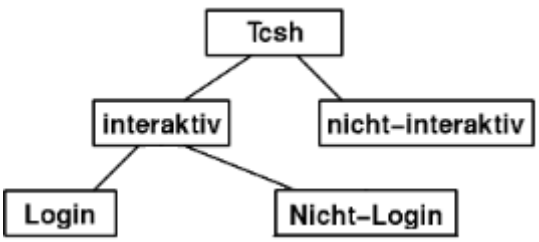


Abbildung 1: Unterteilung der Tcsh

Interaktive und Nicht-Interaktive Tcsh

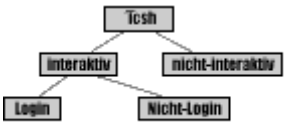
Ist die Tenex-C-Shell mit der Standardein- und Standardausgabe (sprich »mit einer (virtuellen) Konsole«) verbunden, so handelt es sich um eine **interaktive Shell**. Die interaktive Bash teilt sich wiederum ein in die Login Tcsh und die Nicht-Login Tcsh. Eine Shell mit eingeschränkten Nutzungsbefugnissen wie in der Bash gibt es nicht.

Die Tcsh fungiert als Login-Shell, wenn sie entweder explizit mit der Option »-l« ([kleines »L«] und keiner weiteren Option!) gestartet wurde oder wenn das erste Argument (Argument 0) »-« ist.

Die Login-Shell führt zunächst Kommandos aus den System weiten Dateien »/etc/csh.cshrc« und »/etc/csh.login« aus. Die Reihenfolge der Betrachtung der Nutzer definierten Startdateien ist abhängig von der konkreten Version. Meist wird die Datei »~/.tcshrc« betrachtet und nur falls diese nicht existiert die Datei »~/.cshrc«. Falls vorhanden, werden der Reihe die Dateien »~/.history«, »~/.login« und »~/.cshdirs« eingelesen.

Jede Nicht-Login-Shell führt nur Kommandos der Dateien »/etc/csh.cshrc« und »~/.tcshrc« bzw. »~/.cshrc« aus.

Optionen beim Start



Wir beschränken uns hier auf die wichtigsten Optionen.

-c *Kommando*

Die Tcsh führt das angegebene Kommando aus und endet anschließend. Das Kommando muss ein einzelne Argument sein; umfasst es Optionen, ist der gesamte Ausdruck zu quoten:

```
user@sonne> tcsh -c 'ls -l ~/ ../ '
insgesamt 8
drwxr-xr-x22 user       users     4096 Aug 22 11:12 user
drwxr-xr-x   22 tux     users     4096 Aug 22 11:12 tux
```

-d

Die Shell lädt den Verzeichnisstack aus »~/.cshdirs«; sinnvoll nur bei einer Nicht-Login-Tcsh.

-e

Die Tcsh endet sobald eines der in ihr ausgeführten Kommandos mit einem Fehler zurückkehrt; nützlich in Shell-Skripten.

-f

Ignoriert die Datei »~/.tcshrc«.

-i

Die Shell ist interaktiv selbst dann, wenn ihre Ein- und Ausgabe nicht mit einem Terminal verbunden ist.

-l

Die Tcsh arbeitet als Login-Shell. Die Option wird ignoriert, wenn außer ihr weitere Argumente angegeben werden.

-n

Die Shell parst Kommandos ohne sie auszuführen; sinnvoll bei der Fehlersuche in Shell-Skripten.

-v

Die Shellvariable **verbose** wird gesetzt. Jedes Kommando wird nach der History-Substitution nochmals angezeigt.

-x

Die Shellvariable **echo** wird gesetzt. Jedes Kommando wird unmittelbar vor seiner Ausführung (also nach Vollzug sämtlicher Substitutionen) nochmals angezeigt.

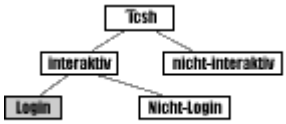
-V

Wie »-v«, nur wird die Variable bereits vor der Verarbeitung der Datei »~/.tcshrc« gesetzt.

-X

Wie »-X«, nur wird die Variable bereits vor der Verarbeitung der Datei »~/.tcshrc« gesetzt.

Tcsh als Login-Shell



In den meisten Konfigurationen, die die Distributoren ausliefern, dürfte die Bash als Login-Shell voreingestellt sein. Bemühen Sie das Kommando chsh, um die Tcsh in der Passwortdatei als Default-Shell einzutragen.

Dies funktioniert jedoch nur, wenn die Tcsh in der Datei /etc/shells eingetragen ist. Ist dem nicht so, könnte Ihnen entweder der Administrator aus der Patsche helfen, indem er den Eintrag in obiger Datei ergänzt oder Sie bauen in einem der Startup-Skripte Ihrer bisherigen Login-Shell den Aufruf »exec /usr/bin/tcsh -l« ein.

Beenden der Tcsh

Zum Beenden stehen mehrere Möglichkeiten zur Verfügung. Ein [Ctrl]-[D] auf einer leeren Eingabezeile beendet sowohl die Login- als auch die Nicht-Login-Shell. Die Kommandos »logout« und »login« stehen allerdings nur in einer Login-Shell zur Verfügung. Letzteres ersetzt den Shellprozess durch das gleichnamige Kommando.

Eine Besonderheit der Tcsh ist der Autologout-Mechanismus, der über die Belegung der Shellvariable **autologout** gesteuert wird. Mit dieser können Sie zwei verschiedene Zeiten (in Minuten) definieren. Zum einen die Zeitspanne, nach der die Shell bei Inaktivität automatisch abgemeldet wird und zum anderen (optional) die Dauer der Inaktivität, nach dem die Shell automatisch gesperrt wird. Eine gesperrte Shell kann nur durch Eingabe des Passworts reaktiviert werden. Fünfmalige falsche Passworteingabe führt automatisch zu einem Autologout.

```
user@sonne> tcsh -l
sonne / home/ user> set autologout = 1
# Nach 1 Minute des Nichtstuns...
user@sonne> auto-logout
```

Beim Beenden der Tcsh setzt diese in Abhängigkeit vom verwendeten Mechanismus die »logout« Shellvariable auf »normal« oder »automatic«. Abschließend führt sie Kommandos aus den Dateien »/etc/csh.logout« und »~/.logout« aus.

Syntax der Tcsh

Der weitere Text erklärt:

- Die Verwendung von Kommentaren
- Die Syntax zur Definition von Variablen

Kommentare

Normalerweise leitet das Doppelkreuz # in der Tcsh einen Kommentar ein. Alles, was diesem Zeichen folgt, wird von der Tcsh ignoriert bis zum nächsten Zeilenumbruch. Dies gilt sowohl für die interaktive als auch für die nicht-interaktive Tcsh.

Eine **Ausnahme** sind Shellskripte, in denen die erste Zeile mit **#!...** beginnt. Für die Tcsh ist es die Anweisung, die nachfolgenden Zeilen mit dem hinter »#!« angegebenen Interpreter (eine Shell, Perl, awk, ...) auszuführen.

Die Bedeutung von # kann durch Quoten (siehe später) aufgehoben werden.

Variablen

Variablennamen bestehen aus Buchstaben, Ziffern und dem Unterstrich, wobei ein Name nicht mit einer Ziffer beginnen darf. In der Tcsh existiert keine (praktische) Begrenzung der Länge von Variablennamen. Als Faustregel gilt, dass Umgebungsvariablen aus Groß- und lokale Variablen aus Kleinbuchstaben bestehen, aber die Shell selbst setzt diesbezüglich keine Vorschriften.

Expansionen der Tcsh

Quoting

Initialisierung

Die Initialisierungsdateien

Die Shellvariablen

Spezielle Shellvariablen

Die Prompts

Bei der interaktiven Arbeit mit der Tcsh zeigt die Shell ihre Bereitschaft zur Entgegennahme von Eingaben anhand eines Prompts (»Eingabeaufforderung«) an. Drei Prompts werden unterschieden, wobei zwei die Eingabeaufforderung symbolisieren:

1. **prompt1:** Dieses zeigt an, dass die Tcsh eine neue Eingabe erwartet.
2. **prompt2:** Das »Fortsetzungsprompt« erscheint bei unvollständigen *while*- oder *foreach*-Schleifen sowie nach Zeilen, die mit dem Backslash **** enden.
3. **prompt2:** Das Prompt erscheint im Fehlerfall und bietet eine kleine Korrekturhilfe durch die Tcsh.

Formatsequenzen:

Aliasse

Eingebaute Kommandos

:

Dieses »Kommando« tut nichts, außer einen Rückgabewert »0« zu erzeugen (»0« ist der übliche Rückgabe eines Kommandos unter Unix, wenn seine Ausführung erfolgreich war). Nützlich ist es in Shellskripten, falls in Bedingungen einen wahren Wert (»true«) benötigen oder an Positionen, wo syntaktisch ein Kommando

erwartet wird, Sie aber keines benötigen:

```
user@sonne> while : ;do echo "Eine Endlosschleife"; done
Eine Endlosschleife
Eine Endlosschleife
Eine Endlosschleife
...
Eine Endlosschleife[Ctrl]+[C]
user@sonne> if test -a foo ; then :; else echo "Datei nicht existent"; fi
Datei nicht existent
```

Interaktive Tcsh

Die History

Der Kommandozeilenspeicher

Der Kommandozeilenspeicher (History) verwaltet die Liste vormals eingegebener Kommandos. Sie kann sowohl als Referenz der bisherigen Kommandofolge als auch als eine Art Baukasten zur komfortablen Zusammenstellung neuer Kommandozeilen dienen.

Zum Betrachten des aktuellen Inhalts des Kommandozeilenspeichers steht das eingebaute Kommando **history** zur Verfügung:

```
sonne / home/ user> history | head
1  12:46   setenv TERM '`/bin/ls -l / > /dev/tty`'
2  12:46   set autologout = "10 5"
3  12:46   echo $status
4  12:46   repeat 3 sleep 1
5  12:46   time | echo
6  13:01   set foo = "this \
and that"
7  13:01   set foo = "this \
and that" ;
8  13:02   echo "$foo" ;
```

In der Voreinstellung werden die letzten 100 Kommandos aufgelistet. Durch Angabe der Anzahl als letztes Argument von »history« kann die Länge der Ausgabe variiert werden.

Des Weiteren versteht »history« folgende Optionen:

-c

Löscht den Inhalt des Kommandozeilenspeichers.

-h [n]

Unterdrückt die Ausgabe der Zeilennummerierung. Normalerweise sind alle Einträge nummeriert, wobei der älteste Eintrag die Nummer 1 trägt. Anhand solcher Nummern kann gezielt auf einen Eintrag des Kommandozeilenspeichers zugegriffen werden. Die Angabe von *n* spezifiziert die Anzahl auszugebender Einträge.

-r [*n*]

Schreibt die *n* letzten Einträge in umgekehrter Reihenfolge aus (den Ältesten zuletzt).

-T [*n*]

In Verbindung mit »-h« wird der Zeitstempel der Einträge auf einer extra Zeile vor dem jeweiligen Eintrag ausgegeben. Die Zeitstempel sind als Kommentare markiert. Diese Ausgabe entspricht dem Format einer

History-Datei.

-S [*Dateiname*]

Speichern der History. Ist ein Dateiname angegeben, wird die History dorthin geschrieben, ansonsten lande in der »normalen« History-Datei (Spezifiziert durch die Shellvariable »histfile» oder, falls diese nicht gesetzt »~/.history«). Wie viele Zeilen geschrieben werden und ob der Inhalt einer vorhandene Datei mit den neue Einträgen gemischt wird, hängt von der Belegung der Shellvariablen **savehist** ab (siehe nachfolgend).

-L *Dateiname*

Lädt die angegebene Datei und hängt die Einträge an den Kommandozeilenspeicher an. Die Datei sollte als History-Format vorliegen (vergleiche Option »-T«).

-M *Dateiname*

Wie »-L«, jedoch wird der Inhalt mit dem existierenden anhand der Zeitstempel gemischt anstatt ihn »hinte anzuhängen.

History-Variablen

Zahlreiche Shellvariablen beeinflussen den Kommandozeilenspeicher. Dazu zählen:

histchars

Mit dieser Variablen kann die Vorbelegung des »History-Substitutionszeichens« (!) und des Ersatzzeichens f das erste Argument (^) modifiziert werden. Weisen Sie hierzu die beiden neu zu verwendenden Zeichen oh Leerreichen der Variablen zu.

histdup

Die Variable steuert den Umgang mit identischen Einträgen im Kommandozeilenspeicher. Steht ihr Inhalt au »all«, wird das aktuelle Kommando nur in die History aufgenommen, wenn noch kein identischer Eintrag existiert. Steht er auf »prev«, wird ein Kommando nur aufgenommen, wenn es nicht dem letzten Eintrag entspricht. Mit »erase« wird ein vorhandener identischer Eintrag entfernt und das aktuelle Kommando land am Ende des Speichers. Bei jedem anderen Wert oder falls die Variable nicht gesetzt ist, wird jede Kommandozeile in die History eingetragen.

histfile

Enthält den Namen der Datei zum Abspeichern der History-Einträge. Ist die Datei leer, wird »~/.history« angenommen, was der üblichen Verfahrensweise entspricht.

histlit

????

history

Die Variable enthält einen oder zwei Werte. Der erste Wert gibt die Anzahl der Einträge an, die im Kommandozeilenspeicher gehalten werden sollen. Der zweite optionale Wert ist eine Formatzeichenkette, d angibt, wie Einträge der History für die Ausgabe zu formatieren sind. Als Formatierer kommen die unter Prompts genannten Zeichen in Frage.

savehist

Sobald die Variable gesetzt ist, ruft eine Shell, bevor sie endet, »history -S« auf, womit der aktuelle Zustand Kommandozeilenspeichers in die durch Datei »histfile« sichert wird. »savehist« kann zwei Werte enthalten. erste gibt an, wie viele Einträge des Kommandozeilenspeichers in die Datei zu sichern sind. Dieser Wert mu kleiner oder maximal gleich dem ersten Wert der »history«-Shellvariable sein. Ein optionaler zweiter Wert »merge« bewirkt, dass die aktuellen Werte der History mit dem Inhalt einer existierenden Datei (bezeichne

durch »histfile«) gemischt werden, wobei der Zeitstempel der Einträge als Kriterium der Sortierung dient.

Navigation in Einträgen des Kommandozeilenspeichers

Durch die Liste der Kommandos in der History und innerhalb einer Kommandozeile können Sie komfortabel mit Hilfe der Pfeiltasten und von Tastenkombinationen navigieren:

↑, [Ctrl][P]

Geht zum vorhergehenden Eintrag.

↑, [Ctrl][N]

Geht zum nachfolgenden Eintrag.

←

Ein Zeichen nach links in der aktuellen Zeile.

→

Ein Zeichen nach rechts in der aktuellen Zeile.

[Home]

Beginn aktuellen Zeile.

[End]

Ende aktuellen Zeile.

Obige Tastenkombinationen beschreiben die Voreinstellung. Mittels dem eingebauten Kommando **bindkey** ist eine Änderung der Belegung möglich.

Eine Suche im Kommandozeilenspeicher wird ebenso angeboten:

***Muster*[Alt][p]**

Sucht rückwärts nach einem mit dem Muster beginnenden Eintrag.

***Muster*[Alt][n]**

Sucht vorwärts nach einem mit dem Muster beginnenden Eintrag.

Das *Muster* kann die Metazeichen »*«, »?«, »[]« und »{}« umfassen, allerdings betrifft dann die Suche nicht den Zeilenanfang sondern die gesamte Zeile. »echo« und »echo*« führen bspw. beide zur letzten Kommandozeile, die mit »echo« startete. »*echo« listet jedoch einzig eine Zeile auf, die mit »echo« endete. Um die Kommandozeile zu finden, die »echo« an beliebiger Position enthielt, ist folgende Eingabe erforderlich:

```
sonne / home/ user> *echo*[Alt][p]
sonne / home/ user> grep -q foo bla |  echo "huchuuuu"
```

Zugriff über die History-Substitution

Eine History-Substitution wird mit dem so genannten »Bang-Operator« (!) eingeleitet. Die Art der Substitution schreiben die nachfolgenden Zeichen vor. Um keine History-Substitution handelt es sich jedoch, wenn dem »Bang-Operator« ein Whitespace, ein Gleichheitszeichen oder eine öffnende runde Klammer folgt (dann handelt es sich um einen unären Operator [Negation]).

Folgende Ausdrücke gestatten die Ausführung von Kommandozeilen aus der History ohne deren vorherige Manipulation:

!!

Startet das vorhergehende Kommando (letzter Eintrag im Kommandozeilenspeicher).

!*n*

Startet das Kommando von Zeile *n* der History.

!-*n*

Startet das Kommando der *n*-letzten Zeile des History (es wird also von »hinten« gezählt; »!-1« entspricht »!!«).

!*Präfix*

Startet das aktuellste in der History eingetragene Kommando, das mit dem *Präfix* beginnt.

!?Muster?

Startet das aktuellste in der History eingetragene Kommando, das das *Muster* enthält.

Einige Beispiele dazu:

```
sonne / home/ user> cat foo
# Ausgabe des Inhalts von »foo«
sonne / home/ user> !! | wc -l
cat foo | wc -l
     26
sonne / home/ user> history | head -5
 1  12:39   cal
 2  12:39   ( cd ; make )
 3  12:50   vi ~/foo.cpp
 4  12:50   history --help
 5  12:50   g++ ~/foo.cpp -o ~/foo
sonne / home/ user> !4
history --help
Benutzung: history [-chrSLMT] [# Anzahl der Befehle].
sonne / home/ user> !vi
vi ~/foo.cpp
```

Die Wiederverwendung alter Kommandos nach obigem Substitutionsschema reißt sicher niemanden vom Hocker. Worin sollte der Vorteil gegenüber der Navigation im Kommandozeilenspeicher bestehen, die sogar das Editieren alter Einträge ermöglicht?

Die weit reichenden Möglichkeiten der Substitution nutzen erst die Erweiterungen zum Bang Operator - so genannte *Wortselektoren*. Sie gestatten die Verwendung von Teilen eines vorherigen Kommandos in der aktuellen Kommandozeile. Eingeleitet wird eine solche Erweiterung durch einen Doppelpunkt, der dem Bang Operator und dem Muster, das das alte Kommando referenziert, folgt. Ein Beispiel ist hoffentlich verständlicher als meine Umschreibung:

```
sonne / home/ user> echo "1 2" 3 "4 5 6"
1 2 3 4 5 6
sonne / home/ user> echo !!:3
4 5 6
```

Der Ausdruck »!!:3« wurde durch das dritte Argument des vorhergehenden Kommandos substituiert. Zugegeben... es ist kein sinnvolles Beispiel. Aber es ist einfach und sollte leicht zu verstehen sein. Der Zweck heiligt die Mittel.

Folgende Aufzählung benennt sämtliche Erweiterungen, mit denen Bestandteile von History-Einträgen extrahiert

werden können.

n

Nummer des Arguments. »0« steht für das erste Wort der Kommandozeile selbst, was i.d.R. der Kommandoname ist.

-n

Argumente 0 bis *n*.

n-

Argumente von *n* bis zum Vorletzten (!).

m-n

Argumente von *m* bis *n* (*m* und *n* eingeschlossen).

^

Kurzform für das erste Argument (wie »:1«).

$

Das letzte Argument.

*****

Alle Argumente ab dem ersten (also ohne den Kommandonamen).

%

Nur in Verbindung mit »!?Muster?« erlaubt. »%« enthält das Wort, das das *Muster* enthielt.

Hierzu wieder einige Beispiele:

```
# Netzwerkschnittstelle initialisieren
sonne /root> ifconfig eth0 192.168.100.100 up
...
# ... Netzwerkschnittstelle runter fahren
sonne /root> !ifc:-1 down
ifconfig eth0 down

# PDF-Version der Fibel erzeugen
sonne / home/ user> htmldoc --book --webpage -f Linuxfibel.pdf toc.htm einleitung.htm # ...125 weitere Dateien
# Upps... eine Datei vergessen;-)
sonne / home/ user> !!:-4 default.htm !!:5-$
```

Ganz schön verwirrend... oder? Aber es kommt noch besser! Sie sind nicht einmal darauf angewiesen, die Argumente in ursprünglicher Form zu verwenden, sondern Sie können diese mit verschiedensten Modifizierern auch noch zuvor bearbeiten. Vorab die Liste der Modifizierer:

:e

Betrachtet das Argument als Dateiname und liefert - falls vorhanden - dessen Suffix.

:h

Entfernt in Pfadangaben die letzte Komponente, »:h:h« würde die beiden letzten Komponenten entfernen u

:p

:q

Quotet im Ergebnis das substituierte Wort (schützt somit vor weiteren Expansionen).

:r

Entfernt den Suffix von Dateinamen. Der trennende Punkt bleibt erhalten. Mehrere Suffixe können durch entsprechend viele Angaben des Modifizierers abgeschnitten werden.

:s/ *x*/ *y*/

Ersetzt im substituierten Wort die Zeichenkette *x* durch die Zeichenkette *y*. *x* darf kein Regulärer Ausdruck s In *y* steht **&** zur Verfügung, um auf die substituierte Zeichenkette zugreifen zu können. Anstatt dem Trennzeichen **/** kann jedes beliebige andere Zeichen, welches nicht in den beiden Zeichenketten vorkommt Verwendung finden.

:t

Entfernt in Dateinamen den kompletten Pfadanteil.

:x

Trennt das substituierte Wort an jedem Whitespace.

:&

Verwendet den zuletzt genutzten Zeichenkettenmodifizierer erneut.

Und zum Abschluss wiederum einige Anwendungsbeispiele obiger Modifizierer:

```
sonne /home/user> ls /home/tux/komplizierterPfad/komplexerName
/home/tux/komplizierterPfad/komplexerName
sonne /home/user> cp !!^ !!^:t
cp /home/tux/komplizierterPfad/komplexerName komplexerName

sonne /root> cp -r /etc/skel /home/user
sonne /root> !!:-$:s/user/tux/
cp -r /etc/skel /home/tux
sonne /root> !!:-$:s?tux?&racer?
cp -r /etc/skel /home/tuxracer
```

Für die Zeichenkettenersetzung im vorhergehenden Kommando existiert ein spezieller Modifizierer. Roots Kopieroperationen in obigem Beispiel hätten auch wie folgt substituiert werden können:

```
sonne /root> cp -r /etc/skel /home/user
sonne /root> ^user^tux
cp -r /etc/skel /home/tux
sonne /root> ^tux^&racer
cp -r /etc/skel /home/tuxracer
```

Diese Kurzform eignet sich also wirklich nur, wenn das vorherige Kommando mit genau einem geänderten Argument zu wiederholen ist.

Die Programmierung der C-Shell und Tcsh

Die Korn Shell

Übersicht

Die Umgebung

Die Kommandozeile

Parsen der Kommandozeile

Variablen

Sonderzeichen quoten

Funktionen

E/ A-Umleitung

Pipes

Die Programmierung der Korn Shell

Unix-Werkzeuge

Überblick

Es ist müßig, über den Sinn von Werkzeugen zu philosophieren. Jeder kennt ihren Nutzen bei der Bewältigung des täglichen Handwerks. Und das Handwerk des Systemadministrators ist die Konfiguration des Systems. Dass hierzu Editoren nützlich sein sollen, hat sich wohl schon herumgesprochen. Sie sind die elementaren Hilfsmittel, derer sich ein Administrator bedienen muss. Die grundlegende Verwendung der auf nahezu allen Unixen verfügbaren Editoren vi und emacs steht deshalb zuvorderst in diesem Abschnitt.

Natürlich kann man auf einen Schraubendreher zurückgreifen, will man mit den zweihundertdreiundvierzig Schräubchen die Metallkonstruktion montieren. Man könnte sich aber auch eines Akkuschraubers bedienen...

Unter Unix sind die Dateien die Schrauben und ein Editor der Schraubendreher. Effektiv arbeitet dieser bei einer einzelnen Datei. Zufriedenstellend ist sein Einsatz auch bei zwei, oder drei, ... Dateien. Was aber wenn 20 Dateien zu durchsuchen und unter bestimmten Voraussetzungen zu ändern sind? Ein Werkzeug, effektiv wie ein Akkuschrauber, müsste her...

Sieht man von den Editoren einmal ab, ist die Kenntnis der hier beschriebenen Unix-Werkzeuge nicht zwingend notwendig. Wer allerdings einen größeren Rechnerpool zu administrieren hat, der wird ihre Vorteile schnell zu schätzen lernen.

Grep, Sed und Awk vereint eine Gemeinsamkeit. Sie suchen nach Mustern in ihrer Eingabe und veranlassen darauf hin bestimmte Aktionen. Bei den Mustern handelt es sich meist um eine Teilmenge der so genannten »Regulären Ausdrücke«, deren Beherrschung das Verständnis der Arbeitsweise der verschiedenen Tools schärft. Eine Einführung in diese Thematik steht deshalb den einzelnen Werkzeugen voran.

Neben den drei beschriebenen Werkzeugen zur Datensuche und -manipulation erhalten Sie in diesem Abschnitt einen Überblick in die Möglichkeiten des Einsatzes von Gnu make, dessen Fähigkeiten zu wesentlich mehr eingesetzt werden können, als einzig als Werkzeug eines Programmierers.

Ziel des Kapitels

Die Grundlagen in der Verwendung der wichtigen Werkzeuge sollte der Leser im Anschluss an das Kapitel beherrschen. Dazu gehören:

- Beherrschung eines Editors
- Kenntnisse im Umgang mit Reguläre Ausdrücken
- Die Verwendung von Grep-Tools
- Die Programmierung des Stream Editors Sed
- Beherrschung der Mustersuche und -manipulation mit Gnu Awk
- Kenntnis der Bedeutung und Anwendung von Make
- Das Schreiben einfacher Makefiles

Inhalt des Kapitels

Der Editor Vi

Übersicht

Der **vi** selbst ist ein kommerzielles Produkt und seine Verwendung unterliegt entsprechenden Lizenzen. Aus diesem Grund existieren eine ganze Reihe von Klones, wie der in diesem Abschnitt beschriebene **vim**.

Wundern Sie sich nicht , falls die eine oder andere Option, die in den folgenden Abschnitten Erwähnung findet, womöglich nicht das beschriebene Resultat erbringt. Der **vi** ist hochgradig konfigurierbar und die verschiedenen Distributionen liefern häufig abweichende Voreinstellungen aus. Auch spielt die Aktualität der Programmversion eine große Rolle, welche Fähigkeiten der **vi(m)** unterstützt.

Modiwechsel

Wenn Sie den Vi starten, befindet sich der Editor automatisch im so genannten Kommandomodus (nachfolgend beschrieben). Eine Reihe Kommandos (Editieren) versetzt ihn erst in den Einfügemodus, der primär zur Eingabe von Text dient. Um zurück in den Kommandomodus zu gelangen, müssen Sie entweder die Taste [ESC] oder die Tastenkombination [Ctrl]-[[ ] betätigen.

Wichtige Optionen im Kommandomodus

Beginnt in nachfolgender Aufzählung eine Option **nicht** mit einem Doppelpunkt, dann wird diese mittels

```
:set [Option]
```

gesetzt und mit

```
:set no[Option]
```

wieder abgeschaltet.

Die wichtigsten Optionen sind:

:se[opt]
: Abfrage des Wertes einer Option.

:se all
: Anzeige der Werte aller Optionen.

:syntax on| off
: Farbliche (unter X) bzw. durch verschiedene Fontarten hervorgehobene Syntax verschiedener Quelldateien (C, C++, Shellskripte, HTML...).

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

Zeilennummerierung einschalten.

scroll

Bestimmt den Scrollbereich.

sh

Bestimmt die Shell, deren Kommandos innerhalb des Editors aufgerufen werden können.

ts

Setzt die Länge des Tabulators (z.B. `:set ts=3`).

wa

Erlaubt das Zurückschreiben des Puffers in die Originaldatei.

Anlegen einer Ressourcendatei

Mitunter wird es lästig, bei jedem Start des Editors die gewünschten Einstellungen vornehmen zu müssen. Einfacher ist es, eine eigene Ressourcendatei `.vimrc` in seinem Heimatverzeichnis abzulegen und in dieser die eigenen Optionen zu setzen. Die folgende Beispieldatei schaltet das Syntax-Highlighting ein und setzt die Schrittweite des Tabulators auf 3.

```
user@sonne> less ~/.vimrc
syntax on
set ts=3
```

Starten und Beenden

Start des Vi

Der **vi** kann auf verschiedene Arten gestartet werden. Dabei können ihm beliebig viele Dateien als Argumente mitgegeben werden (die Anzahl ist allerdings durch die Beschränkung offener Filedeskriptoren begrenzt):

vi

Start ohne Dokument.

vi Dokument

Start mit dem angegebenem Dokument. Existiert *Dokument* nicht, wird die Datei neu angelegt.

vi + Dokument

Der Vi arbeitet mit einer Kopie von *Dokument* und öffnet die Datei selbst erst beim Beenden und Zurückschreiben.

vi -r Dokument

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 Dokument

Eröffnet *Dokument* nur zum Lesen.

Öffnen und Schließen von Dateien im Vi

Weitere Möglichkeiten zum Schließen von Dateien stehen unter »Beenden«.

:e *Dokument*

Öffnet die Datei »Dokument«. Bei der Eingabe des Dateinamens kann der Expansionsmechanismus der Bash verwendet werden. Die aktuell bearbeitete Datei darf seit dem letzten Abspeichern nicht modifiziert worden sein.

:e! *Dokument*

Wie »:e *Dokument*«; die Änderungen in der aktuell bearbeiteten Datei werden ggf. verworfen.

:r *Dokument*

Fügt die Datei »*Dokument*« ab der Position des Cursors ein. Bei der Eingabe des Dateinamens kann der Expansionsmechanismus der Bash verwendet werden.

:w »*Dokument*«

Schreibt den aktuellen Puffer in die Datei »*Dokument*«.

:w >> »*Dokument*«

Fügt den aktuellen Puffer an die Datei »*Dokument*«.

Beenden des Vi

Das Beenden kann u.a. auf folgende Art und Weise erfolgen:

: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.

Positionierung des Cursors

Neben der Positionierung mit den Cursortasten stehen zum Scrollen durch den Text noch folgende Kommandos zur Verfügung:

[Ctrl]+[F] oder **l**

Ein Zeichen nach rechts.

[Ctrl]+[B] oder **h**

Ein Zeichen nach links.

[Ctrl]+[P] oder **k**

Eine Zeile nach oben.

[Ctrl]+[N] oder **j**

Eine Zeile nach unten.

[Ctrl]+[V]

Eine Bildschirmseite nach unten.

[Alt]+[V]

Eine Bildschirmseite nach oben.

w

Beginn des nächsten Wortes.

b

Beginn des vorherigen Wortes.

Zum Einfügen von Text an der aktuellen Cursorposition kann auch die Maus (mittlere Taste) genutzt werden.

In neueren Versionen des Vim kann der Cursor auch mit der Maus positioniert werden. Hierzu ist die Option »:set mouse=a« zu setzen.

Editieren

Die nachfolgenden Kommandos (Ausnahme »r*c*«) schalten den Vi vom Kommandomodus in den Eingabemodus. Um zurück in den Kommandomodus zu gelangen, ist die Taste [ESC] zu betätigen.

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.

r*c*

Ersetze ein Zeichen unter Cursor durch das Zeichen *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*.

Text kopieren

Das Kopieren folgt immer dem gleichen Schema:

- Text in einen Puffer kopieren
- Text aus (einem bestimmten) Puffer einfügen

Text in einen Puffer kopieren

yy

Kopiert aktuelle Zeile in einen Puffer.

***n*y**

Kopiert *n*+1 Zeilen in einen Puffer.

yw

Kopiert ein Wort rechts vom Cursor in Puffer.

yb

Kopiert ein Wort links vom Cursor in Puffer.

Das Kommando »y« kann in Verbindung mit jedem Befehl zur Cursorpositionierung verwendet werden.

Um einen bestimmen Bereich zu kopieren, kann man wie folgt vorgehen:

1. Setzen des Cursors an den Beginn (oder Ende) des zu markierenden Textes
2. Umschalten in den »Visual«-Modus durch Eingabe von »v«
3. Setzen des Cursors an das Ende (oder Beginn) des zu markierenden Textes
4. Eingabe von »y« (der »Visual«-Modus wird automatisch beendet)

Text aus einem Puffer einfügen

P

Fügt Inhalt des aktiven Puffers

- vor der aktuellen Zeile ein, falls sich eine Zeile im Puffer befindet
- vor dem aktuellen Wort ein, falls sich ein Wort im Puffer befindet

p

Fügt Inhalt des aktiven Puffers

- nach der aktuellen Zeile ein, falls sich eine Zeile im Puffer befindet
- nach dem aktuellen Wort ein, falls sich ein Wort im Puffer befindet

:reg

Anzeige aller Pufferinhalte (siehe Beispiel im Anschluss an diese Liste).

:"*c*

Nutzt für die **nächste (!)** Operation (Löschen/Markieren/Einfügen) den durch *c* benannten Puffer (siehe Beispiel unter dieser Tabelle).

Beispiel zum Umgang mit Puffern

Angenommen wir haben 4 Texte bereits in einen Puffer kopiert. Uns interessiert jetzt, was im Puffer steht. Also geben wir im **vi** `:reg` ein:

```
:reg
--- Registers ---
"" eine kurze Zeile
"0 eine kurze Zeile
"- ein_wort
": reg
"% /home/user/.vimrc
Press RETURN or enter command to continue
```

Der aktive Puffer ist `"" eine kurze Zeile`. Um nun mit der folgenden Einfügeoperation mittels `p` den Puffer `ein_wort` zu wählen, tippt man zuvor `"-` ein.

Text löschen

Befindet sich der Vi im Einfügemodus, können die *zuletzt eingegebenen* Zeichen nacheinander mittels der Taste »[Backspace]« gelöscht werden. Bei entsprechender Konfiguration steht auch die Taste »[Del]« im Einfügemodus zur Verfügung.

x
: Zeichen unter Cursor

***n*x**
: *n* Zeichen ab Cursorposition

X
: Zeichen vor dem Cursor

***n*X**
: *n* Zeichen vor der Cursorposition

***n*J**
: Entfernt die Zeilenendezeichen der nächsten *n* Zeilen

dd
: Aktuelle Zeile

***n*dd**
: *n* Zeilen ab aktueller Zeile

dL
: Bis zum unteren Bildschirmrand

dw
: Ein Wort

d)
: Bis Absatzende

D
: Bis Zeilenende

***n*D**
: Alles bis zum Ende der aktuellen Zeile und die *n-1* nächsten Zeilen

:[Bereich]d [x]
: Löscht die durch *Bereich* angegebenen Zeichen und speichert diese ins Register *x*. Beispiel: Die Eingabe von `:1,$d z` löscht die ganze Datei und kopiert den Inhalt in das Register *z*.

Zum Löschen von Bereichen kann wiederum der »Visual«-Modus verwendet werden:

1. Cursor an Beginn des zu löschenden Textes setzen
2. Mit »v« in den »Visual«-Modus schalten
3. Mit den Cursortasten den zu entfernenden Bereich markieren
4. Mit »d« löschen

Suchen und Ersetzen

/ muster
: Suche nach Muster vorwärts im Text.

/
: Wiederholt die Suche vorwärts.

?muster
: Suche nach Muster rückwärts im Text.

?
: Wiederholt die Suche rückwärts.

n

Wiederholt letztes Suchkommando.

:s/ alt/ neu

Sucht und ersetzt »alt« durch »neu« (nur das erste Auftreten in aktueller Zeile).

:s/ alt/ neu/ g

Sucht und ersetzt alle »alt« durch »neu« in aktueller Zeile.

:1,$s/ alt/ neu

Ersetzen im gesamten Dokument

Für Informationen zu weiteren Kommandos rufen Sie im *vi* »:h« auf.

(X)emacs - Kurzanleitung

Übersicht

Die folgende Kurzanleitung für den Gebrauch des *(x)emacs* 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 grafisches Frontend für den Debugger *gdb* oder als Browser für die info-Seiten einsetzen. Außerdem kommt der Emacs mit einer kleinen Spielesammlung daher...

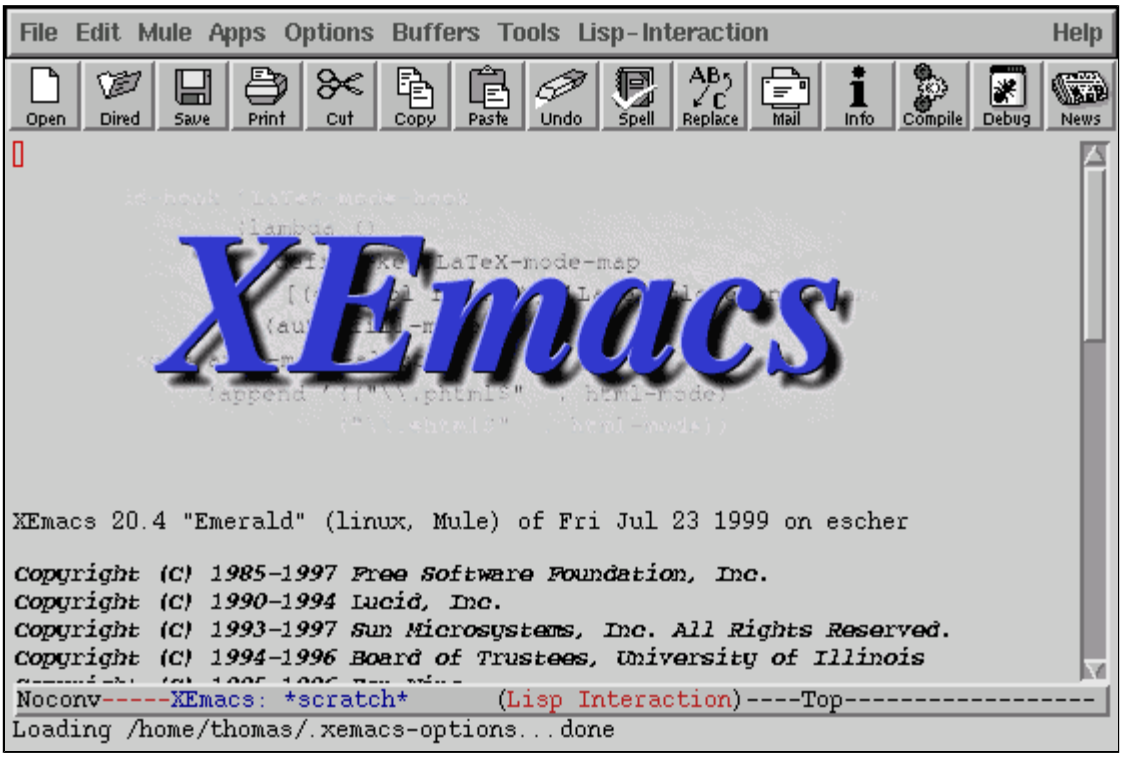


Abbildung 1: Startbildschirm des Xemacs

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.

Laden und Speichern

Nachfolgende Tabelle enthält die verfügbaren Tastenkombinationen zum Speichern und Laden von Dateien, sowie zum Verlassen von Emacs. Anstelle mancher Tastenkombinationen können in der X-Variante auch die entsprechenden Menüeinträge verwendet werden:

[Ctrl]+[X],[Ctrl]+[F] ***Datei*** **[Enter]**
: Datei laden.

[Ctrl]+[X],[4],[F] ***Datei*** **[Enter]**
: Datei in neuem Fenster öffnen.

[Ctrl]+[X],[I] ***Datei*** **[Enter]**
Datei unter Cursor einfügen.

[Ctrl]+[X],[Ctrl]+[S]
Datei speichern.

[Ctrl]+[X],[Ctrl]+[W] ***Datei*** **[Enter]**
Datei als »Datei« speichern.

[Ctrl]+[X],[S]
Alle geöffneten Dateien speichern.

[Ctrl]+[X],[Ctrl]+[C]
(x)emacs beenden.

[Ctrl]+[Z]
(x)emacs unterbrechen, Fortsetzung mit %(x)emacs

Positionierung des Cursors

Die angegebenen Tastaturkommandos betreffen den Vi-Modus!

^
Positioniert Cursor auf das erste Zeichen der aktuellen Zeile.

0
Positioniert Cursor auf die erste Spalte der aktuellen Zeile.

$
Positioniert Cursor auf das letzte Zeichen der aktuellen Zeile.

h
Rückt den Cursor um ein Zeichen nach links.

l
Rückt den Cursor um ein Zeichen nach rechts.

k
Positioniert den Cursor um eine Zeile nach oben.

j
Positioniert den Cursor um eine Zeile nach unten.

w
Setzt den Cursor an den Anfang des nächsten Wortes.

b
Setzt den Cursor an den Anfang des vorhergehenden Wortes.

e
Setzt den Cursor an das Ende des aktuellen Wortes.

[Ctrl] n
Setzt Cursor auf die gleiche Spalte auf der nächsten Zeile.

[Ctrl] p
Setzt Cursor auf die gleiche Spalte auf der vorhergehenden Zeile.

***n*G**
Setzt Cursor auf die *n*-te Zeile des Dokuments.

+
Positioniert auf Textanfang der nächsten Zeile.

-
Positioniert auf Textanfang der vorhergehenden Zeile.

H
Setzt den Cursor auf erste Zeile des Bildschirms.

M
Setzt den Cursor auf mittlere Zeile des Bildschirms.

L
Setzt den Cursor auf letzte Zeile des Bildschirms.

(
Setzt Cursor auf den Anfang des aktuellen Satzes.

)
Setzt Cursor auf das Ende des aktuellen Satzes.

{
Setzt Cursor auf den Anfang des aktuellen Absatzes.

}
Setzt Cursor auf das Ende des aktuellen Absatzes.

Text markieren, löschen, einfügen

Löschen von Text

[Ctrl]+[D]
Löscht ein Zeichen des Wortes rechts ab Cursor.

[Ctrl]+[=]
Löscht ein Zeichen des Wortes links ab Cursor.

[Ctrl]+[K]
Löscht ab Cursor bis Zeilenende.

[Ctrl]+[0],[Ctrl]+[K]
Löscht vom Zeilenanfang bis Cursor.

[Alt]+[M]
Löscht den nächsten Absatz.

[Alt]+[Z]*xxx*
Löscht alle Zeichen von Cursorposition bis zum ersten Auftreten von *xxx*.

[Ctrl]+[Y]
Fügt zuletzt gelöschten Text ab Cursor ein.

Markieren von Text

[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 Cursorposition ein.

Vertauschen von Text

[Ctrl]+[T]
Vertauscht das Zeichen unter dem Cursor mit dem Zeichen links davon.

[Alt]+[T]
Vertauscht zwei Wörter:

- Steht der Cursor am Wortanfang, wird das Wort mit dem vorhergehenden vertauscht.

- Steht der Cursor mitten im Wort, wird dieses mit dem nachfolgenden Wort vertauscht.

[Ctrl]+[U]

Letzte Vertauschung rückgängig machen.

[Ctrl]+[X],[Ctrl]+[T]

Vertauscht die aktuelle Zeile mit der nachfolgenden.

Suchen und Ersetzen von Text

Suchen von Text

[Ctrl]+[S]

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).

Suche mit Wildcards

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, das mit *Taste* beginnt. Einige Musterkombinationen sind in folgender Tabelle aufgeführt:

\ >

Sucht am Wortanfang.

\ <

Sucht am Wortende.

~

Sucht am Zeilenanfang.

$

Sucht am Zeilenende.

.

Platzhalter für ein beliebiges Zeichen.

.*

Platzhalter für beliebig viele (auch Null) beliebige Zeichen.

.+

Platzhalter für beliebig viele (nicht Null) beliebige Zeichen.

.?

Platzhalter für 0 oder ein beliebiges Zeichen.

[abc]

Platzhalter für eines der in Klammern eingeschlossenen Zeichen.

[^ abc]

Platzhalter für alle Zeichen außer den in Klammern eingeschlossenen.

\ (

Beginn einer Gruppe.

\ )

Ende einer Gruppe.

\ Sonderzeichen

Suche nach »Sonderzeichen« (das dem Slash folgende Zeichen verliert seine Sonderbedeutung).

Ersetzen von Text

Zum Suchen und Ersetzen von Text 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

Spezialitäten

Um die Beschreibung nicht unendlich auszudehnen (allein die Beschreibung der Emacs-Konfiguration mittels LISP umfasst mehrere hundert Seiten), soll hier nur eine Beschreibung ausgewählter Modi erfolgen.

Die Bearbeitungsmodi des 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] indended-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.

[Alt]+[X] font-lock-mode [Enter]

Aktivieren von Syntax-Highlighting.

TeX-und LaTeX-Modus

Die wichtigsten zusätzlichen Funktionen sind:

[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.

Die Emulation anderer Editoren

Alle bisher beschriebenen Kommandos arbeiten im "normalen" Modus. Der Emacs unterstützt ebenso die Emulation der Editoren EDT (Dec VMS Editor) und des Vi (in mehreren Varianten).

Um z.B. den Viper-Modus zu benutzen, gibt man im emacs Folgendes ein:

```
M-x viper-mode
```

`M-x` (sprich: Meta-x) bedeutet dabei das gleichzeitige Drücken der Tasten `[Alt][x]`.

Abkürzungen

Für immer wiederkehrenden (lange) Textpassagen kann eine Abkürzung definiert werden, die bei entsprechender Einstellung des Emacs dann automatisch zum vollen Text expandiert wird. Bevor wir ein Beispiel betrachten, seien die verschiedenen Möglichkeiten in Zusammenhang mit Abkürzungen aufgelistet:

[Ctrl]+[X],[A],[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 zur Definition von Abkürzungen

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 von `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!

Sonstiges

Emacs als Newsreader

In der X-Variante muss nur der verwendete Newsserver eingetragen werden:

Options ➜ Customize ➜ Emacs ➜ Applications ➜ News ➜ Gnus ➜ Server.

Alternativ lässt sich in der Datei `~/.emacs` eine Zeile in der Art

```
(setq nntp-address "news.irgendwo.de")
```

aufnehmen.

Aufruf der Kommandos

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.

Reguläre Ausdrücke

Definition

Ein regulärer Ausdruck ist nichts anderes als ein Suchmuster, um übereinstimmende Muster in einer Eingabe zu finden.

In der theoretischen Informatik wird der Begriff des regulären Ausdrucks im Zusammenhang mit endlichen Automaten in einer anderen Terminologie verwendet.

Ein Beispiel

Einfache reguläre Ausdrücke sind Ihnen sicher schon begegnet, ohne dass Sie es bewusst wahr genommen haben. Zum Beispiel bei der Suche nach einem Zeichenmuster »Text« im Editor **vi** (mit »/Text«).

Bleiben wir beim Beispiel des **vi** und betrachten weitere Möglichkeiten »regulärer Ausdrücke« anhand eines Verses von Eugen Roth:

```
Übelkeit

Du magst der Welt oft lange trotzen,
Dann spürst du doch: es ist zum ---.
Doch auch wenn deine Seele bricht,
Beschmutze deinen Nächsten nicht!
```

Aus irgend einem Grund sei der Ersatz aller Textpassagen »dein« durch »sein« erwünscht. Im Kommandomodus des **vi** verrichtet folgende Zeile Suche und Ersatz:

```
:1,$s/ dein/ sein/ g
```

Als Ergebnis erhalten wir:

```
Übelkeit

Du magst der Welt oft lange trotzen,
Dann spürst du doch: es ist zum ---.
Doch auch wenn seine Seele bricht,
Beschmutze seinen Nächsten nicht!
```

Ein »statischer« Text ist ein einfacher Spezialfall eines regulären Ausdrucks. Erst unser nächstes Ansinnen, beide Auftreten des Worts »doch« zu streichen, lässt die Flexibilität des Konzepts erahnen:

```
:1,$s/ [Dd]och/ / g
```

Als Text erzielen wir:

```
Übelkeit

Du magst der Welt oft lange trotzen,
Dann spürst du : es ist zum ---.
 auch wenn seine Seele bricht,
Beschmutze seinen Nächsten nicht!
```

Der praktische Nutzen?

Stellen Sie sich vor, auch Sie schreiben eine Einführung über irgendetwas. Und auch Sie veröffentlichen das Ganze in den zwei Formaten HTML und Postscript. Sie schreiben auch noch in Deutsch und müssen in LaTeX jedes 'ü' durch '"u' ausdrücken, jeden 'ö' durch '"o' usw. Vielleicht machen Sie sich die Arbeit, in HTML die Umlaute durch den Unicode auszudrücken... Dann könnten Sie alles per Hand anpassen, oder das Ganze durch ein paar wenige Zeilen regulärer Ausdrücke erledigen.

Bevor wir uns den einzelnen Werkzeugen zuwenden, sollen alle regulären Ausdrücke tabellarisch zusammen gefasst werden.

Alle Regulären Ausdrücke auf einen Blick

^

Sucht das Muster am Zeilenanfang.

```
/^ beginnt hier/
```

$

Sucht das Muster am Zeilenende.

```
/endet hier$/
```

*****

Beliebig viele Auftreten des vorangegangenen Zeichens; im Beispiel suchen wir Zeilen, die mit »schon« beginnende Muster enthalten (beiliebig viele führende Leerzeichen):

```
/ *schon/
```

.

Genau ein beliebiges Zeichen.

```
/.och/
```

[]

Genau eines der eingeschlossenen Zeichen.

```
/[Dd]och/
```

[a-z]

Eines der Zeichen aus dem Bereich.

```
/[A-X]och/
```

[^]

Keines der eingeschlossenen Zeichen.

```
/[^ ln]och/
```

****

Sperrt die Sonderbedeutung des nachfolgenden Zeichens.

```
/x\$\>y/
```

\ <

Muster am Wortanfang suchen.

```
/\<doch/
```

\ >

Muster am Wortende suchen.

```
/ung\>/
```

\ (..\ )

Eingeschlossenes Muster vormerken; auf dieses kann später über `\1` zugegriffen werden. Bis zu neun Muster können auf diese Weise gespeichert werden (Beispiel).

```
/\(aff\)ig \1enstark/
```

***x*\ { *m*\ }**

m-faches Auftreten des Zeichens *x*.

```
/s\{3\}/
```

x\ { *m*,*n*\ }

Mindestens *m*-, maximal *n*-maliges Auftreten des Zeichens *x*

```
/+\{5,8\}/
```

Zwar vermag nicht jedes der nachstehenden Tools mit jedem der Ausdrücke etwas anzufangen, aber die meisten Mechanismen lassen sich schon in einem Pager wie **less** nachvollziehen.

Beispiel zu \(. . \)

In einem Leserbrief an eine beliebige Zeitung verwenden wir mehrfach die Redewendung »Herren und Damen«. Später wird uns bewusst, dass wir die guten Manieren etwas vernachlässigt hatten und wollen die Ansprache nun in »Damen und Herren« ändern:

```
Meine Herren und Damen!
...
bitte ich die Herren und Damen, sich mit den Mechanismen der regulären Ausdrücke vertraut zu machen.
...
Ich danke Ihnen, meine Herren und Damen!
```

Im Vi erledigt eine einzige Zeile die Arbeit:

```
:1,$s/ \ (Herren\ ) \ (und\ ) \ (Damen\ )/ \ 3 \ 2 \ 1/ g
```

Die drei uns interessierenden Muster merken wir uns vor, und greifen während der Ersetzung in umgekehrter Reihenfolge auf diese zu.

```
Meine Damen und Herren!
...
bitte ich die Damen und Herren, sich mit den Mechanismen der regulären Ausdrücke vertraut zu machen.
...
Ich danke Ihnen, meine Damen und Herren!
```

Unix Werkzeuge - Grep und verwandte Tools

Übersicht

»**G**lobal search for a **r**egular **e**xpression and **p**rint out matched lines« - kurz **'g/ re/ p'** ist das gebräuchlichste Kommando, um in Dateien nach bestimmten Mustern zu suchen. Die Grep-Familie umfasst die drei Kommandos **egrep**, **grep** und **fgrep**. Das erste »Extended Grep« (erweitertes grep) versteht »ein paar mehr« der Regulären Ausdrücke als »grep«. »fgrep« (Fixed Grep) hingegen unterstützt nur eine eingeschränkte Teilmenge, womit sich die Suche vor allem in großen Dateien erheblich beschleunigen lässt.

Grep arbeitet bei der Suche wesentlich effizienter als das in einem Editor geschehen würde. Per Voreinstellung schreibt das Kommando alle Zeilen der Eingabe, die das gesuchte Muster enthalten, auf die Standardausgabe. Dabei kann die Eingabe beliebig viele Dateien, als auch die Standardeingabe betreffen. Zudem liefern die Kommandos der Grep-Familie einen Rückgabewert an das Betriebssystem, was sie für die Verwendung in Shellprogrammen bevorzugt.

Der Rückgabewert von Grep

Häufig interessiert man sich nicht für den exakten Inhalt der Zeile, die das Muster enthält, sondern einzig ob das Muster überhaupt existiert. Vor allem in Shellskripten wird man die Ausgaben der Kommandos unterdrücken und anschließend anhand des Rückgabewertes den weiteren Programmablauf steuern.

Rückgabewert 0

Muster wurde gefunden:

```
# "root" sollte es eigentlich in jeder "passwd" geben
user@sonne> grep root / etc/ passwd > & / dev/ null
user@sonne> echo $?
0
```

Rückgabewert 1

Muster wurde nicht gefunden:

```
# "ROOT" gibt es hoffentlich nicht
user@sonne> grep ROOT / etc/ passwd > & / dev/ null
user@sonne> echo $?
1
```

Rückgabewert 2

Datei nicht gefunden:

```
# die Datei "/bla" gibt es hoffentlich nicht
user@sonne> grep text / bla > & / dev/ null
user@sonne> echo $?
2
```

Wichtige Optionen von Grep

Optionen beeinflussen die Arbeitsweise aller Kommandos der Grep-Familie. Welche Optionen es gibt, beschreibt die folgende Tabelle:

-c

Anzeige der Anzahl Zeilen, in denen das Muster gefunden wurde:

```
user@sonne> grep -c bash / etc/ passwd
38
```

-i

Groß- und Kleinschreibung werden nicht unterschieden:

```
user@sonne> grep -i ROot / etc/ passwd
root:x:0:0:root:/root:/bin/bash
```

-l

Nur Anzeige der Namen der Dateien, in denen das Muster gefunden wurde:

```
user@sonne> grep -l tcp / etc/ host*
/etc/hosts.allow
/etc/hosts.deny
```

-n

Zeigt die Zeilennummer an, in der das Muster gefunden wurde:

```
user@sonne> grep -n root / etc/ passwd
1:root:x:0:0:root:/root:/bin/bash
```

-s

Unterdrückt die Fehlerausgaben (Standardfehler); sinnvoll in Skripten.

-v

Zeigt alle Zeilen an, die das Muster **nicht** enthalten:

```
# ohne diese Option
user@sonne> ps ax | grep inetd
  133 ?        S      0:00 /usr/sbin/inetd
  762 pts/2    S      0:00 grep inetd

# die Ausgabe "grep" herausfiltern
user@sonne> ps ax | grep inetd | grep -v grep
  133 ?        S      0:00 /usr/sbin/inetd
```

-w

Das Suchmuster muss ein einzelnes Wort sein (also kein Bestandteil eines anderen Wortes).

```
user@sonne> echo -e "Automaten\ n essen\ n keine Tomaten" | grep -i Tomaten
Automaten
keine Tomaten

user@sonne> echo -e "Automaten\ n essen\ n keine Tomaten" | grep -iw Tomaten
keine Tomaten
```

-A [Anzahl]

Zeigt »Anzahl« Zeilen an, die der Zeile mit dem Muster folgen.

```
user@sonne> grep -A 2 root / etc/ passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
```

-B [Anzahl]

Zeigt »Anzahl« Zeilen an, die vor der Zeile mit dem Muster liegen.

Von Grep unterstützte Reguläre Ausdrücke

Nachfolgend sind alle Reguläre Ausdrücke aufgeführt, die die Kommandos der Grep-Familie unterstützen. Hinter jedem Muster ist angegeben, welches grep-Kommando diese Syntax beherrscht. Dabei wird nicht zwischen grep und fgrep unterschieden, da beide Kommandos dieselbe Sprache sprechen (also grep=fgrep).

^ (grep, egrep)

Beginn der Zeile

$ (grep, egrep)

Ende der Zeile

. (grep, egrep)

Genau ein beliebiges Zeichen

*** (grep, egrep)**

Beliebig viele des vorangegangenen Zeichens

[] (grep, egrep)

Ein Zeichen aus dem Bereich. Anstelle von Zeichen können vordefinierte Klassen von Zeichen verwendet werden: [:alnum:], [:alpha:], [:cntrl:], [:digit:], [:graph:], [:lower:], [:print:], [:punct:], [:space:], [:upper:] und [:xdigit:].

[^] (grep, egrep)

Kein Zeichen aus dem Bereich

\ < (grep)

Muster am Wortanfang suchen

\ > (grep)

Muster am Wortende suchen

\ (..\ ) (grep)

Eingeschlossenes Muster vormerken; auf dieses kann später über \1 zugegriffen werden. Bis zu neun Muste können auf diese Weise gespeichert werden (ein Beispiel steht im Abschnitt Reguläre Ausdrücke).

x\ { m\ } (grep)

x\{m,n\} (grep)

mindestens m-, maximal n-maliges Auftreten des Zeichens x

+ (egrep)

Mindestens ein Auftreten des vorangegangenen Zeichens

? (egrep)

Höchstens ein Auftreten des vorangegangenen Zeichens

x| y (egrep)

Zeichen "x" oder Zeichen "y"

(abc| xyz) (egrep)

Zeichenkette "abc" oder Zeichenkette "xyz". Die runden Klammern können entfallen.

Grep - Beispiele

Einfache Beispiele zur Anwendung von **grep** begegneten uns schon an mehreren Stellen dieses Buches. Nun möchte ich versuchen, anhand typischer Anforderungen bei der alltäglichen Arbeit mit einem Unix-System, die Verwendung der komplexeren Mechanismen zu erläutern.

Beispiel 1

Bei der Systemadministration fragt man sich häufig, in welcher Datei eigentlich welche Shell-Variable gesetzt wird? Die globalen Vorgaben erfolgen zum Großteil in den Dateien des Verzeichnisses **/etc**. Also interessieren uns die Namen der Dateien, in denen z.B. die **PATH**-Variable modifiziert wird:

```
user@sonne> grep -l PATH /etc/* 2>/dev/null
/etc/csh.cshrc
/etc/login.defs
/etc/manpath.config
/etc/profile
/etc/profile.rpmsave
/etc/rc.config
/etc/squid.conf
```

Die Umleitung der Fehlerausgabe nach /dev/null ist sinnvoll, da »grep« nicht auf Verzeichnisse anwendbar ist.

Beispiel 2

Wie viele Nutzer sind Mitglied in der default-Gruppe users (GID 100)?

```
user@sonne> grep -c ':[0-9]\{1,\}:100:' /etc/passwd
9
```

Bei der Angabe des Suchmusters hilft uns die Kenntnis des Aufbaus der Datei »/etc/passwd«. Dabei steht die GruppenID immer zwischen zwei Doppelpunkten. Allerdings könnte es sein, dass auch die NutzerID (UID) 100 vergeben ist - der Ausdruck :[0-9]\{1,\}:100: garantiert, dass :100: das zweite rein numerische Feld betrifft. Eine andere Schreibweise ist:

```
user@sonne> grep -c ':[[:digit:]]\{1,\}:100:' /etc/passwd
9
```

Beispiel 3

Welche Netzwerkdienste über UDP sind auf unserem System verfügbar (Datei /etc/inetd.conf)?

```
user@sonne> grep '^[^#].*[[:space:]]udp' / etc/ inetd.conf
time    dgram   udp     wait    root    internal
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
netbios-ns      dgram   udp     wait    root    /usr/sbin/nmbd  nmbd
```

Jede Zeile, die mit einem # beginnt, ist ein Kommentar. Also filtern wir solche Zeilen aus (^[^#]). Das gesuchte Protokoll ist "udp". Vor diesem Schlüsselwort können beliebig viele Zeichen (.*) gefolgt von einem Leerzeichen oder Tabulator ([[:space:]]) stehen.

Beispiel 4

Je gezielter man nach Informationen fahndet, desto verwirrender wird die Angabe der Suchmusters. In zahlreichen Fällen wird die Verwendung von Pipes einleuchtender sein. Das Ergebnis aus obigen Beispiel erhält man auch mit folgender Befehlsfolge:

```
user@sonne> grep -w udp / etc/ inetd.conf |  grep -v ^#
time    dgram   udp     wait    root    internal
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
netbios-ns      dgram   udp     wait    root    /usr/sbin/nmbd  nmbd
```

Egrep - Beispiele

Egrep ist hilfreich, wenn Sie nach Zeilen in der Eingabe suchen, die mindestens eine von mehreren Zeichenketten enthalten.
So findet das folgende Beispiel alle Zeilen der Datei /etc/fstab, in denen »floppy« und »cdrom« auftauchen:

```
user@sonne> egrep 'floppy| cdrom' / etc/ fstab
/dev/hdc    /cdrom          iso9660   ro,noauto,user,exec 0   0
/dev/fd0    /floppy         auto      noauto,user         0   0
```

Eine weitere interessante Anwendung ist die Suche nach »geteilten« Mustern, d.h. die bekannten Teile stehen auf einer Zeile, aber der Zwischenraum ist unbekannt. Zur Demonstration dient folgende Datei:

```
user@sonne> cat beispiel.txt
1 ein  Zwischenraum
2 ein          Zwischenraum
3 ein    Zwischenraum
4 ein          Zwischenraum
```

Gesucht werden sollen alle Zeilen, die »einen Zwischenraum« enthalten; jedoch ist die Zusammensetzung des Zwischenraums nicht bekannt (und besteht teils aus Leerzeichen, teils aus Tabulatoren und teils aus beidem). Mit dem normalen **grep** könnte man sich mit folgendem Konstrukt behelfen:

```
user@sonne> grep "ein[[:space:]][[:space:]]* Zwischenraum" beispiel.txt
1 ein  Zwischenraum
2 ein          Zwischenraum
3 ein    Zwischenraum
4 ein          Zwischenraum
```

Die doppelte Anwendung des [:space:]-Musters ist für diesen Fall notwendiger Ballast, da wir ja mindestens einen

Zwischenraum benötigen. Eleganter ist da die Möglichkeit von "**+**" in Verbindung mit **egrep**:

```
user@sonne> egrep "ein[[:space:]]+Zwischenraum" beispiel.txt
1 ein  Zwischenraum
2 ein        Zwischenraum
3 ein   Zwischenraum
4 ein        Zwischenraum
```

Fgrep - Beispiele

Fgrep kann immer anstelle von grep verwendet werden, falls das zu suchende Muster keine regulären Ausdrücke enthält. Alle Sonderzeichen in der Musterzeichenkette verlieren ihre Sonderbedeutung und werden als Bestandteil des Musters verstanden. Fgrep arbeitet dadurch etwas schneller als grep und ist vor allem beim Durchsuchen großer Datenmengen nützlich.

Unix Werkzeuge - Der Stream Editor

Übersicht

Der Stream Editor ist kein herkömmlicher Editor wie Vi oder Emacs. Der *Sed* arbeitet *nicht interaktiv*, er wird mittels Kommandozeilenoptionen oder durch ein Skript gesteuert. Der Stream Editor modifiziert niemals das Original, sondern schreibt das Ergebnis auf die Standard-Ausgabe. Die aktuell betrachtete Zeile lädt der *Sed* in einen temporären Puffer - nachfolgend als *Arbeitspuffer* bezeichnet.

Der Aufruf des Stream Editors auf der Kommandozeile besitzt immer folgendes Format:

```
sed 'Kommando' Dateiname
```

Die Kommandos des Sed

Die Aktionen von *Sed* werden durch Kommandos gesteuert. Diese Kommandos können Zeilenangaben oder -bereiche enthalten, dann betrachtet der Editor nur die Zeilen der zu bearbeitenden Datei. Fehlt eine solche Angabe, bearbeitet *Sed* die gesamte Datei.

Bevor wir das Verhalten anhand von Beispielen kennen lernen, seien alle Kommandos aufgeführt:

a

Fügt eine oder mehrere Zeilen an die aktuelle Zeile an

c

Ersetzt Text in der aktuellen Zeile

d

Löscht Zeile(n)

g

Kopiert den Inhalt eines temporären Puffers in den Arbeitspuffer (dessen alter Inhalt geht verloren)

G

h

Kopiert den Inhalt des Arbeitspuffers in einen temporären Puffer

H

Fügt den Inhalt des Arbeitspuffers an einen temporären Puffer an

i

Fügt Text oberhalb er aktuellen Zeile ein

l

Zeigt nicht druckbare Zeichen an

n

Wendet das nächste Kommando anstelle des aktuellen Kommandos auf die nächste Zeile an

p

Druckt Zeile(n)

q

Beendet den Editor

r

Liest Zeilen aus einer Datei

!

Wendet das Kommando auf Zeilen an, die nicht zutreffen

Substitutionskommandos des Sed

Im Zusammenhang mit Substitutionen werden die Kommandos häufig als *Flags* bezeichnet. Beachten Sie, dass die Wirkung mancher Kommandos (»g«) aus dem Kontext entschieden wird.

g

Globale Ersetzung (jedes Vorkommen des Musters auf der Zeile)

p

Ausgabe der Zeile in Verbindung mit "s".

s

Ersetzen eines Musters durch ein anderes

w

Ausgabe der bearbeiteten Zeilen in eine Datei

x

y

Ersetzen eines Zeichens durch ein anderes

Reguläre Ausdrücke des *Sed*

Von den Regulären Ausdrücken versteht der Stream Editor Folgende:

^

Zeilenanfang

$

Zeilenende (bei der Adressierung steht das Zeichen für die letzte Zeile)

.

Ein Zeichen (Ausnahme ist der Zeilenumbruch)

*****

Keine, eine oder mehrere Wiederholungen des vorhergehenden Buchstabens / der vorhergehenden Gruppe

[...]

Ein Zeichen aus der Menge

[^ ...]

Kein Zeichen aus der Menge

\ (...\ )

Speichern des enthaltenen Musters

&

Enthält das Suchmuster

\ <

Wortanfang

\ >

Wortende

x\ { m\ }

m-fache Wiederholung des Zeichens x

x\ { m,\ }

Mindestens m-fache Wiederholung des Zeichens x

x\ { m,n\ }

Mindestens m-, maximal n-fache Wiederholung des Zeichens x

Adressierung und Beispieltext

Die Handhabung des Stream Editors erlernen Sie vermutlich nur durch Beispiele. Es liegt also nahe, dem Leser mittels des Beispieltextes gleichzeitig neues Wissen einzuimpfen. Die folgende Abhandlung diskutiert die Adressierungsmöglichkeiten des *Sed* (das Schema lässt sich u.a. auch im Vi anwenden):

```
user@sonne> cat sedtest.txt
Der Aufruf des Stream Editors besitzt immer das Format:
    sed 'Kommando' Dateiname
Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
Mechanismen in Frage:
Keine Angabe  Alle Zeilen
Nummer        Genau diese Zeile
Start, Ende   Alle Zeilen von "Start" bis "Ende"
$             Symbolisiert die letzte Zeile
RegEx         Zeilen, die den Regulären Ausdruck enthalten
1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Um die spätere Arbeit des Editors besser verfolgen zu können, nummerieren wir noch die Datei:

```
user@sonne> nl -w 2 -b a sedtest.txt |  tee test.txt
 1    Der Aufruf des Stream Editors besitzt immer das Format:
 2
 3        sed 'Kommando' Dateiname
 4
 5    Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
 6    Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
 7    Mechanismen in Frage:
 8
 9    Keine Angabe  Alle Zeilen
10    Nummer        Genau diese Zeile
11    Start, Ende   Alle Zeilen von "Start" bis "Ende"
12    $             Symbolisiert die letzte Zeile
13    RegEx         Zeilen, die den Regulären Ausdruck enthalten
14    1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Ausgabe - Das p-Kommando

Zunächst wenden wir das Kommando p auf die ersten 3 Zeilen der Datei an:

```
user@sonne> sed '1,3p' test.txt
 1    Der Aufruf des Stream Editors besitzt immer das Format:
 1    Der Aufruf des Stream Editors besitzt immer das Format:
 2
 2
 3        sed 'Kommando' Dateiname
 3        sed 'Kommando' Dateiname
```

```
 4
 5      Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
 6      Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
 7      Mechanismen in Frage:
 8
 9      Keine Angabe  Alle Zeilen
10      Nummer        Genau diese Zeile
11      Start, Ende   Alle Zeilen von "Start" bis "Ende"
12      $             Symbolisiert die letzte Zeile
13      RegEx         Zeilen, die den Regulären Ausdruck enthalten
14      1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Offensichtlich zeigt der Stream Editor etwas mehr an, als uns lieb ist; er gibt einfach die gesamte Datei aus und wiederholt nur die Zeilen, die durch das Kommando »1,3p« bearbeitet wurden. Um solche »überflüssigen« Ausgaben zu unterbinden, muss die Option »-n« verwendet werden:

```
user@sonne> sed -n '1,3p' test.txt
1      Der Aufruf des Stream Editors besitzt immer das Format:
2
3          sed 'Kommando' Dateiname
```

Bei der Adressierung mittels regulärer Ausdrücke müssen diese in Schrägstriche (Slashes) eingeschlossen sein:

```
user@sonne> sed -n '/ RegEx/ ,/ RegEx/ p' test.txt
13      RegEx         Zeilen, die den Regulären Ausdruck enthalten
14      1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Löschen - Das d-Kommando

Der nachfolgende Aufruf löscht alle Zeilen ab der (einschließlich) 4. bis zum Dateiende:

```
user@sonne> sed '4,$d' test.txt
1      Der Aufruf des Stream Editors besitzt immer das Format:
2
3          sed 'Kommando' Dateiname
```

Das Entfernen aller Zeilen, die mit einem Leerzeichen beginnen, erledigt dieser Aufruf:

```
user@sonne> sed '/ ^  / d' test.txt
10      Nummer        Genau diese Zeile
11      Start, Ende   Alle Zeilen von "Start" bis "Ende"
12      $             Symbolisiert die letzte Zeile
13      RegEx         Zeilen, die den Regulären Ausdruck enthalten
14      1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Ersetzen - Das s-Kommando

Das dem s-Kommando folgende Zeichen wird als *Trennzeichen* angesehen. Anschließend folgt das Suchmuster

und, getrennt durch das Trennzeichen, das Ersatzmuster, welches wiederum mittels des Trennzeichens abgeschlossen wird. Prinzipiell kann jedes druckbare Zeichen als Trennzeichen Verwendung finden, es selbst darf allerdings kein Bestandteil eines Musters sein! Eine Substitution sieht demnach wie folgt aus:

```
sed 's/altes Muster/neues Muster/' datei
sed 's?altes Muster?neues Muster?' datei
```

Im Beispiel ersetzen wir »RegEx« durch »Regulärer Ausdruck«:

```
user@sonne> sed 's# RegEx# Regulärer Ausdruck# ' test.txt
 1    Der Aufruf des Stream Editors besitzt immer das Format:
 2
 3         sed 'Kommando' Dateiname
 4
 5    Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
 6    Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
 7    Mechanismen in Frage:
 8
 9    Keine Angabe      Alle Zeilen
10    Nummer            Genau diese Zeile
11    Start, Ende       Alle Zeilen von "Start" bis "Ende"
12    $                 Symbolisiert die letzte Zeile
13    Regulärer Ausdruck      Zeilen, die den Regulären Ausdruck enthalten
14    1, Regulärer Ausdruck   Von Zeile 1 bis zur ersten Zeile, die Regulärer Ausdruck enthält
```

Wer genau hinschaute, wird im letzten Beispiel eine fehlende Ersetzung von »RegEx« bemerkt haben (Zeile 14). Der Editor bearbeitet in jeder Zeile nur das erste Vorkommen. Um alle Muster zu ersetzen, ist das Kommando »g« nachzustellen:

```
user@sonne> sed -n 's# RegEx# Regulärer Ausdruck# gp' test.txt
13    Regulärer Ausdruck      Zeilen, die den Regulären Ausdruck enthalten
14    1, Regulärer Ausdruck   Von Zeile 1 bis zur ersten Zeile, die Regulärer Ausdruck enthält
```

Da wir nur an den modifizierten Zeilen interessiert sind, haben wir das *Sed* mitgeteilt (Option -n). Allerdings würde nun das Substitutionskommando die gesamte Ausgabe unterdrücken, hätten wir dem nicht mit dem p-Kommando entgegen gewirkt.

Ein (zugegeben... etwas konstruiertes) Beispiel soll das Speichern von Mustern und den späteren Zugriff darauf demonstrieren. Es soll die Nummerierung der Zeilen von Einer- auf Zehnerschritte erhöht werden:

```
user@sonne> sed 's/ ^ \ (\ [[:space:]]* [1-9]\ { 1,\ }\ )/ \ 10/ ' test.txt
  10    Der Aufruf des Stream Editors besitzt immer das Format:
  20
  30         sed 'Kommando' Dateiname
  40
  50    Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
  60    Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
  70    Mechanismen in Frage:
  80
  90    Keine Angabe  Alle Zeilen
```

100	Nummer	Genau diese Zeile
110	Start, Ende	Alle Zeilen von "Start" bis "Ende"
120	$	Symbolisiert die letzte Zeile
130	RegEx	Zeilen, die den Regulären Ausdruck enthalten
140	1, RegEx	Von Zeile 1 bis zur ersten Zeile, die RegEx enthält

Das Beispiel profitiert von dem Wissen, dass die Zeilenummer am Beginn der Zeile zu finden ist. Das Muster, dem unser Interesse gilt, sind alle Ziffern zu Beginn der Zeile, wobei führende Leerzeichen durchaus möglich sind. Genau jenes Muster merken wir uns für den späteren Gebrauch vor, indem wir es in »\(...\)« einschlie&szlIg;en. Der Zugriff auf dieses *erste* gespeicherte Muster im Ersatzmuster erfolgt durch »\1«.

Bis zu 9 Muster lassen sich pro Zeile speichern, die entsprechend ihrer Reihenfolge mittels \1, \2,... \9 referenziert werden.

Mehrere Kommandos - Die e-Option

Das Problem mit Kommandos wie »s« ist, dass sie keine Adressierung zulassen. Sicher gibt es Situationen, wo nur ein Teil einer Datei zu bearbeiten ist. Hier hilft das e-Kommando, mit dem sich beliebig viele Kommandos kombinieren lassen:

```
user@sonne> sed -e '3,$d' -e 's#  # .# g' test.txt
.1      Der.Aufruf.des.Stream.Editors.besitzt.immer.das.Format:
.2
```

Zuerst werden alle Zeilen ab der 3. Zeile entfernt und anschließend die Leerzeichen durch Punkte ersetzt. Aber Vorsicht... mitunter beeinflusst die Reihenfolge der Kommandos das Ergebnis!

Eine alternative Angabe ist die Gruppierung mehrerer Kommandos. Hierzu werden diese in geschweifte Klammern gesetzt und ein Semikolon nach jedem Kommando eingefügt

```
user@sonne> sed '{ s/  / ./ g;3,$d} ' test.txt
.1      Der.Aufruf.des.Stream.Editors.besitzt.immer.das.Format:
.2
```

Einfügen - Das i- und das a-Kommando

Die Syntax zum Einfügen ist wohl etwas gewöhnungsbedürftig. Der einzufügende Text muss auf einer neuen Zeile stehen, wobei jede Zeile bis auf die letzte durch einen Backslash abzuschließen ist. Das Kommando »i« (insert) fügt den Text **vor** der betreffenden Zeile ein, »a« (append) schreibt den neuen Text **nach** der Zeile.

```
user@sonne> sed '8i\
==========================================\
            Angabe              Bereich von Zeilen\
==========================================' test.txt
        1   Der Aufruf des Stream Editors besitzt immer das Format:
        2
        3       sed 'Kommando' Dateiname
        4
        5   Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
        6   Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
        7   Mechanismen in Frage:
```

```
      8
==========================================
            Angabe              Bereich von Zeilen
==========================================
      9     Keine Angabe        Alle Zeilen
     10     Nummer              Genau diese Zeile
     11     Start, Ende         Alle Zeilen von "Start" bis "Ende"
     12     $                   Symbolisiert die letzte Zeile
     13     RegEx               Zeilen, die den Regulären Ausdruck enthalten
     14     1, RegEx            Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Einfügen aus einer Datei - Das r-Kommando

Wohl in seltenen Fällen wird ein Text nur in eine einzige Datei eingefügt werden (sonst wäre der Griff zu einem herkömmlichen Editors der effizientere Weg). Eine gangbare Methode ist, den einzusetzenden Text in einer separaten Datei zu erfassen und *Sed* jene unterzuschieben.

```
user@sonne> cat ins.tex
==========================================
 Angabe                 Bereich von Zeilen
==========================================
```

Dieser Text lässt sich mittels des r-Kommandos einfach an beliebiger Stelle einordnen:

```
user@sonne> sed '8r ins.txt' test.txt
 1    Der Aufruf des Stream Editors besitzt immer das Format:
 2
 3         sed 'Kommando' Dateiname
 4
 5    Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
 6    Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
 7    Mechanismen in Frage:
 8
==========================================
      Angabe        Bereich von Zeilen
==========================================
 9    Keine Angabe  Alle Zeilen
10    Nummer        Genau diese Zeile
11    Start, Ende   Alle Zeilen von "Start" bis "Ende"
12    $             Symbolisiert die letzte Zeile
13    RegEx         Zeilen, die den Regulären Ausdruck enthalten
14    1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Schreiben in eine Datei - Das w-Kommando

Das Ergebnis des Stream Editors lässt sich in einer Datei speichern:

```
user@sonne> sed -n '/ ^ 1/ w out.txt' test.txt
```

```
user@sonne> cat out.txt
10    Nummer        Genau diese Zeile
11    Start, Ende   Alle Zeilen von "Start" bis "Ende"
12    $             Symbolisiert die letzte Zeile
13    RegEx         Zeilen, die den Regulären Ausdruck enthalten
14    1, RegEx      Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
```

Die nächste Zeile - Das n-Kommando

Soll erst die dem Suchmuster folgende Zeile manipuliert werden, ist das n-Kommando der beste Kandidat::

```
user@sonne> sed -n '8,${ n;s/ \ (.\ { 1,\ } \ )/ * \ 1/ p;} ' test.txt
 * 9    Keine Angabe Alle Zeilen
*11    $            Symbolisiert die letzte Zeile
*13    RegEx        Zeilen, die den Regulären Ausdruck enthalten
```

Die Kommandozeile ist schwer verdaulich... aber der Reihe nach:

»-n« als Kommandozeilenoption besagt, dass de Ausgabe einzig die bearbeiteten Zeilen betreffen soll. Da jedoch das Flag »s« sämtliche Ausgaben »verschluckt«, muss »p« am Ende bemüht werden.

»8,$« adressiert die Zeilen 8 bis zum Ende der Datei. Die erste Zeile, die also gefunden wurde, ist die 8. »-n« als Substitutionskommando bewirkt nun, dass die nächste Zeile in die Mangel genommen wird - Zeile 9.

Diese 9. Zeile wird nun substituiert. ».\{1,\}« meint »mindestens ein (\{1,\}) beliebiges (.) Zeichen. Da jede Zeile im Beispiel zumindest die Zeilenummer umfasst und das komplette Muster via »\(...\)« gespeichert wird, erscheint die gesamte Zeile mit vorangestelltem Stern (*\1) in der Ausgabe...

Sed fährt mit der folgenden Zeile (10) fort, die (wegen Flag »n«) übersprungen wird...

Zeichentausch - Das y-Kommando

Einzelne Zeichen lassen sich durch andere einzelne Zeichen ersetzen. Das Zeichen an Position *x* der Liste zu ersetzender Zeichen wird in das Zeichen an Position *x* der Liste der neuen Zeichen transformatiert. Damit ist klar, dass beide Zeichenlisten dieselbe Länge besitzen müssen:

```
user@sonne> sed 'y/ abcdefghijklmnopqrstuvwxyz/ zyxwvutsrqponmlkjihgfedcba/ ' test.txt
 1    Dvi Afuifu wvh Sgivzn Ewrglih yvhrgag rnnvi wzh Flinzg:
 2
 3        hvw 'Klnnzmwl' Dzgvrmznv
 4
 5    Dzyvr pzmm wvn Klnnzmwl nrgtvgvrog dviwvm, dvoxsv Zvrovm wvi
 6    Ermtzyvwzgvr vh yvziyvrgvm hloo. Aoh Awivhhrvifmt plnnvm ulotvmwv
 7    Mvxszmrhnvm rm Fiztv:
 8
 9    Kvrmv Amtzyv  Aoov Zvrovm
10    Nfnnvi        Gvmzf wrvhv Zvrov
11    Sgzig, Emwv   Aoov Zvrovm elm "Sgzig" yrh "Emwv"
12    $             Sbnylorhrvig wrv ovgagv Zvrov
13    RvtEc         Zvrovm, wrv wvm Rvtfo/auml;ivm Afhwifxp vmgszogvm
```

```
14      1, RvtEc        Vlm Zvrov 1 yrh afi vihgvm Zvrov, wrv RvtEc vmgsäog
```

Sed beenden - Das q-Kommando

Manchmal ist es sinnvoll, den Stream Editor vorzeitig zu beenden:

```
user@sonne> sed '3q' test.txt
1       Der Aufruf des Stream Editors besitzt immer das Format:
2
3               sed 'Kommando' Dateiname
user@sonne> sed -n '/ sed/ { p;q;} ' test.txt
3               sed 'Kommando' Dateiname
```

Zeilentausch - Die Kommandos h, g, G und x

Die soeben bearbeitete Zeile hält der *Sed* in einem Zwischenspeicher und bearbeitet diese in diesem »Pattern Space«. Hat der Editor seine Arbeit beendet, gibt er die Zeile aus und lädt die folgende Zeile der Eingabedatei in den Zwischenspeicher.

Mit dem Kommando »h« kann der aktuelle Zwischenspeicher in einen Puffer gesichert werden (»holding buffer«). Das Kommando »G« fügt den Inhalt dieses Sicherungspuffers **hinter** der aktuell bearbeiteten Zeile ein; »g« **ersetzt** die aktuelle Zeile durch den Inhalt des Sicherungspuffers. Den Inhalt der beiden Puffer **vertauscht** das Kommando »x«.

```
user@sonne> sed -e '/ sed/ { h;d;} ' -e '4G' -e '4q' test.txt
1       Der Aufruf des Stream Editors besitzt immer das Format:
2
4
3               sed 'Kommando' Dateiname
user@sonne> sed -e '/ sed/ { h;d;} ' -e '4g' -e '4q' test.txt
1       Der Aufruf des Stream Editors besitzt immer das Format:
2
3               sed 'Kommando' Dateiname
```

Erklärung: Das erste Kommando »-e '/sed/{h;d;}'« in beiden Aufrufen teilt dem Editor mit, die Zeile, die *Sed* enthält, zuerst in den Zwischenspeicher zu sichern und nachfolgend zu löschen. Das zweite Kommando vollzieht das Einfügen des Inhalts des Puffers. »-e '4G'« fügt nach der 4.Zeile ein (Achtung: die gelöschte Zeile wird mitgezählt!); »-e '4Gg'« ersetzt die 4.Zeile. Das letzte Kommando »-e '4q'« beendet die Arbeit des Editors nach der 4. Zeile.

```
user@sonne> sed -e '/ Aufruf/ h' -e '/ Angabe/ x' -e '$G' test.txt
 1      Der Aufruf des Stream Editors besitzt immer das Format:
 2
 3              sed 'Kommando' Dateiname
 4
 5      Dabei kann dem Kommando mitgeteilt werden, welche Zeilen der
 6      Eingabedatei es bearbeiten soll. Als Adressierung kommen folgende
 7      Mechanismen in Frage:
 8
 1      Der Aufruf des Stream Editors besitzt immer das Format:
```

```
10      Nummer          Genau diese Zeile
11      Start, Ende     Alle Zeilen von "Start" bis "Ende"
12      $               Symbolisiert die letzte Zeile
13      RegEx           Zeilen, die den Regulären Ausdruck enthalten
14      1, RegEx        Von Zeile 1 bis zur ersten Zeile, die RegEx enthält
 9      Keine Angabe    Alle Zeilen
```

Erklärung: Enthält eine Zeile das Muster »Aufruf«, wird sie im Zwischenpuffer abgelegt. Steht in einer Zeile »Angabe«, so wird diese Zeile mit dem Inhalt des Zwischenpuffers vertauscht. Der Inhalt des Zwischenpuffers wird hinter der letzten Zeile eingefügt.

Sed-Skripte

Kompliziertere und häufig benötigte *Sed*-Aufrufe schreibt man besser in eine Datei. Ein Aufruf des Editors sieht dann wie folgt aus:

```
sed -f <Skript_Datei> <zu_bearbeitende_Datei>
```

Beim Schreiben eines Skripts gelten folgende Regeln:

- Beginnt eine Zeile mit einem Doppelkreuz # , so handelt es sich um einen Kommentar
- Vor und nach einem Kommando dürfen keine Leerzeichen, Tabulatoren... stehen
- Stehen mehrere Kommandos auf einer Zeile, sind sie durch Semikola voneinander zu trennen

Der Stream Editor wird das gesamte Skript auf jede Zeile der Eingabedatei anwenden.

Als Beispiel dient ein kleines Skript, dass alle deutschen Umlaute in der Eingabedatei durch den entsprechenden Unicode (für html) ersetzt.

```
user@sonne> cat umlaut
# Ersetzen der äüö... durch Unicode
s/ä/\&auml;/g
s/ü/\&uuml;/g
s/ö/\&ouml;/g
s/Ä/\&Auml;/g
s/Ü/\&Uuml;/g
s/Ö/\&Ouml;/g
s/ß/\&szlig;/g
```

Das Anwendungsbeispiel demonstriert die Möglichkeit der Kopplung von Kommandozeilenbefehlen und einem Skript:

```
user@sonne> sed -e '1,12d' -f umlaut test.txt
13      RegEx           Zeilen, die den Regul&auml;ren Ausdruck enthalten
14      1, RegEx        Von Zeile 1 bis zur ersten Zeile, die RegEx enth&auml;lt
```

Unix Werkzeuge - Die Skriptsprache gawk

Überblick

awkward heißt im Englischen soviel wie schwierig, ungünstig. Ganz so problematisch erweist sich das Erlernen der Programmiersprache *Awk* dann doch nicht, auch wenn ihre Handhabung von perfektionierter Einfachheit weit entfernt ist.

Das Gleichnis der Wortstämme ist Produkt des Zufalls, denn der Begriff *Awk* gründet sich auf den Initialen seiner Erfinder Alfred V. **A**ho, Peter J. **W**einberger und Brian W. **K**ernighan und erweiterte erstmals 1978 den Werkzeugfundus von Unix Version 7. Jener Kernighan prägte übrigens gemeinsam mit Dennies Ritchie maßgeblich die Entstehung der Programmiersprache C - wen wundern da noch die Analogien beider Sprachen?

1987 wartete *Awk* mit einer komplett überarbeiteten Fassung auf. Wesentliche Bestandteile fanden im später definierten POSIX-Standard Einzug. Der freie *Awk*-Abkömmling der GNU-Gemeinde *gawk* zeigt sich zu POSIX konform und soll uns in den folgenden Abschnitten interessieren.

Der wesentliche Unterschied von *Awk* zu anderen Programmiersprachen wie den Skriptsprachen der UNIX Shells, C oder Tcl/Tk besteht in der datenorientierten Arbeitsweise, während die typischen Vertreter prozeduraler Programmiersprachen funktionsorientiert wirken. Ein awk-Programm wirkt implizit wie eine endlose Schleife, die fortwährend durchlaufen wird, bis keine Daten mehr in der Eingabe stehen oder das Programm »bewusst« verlassen wird, d.h. der Steuerfluss ist maßgeblich durch die Daten gegeben. In den meisten anderen Programmiersprachen wird das Hauptprogramm aber einmalig initiiert und Funktionen beeinflussen den Fortschritt der Berechnung.

Awk ähnelt somit eher dem Streameditor; er vermag allerdings bedeutend mehr als die »bloße« Modifikation von Textdateien. So kennt *Awk* Variablen, Vergleiche, Funktionen, Schleifen u.a.m. und ermöglicht eine Interaktion mit dem System.

Da in Linuxinstallationen nahezu ausschließlich die GNU-Implementierung des Werkzeugs vorzufinden ist, werden wir nachfolgend immer die Bezeichnung »awk« verwenden und meinen damit eigentlich »gawk«. Zeigen die von uns präsentierten Beispiele auf Ihrem System abweichende oder gar fehlerhafte Reaktionen, so überprüfen Sie ggf., ob »awk« in ihrem System ein Link auf »gawk« ist.

Allgemeiner Programmaufbau

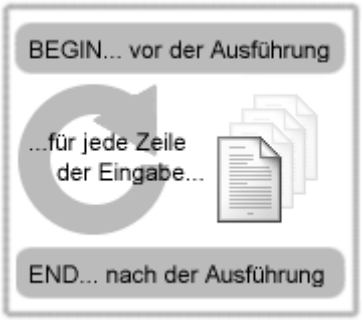


Abbildung 1: Die 3 Bestandteile eines Awk-Programms

Ein *Awk*-Programm lässt sich in **drei wesentliche Bestandteile** zerlegen:

1. Eine optionale Anweisung, die **einmalig vor** der Verarbeitung der Eingabedaten durchlaufen wird
2. Ein **Hauptprogramm**, bestehend aus beliebig vielen Anweisungen, das für jede Eingabezeile erneut ausgeführt wird, es sei denn, eine bestimmte Bedingung führt zum vorzeitigen Abbruch
3. Eine optionale Anweisung, die **einmalig nach** der Verarbeitung der Eingabedaten durchlaufen wird

Eine **Anweisung** besteht wiederum aus einem optionalen **Muster**, gefolgt von einem **Kommandoblock**, der in geschweifte Klammern eingeschlossen wird.

```
/Muster/   { Kommando [; Kommando] }
```

Der Kommandoblock darf durchaus auf mehrere Zeilen verteilt werden. Er endet mit der abschließenden geschweiften Klammer, sodass die folgende Aufteilung legitim und aus Gründen der Übersichtlichkeit bei größeren Kommandoblöcken zu empfehlen ist:

```
/Muster/   {
              Kommando;
              Kommando;
              ...
           }
```

Semikola und Zeilenumbrüche trennen Kommandos, sodass dem letzten Kommando auf einer Zeile kein Semikolon folgen muss. Da es aber auch nichts schadet, werden wir es in den weiteren Beispielen verwenden.

Die **Muster** können reguläre Ausdrücke oder Variablenvergleiche sein. Sie werden mit der aktuellen Eingabezeile verglichen, woraufhin bei Übereinstimmung der Kommandoblock betreten wird. Fehlt das Muster, wird der Kommandoblock auf jeden Fall ausgeführt.

Zwei **spezielle Muster** kennzeichen die anfänglich bzw. abschließend auszuführenden Anweisungen. Die Kommandos hinter dem Muster **BEGIN** werden zumeist zu Variableninitialisierungen oder einführenden Ausgaben genutzt. Dementsprechend ermöglicht das Muster **END** finale Aktionen, wie die Ausgabe von Ergebnissen.

Mit diesen Vorkenntnissen sollte die Funktionsweise des ersten Programms leicht nachvollziehbar sein; es zählt einfach nur die Zeilen der Eingabe und gibt das Resultat aus:

```
BEGIN   {
      print "Zählen von Eingabezeilen";
      zaehler=0;
}

{ zaehler++; }

END   { print "Ergebnis: " zaehler; }
```

Wie Sie das Programm starten können? Nach Studium des folgenden Abschnitts werden Sie es wissen...

Programmstart

Kurze Awk-Programme...

Die Möglichkeiten zum Aufruf von *Awk* sind vielfältig. Für kurze und einmalig verwendete Programme bietet es sich an, diese unmittelbar auf der Kommandozeile anzugeben:

```
awk 'Programm' <Datei> [<Datei>]
```

Das eigentliche Awk-Programm muss vor der Auswertung durch die Shell geschützt werden, deshalb die Einbettung in einfache Hochkommata. Alle folgenden Elemente der Kommandozeile werden von **awk** als Dateinamen interpretiert. Fehlt ein solcher Name, erwartet *Awk* die Daten von der Standardeingabe.

Mit Hilfe dieses Schemas lassen sich auf einfache Art und Weise Felder aus den Zeilen der Eingabe extrahieren. Zur Demonstration lassen wir das erste Feld der Passwortdatei ausgeben, wobei die Zeilen nummeriert werden (der Feldseparator ist der Doppelpunkt und ist durch die eingebaute Variable FS festgelegt):

```
user@sonne> awk 'BEGIN {FS = ":"} {print NR,$1}' /etc/passwd
1 root
2 bin
3 daemon
4 lp
5 news
...
```

Möchte man ein awk-Programm auf die Ausgabe eines Kommandos anwenden, lässt sich dies über eine Pipe realisieren:

```
Kommando | awk 'Programm'
```

Zur Demonstration »verschönert« nachfolgende Kommandofolge die Ausgabe von date:

```
user@sonne> date | awk '{print "Der " $3".",$2".", "des Jahres", $6}'
Der 3. Nov. des Jahres 2000
```

Umfangreiche Awk-Programme...

Einfache Programme erledigen i.d.R. auch nur einfachste Aufgaben. Aber die Anforderungen sind meist doch komplexer. Und so wird man umfangreiche awk-Programme in separaten Dateien nieder schreiben, so wie man in der Shellprogrammierung komplexere Konstrukte einzig in Shellskripten formuliert. *Awk* bezieht seine Instruktionen aus einer Steuerdatei, wenn deren Namen explizit mit der Option **-f** angegeben wurde:

```
awk -f <Programm.awk> <Datei> [<Datei>]
```

Bzw. bei Verwendung einer Pipe:

```
Kommando | awk -f <Programm.awk>
```

Awk-Anwendungen, die man immer wieder benötigt, wird man wohl bevorzugt in Dateien fassen; Beispiele werden uns im weiteren Text noch reichlich begegnen.

Gar noch einfacher gestaltet sich der awk-Programmaufruf bei Verwendung von **awk-Skripten**. Man bedient sich der Aufrufsyntax der entsprechenden UNIX Shell und weist die Shell an, die nachfolgenden Anweisungen dem *Awk*-Interpreter zuzuführen. Zuoberst muss einem awk-Skript nur die Zeile **#!/usr/bin/awk -f** (bzw. der korrekte Zugriffspfad zum awk-Programm) stehen. Versieht man eine solche Datei noch mit den entsprechenden Ausführungsrechten (»chmod u+x Programm.awk«), genügt ein simpler Aufruf:

```
<Programm.awk> <Datei>
```

Bzw. bei Verwendung einer Pipe:

```
Kommando | <Programm.awk>
```

Tipp: Wenn Sie beim Editieren eines awk-Skripts den vim verwenden, so stellt dieser Editor den Text mit Syntax-Highligthing dar, wenn der Skriptname auf ».awk« endet. Davon abgesehen, ist die Namensgebung des Skripts

Ihnen überlassen.

Die Kommandozeilenoptionen

Einige Optionen wurden schon verwendet, ohne ihre konkrete Bedeutung zu erläutern. Dies soll nun nachgeholt werden:

-F *Feldtrenner*

awk arbeitet auf Feldern einer Eingabezeile. Normalerweise dient das Leerzeichen/Tabulator zur Trennung einzelner Felder. Mit der Option -F wird der Wert der internen Variable **FS** (field separator) verändert.

Das bereits erwähnte Beispiel zur Ausgabe des ersten Feldes der Passwortdatei, indem das BEGIN-Muster zum Setzen des Feldtrenners genutzt wurde, lässt sich somit auch wie folgt realisieren:

```
user@sonne> awk -F: '{print NR,$1}' /etc/passwd
1 root
2 bin
3 daemon
4 lp
5 news
...
```

-v *Variable=Wert*

Eine im Programm verwendete Variable kann somit »von außen« initialisiert werden (eine interne Initialisierung wird damit nicht überschrieben).

-f *Programmdatei*

awk liest den Quellcode aus der angegebenen Datei.

-W compat

GNU awk verhält sich wie UNIX awk, d.h. die GNU-Erweiterungen werden nicht akzeptiert.

-W help

Eine Kurzhilfe erscheint.

-W posix

GNU awk hält sich exakt an den POSIX-Standard.

Kommandozeilenparameter

Awk' s Umgang mit Kommandozeilenparametern ist etwas eigenwillig. Dem C-Programmierer sind sicherlich die Variablennamen **argc** und **argv** geläufig, die bevorzugt gewählt werden, um Kommandozeilenargumente an das Hauptprogramm zu übergeben. *Awk* verfügt über zwei *builtin*-Variablen **ARGC** und **ARGV**, die die Anzahl auf der Kommandozeile stehenden Parameter (ARGC) angeben und den Zugriff auf jene über ein Feld (ARGV) ermöglichen.

»Und was soll daran verwirrend sein?« *Awk* zählt seine eigenen Kommandozeilenoptionen nicht mit. Dazu ein Beispiel (arguments.awk):

```
#!/usr/bin/awk -f

BEGIN  {
    print "Anzahl Argumente: ", ARGC;
    for (i=0; i < ARGC; i++)
       print i, ". Argument: ", ARGV[i];
}
```

Wie auch in den nachfolgenden Programmfragmenten verzichten wir an dieser Stelle auf eine Diskussion verwendeter Kontrollkonstrukte und eingebauter Variablen. Wir kommen später darauf zurück.

Achten Sie in den folgenden Testläufen auf die Reihenfolge der Argumente:

```
user@sonne> ./ arguments.awk -F : -W posix n= 3 "Die Linuxfibel"
Anzahl Argumente: 3
0. Argument: awk
1. Argument: n= 3
2. Argument: Die Linuxfibel
user@sonne> ./ arguments.awk n= 3 -F : -W posix "Die Linuxfibel"
Anzahl Argumente: 7
0. Argument: awk
1. Argument: n= 3
2. Argument: -F
3. Argument: :
4. Argument: -W
5. Argument: posix
6. Argument: Die Linuxfibel
```

Aus dem Beispiel sind mehrere Regeln abzuleiten:

1. **ARGC** ist mindestens 1 (da der Programmname immer als erstes Argument übergeben wird)
2. Die Indizierung der Argumente in ARGV beginnt bei 0
3. *Awk* übernimmt die eigenen Aufrufoptionen nicht in die Argumentenliste
4. Sobald *Awk* ein Argument als »nicht-eigene Option« erkennt, behandelt es alle weiteren Argumente als »fremde Optionen« (diese verlieren damit auch ihre übliche Wirkung)
5. Für *Awk* ist jedes Argument zunächst der Name einer Datei

Letztere Eigenschaft ist sofort anhand von Fehlermeldungen ersichtlich, sobald das Beispielprogramm »normale Anweisungen« (außer BEGIN und END) umfasst.

Argumente würden allerdings jeglichen Nutzen entbehren, ließen sie sich nicht doch vor **Awk's** Interpretation schützen. Da *Awk* die so übergebenen Dateien erst mit dem Eintritt in die Hauptschleife zu öffnen versucht, bleibt das BEGIN-Muster als der Ort der Einflussnahme übrig. Ein Argument sollte hier ausgewertet und anschließend aus dem Array ARGV gelöscht werden. Das folgende Beispiel nutzt das erste Kommandozeilenargument, um die eingebaute Variable **FS** neu zu belegen:

```
#!/usr/bin/awk -f

BEGIN  {
    if (ARGC > 1) {
        FS= ARGV[1];
        delete ARGV[1]
    }
}
...
```

Nehmen Sie das Beispiel nicht zu ernst... den *Field Separator FS* würde jeder erfahrene Awk-Programmierer mittels der Option -F setzen. Überhaupt ist im Beispiel die Annahme einer festen Position des Arguments eine unsaubere Methode und sollte vermieden werden. Besser wäre eine positionsunabhängige Behandlung aller Argumente.

Dienen Argumente einzig dazu, im Programm verwendeten Variablen initiale Werte zuzuweisen, so kann dies vorteilhaft erfolgen, indem dem Variablennamen auf der Kommandozeile der Startwert per Gleichheitszeichen zugewiesen wird. Allerdings stehen derartige Variablen erst in der Hauptschleife und *nicht* im BEGIN-Block zur Verfügung (es sei denn, auf sie wird über ARGV zugegriffen).

Als Beispiel dient eine awk-basierte Variante des Kommandos »head«, das die ersten *n* Zeilen der Eingabe (10 in der Voreinstellung) ausgibt:

```
#!/usr/bin/awk -f

BEGIN  {
    n=10;
}
```

```
{  if (n < FNR) exit;  }
{  print $0;  }
```

Speicherten wir nun das Programm unter dem Namen »head.awk« (chmod nicht vergessen!) und bringen es ohne weitere Argumente zur Ausführung, so werden - gemäß der Voreinstellung »n=10« im BEGIN-Muster - die ersten 10 Zeilen der Eingabedatei ausgegeben:

```
user@sonne> ./ head.awk < Datei>
```

Um eine abweichende Anzahl Zeilen zur Ausgabe zu bringen, muss der Variablen *n* beim Kommandoaufruf der entsprechenden Wert mitgegeben werden:

```
user@sonne> ./ head.awk n= 3 head.awk
#!/usr/bin/awk -f

BEGIN  {
```

Die Zuweisung an die Variable muss vor dem Dateinamen stehen; im anderen Fall wäre *n* noch nicht bekannt, wenn *Awk* die Datei betrachtet. Mehrere Variablen lassen sich durch Leerzeichen getrennt angeben. Zwischen Variablennamen und Wert darf sich nur das Gleichheitszeichen befinden (auch keine Leerzeichen!):

```
# Syntaxfehler!
user@sonne> ./ head.awk n =  3 head.awk
awk: ./head.awk:4: fatal: cannot open file `n' for reading (No such file or directory)
```

Reguläre Ausdrücke in Mustern

Ein Muster ist die maßgebliche Methode, um den Programmfluss von *Awk* zu steuern. Nur wenn der Inhalt des aktuell bearbeiteten Datensatzes mit dem angegebenen Muster übereinstimmt, wird der zugehörige Kommandoblock ausgeführt.

Um dieses Schema flexibel zu gestalten, verhelfen **reguläre Ausdrücke** zur Formulierung der Muster. Anstelle starrer Vergleichsmuster treten Platzhalter, sodass von der Eingabe quasi nur noch eine »Ähnlichkeit« mit dem Muster gefordert wird, um die Kommandofolge anzuwenden. Die von *Awk* unterstützten regulären Ausdrücke sind:

^

Die betrachtete Eingabe beginnt mit dem Muster

.

Ein beliebiges Zeichen (auch Zeilenumbruch)

$

Die betrachtete Eingabe endet mit dem Muster

Beispiel: Um alle Leerzeilen der Datei /etc/inittab auszugeben, hilft:

```
user@sonne> awk '/ ^ $/  { print "Zeile" FNR "ist leer"} ' / etc/ inittab
Zeile 15 ist leer
Zeile 18 ist leer
Zeile 22 ist leer
```

Die interne Variable **FNR** enthält die Nummer der aktuell bearbeiteten Zeile.

*****

Kein oder mehrere Auftreten des vorangegangenen Zeichens

+

Ein oder mehrere Auftreten des vorangegangenen Zeichens

Beispiel: Steht eine gültige Zahl (Integer) in der Eingabe?

```
user@sonne> awk '/^[[:digit:]]+$/ {print "Ja!"}'
123457900
Ja!
12a
[Ctrl]-[D]
```

Die enthaltene Anwendung von Zeichenklassen ([:digit:]) wird später behandelt.

?

Ein oder kein Auftreten des vorangegangenen Zeichens

[abc]

Genau ein Zeichen aus der Menge

[^abc]

Kein Zeichen aus der Menge

[a-z]

Ein Zeichen aus dem Bereich (hier der Kleinbuchstaben)

a|b

Alternative (entweder a oder b)

(ab)+

Mindestens ein Auftreten der Mengen »ab«

\?

Zeichen ? (ein vorgestellter Backslash hebt die Bedeutung eines jeden Sonderzeichens auf)

Metazeichen vs. Regulärer Ausdruck

Verwechseln Sie die regulären Ausdrücke nicht mit den Metazeichen der Shells! Zwar ist das Funktionsprinzip identisch, selten aber die konkrete Semantik der Zeichen (bspw. * oder ?):

```
# [Shell] Liste alle mit "a" beginnenden Dateien auf:
user@sonne> ls -1 a*
a.out
ascii2short.c
awk.txt
# [awk] Suche in der Ausgabe von "ls" nach mit "a" beginnenden Dateien
user@sonne> ls | awk '/^a+/ {print}'
a.out
ascii2short.c
awk.txt
```

Das Beispiel verdeutlicht die unterschiedliche Syntax zwischen Shell und awk bei der Realisierung derselben Aufgabe. Ersetzen Sie einmal das awk-Muster durch »/a*/« und vergleichen die Resultate!

Muster und Zeichenketten

Bislang arbeitete die Mustererkennung stets über die gesamte Eingabezeile. Mit

```
"Zeichenkette ~ /Muster/"
```

lässt sich dieses Schema auf beliebige Zeichenketten ausdehnen.

So extrahiert nachfolgendes Awk-Programm alle Benutzernamen aus der Datei /etc/passwd, deren Heimatverzeichnisse unterhalb von /home liegen:

```
user@sonne> awk -F':' '$6 ~ /^\/home/ {print $1}' /etc/passwd
```

```
tux
user
```

Die Zeichenkette ist hier das 6.Feld ($6) der Passwortdatei (Heimatverzeichnis); die Syntax des Musters sollten Sie sich anhand der einleitenden Tabelle der regulären Ausdrücke selbst erklären können. Felder und deren Indizierung sind Gegenstand des folgenden Abschnitts.

Der Zeichenkettenvergleich ist ebenso hilfreich, wenn Sie das zu betrachende Muster flexibel gestalten möchten. Die Zeichenkette ist dann die aktuell bearbeitete Zeile ($0, wird später behandelt); als Muster dient der Inhalt einer Variablen (*muster* im Beispiel):

```
user@sonne> cat sinnlos.awk
#!/usr/bin/awk -f

$0 ~ muster { print $0; }
```

Die Variable *muster* wird dann als Kommandozeilenargument geeignet belegt:

```
user@sonne> ./ sinnlos.awk muster='.+' <Datei>
```

Zeichenklassen

Der Begriff »Wort« ließe sich einfach als eine Folge von Buchstaben umschreiben; eine »ganze Zahl« demzufolge als Ansammlung von Ziffern. Ein simples awk-Programm, das Zeichenketten aus der Eingabe in die Kategorien »Wort« oder »Zahl« eingliedert, könnte wie folgt verfasst sein:

```
# Testet die Eingabe auf Wort, Zahl oder Sonstiges.
/[0-9]+/     { print "Eine Zahl" }
/[A-Za-z]+/  { print "Ein Wort" }
```

Auf den ersten Blick ist kein Fallstrick zu erkennen. Wer allerdings garantiert, dass ein Skript tatsächlich nur in Umgebungen eingesetzt wird, die denselben Zeichensatz verwenden wie der Skriptentwickler? Nicht in jedem Zeichensatz bilden Ziffern bzw. Buchstaben eine zusammenhängende Folge. Und Angaben »[von-bis]« beruhen genau auf jenem Prinzip.

Um Skripte portabel (nach POSIX-Standard) zu halten - und mitunter auch kürzer - sollte daher **Zeichenklassen** der Vorzug vor Bereichsangaben gegeben werden. Es ist dann Aufgabe der konkreten Awk-Implementierung, eine Zeichenklasse auf den verwendeten Zeichensatz abzubilden. Unser Beispiel mit Zeichenklassen schreibt sich dann so:

```
# Testet die Eingabe auf Wort, Zahl oder Sonstiges.
/[[:digit:]]+/  { print "Eine Zahl" }
/[[:alpha:]]+/  { print "Ein Wort" }
```

Vorausgesetzt Ihr System wurde sauber konfiguriert (Belegung der Shellvariablen $LANG), sollten fortan auch die deutschen Umlaute korrekt erfasst werden:

```
user@sonne> echo Überprüfung |  awk '/[[:alpha:]]+/ { print "Ein Wort";}'
Ein Wort
```

Darüber hinaus existieren weitere Zeichenklassen:

[:alnum:]
: Alphanumerische Zeichen

[:alpha:]
: Alphabetische Zeichen.

Um zu testen, ob in einer Variablen eine gültige Zahl (ganzzahlig) gespeichert ist, bietet sich folgendes Konstrukt an:

```
user@sonne> echo "0815" | awk '/ ^[[:digit:]]+$/ {print "eine Zahl"}'
Eine Zahl
```

[:blank:]

Leerzeichen und Tabulatoren

[:cntrl:]

Steuerzeichen

[:digit:]

Nummerische Zeichen

[:graph:]

Druck- und sichtbare Zeichen (Ein Leerzeichen ist druckbar aber nicht sichtbar, wogegen ein »a« beides ist)

[:lower:]

Kleingeschriebene alphabetische Zeichen

[:print:]

Druckbare Zeichen (also keine Steuerzeichen)

[:punct:]

Punktierungszeichen *Punctuation characters* (Zeichen die keine Buchstaben, Zahlen, Steuerzeichen oder Leerzeichen sind) (".", "," ":")

[:space:]

Druck- aber keine sichtbaren Zeichen (Leerzeichen, Tabulatoren, Zeichenende ..)

[:upper:]

Großbuchstaben

[:xdigit:]

Hexadezimale Zeichen

Datenfelder und Variablen

Die wichtigen eingebauten Variablen

Awk arbeitet unter der Annahme, dass die Eingabe strukturiert ist. Im einfachsten Fall interpretiert **awk** jede Eingabezeile als Datensatz und jedes enthaltene Wort als **Feld**. Ein Wort ist dabei jede von Feldseparatoren begrenzte Zeichenfolge. In der Voreinstellung trennen Leerzeichen und Tabulatoren Wörter; durch Belegung der *builtin*-Variablen **FS** lassen sich beliebige Separatoren definieren.

Über den »Feldoperator $« gestattet *Awk* den Zugriff auf die Felder der zuletzt eingelesenen Eingabezeile. Im Unterschied zur Programmiersprache C beginnt die Nummerierung der Felder allerdings bei 1 und - im Gegensatz zu den Positionsparametern der Shell - endet sie *nicht* bei 9 (implementierungsabhängig werden 100 Felder garantiert; *gawk* auf x86-Architektur kann vermutlich 2^{32} Felder indizieren). Um bspw. das 1. und 11. Feld einer Eingabezeile auszugeben, hilft Folgendes:

```
user@sonne> echo "0 1 2 3 4 5 6 7 8 9 10 11" | awk '{print $1,$11;}'
0 10
```

Der Feldoperator **$0** steht für die gesamte Zeile; die Anzahl der Felder der aktuell bearbeiteten Zeile ist in der Variablen **NF** gespeichert.

```
user@sonne> echo "0 1 2 3 4 5 6 7 8 9 10 11" | awk '{print NF,$11;}'
12 0 1 2 3 4 5 6 7 8 9 10 11
```

Die aktuelle Zeilennummer hält *Awk* in der Variablen **NR**. Um die Zeilen einer Datei zu nummerieren, könnte somit

folgender Aufruf dienen (spontan ist Ihnen sicherlich »nl« für diesen Zweck eingefallen?):

```
user@sonne> awk '{ print NR,$0;}' < zu_nummerierende_Datei>
```

Bei aufmerksamer Betrachtung des einführenden Beispiels »head.awk« ist Ihnen vermutlich aufgefallen, dass wir dort **FNR** anstatt **NR** zur Nummerierung verwendeten. Der Unterschied ist, dass Letzteres (NF) fortlaufend die Anzahl der Durchläufe der Hauptschleife zählt, während FNR im Falle mehrerer Eingabedateien die Zählung für jede Datei von vorn beginnt. Um die Worte mit einem Beispiel zu untermauern, implementieren wir »wc -l« mit Mitteln von *Awk*, wobei wir außerdem die Variable **FILENAME** verwenden, die den Namen der aktuell bearbeiteten Datei enthält :

```
#!/usr/bin/awk -f

BEGIN   { zeile=0; }
{
  if ( zeile > FNR ) {
    print n,FILENAME;
    n=0;
  }
  else n++;
}
END   {
  print FNR, FILENAME
  if ( FNR != NR ) { print NR, "insgesamt"; }
}
```

Als Testfall wenden wir das kleine Programm auf sich selbst an und vergleichen die Ausgabe mit der von »wc -l«:

```
user@sonne> ./ wc-l.awk wc-l.awk wc-l.awk
15 wc-l.awk
15 wc-l.awk
30 insgesamt
user@sonne> wc -l wc-l.awk wc-l.awk
    15 wc-l.awk
    15 wc-l.awk
    30 insgesamt
```

Abgesehen von einem »kleinen Formatierungsproblem«, das wir später beheben werden, und einem Sonderfall (finden Sie ihn selbst heraus) verhält sich unser Programm exakt wie das Vorbild.

Eher unüblich ist die Manipulation der Variablen **RS**, die den Zeilenseparator spezifiziert. Sinn macht es für Datensätze, die über mehrere Zeilen verteilt stehen und wobei ein anderes Zeichen (bspw. eine Leerzeile) eine eindeutige Strukturierung erlaubt (ein anderes Beispiel zur Anwendung finden Sie im Abschnitt zur Bashprogrammierung (Skript »Symbolsuche«)). Das die Ausgabe von *Awk* betreffende Pedand zu RS ist **ORS**. In der Voreinstellung dient der Zeilenumbruch zur Separierung der Ausgaben; durch Neubelegung von ORS kann dies geändert werden. Nachfolgender Kommandoaufruf ersetzt die Unix-Dateiendung (Newline) durch das DOS-Äquivalent (Carriage Return, Newline):

```
user@sonne> awk 'BEGIN { ORS= "\r\n";} { print $0;}' < Datei>
```

Natürlich hätte es auch »recode« getan...

Neben ORS zum Ändern des Zeilenseparators existiert mit **OFS** eine Variable, die das Trennzeichen für einzelne Datenfelder (Voreinstellung: Leerzeichen) festlegt.

Weitere eingebaute Variablen

Die weiteren von *Awk* verwendeten internen Variablen sollen an dieser Stelle nur kurz benannt werden. Einigen Vertretern werden wir später erneut begegnen.

CONVFMT

Diese Variable steuert die Konvertierung von Zahlen in Zeichenketten. Die Voreinstellung »%.6g« bewirkt, dass Gleitkommazahlen mit einer Genauigkeit von bis zu 6 Ziffern konvertiert werden. Die zulässigen Werte sind genau jene, die auch »printf« zur Zahlenausgabe verwendet:

```
user@sonne> awk 'BEGIN { print "Pi = " 3.1415926532; }'
Pi = 3.14159
user@sonne> awk 'BEGIN { CONVFMT="%.10g"; print "Pi = " 3.1415926532; }'
Pi = 3.141592653
```

ENVIRON

Die Feldvariable ENVIRON enthält alle Umgebungsvariablen. Der Index ist hierbei der Name der Umgebungsvariablen:

```
user@sonne> awk 'BEGIN { print ENVIRON["HOME"]; }'
/home/user
```

IGNORECASE

Steuert die Unterscheidung von Klein- und Großschreibung beim Mustervergleich. Steht die Variable auf »0« sind bspw. "ab" und "Ab" unterschiedliche Zeichenketten. Bei jedem Wert ungleich »0«, spielt Klein- und Großschreibung keine Rolle.

OFMT

OFMT steuert das Ausgabeformat von Zahlen im Zusammenhang mit der Ausgabe durch das Kommando **print**. Die Verwendung erfolgt analog zu CONVFMT.

RLENGTH

Im Zusammenhang mit der Zeichenkettenfunktion **match()** gibt diese Variable die Länge der übereinstimmenden Teilzeichenkette an (später dazu mehr).

RSTART

Im Zusammenhang mit der Zeichenkettenfunktion **match()** gibt diese Variable den Index des Beginns der übereinstimmenden Teilzeichenkette an (später dazu mehr).

SUBSEP

Das Zeichen, das in Feldvariablen die einzelnen Elemente trennt (\034).

Eigene Variablen

Awk unterscheidet einzig (Gleitkomma-)Zahlen und Zeichenketten. Von welchem Typ eine Variable ist, hängt einzig vom aktuellen Kontext ab.

Findet eine numerische Berechnung statt, werden die beteiligten Variablen als Zahlen interpretiert und bei Zeichenkettenoperationen eben als Zeichenketten. Notfalls lässt sich ein konkreter Kontext erzwingen, indem bspw. der Wert '0' zu einer Variable addiert oder die leere Zeichenkette "" an eine solche angehangen wird.

Für den Awk-Programmierer von Interesse sind die internen Konvertierungen, die beim Vergleich von Variablen stattfinden. Zwei numerische Variablen werden als Zahlen verglichen, genauso wie zwei Zeichenkettenvariablen als Zeichenketten verglichen werden. Ist eine Variable eine Zahl und die andere eine numerische Zeichenkette, so wird letztere in eine Zahl konvertiert. Handelt es sich um keine numerische Zeichenkette, wird die Variable mit der Zahl in eine Zeichenkette gewandelt und nachfolgend ein Vergleich der Zeichenketten vorgenommen.

Variablennamen in *Awk* dürfen aus Buchstaben, Ziffern und dem Unterstrich bestehen, wobei zum Beginn keine Ziffer stehen darf. Klein- und Großschreibung werden unterschieden, sodass bspw. *Variable*, *variable*, und *VARIABLE* unterschiedliche Bezeichner sind.

Eine Variable wird definiert, indem sie benannt und ihr ein Wert zugewiesen wird. Die in anderen Programmiersprachen üblichen Typbezeichner kennt *Awk* nicht. Eine Variable kann auch nur deklariert werden. *Awk* initialisiert sie dann selbsttätig mit der leeren Zeichenkette.

Operatoren

Awk kennt nahezu das komplette Repertoire an Operatoren, die die Programmiersprachen zur Manipulation von Ausdrücken zur Verfügung stellen.

Arithmetische Operatoren

Als arithmetische Operatoren beinhaltet *Awk*:

- **+** Addition
- **-** Subtraktion
- ***** Multiplikation
- **/** Division
- **%** Modulo
- **^** Potenzieren (Posix)
- ****** Potenzieren (gawk u.a.)

Da letzterer Potenzoperator (**) nicht im POSIX-Standard enthalten ist, muss eine konkrete Awk-Implementierung diesen nicht zwangläufig verstehen. Verwenden Sie daher stets »^ «, um Ihre Awk-Skripte portabel zu halten.

Im Falle der Division sollten Sie prüfen, dass der Divisor nicht 0 wird, da *Awk* sonst mit einem Ausnahmefehler das Programm abbricht:

```
user@sonne> awk 'BEGIN { x=5; y; print x/y; }'
awk: cmd. line:1: (FILENAME=- FNR=1) fatal: division by zero attempted
```

Für arithmetische Operationen gelten die üblichen Vorrangregeln.

```
user@sonne> awk 'BEGIN { print 5*4+20, 5*(4+20); }'
40 120
```

Zuweisungsoperatoren

Das Gleichheitszeichen zur Zuweisung eines Wertes an eine Variable sollte hinreichend bekannt sein; dem C erfahrenen Programmierer sollten auch die folgenden Operatoren vertraut erscheinen:

x++ bzw. ++x	Kurzschreibweise für: x = x + 1 (vergleiche Beispiel im Anschluss)
x-- bzw. --x	Kurzschreibweise für: x = x - 1 (vergleiche Beispiel im Anschluss)
x+=y	Kurzschreibweise für: x = x + y
x-=y	Kurzschreibweise für: x = x - y
x*=y	Kurzschreibweise für: x = x * x
x/=y	Kurzschreibweise für: x = x / y
x%=y	Kurzschreibweise für: x = x % y
x^=y	Kurzschreibweise für: x = x ^ y
x=y**	Kurzschreibweise für: x = x ** y

Mit Ausnahme von Inkrement- und Dekrement-Operatoren bleibt es Ihnen überlassen, ob Sie die Kurzform der ausführlichen Schreibweise vorziehen. Für Inkrement bzw. Dekrement entsprechen genau genommen nur *++x* und *--x* (Prefix-Operatoren) der Semantik der langen Darstellung. Denn nur hier werden die Zuweisungen zuerst ausgeführt und erst anschließend der Wert von *x* evaluiert. Als Postfix-Operatoren (*x++, x--*) verwendet, geht der alte Wert der Variable *x* in einem Ausdruck ein und erst nach Auswertung erfolgt die Zuweisung des neuen Wertes an *x*. Die folgenden Beispiele sind vermutlich einleuchtender als mein Erklärungsversuch:

```
user@sonne> awk 'BEGIN { x=5; print x = x + 1, x; }'
6 6
user@sonne> awk 'BEGIN { x=5; print x++, x; }'
```

```
5 6
user@sonne> awk 'BEGIN { x=5; print ++x, x; }'
6 6
```

Das folgende Beispiel demonstriert, dass tatsächlich in scheinbar gleichwertigen Ausdrücken abweichende Resultate erzielt werden:

```
user@sonne> awk 'BEGIN { while ( ++x < 5 ) print x }'
1
2
3
4
user@sonne> awk 'BEGIN { while ( x++ < 5 ) print x }'
1
2
3
4
5
```

Das nächste Beispiel zählt die Dateien im aktuellen Verzeichnis und berechnet ihren Speicherbedarf:

```
user@sonne> ls -l | awk '{ ++files; sum+=$5 } END { print $sum, "Bytes in", files, "Dateien"; }'
```

Vergleichsoperatoren

- **<** Kleiner als
- **>** Größer als
- **<=** Kleiner als oder gleich
- **>=** Größer als oder gleich
- **==** Gleich
- **!=** Ungleich
- **~** Übereinstimmung
- **!~** Keine Übereinstimmung

Der Zweck der ersten 6 Operatoren sollte leicht erkenntlich sein. Beachten Sie, dass der Test auf Gleichheit »==« anstatt »=« verwendet. Eine Verwechslung wäre kein Syntaxfehler (und ist daher als Fehler schwer zu erkennen), führt aber zu abweichenden Ergebnissen!

Bei den Operatoren ~ und !~ dürfen als rechtsseitige Operanden auch reguläre Ausdrücke stehen, während bei allen anderen Operatoren einzig Variablen und Konstanten zulässig sind.

```
# Finde alle Benutzer, deren Heimatverzeichnis unter /home liegt:
user@sonne> awk '$6 ~ /^\/home.*/ { print $1; }' /etc/passwd
user
tux
```

Logische Operatoren

- **&&** Logisches UND
- **||** Logisches ODER
- **!** Negation

Logische oder boolesche Operatoren dienen maßgeblich zur Verknüpfung mehrerer Vergleichsoperatoren oder mehrerer regulärer Ausdrücke.

Das wenig konstruktive Beispiel listet alle Dateien des aktuellen Verzeichnisses auf, die zwischen 4096 und 8192

Bytes groß sind:

```
user@sonne> ls -l | awk '$5 >= 4096 && $5 <= 8192 { print $NF; }'
Linuxfibel
linuxbuch
```

Ein weiteres Beispiel veranschaulicht die Verwendung der Negation, indem im aktuellen Verzeichnis alle Dateien gesucht werden, die nicht dem aktuell angemeldeten Benutzer gehören:

```
user@sonne> ls -l | awk 'FNR > 1 && !($3 == ENVIRON["USER"]) { print $NF; }'
```

Natürlich hätten wir auch gleich mit »!=« vergleichen können...

Verknüpfen Sie mehrere Ausdrücke per logischer Operatoren, so kann die Priorität der Auswertung eine Rolle spielen. Eine Negation wird vor dem logischen UND und dieses vor dem logischen ODER betrachtet. Nutzen Sie die Klammerung, um ggf. eine andere Bearbeitungsreihenfolge zu erzwingen.

Kontrollstrukturen

Der bislang kennengelernte Mustervergleich arbeitet über die Eingabedaten und regelte, ob für den aktuellen Datensatz Aktionen durchzuführen waren. Die im weiteren Text vorgestellten Konstrukte gestatten quasi den Kontrollfluss innerhalb einer solchen Aktionsfolge.

Bedingte Ausführung

if-else

Wohl jede Programmiersprache verfügt über ein *if-else*-Konstrukt und **Awks** Realisierung orientiert sich exakt an der C-Syntax:

```
if (Bedingung) {
    # diese Aktionen
}
[ else {
    # jene Aktionen
} ]
```

Evaluiert die Bedingung zu *wahr* (!=0), so werden die Aktionen des **if**-Zweigs ausgeführt, anderenfalls wird der optionale **else**-Zweig betreten. Folgt einem Zweig nur eine einzelne Aktion, können die geschweiften Klammern entfallen.

Das nachfolgende Beispiel widerspricht eigentlich dem Einsatzzweck von Awk, bedarf es doch keinerlei Eingabedaten. Es testet, ob das Skript mit (mind.) einem Argument aufgerufen wurde. Fehlt dieses, endet das Skript mit einer Fehlerausgabe. Anderenfalls wird der Wert des ersten Arguments potenziert:

```
user@sonne> cat potenz.awk
#!/usr/bin/awk -f

BEGIN {
   if (ARGV[1] == "") {
      print "Benutze: ./potenz.awk 'beliebiger_Wert'";
      exit 1;
   }
   else {
       print "x^ x =", ARGV[1]**ARGV[1];
   }
}
```

Anmerkung: Während der Abhandlung zu Variablen wurde erwähnt, dass nicht initialisierte Variablen intern mit »0« (numerischer Kontext) bzw mit »""« (Zeichenkettenkontext) belegt werden. Da »0« eine erlaubte Eingabe des Skripts darstellt, erzwingen wir mit dem Vergleich mit der leeren Zeichenkette letzteren Kontext.

Awk kennt kein zu *case* äquivalentes Konstrukt, sodass das zur Bewertung mehrerer Bedingungen verschachtelte if-else-Anweisungen verwendet werden:

```
if (Bedingung 1) {
    # Aktionen
}
else if (Bedingung 2) {
    # Aktionen
}
[...]
else {
    # Aktionen
}
```

Bei mehreren erfüllten Bedingungen werden einzig die Aktionen der ersten zutreffenden ausgeführt. Wird keine Bedingung wahr, werden alle Aktionen des optionalen else-Zweigs abgearbeitet.

Conditional-if

Eine brauchbare deutsche Übersetzung dieses Konstrukts ist mir nicht bekannt. *Bedingte Zuweisung* wäre vermutlich eine sinnvolle Umschreibung...

Manchmal soll einer Variable in Abhängigkeit vom Ergebnis einer Bedingung ein Wert zugewiesen werden. Mit »if-else« wäre das folgende Fragment denkbar:

```
if (Bedingung)
    variable = Wert_1
else
    variable = Wert_2
```

Prägnanter ist die Schreibweise des Conditional-if:

```
variable = Bedingung ? Wert_1 :Wert_2
```

Letzlich bleibt es Ihnen überlassen, welche der beiden Varianten Sie wählen. Für meinen Geschmack unterstreicht letztere Schreibweise deutlicher das Ansinnen, einer Variable einen Wert zuzuweisen. Da conditional-if im Gegensatz zu if-else einen Wert liefert, finden sich auch hin und wieder Situationen, wo eine Umschreibung per if-else umständlich, wenn nicht gar unmöglich ist:

```
user@sonne> awk 'BEGIN { x=1; while (x<=3) printf("%d Osterei%s\n", x, x++>1 ?"er":"") }'
1 Osterei
2 Ostereier
3 Ostereier
```

printf wird im Abschnitt Ein/Ausgabe behandelt.

Schleifen

for

Die for-Schleife wird bevorzugt, wenn eine konkrete Anzahl Durchläufe der Anweisungen im Schleifenrumpf erfolgen soll:

```
for (Zähler initialisieren; Zähler testen; Zähler verändern) {
```

```
    # Aktionen
}
```

Die geschweiften Klammern dürfen entfallen, wenn nur eine einzelne Aktion zum Schleifenrumpf gehört.

Die mit *Zähler initialisieren*, *Zähler testen* und *Zähler verändern* bezeichneten Ausdrücke beschreiben den »gebräuchlichen« Verwendungszweck. Wie auch in C wird der erste Ausdruck (*Zähler initialisieren*) einmalig vor Eintritt in die Schleife ausgeführt. Der zweite Ausdruck (*Zähler testen*) wird jeweils vor dem nächsten Durchlauf der Schleife betrachtet und Ausdruck Nummer 3 (*Zähler verändern*) wird nach Bearbeitung der Anweisung des Schleifenrumpfes berechnet:

```
user@sonne> awk 'BEGIN { for ( i=0; i<3; i++ ) print i;}'
0
1
2
user@sonne> awk 'BEGIN { for ( i=0; i++<3; ) print i;}'
1
2
3
```

Die Beispiele zeigen auch, dass die Angabe der Ausdrücke optional ist. Analog zu C kann eine Endlosschleife wie folgt angegeben werden:

```
user@sonne> awk 'BEGIN { for (;;) printf(".") }'
```

Was *Awk* im Gegensatz zu C allerdings nicht gestattet, ist die Angabe einer kommaseparierten Liste von Ausdrücken (bspw. »for (i=0,j=0;...;...)«).

while

while-Schleifen werden bevorzugt, wenn die Anzahl der Schleifendurchläufe vom Ergebnis eines Ausdrucks abhängt.

```
while (Bedingung) {
    # Aktionen
}
```

Die geschweiften Klammern dürfen entfallen, wenn nur eine einzelne Aktion zum Schleifenrumpf gehört.

Das folgende Beispiel berechnet die Fakultät zu einer per Argument übergebenen Zahl:

```
user@sonne> cat fakultaet.awk
#!/usr/bin/awk -f

BEGIN {
    if (ARGC < 2 || (ARGV[1] !~ /^[0-9]+$/)){
        print "Aufruf: ./fakultaet.awk <Zahl>";
        exit 1;
    }

    fakultaet=1;
    zahl=ARGV[1];

    while (zahl > 1)
        fakultaet*=zahl--;

    print "Fakultaet von", ARGV[1], "ist", fakultaet;
}
```

do-while

In Situationen, in denen ein Programmabschnitt mindestens einmal - und in Abhängigkeit von einer Bedingung weitere Male - zu durchlaufen ist, bietet sich die **do-while**-Schleife an:

```
do {
   # Aktionen
} while (Bedingung)
```

Eine Anwendung könnte der Test einer Nutzereingabe sein, die solange zu wiederholen ist, bis sie dem geforderten Format entspricht:

```
do {
   printf("Geben Sie eine Zahl ein: ");
   getline zahl < "-";
} while ( zahl !~ /^[[:digit:]]+$/ )
```

getline wird im Abschnitt Ein/Ausgabe behandelt.

Weitere Kontrollanweisungen

Betrachtet man das Hauptprogramm von *Awk* als eine Schleife (die über alle Zeilen der Eingabe läuft), so besitzen die nachfolgend vorgestellten Anweisungen eine Gemeinsamkeit: Sie verändern den Kontrollfluss von Schleifen.

break und **continue** wirken mit **for** und **[do-]while** zusammen; **exit**, **next** und **nextfile** betreffen die Hauptschleife.

break

Mittels **break** kann eine for-, while- oder do-while-Schleife vorzeitig verlassen werden. Die Programmausführung fährt mit der der Schleife unmittelbar folgenden Anweisung fort.

```
user@sonne> awk 'BEGIN { for (;;) if (x++ == 5) break; print x}'
6
```

continue

Wird **continue** innerhalb einer for-, while- oder do-while-Schleife ausgeführt, wird die Bearbeitung des aktuellen Schleifendurchlaufs abgebrochen und der nächste Durchlauf begonnen.

```
user@sonne> awk 'BEGIN { for (i=1;i<10;i++) {if (i%2) continue; print i } }'
2
4
6
8
```

exit

Der **exit**-Befehl dient zum (vorzeitigen) Beenden der Hauptschleife von *Awk*. Das Programm wird unverzüglich mit Ausführung des optionalen END-Statements fortgeführt.

Als Argument kann **exit** ein Ausdruck mitgegeben werden. Der Wert des Ausdrucks ist der Rückgabewert von *Awk*. Ohne Angabe des Arguments wird implizit 0 angenommen. Eine diesbezügliche Ausnahme bildet die Verwendung von **exit** innerhalb der END-Anweisungen. Fehlt hier das Argument, gilt ein ggf. in der Hauptschleife (bzw. innerhalb von BEGIN) gesetzter Wert:

```
user@sonne> awk 'BEGIN { exit 3 }; END { exit }'; echo $?
3
user@sonne> awk 'BEGIN { exit 3 }; END { exit 0 }'; echo $?
```

```
0
```

next

next wirkt wie **continue**, nur dass jetzt der aktuelle Durchlauf des Hauptprogramms unterbrochen und mit dem nächsten Durchlauf - sprich: mit der nächsten Zeile der Eingabe - fortgefahren wird.

nextfile

Diese Erweiterung von *GNU-Awk* (nicht POSIX) wirkt wie **next**, d.h. es wird zum Beginn des Hauptprogramms gesprungen. Die aktuelle Datei wird geschlossen und als Datensatz die erste Zeile der nächsten Eingabedatei geladen.

Ein- und Ausgabe

Eingabe

Das Erfassen von Nutzereingaben ist nur eine Anwendungsvariante der Funktion **getline**. Und genau genommen vermag sie das auch nur, weil ihr die Standardeingabe als Quelldatei untergeschoben werden kann.

```
getline [variable] [< "<Datei>"]


<Kommando> | getline [variable]
```

Im einfachsten Fall des Funktionsaufrufs *ohne Argumente* lädt **getline** einfach die nächste Eingabezeile. Alle im Skript folgenden Anweisungen arbeiten folglich mit diesem neuen Datensatz. In dieser Form manipuliert **getline** die Awk-Variablen *$0, NR, NF* und *FNR*

Folgt dem Funktionsaufruf eine Variable, so landet der Inhalt der Eingabe in dieser. Der Arbeitspuffer mit dem aktuellen Datensatz (*$0*) wird hierbei nicht verändert, jedoch werden die gelesenen Daten aus der Eingabe entfernt. Das folgende Beispiel nutzt das Verhalten, um jeweils zwei Zeilen der Eingabe zu einer Zeile in der Ausgabe zusammenzufassen:

```
user@sonne> cat 2to1.awk
#!/usr/bin/awk -f
{
   if ( (getline nextLine) < 1)
      nextLine="";
   print $0, nextLine;
}
```

getline liefert bei erfolgreich gelesener Eingabe eine »1« zurück. Das Beispiel nutzt den Rückgabewert, um die Variable zurückzusetzen, falls in der Eingabe eine ungerade Anzahl Datensätze steht. Eine »-1« liefert **getline** im Fehlerfall; eine »0« bei Erreichen des Dateiendes.

getline in Verbindung mit *< "<Datei>"* versucht den nächsten Datensatz aus der mit *Datei* bezeichneten Datei zu lesen. Dieser steht in *$0* zur Verfügung, *FN* enthält die Anzahl der Felder dieses Dateisatzes. Mit jedem Aufruf wird ein weiterer Datensatz aus der Datei geliefert. Um von der Standardeingabe zu lesen, ist als Dateiname »-« anzugeben:

```
user@sonne> awk 'BEGIN { getline < "-"; print "Die Eingabe war ", $0; }'
```

Anmerkung: Speziell in *gawk* steht der Dateiname »/dev/stdin« zur Verfügung, der anstatt dem Minus verwendet werden kann (siehe weiter unten im Text).

Als weitere Quelle kann **getline** seine Eingaben auch aus einer Pipe beziehen. Auf der linken Seite der Pipe muss

ein Kommando stehen, das in doppelte Anführungszeichen einzuschließen ist:

```
user@sonne> awk 'BEGIN {"date" | getline datum; close("date"); print "Aktuelles Datum: ", datum;}'
Aktuelles Datum: Mit Dez 12 21:42:35 CET 2001
```

close wird im Anschluss behandelt.

Escape-Sequenzen

Escape-Sequenzen sind nicht auf die Verwendung in Ausgaben beschränkt, werden aber in diesem Zusammenhang häufig genutzt.

\ a	Alert (Piepton)
\ b	Backspace (Cursor ein Zeichen nach links)
\ f	Formfeed (Cursor auf nächste Zeile, eine Position weiter)
\ n	Zeilenumbruch
\ r	Carriage return (Cursor an Anfang der aktuellen Zeile)
\ t	Horizontaler Tabulator
\ v	Vertikaler Tabulator
\ *ddd*	1-3 oktale Ziffern
\ x*HEX*	Hexadezimale Zahl
\ *c*	Das Zeichen *c* selbst (i.A. zur Darstellung von Sonderzeichen verwendet)

```
user@sonne> awk 'BEGIN {print "\ \ f\ f führt\ f zu\ f einem\ f Treppeneffekt:)";}'
\f
   führt
        zu
          einem
               Treppeneffekt:)
```

Ausgabe mit print

```
print  [Parameter [, Parameter] ]
print  ([Parameter [, Parameter] ])
```

print ist für die Ausgabe zu bevorzugen, wenn Sie keine Anforderungen an die Formatierung stellen. Geben Sie hierzu einfach die Liste der auszugebenden Ausdrücke, jeweils getrennt durch ein Komma, an. Jedes Element wird in der Ausgabe vom Folgenden durch ein Leerzeichen getrennt werden. Dem letzten Element lässt **print** einen Zeilenumbruch folgen (das voreingestellte Verhalten kann mittels der Variablen OFS und ORS geändert werden)

Sie können gar auf jegliche Elemente verzichten. In dem Fall gibt **print** die aktuell bearbeitet Zeile aus, d.h. zwischen »print« und »print $0« besteht bez. des Ergebnisses kein Unterschied.

Die an **print** zu übergebenden Argumente können wahlweise in runde Klammern eingeschlossen werden. Notwendig ist dies, wenn die Ausdrücke den Vergleichsoperator »>« enthalten, da dieser sonst als Ausgabeumleitung verstanden wird.

Es ist kein Syntaxfehler, wenn Sie auf die Kommata zwischen den auszugebenden Ausdrücken verzichten. *Awk* interpretiert dies als das Zusammenhängen von Zeichenketten, sodass als ersichtlicher Unterschied zumeist das trennende Leerzeichen fehlt:

```
user@sonne> awk 'BEGIN { print "foo", "bar";}'
foo bar
user@sonne> awk 'BEGIN { print "foo" "bar";}'
foobar
```

Eine begrenzte Formatierung lässt sich für numerische Werte erzwingen, indem das in OFMT gespeicherte Ausgabeformat (Voreinstellung "%.6g") geändert wird:

```
user@sonne> awk 'BEGIN { print rand();}'
0.487477
user@sonne> awk 'BEGIN { OFMT="%.8f"; print rand();}'
0.48747681
user@sonne> awk 'BEGIN { OFMT="%.1f"; print rand();}'
0.5
```

Beachten Sie die enthaltene mathematisch korrekte Rundung der Werte!

Formatierte Ausgabe mit printf

Wer C beherrscht, ist klar im Vorteil... die Syntax von **printf** kann ihre Anlehnung an die verbreitete Sprache nicht verbergen.

printf (*Formatzeichenkette*[, *Parameter(liste)*])

Zwar dürfen in *Awk* die runden Klammern auch entfallen, aber das ist fast schon der einzige Unterschied zu C. Die Formatzeichenkette kann jedes ASCII-Zeichen, Escape-Sequenzen oder einen der folgenden Platzhalter umfassen:

- **%c** Ein einzelnes ASCII-Zeichen
- **%d** Eine ganze Zahl
- **%i** Eine ganze Zahl (wie d, aber konform zu POSIX)
- **%e** Gleitkommazahl in Exponentendarstellung (2.e7)
- **%E** Gleitkommazahl in Exponentendarstellung (2.E7)
- **%f** Gleitkommazahl in Fließdarstellung
- **%g** e oder f, je nachdem, was kürzer ist
- **%G** E oder f, je nachdem, was kürzer ist
- **%o** Oktale Zahl
- **%s** Zeichenkette
- **%x** Hexadezimale Zahl mit a-e
- **%X** Hexadezimale Zahl mit A-E
- **%%** % selbst

Die Anzahl der Elemente der Parameterliste muss mindestens die Anzahl Platzhalter in der Formatzeichenkette umfassen. Überschüssige Angaben werden schlicht ignoriert. Der Typ des *i*-ten Parameters sollte zum *i*-ten Platzhalter »passen«, allerdings reagiert *Awk* recht großzügig und wandelt die Typen ggf. ineinander um. Meist weicht das Resultat aber vom Gewünschten ab.

```
user@sonne> awk 'BEGIN { printf("%i\n", 11.100); }'
11
user@sonne> awk 'BEGIN { printf("%e\v%E\n", 11.100, 11.100); }'
1.110000e+01
          1.110000E+01
user@sonne> awk 'BEGIN { printf("%f\n", 11.100); }'
11.100000
user@sonne> awk 'BEGIN { printf("%x\n", 11.100); }'
b
user@sonne> awk 'BEGIN { printf("%s\t%s\n", 11.100, "11.100"); }'
11.1    11.100
```

Eine exakte Positionierung und angepasstes Format der auszugebenden Parameter wäre mit den Platzhaltern allein nicht immer möglich, daher gestattet **printf** - C lässt grüßen - die Angabe von *Breite* und *Ausrichtung* eines Parameters sowie eines *Modifizierers*. Für Gleitkommaangaben kommt noch die Genauigkeit der Nachkommastellen hinzu. Die optionalen Angaben stehen zwischen Prozentzeichen und Formatidentifikator:

```
% ModifiziererBreite.GenauigkeitFormatidentifikator
```

Der wohl gebräuchlichste Modifizierer beeinflusst die horizontale Anordnung eines Parameters. In der Voreinstellung rechtsbündig, erzwingt ein dem Prozentzeichen folgendes Minus die linksbündige Ausrichtung:

```
user@sonne> awk 'BEGIN { for (i=0; i<6; i++) printf("%4i\n", i**i) }'
   1
   1
   4
  27
 256
3125
user@sonne> awk 'BEGIN { for (i=0; i<6; i++) printf("%-4i\n", i**i) }'
1
1
4
27
256
3125
```

Als Modifizierer stehen zur Verfügung:

-	Linksbündige Ausrichtung
+	Das Vorzeichen bei nummerischen Werten wird stets angegeben
Leerzeichen	Bei positiven nummerischen Werten wird ein Leerzeichen vorangestellt
0	Nummerische Werte werden von links her mit Nullen aufgefüllt
#	Druckt bspw. bei "%x" ein "0x" vor den Wert

```
user@sonne> awk 'BEGIN { printf ("%#x %04i %+4i\n", 12, 12, 12)}'
0xc 0012  +12
```

Von C's printf hat *Awk* ebenso die variable Angabe der Breiten- und Genauigkeitswerte übernommen. Anstatt des Wertes steht im Formatierer nun ein Stern. Der zu wählende Wert erscheint in der Argumentenliste vor dem jeweiligen Argument (Reihenfolge und Anzahl sollte übereinstimmen!):

```
# Die »übliche« Form:
user@sonne> awk 'BEGIN { printf("%12.11f\n", 11/13 ) }'
# Die »dynamische« Form:
user@sonne> awk 'BEGIN {width=12; precision=11; printf("%*.*f\n", width, precision, 53/17 ) }'
0.84615384615
```

Umleitung der Ausgabe

Die nachfolgend am Beispiel von **print** demonstrierten Mechanismen funktionieren ebenso mit **printf**.

Normalerweise landen die Ausgaben auf dem Terminal (Standardausgabe). Ihr Ziel kann mittels der schon von den Shells bekannten Umleitungen ebenso eine *Datei* oder eine *Pipe* sein.

print *Argument* > *Ausgabedatei*
: Schreiben in Datei; existiert diese, wird ihr Inhalt überschrieben

print *Argument* >> *Ausgabedatei*
: Anfügen ans Ende der Datei; existiert sie nicht, wird sie erzeugt

print *Argument* | *Kommando*
: Schreiben in eine Pipe, aus der ein Kommando liest

Anwendungsbeispiele für den Nutzen der Umleitung von Ausgaben in einer Datei finden sich reichlich in der alltäglichen Administration. Bspw. bei der Auswertung der Logdateien. Gerade bei Servern häufen sich die

Meldungen in der Datei /var/log/messages nur allzu schnell. Per Hand regelmäßig nach verdächtigen Zeilen Ausschau zu halten, wird rasch zur Last. So könnte *Awk* eine vorherige Selektion treffen und thematische Logdateien anlegen. Das folgende Beispiel zeigt eine Anwendung, die die Nachrichten des sshd und des telnetd filtert:

```
#!/usr/bin/awk -f
# 'filter.awk' filtert Meldungen des sshd und telnetd
# Aufruf: filter.awk /var/log/messages

/.*sshd.*/          { print >> ENVIRON["HOME"]"/sshd.log"}
/.*in.telnetd.*/    { print >> ENVIRON["HOME"]"/telnet.log"}
```

Verwenden Sie in einem der Argumente von print(f) den Vergleichsoperator »>«, so müssen Sie die Argumente klammern, da der Operator ansonsten als Umleitung verstanden wird!

Die Ausgabe von print(f) in eine Pipe zu speisen, bietet sich an, falls das Ergebnis durch ein Kommando weiter bearbeitet werden soll. Schließen Sie dazu das Kommando inklusive seiner Argumente in doppelte Anführungszeichen ein. Als Beispiel werden Funde von telnet-Nachrichten sofort per Mail an den lokalen Administrator gemeldet:

```
#!/usr/bin/awk -f
# 'mailer.awk' meldet Nachrichten des telnetd an Root
# Aufruf: mailer.awk /var/log/messages

/.*in.telnetd.*/    { print | "mail -s 'Telnet-Kontakt' root" }
```

Derartige Kommandoaufrufe lassen sich auch in einer Variable speichern und darüber referenzieren:

```
...
BEFEHL="mail -s 'Telnet-Kontakt' root"
...
/.*in.telnetd.*/    { print | BEFEHL }
```

Spezielle Dateinamen (nicht nur) für die Umleitung

Analog zu den Dateideskriptoren der Shells kennt *GNU-Awk* (nicht POSIX!) spezielle Dateinamen, um die Umleitung in konkrete Kanäle zu realisieren. Wenn Sie den letzten Befehl mit der Pipe als Muster hernehmen, sollte Ihnen zur Ausgabe über den Standardfehler-Deskriptor zumindest eine trickreiche Variante einfallen:

```
{ print | "cat 1>&2" }
```

Eleganter geht es mit der »Datei« /dev/stderr:

```
{ print | "/dev/stderr" }
```

Awk kennt folgende Dateien:

/dev/stdin Standardeingabe
/dev/stdout Standardausgabe
/dev/stderr Standardfehlerausgabe
/dev/fd/*x* Die mit dem Dateideskriptor *x* verbundene Datei

Ein Zugriff auf bspw. /dev/fd/5 bedingt, dass zuvor eine Datei mit dem Deskriptor verbunden wurde. Dazu ein Beispiel:

```
user@sonne> exec 5>test.log
user@sonne> awk 'BEGIN {print "Testausgabe" > "/dev/fd/5"} '
```

Nun raten Sie einmal, was in der Datei »test.log« drin steht?

Vollständigkeitshalber sei noch erwähnt, dass *Awk* einige Dateien kennt, um Informationen über die Prozessumgebung auszulesen. Dies sind:

/dev/pid	Prozess-ID
/dev/ppid	Prozess-ID des Eltern-Prozesses
/dev/pgrpid	Prozess-Gruppen-ID
/dev/user	4 Werte zum Eigentümer des Prozesses

```
user@sonne> awk 'BEGIN{ getline < "/dev/user";
> print "UID", $1
> print "EUID", $2
> print "GID", $3
> print "EGID", $4
> } '
UID 500
EUID 500
GID 100
EGID 100
```

Schließen von Dateien und Pipes

Wenn Sie im Laufe eines *Awk*-Programms mehrfach via **getline** Daten aus ein und derselben Datei lesen, so liefert ein Aufruf die jeweils nächste Zeile dieser.

Bezieht **getline** seine Daten aus einer Pipe und erfolgt auch hierbei der mehrfache Zugriff auf ein und denselben Befehl, so wird der Befehl nur einmalig ausgeführt und jeder **getline**-Aufruf bringt den nächsten Datensatz aus der Ausgabe des Befehls hervor.

Nicht immer jedoch ist dies das gewünschte Verhalten, bspw. wenn ein Befehl nur eine einzige Ausgabezeile liefert oder wir im Laufe der Berechnung an der ersten Zeile einer Datei interessiert sind:

```
user@sonne> awk 'BEGIN {
> for (i=0; i<2; i++) {
> "date" | getline x;
> print x;
> system("sleep 65")}
> }'
Don Dez 13 16:00:30 CET 2001
Don Dez 13 16:00:30 CET 2001
```

Dass im Beispiel beide Male dieselbe Zeit ausgegeben wird, war sicherlich nicht das beabsichtigte Ergebnis. Abhilfe bringt das Schließen der Pipe mit **close**:

```
user@sonne> awk 'BEGIN {
> for (i=0; i<2; i++) {
> "date" | getline x
> print x
> system("sleep 65")
> close("date") }
> }'
Don Dez 13 16:01:10 CET 2001
Don Dez 13 16:02:15 CET 2001
```

Verwenden Sie **close** im Zusammenhang mit Dateien, wenn Sie aus diesen »von vorn« lesen möchten.

Näheres zum Awk-Befehl **system** erfahren Sie unter Weiteres.

Arrays

Ein Array ist eine Variable, die einen Satz von Daten - *Elemente* genannt - aufnehmen kann. Der Elementzugriff erfolgt über einen Index, der eine Nummer oder eine Zeichenkette sein kann, wobei - im Falle von Zeichenketten - die Groß- und Kleinschreibung stets *keine* Rolle spielt, unabhängig vom Wert der internen Variablen IGNORECASE! Der Name eines Arrays darf nicht mit dem Namen einer »einfachen« Variablen kollidieren.

Die Größe eines Arrays muss nicht vorab angegeben werden. Jederzeit lassen sich weitere Elemente zum Array hinzufügen, löschen oder der Wert eines Elements verändern.

Einfügen und Löschen von Elementen

Mit jeder Zuweisung eines *Wert*es an einen neuen *Index* wächst das *Array* um ein weiteres Element. Im Falle eines existierenden Indizes wird der alte Wert durch den neuen ersetzt:

```
Array [Index] = Wert
```

Neben eindimensionalen Arrays gestattet *Awk* auch die Verwendung mehrdimensionaler Felder. Im zweidimensionalen Fall sind zwei Indizes zur Adressierung eines Elements notwendig:

```
ZweiDimensionalesArray [Index1, Index2] = Wert
```

Intern bildet *Awk* mehrdimensionale Felder auf ein eindimensionales ab, indem die einzelnen Indizes implizit zu einer einzelnen Zeichenkette verkettet werden. Um die Indizes identifizieren zu können, wird der durch **SUBSEP** definierte Separator als Trennzeichen zwischen diese gesetzt. In der Voreinstellung von SUBSEP wird bspw. aus »f[1,2]« intern ein »f["1@2"]«. Beide Angaben sind äquivalent.

Zum Entfernen eines Elements dient der **delete**-Operator. Erforderlich ist die Angabe des Indexes des zu löschenden Elements:

```
delete array [Index]
```

Wird **delete** einzig der Name eines Array übergeben - also ohne einen Index - werden sämtliche Elemente des Feldes gelöscht.

Index-Zugriff

Der Zugriff auf ein Element eines Feldes ist stets über seinen Index möglich. Dies setzt voraus, dass der Index bekannt ist. Erfolgt eines Referenzierung mittels eines unbekannten Indizes, wird eine leere Zeichenkette als Ergebnis geliefert und gleichzeitig diese als Element des Arrays angelegt:

```
user@sonne> awk 'BEGIN { f[A]="foo";
> printf("_%s_%s_\n", f[A], f[B]);}'
_foo_foo_
```

Eventuell haben Sie mit dem Verständnis des Beispiels so Ihre Probleme? Kein Wunder, es wurde bewusst ein Fehler eingebaut. Und zwar werden *A* und *B* als Namen von Variablen betrachtet und beide zur leeren Zeichenkette evaluiert. Hieraus resultiert, dass jeweils auf »f[""]« - also auf einundenselben Feldindex - zugegriffen wird. Die korrigierte Variante bringt das vorhergesagte Ergebnis:

```
user@sonne> awk 'BEGIN { f["A"]="foo";
> printf("_%s_%s_\n", f["A"], f["B"]);}'
_foo__
```

Das automatische Anlegen eines neuen Indizes im Falle dessen Nichtexistenz ist vermutlich in etlichen Fällen

unerwünscht. Aus diesem Grund bietet *Awk* eine Abfrage an, um festzustellen, ob ein Index existiert:

```
if ( Index in Array ) {
    # tue etwas
}
```

Ein komplexeres Beispiel zum Indexzugriff druckt die Felder der Datei /etc/passwd in tabellarischer Form, wobei die notwendige Spaltenbreite berechnet wird:

```
#!/usr/bin/awk -f

BEGIN   {
    FS=":";
    while ( getline < "/etc/passwd" )
        for (i=1; i<=6; i++)
            if (max[i] < length($i)) { max[i] = length($i)}

    close("/etc/passwd" )

    while ( getline < "/etc/passwd" )
        printf("%-*s %*s %*s %-*s %-*s\n",
            max[1], $1, max[3], $3,
            max[4], $4, max[5], $5,
            max[6], $6);
}
```

Scannen eines Arrays

Unter »Scannen eines Arrays« wollen wir den sequentiellen Zugriff auf alle enthaltenen Elemente in der Reihenfolge ihres Einfügens verstehen.

Im Falle eindimensionaler Felder wird in *Awk* einfach in einer **for**-Schleife eine Variable der Reihe nach mit jedem *Index* aus dem Array verbunden. Im Schleifenrumpf kann nachfolgend auf das mit dem Index verbundene Elemente zugegriffen werden.

```
for ( Index in Array ) {
    # tue etwas (mit Array[Index])
}
```

Das nachfolgende Beispiel demonstriert die Anwendung des Scannens, indem eine Liste der Häufigkeiten des Auftretens von Wörtern aus der Eingabe generiert wird:

```
#!/usr/bin/awk -f

BEGIN   {
    RS="[[:space:]]"
}
{ ++wort[$0] }
END {
    for (x in wort)
        printf("%-20s %d\n", x, wort[x]);
}
```

Ein simpler Test beweist, dass das Skript (»WorkCounter.awk« genannt) tatsächlich funktioniert:

```
user@sonne> echo drei eins drei zwei zwei drei | WordCounter.awk
zwei                 2
drei                 3
eins                 1
```

Der Zugriff auf die Elemente in einem mehrdimensionalen Feld kann analog erfolgen, wenn der Elementzugriff über

den verketteten Index genügt. Werden hingegen die originalen Indizes benötigt, ist etwas mehr Aufwand zu betreiben:

```
for ( VollerIndex in Array ) {
  split ( VollerIndex, Hilfsfeld, SUBSEP )
        # Index1 steht in Hilfsfeld[1],
        # Index2 steht in Hilfsfeld[2]) usw.
    }
}
```

Die Zeichenkettenfunktion **split** soll im folgenden Abschnitt behandelt werden. Sie trennt den Inhalt von *VollerIndex* an den durch *SUBSEP* vorgegebenen Positionen auf und speichert die einzelnen Bestandteile in *Hilfsfeld*.

Ein sinnfreies Beispiel veranschaulicht die Methodik der Index-Aufsplittung:

```
user@sonne> awk 'BEGIN {
> f[1,"foo",0815] = "egal";
> for (i in f) {
>   split (i,hf,SUBSEP);
>   for (x in hf) print hf[x]
> } }
1
foo
815
```

Eingebaute Funktionen

Der bevorzugte Einsatzbereich von *Awk* ist die automatische Generierung von Reports und Statistiken. So existieren zahlreiche eingebaute Funktionen, die *Awk* zur Auswertung von Daten jeglicher Art prädestinieren.

Eine Funktion ist gekennzeichnet durch einen Namen und der Liste der Argumente, die dem Namen, eingeschlossen in runde Klammern, folgen. Es ist bei einigen Funktionen zulässig, weniger Argumente anzugeben, als die Funktion eigentlich bedingt; welche voreingestellten Werte dann *Awk* einsetzt, unterscheidet sich von Funktion zu Funktion. Stets ein Syntaxfehler hingegen ist, mehr Argumente einer Funktion mitzugeben, als bei ihrer Definition vereinbart wurden.

Finden Ausdrücke als Argumente Verwendung, so werden diese *vor* der Übergabe an die Funktion ausgewertet, d.h. bspw *Funktion(i++);* liefert (zumeist) ein anderes Ergebnis als *Funktion(i); i++;*. Allerdings ist die Reihenfolge der Auswertung der Argumente unspezifiziert; eine Funktion wie *atan2(x++, x-1);* kann in unterschiedlichen Implementierungen zur unterschiedlichen Ergebnissen führen.

Mathematische Funktionen

Zur numerischen Berechnung stellt *Awk* folgende Funktionen zur Verfügung:

atan2 (x,y)	Liefert den Arcus-Tangens (in *rad*) von x/y
cos(x)	Liefert den Consinus von x
exp(x)	Berechnet e^x
int(x)	Ganzzahliger Wert von x, wobei »in Richtung 0« gerundet wird
log(x)	Liefert den natürlichen Logarithmus von x
rand()	Liefert eine Zufallszahl im Bereich von [0,1]
sin(x)	Liefert den Sinus von x
sqrt(x)	Liefert die Quadratwurzel von x
srand ([x])	Setzt den Startwert für die Zufallszahlen-Generierung [mittels rand()]. Geliefert wird der »alte« Startwert

Die Funktion zur Erzeugung von Zufallszahl liefert nicht wirklich zufällige Zahlen, sondern eine Folge relativ gleichverteilter Zahlen im Intervall [0,1]. Mit jedem Start eines Awk-Programms wird somit stets dieselbe Folge von Zufallszahlen generiert. Genügt dieser »Zufall« nicht aus, muss mit **srand()** ein neuer Bezugspunkt für **rand()** gesetzt werden. **srand()** ohne Argument erzeugt diesen Startwert aus aktuellem Datum und Uhrzeit, womit **rand()** tatsächlich den Eindruck zufälliger Werte erweckt.

Einige Anwendungsbeispiele sollen die Verwendung der Funktionen demonstrieren:

```
# Berechnung von π
user@sonne> awk 'BEGIN { printf("π= %.50f\n", 4*atan2(1,1)); }'
π= 3.14159265358979311599796346854418516159057617187500

# Ganzzahliger Anteil eines Wertes
user@sonne> awk 'BEGIN { print int(-7), int(-7.5), int (7), int(7.5)}'
-7 -7 7 7

# Der natürliche Logarithmus von e[1]
user@sonne> awk 'BEGIN { print log(exp(1))}'
1
```

Anmerkung: Das Beispiel der Berechnung von π zeigt die Genauigkeitsschranke der verwendeten *Awk*-Implementierung; ab der 48. Nachkommastelle ist Schluss. Allerdings genügt die interne Darstellung, um bei Umkehrrechnungen auf den Ausgangswert zu kommen (log(exp(1))).

Zeichenkettenfunktionen

Die nachfolgend diskutierten Funktionen durchsuchen oder manipulieren Zeichenketten.

gsub(Regex,Ersatz, [String])	Jedes Vorkommen des Regulären Ausdrucks *Regex* in *String* wird durch *Ersatz* ersetzt. Wird *String* nicht angegeben, wird $0 bearbeitet. Rückgabewert ist die Anzahl der Ersetzungen.
index (Suchstring,Muster)	Liefert die Position, an der *Muster* in *Suchstring* erstmals vorkommt oder 0
length([String])	Liefert die Länge von *String*; fehlt *String*, wird die länge von $0 zurückgegeben
match(String,Regex)	Liefert die Position des ersten Auftretens des Regulären Ausdrucks *Regex* in *String*; setzt RSTART und RLENGTH
split(String,Feld, [Seperator])	Zerlegt *String* in einzelne Elemente und liegt diese in *Feld* ab. Als Trennzeichen dient *Seperator* oder FS, falls dieser nicht angegeben wurde. Rückgabewert ist die Anzahl der Elemente.
sprintf ("Format",Ausdruck)	Anwendung wie **printf**, anstatt einer Ausgabe gibt die Funktion die resultierende Zeichenkette zurück
sub(Regex,Ersatz, [String])	Arbeitsweise wie **gsub**, es wird aber nur das erste Muster ersetzt
substr(string,p,[l])	Gibt eine Teilzeichenkette von *string* der Länge *l*, beginnend an Position *p* zurück
tolower(string)	Wandelt Groß- in Kleinbuchstaben um, Rückgabewert ist die neue Zeichenkette
toupper(string)	Wandelt Klein- in Großbuchstaben um, Rückgabewert ist die neue Zeichenkette

Wieder demonstrieren Beispiele die Anwendung:

```
# Ermitteln der längsten Zeile einer Datei
user@sonne> expand default.htm | awk '{x = (x < length()) ? length():x;} END {print x}'
566

# In welcher Zeile und welcher Position erscheint ein Muster:
user@sonne> awk '/ Partitionstabelle/ {print "Zeile:", NR, "Spalte", index($0,"Partitionstabelle")}'
Linuxfibel/ installbefore.htm
Zeile: 804 Spalte 14
Zeile: 818 Spalte 81
Zeile: 837 Spalte 57
```

```
Zeile: 838 Spalte 41
Zeile: 843 Spalte 41
Zeile: 972 Spalte 12
```

Sonstige Funktionen

close(Datei)	Schließt eine zuvor zum Lesen oder Schreiben geöffnete Datei oder die Pipe, die als Eingabe oder Ausgabe eines Kommandos diente (dann muss anstatt eines Dateinamens der Kommandoaufruf angegeben werden)
fflush(Datei)	Leert unverzüglich den Puffer einer »gepufferten« Ausgabe. D.h. veränderte Daten einer Datei werden sofort zurückgeschrieben oder die in eine Pipe geleitete Ausgabe eines Kommandos wird sofort aus dieser entnommen. Bei fehlendem Argument entleert *Gawk* die Standardausgabe; bei Angabe der leeren Zeichenkette ("") werden die Puffer sämtlicher offener Ausgabedateien und Pips entleert.
system(Kommando)	Gestattet die Ausführung von beliebigen Kommandos. Nach Ende eines solchen wird im *Awk*-Programm fortgefahren. **system** liefert den Status des letzten gestarteten Kommandos zurück.
systime()	Liefert die Systemzeit in Sekunden (seit 1.1.1970)
strftime([Format [,Zeitstempel]])	Liefert eine Datumszeichenkette. Ohne Zeitstempel wird die aktuelle Systemzeit verwendet, das Format entspricht dem der gleichnamigen C-Funktion.

Beispiele zur Anwendung von **close()** finden sich im Text genügend. Hier soll eine unnütze Anwendung von **systime()** den Abschnitt beschließen.

```
user@sonne> awk 'BEGIN { start = systime();
> for (i = 1; i < 10000000; i++) ;
> end = systime();
> print "Gesamtzeit betrug", end - start, "Sekunden.";
}'
Gesamtzeit betrug 6 Sekunden.
```

Eigene Funktionen

Die eingebauten Funktionen decken einen weiten, aber doch nicht jeden Anwendungsbereich ab. Erst eigene Funktionen eröffnen den Weg zu effizienten Programmen, vor allem dann, wenn identische Abläufe wiederholt im Programm auftreten.

Funktionsdefinition

Eine Funktion muss *vor* ihrer ersten Verwendung definiert sein. Ihre Definition erfolgt zweckmäßig zu Beginn eines Awk-Skripts, erlaubt ist sie auch zwischen BEGIN-Block und Hauptprogramm bzw. zwischen Hauptprogramm und END-Block.

Der allgemeine Aufbau einer Funktion ist:

```
function name([Parameter [, Parameter] ]) {
   # Anweisungen...
}
```

Als Funktionsname sind alle Kombinationen aus Buchstaben, Ziffern und dem Unterstrich zulässig, wobei zu Anfang keine Ziffer stehen darf. In einem Programm darf eine Funktion nicht gleich lauten wie der Bezeichner einer Variable. Als Argumente können beliebig viele Parameter an eine Funktion übergeben werden, wobei das Komma als Trennzeichen dient.

Als erstes Beispiel berechnet eine Funktion **facultiy** die Fakultät zu einer Zahl *x*:

```
function faculty(x) {
   if (x == 0) return 1;
   return x*faculty(x-1)
}
```

Unsere Version von **faculty** ist rekursiv realisiert, was *Awk* problemlos akzeptiert. Zur Rückkehr aus einer Funktion kann **return** mit optionalem Rückgabewert verwendet werden. Fehlt **return**, kehrt die Funktion nach Ausführung der letzten Anweisung zurück.

Funktionsaufruf

Zur Verwendung der Funktion rufen Sie diese einfach auf. Im Gegensatz zu den eingebauten Funktionen darf bei nutzerdefinierten Funktionen *kein* Leerzeichen zwischen Funktionsname und der öffnenden runden Klammer stehen! Die Angabe von weniger Parametern als definiert ist zulässig; abhängig vom Kontext werden diese als leere Zeichenkette oder als 0 interpretiert.

Im folgenden Programm wird der zu berechnende Wert per Kommandozeilenargument übergeben:

```
#!/usr/bin/awk -f

function faculty(x) {
   if (x == 0) return 1;
   return x*faculty(x-1)
}

BEGIN  {
  if (ARGC < 2) {
    print "Fehlendes Argument!";
    exit 1;
  }
  if (ARGV[1] !~ /^[[:digit:]]+$/) {
    print "Unzulaessiges Argument!";
    exit 2;
  }
  print faculty(ARGV[1]);
}
```

Lokale und globale Variablen, Parameterübergabe

Damit haben Sie fast das Rüstzeug beisammen, um eigene komplexe Funktionen zu verfassen. Doch die eigenwillige Semantik Semantik der Verwendung von Variablen kann zumindest für den Programmierneuling schnell zu Stolperfalle werden.

Mit einer Ausnahme hat eine Funktion grundsätzlichen Zugriff auf alle Variablen, die im Programm bis zum Aufruf der Funktion eingeführt wurden. Solche **globalen** Variablen können in der Funktion verändert, gelöscht (nur Array-Elemente [delete]) oder sogar neu eingeführt werden. **Lokale** Variablen sind einzig jene, die in der Parameterliste benannt wurden. Diese Variablen verdecken ggf. vorhandene gleichnamige Variablen des Programms, sodass Änderungen an diesen in der Funktion »außen« nicht sichtbar werden.

Das folgende Programm demonstriert die Verwendung von globalen und lokalen Variablen:

```
user@sonne> cat context.awk
#!/usr/bin/awk -f

function context(x, a)
{
  printf("In der Funktion...\n")
  printf("\tx = %d\n\ta = %d\n\tb = %d\n", x, a, b);
  x = a = b = 99;
}
```

```
BEGIN  {
   x = a = 100;
   printf("Vor Aufruf der Funktion...\n")
   printf("\tx = %d\n\ta = %d\n\tb = %d\n", x, a, b);

   context(10);

   printf("Nach Aufruf der Funktion...\n")
   printf("\tx = %d\n\ta = %d\n\tb = %d\n", x, a, b);
}
user@sonne> ./ context.awk
Vor Aufruf der Funktion...
      x = 100
      a = 100
      b = 0
In der Funktion...
      x = 10
      a = 0
      b = 0
Nach Aufruf der Funktion...
      x = 100
      a = 100
      b = 99
```

Das abschließende Beispiel dreht einem das Wort im Munde um...

```
function reverse (x) {
   if (length(x) == 0)
      return "";
   return substr(x, length(x), 1) reverse(substr(x,1, length(x)-1));
}
```

Unix Werkzeuge - Make und Makefiles

GNU make - Überblick

Die Komplexität verschiedener Programmierprojekte hat schon lange Dimensionen erreicht, in denen das Programm, aus reiner Notwendigkeit heraus, nicht mehr ein einziger großer Klumpen ist, sondern sich aus Hunderten von Modulen zusammensetzt.
Ein durchdacht strukturiertes Projekt ermöglicht den Austausch der einzelnen Teile - bei Wahrung der Schnittstellen -, ohne dass der Funktionalität des Gesamten Einschränkungen widerfährt.
So eine in sich abgeschlossene Einheit eines Programms kann selbst wieder aus mehreren Tausend Quelldateien bestehen. Stellt sich die Frage: Wenn Programmierer *A* die Dateien *X, Y, Z* im Modul *B* verändert... welche Module sind neu zu übersetzen?

Nehmen Sie sich den Netscape her, dessen Grundgerüst zusammen mit allen peripheren Programme an die 100 MByte Quelldateien umfasst... Wird nun eine Schnittstelle modifiziert, welche der Programme oder Bibliotheken sind neu zu generieren?
Das gesamte Paket neu durch den Compiler zu jagen, kann selbst auf schneller Hardware Stunden verschlingen. Und bevor alle Fehler aus einem Programm beseitigt sind, werden einige tausend Compiler-Durchläufe notwendig (und selbst dann sind solche großen Pakete wie der Netscape nicht fehlerfrei).

Hier kommt **make** ins Spiel. Make benötigt für seine Arbeit eine Steuerdatei, üblicherweise **makefile** genannt. In einem solchen Makefile sind nun die Abhängigkeiten der einzelnen Programmteile von den betreffenden Quelldateien beschrieben. Make erkennt anhand der Modifikationszeiten der einzelnen Dateien, in welchen Teilen sich die Quellen geändert haben. Genau für diese Teile (man bezeichnet diese als Targets - Ziele) wird »make« den Compilervorgang anstoßen.

Das Erzeugen eines solchen Makefiles erfordert eine gewisse Disziplin; seine Komplexität wächst mit dem Fortschritt des Projektes. Ein sorgsam gepflegtes Gerüst aus Makefiles kann den Aufwand beim Übersetzen von Projekten enorm reduzieren.

GNU make - Nicht nur für Programmierer

Make kann überall dort eingesetzt werden, wo bestimmte Aktionen notwendig sind, sobald eine konkrete Datei verändert wurde:

- Bei allen Programmierprojekten
- Bei Backups: »Wurde(n) diese Datei(en) modifiziert, erzeuge die Backup-Datei neu.«
- Bei der Systemverwaltung: "Wurde diese Konfigurationsdatei verändert, dann generiere eine neue Datenbasis." (dieses Prinzip wird u.a. für die NIS-Datenbasis angewandt.)
- Bei LaTeX-Dokumenten: "Erzeuge die Postscript-Datei neu, falls eine der Quelldateien geändert wurde."
- ...?

GNU make - Aufruf und Optionen

Im einfachsten Fall wird »make« ohne Argumente gestartet:

```
user@sonne> make
```

Make sucht jetzt im aktuellen Verzeichnis nach einer Datei mit dem Namen »GNUmakefile«, »makefile« oder »Makefile«. Make sucht in beschriebener Reihenfolge und verwendet die **erste** gefundene Steuerdatei (und nur diese!). Findet make kein Makefile, so ist die Antwort:

```
user@sonne> make
make: *** No targets. Stop.
```

In einem Makefile sind die Abhängigkeiten zu den einzelnen Zielen (Targets) beschrieben. Per Voreinstellung beginnt »make« mit der Abarbeitung des ersten Zieles aus der Steuerdatei und endet, sobald alle Aktionen zu **diesem** Ziel erledigt sind.

Soll nun ein anderes als das erste Ziel betrachtet werden, ist dies »make« mitzuteilen:

```
user@sonne> make anderes_ziel
```

Make können mehrere Ziele als Argumente mitgegeben werden; Existieren die Ziele in der Steuerdatei, wird »make« sie in dieser Reihenfolge abarbeiten. Im anderen Fall hagelt es eine Fehlermeldung:

```
user@sonne> make gibts_nicht
make: *** No rule to make target `gibts_nicht'. Stop.
```

Weitere wichtige Optionen sind:

```
user@sonne> make -C / ../ anderer_Pfad/
```

Make wechselt zunächst das Arbeitsverzeichnis und arbeitet dann wie gehabt.

```
user@sonne> make -d
```

Make gibt erweiterte Debug-Informationen zum Vorgang aus.

```
user@sonne> make -f Steuerdatei
```

Make bezieht seine Instruktionen aus der mit *Steuerdatei* benannten Datei.

```
user@sonne> make -j 2
```

Make versucht die Aktionen zu parallelisieren (im Beispiel 2 Prozesse - sinnvoll bei SMP-Maschinen).

```
user@sonne> make -t
```

Make setzt die Modifikationszeit der Ziele (so dass sie **nicht** neu erzeugt werden, selbst wenn ihre Quelldateien modifiziert wurden).

GNU make - Aktionen erzwingen

In manchen Situationen ist es notwendig, die Zielaktionen durchzuführen, selbst wenn sich in den abhängigen Dateien nichts geändert hat. Z.B. werden Sie Ihr Projekt im Endstadium mit optimierenden Compileroptionen übersetzen wollen. Die alleinige Änderung der Compiler-Flags juckt »make« so ziemlich gar nicht. Ebenso werden Sie, um ganz sicher zu gehen, dass Ihnen keine Fehler unterlaufen sind, vor der Auslieferung von Software eine

vollständige Neuübersetzung aller Quellen erzwingen wollen.

Sie haben nun (mind.) zwei Möglichkeiten, eine erneute Abarbeitung aller Aktionen zu erzwingen:

1. Sie könnten alle Zieldateien löschen (dann wird »make« jedes neu erzeugen)
2. Sie können die Modifikationszeiten aller Quelldateien ändern, z.B.:

```
user@sonne> touch *.h *.c
```

Jetzt kennen Sie die wahre Berufung des Kommandos »touch«.

Einführung in Makefiles

Makefiles bestehen aus Regeln (»rules«). Jede Regel besitzt folgendes Format:

```
Ziel:    Abhängigkeit(en)
    Kommando
    Kommando
    ...
```

Ziel	Ziel ("target") ist normalerweise der Name des Programm(moduls), das durch die nachfolgenden Kommandos generiert wird. Ziel kann aber auch der Name einer Aktion sein.
Abhängigkeiten	Hier stehen meist die Namen der Dateien und Ziele, von denen dieses Ziel abhängt.
Kommando	Hier stehen die Aktionen, die im Falle einer erfüllten Abhängigkeit auszuführen sind. Jede Aktion muss auf einer eigenen Zeile stehen und durch einen Tabulator eingerückt sein (**keine Leerzeichen**). Sehr lange Zeilen können umgebrochen werden, indem am Ende jeder Zeile - bis auf die letzte - ein Backslash »\« angefügt wird (auch hier gilt: keine weiteren Zeichen nach dem Backslash!).

Lernen am Beispiel - Die Ausgangssituation

Die wichtigen Bestandteile von Makefiles möchte ich schrittweise anhand eines Beispieles erläutern. Konkret soll es um ein LaTeX-Dokument gehen, das in mehrere Dateien aufgeteilt ist.

Das Makefile soll am Ende folgende Ziele enthalten:

- Die Dvi-Datei (Device independend) soll erzeugt werden, falls eine der Quelldateien modifiziert wurde
- Die Postscript-Datei soll erzeugt werden, wenn sich die DVI-Datei geändert hat
- Es soll ein Archiv mit allen Quelltexten erstellt werden
- Die Postscript-Datei soll in verschiedene Formate konvertiert werden können
- Ein Ziel soll das Löschen aller temporären Dateien ermöglichen

Die Quelldateien nennen sich »ch01.tex«, »ch02.tex«,..., »ch09.tex« und »book.tex«. Weiterhin existieren die Bilddateien »picture1.eps« und »picture2.eps« (Encapsulated Postscript). Die fertige Postscript-Datei soll »book.ps« heißen; das Archiv nennt sich »book.tar.gz«.

Makefile - Erster Versuch

Dvi-Datei: Diese Datei ist nur von den »tex«-Quelldateien abhängig. Um sie zu erzeugen, ist das Kommando »latex« mit dem Namen des Hauptdokuments »book.tex« als Argument aufzurufen. »latex« generiert aus »book.tex« die Datei »book.dvi«. Also lautet der gesamte Eintrag:

```
book.dvi: ch01.tex ch02.tex ch03.tex ch04.tex \
        ch05.tex ch06.tex ch07.tex ch08.tex \
        ch09.tex book.tex
```

```
        latex book.tex
```

Da die Liste der Abhängigkeiten sehr lang ist, wurde diese auf mehrere Zeilen verteilt. Man beachte, dass unmittelbar nach dem Backslash der Zeilenumbruch stehen muss. Ist nur eine der genannten Dateien neuer als das Ziel »book.dvi«, wird die nachfolgende Kommandozeile ausgeführt.

Postscript-Datei: Die Datei muss erzeugt werden, falls entweder eines der Bilder modifiziert wurde oder eine neuere »dvi«-Datei vorliegt:

```
book.ps: picture1.eps picture2.eps book.dvi
        dvips -D 600 book.dvi
```

Da »book.dvi« ein Ziel desselben Makefiles ist, werden automatisch auch dessen Abhängigkeiten überprüft und die Datei eventuell neu erstellt. Die Option »-D 600« bewirkt eine Skalierung der Postscript-Fonts auf 600 dpi (Voreinstellung ist 300 dpi).

Archiv: Das Archiv (Format »tar«, gepackt) des Quelltextes muss erstellt werden, sobald sich eine dieser Dateien geändert hat. Zusätzlich soll das Makefile im Archiv enthalten sein.

```
booksrc.tar.gz: ch01.tex ch02.tex ch03.tex ch04.tex \
        ch05.tex ch06.tex ch07.tex ch08.tex \
        ch09.tex book.texpicture1.eps \
        picture2.eps makefile
          tar czf booksrc.tar.gz ch01.tex ch02.tex ch03.tex \
          ch04.tex ch05.tex ch06.tex ch07.tex ch08.tex ch09.tex\
          book.tex picture1.eps picture2.eps makefile
```

Konvertierung: Beispielhaft soll die Konvertierung der Postscript-Datei in das Pdf-Format vorgenommen werden. Die Pdf-Datei ist nur von der entsprechenden Postscript-Datei abhängig:

```
book.pdf: book.ps
        ps2pdf book.ps book.pdf
```

Löschen: Die Kommandos »latex« und »dvips« legen eine Reihe von Hilfsdateien an (z.B. Index-Verzeichnisse...). Diese sollen bei Aufruf von »make clean« entfernt werden. Weiterhin zu Löschen sind alle Zieldateien und das Archiv:

```
clean:
        rm -f book.[^t][^e][^x] book.ps *~ booksrc.tar.gz
```

Das Ziel »clean« ist von nichts abhängig, deswegen steht es allein auf einer Zeile. Wenn man weiß, dass die von »latex« und »dvips« erzeugten Dateien alle eine dreistellige Erweiterung aufweisen, so sollte erkennbar sein, dass man mit »book.[^t][^e][^x]« genau jene erfasst, nicht aber »book.ps«, da diese nur eine zweistellige Namenserweiterung besitzt. Mit »*~« löscht man Backup-Dateien, so wie sie z.B. der Editor Vi anlegt. Die Option »-f« unterdrückt Fehlerausgaben, falls dem Kommando »rm« einmal keine Dateien übergeben wurden (z.B. bei zweimaligem Aufruf von »make clean«).

Die Reihenfolge der Ziele

Wird das Kommando »make« mit einem Ziel als Argument aufgerufen, so sucht es zunächst nach einem Makefile und anschließend in diesem nach einem Ziel mit diesem Namen. Einzig die Zeilen dieses Zieles werden dann abgearbeitet.

Wird das Kommando »make« ohne Argumente aufgerufen (gemeint sind hier Ziele - keine Optionen), so wird - falls ein Makefile gefunden wurde - das erste Ziel aus diesem Makefile abgearbeitet..

Es liegt also nahe, das »gebräuchlichste« Ziel als erstes zu nennen. Ebenso hat es sich eingebürgert, ein Ziel wie »clean« an das Ende einer Steuerdatei zu verbannen. Damit erhalten wir folgendes Makefile:

```
book.ps: picture1.eps picture2.eps book.dvi
        dvips -D 600 book.dvi

book.dvi: ch01.tex ch02.tex ch03.tex ch04.tex \
        ch05.tex ch06.tex ch07.tex ch08.tex \
        ch09.tex book.tex
          latex book.tex

booksrc.tar.gz: ch01.tex ch02.tex ch03.tex ch04.tex \
        ch05.tex ch06.tex ch07.tex ch08.tex \
        ch09.tex book.tex picture1.eps picture2.eps makefile
          tar czf booksrc.tar.gz ch01.tex ch02.tex ch03.tex \
          ch04.tex ch05.tex ch06.tex ch07.tex ch08.tex ch09.tex \
          book.tex picture1.eps picture2.eps makefile

book.pdf: book.ps
        ps2pdf book.ps book.pdf

clean:
        rm -f book.[^ t][^ e][^ x] book.ps *~ booksrc.tar.gz
```

Wollen wir nun die Postscript-Datei neu erstellen, rufen wir »make book.ps« oder - da dieses Ziel das erste im Makefile ist - »make« auf. Ein Archiv erzeugen wir mit »make booksrc.tar.gz« und zum Aufräumen hilft »make clean«.

Makefile - Variablen

Unser Makefile besitzt einige Schwächen:

- Es wirkt ziemlich verwirrend...
- Wenn die Pfade zu den enthaltenen Kommandos nicht in der »PATH«-Variable des Nutzers stehen, wird »make« das Kommando nicht finden...
- Ein Archiv zu erstellen, wäre einleuchtender mit z.B. »make archiv« zu realisieren...

Dateilisten in Variablen speichern

In unserem Makefile werden mehrfach die Dateien »ch01.tex«,..., »ch09.tex« usw. genannt. Anstatt die Liste an jeder Stelle zu nennen, sollte diese besser einmal zu Beginn des Makefiles definiert werden. Neben der gewonnenen Übersichtlichkeit ist eine solche Liste auch einfacher zu modifizieren - nur an einer einzigen Stelle in der Steuerdatei sind Änderungen erforderlich.

```
TEXFILES = ch01.tex ch02.tex ch03.tex ch04.tex \
          ch05.tex ch06.tex ch07.tex ch08.tex \
          ch09.tex book.tex

PICTURES = picture1.eps picture2.eps
```

Auf eine solche Variable kann anschließend mittels »$(VARIABLENNAME)« zugegriffen werden.

Kommandos und Kommandooptionen in Variablen speichern

Kommandos werden bei verschiedenen Systemen häufig in unterschiedlichen Verzeichnissen abgelegt. Es ist sicherlich kein Fehler, anstelle des Kommandos selbst innerhalb des Makefiles dieses über eine Variable aufzurufen. So muss bei Verwendung des Makefiles auf einem anderen System nur die Zeile der Variablendefinition geändert werden. Eine ähnliche Überlegung legt die Verwendung von Variablen für Optionen nahe, da auch diese oft differieren. Kommandoname und -optionen können auch in einer Variable vereinbart werden:

```
LATEX = /usr/bin/latex
DVIPS = /usr/bin/dvips
DVIPSOPTS = -D 600
```

```
RM = rm -f
TAR = tar -z
```

Prägnante Namen für das Ziel

Um ein Archiv zu erzeugen, ist es sicherlich lästig, immer den Namen »booksrc.tar.gz« angeben zu müssen... Einfach den Namen des Zieles zu ändern, würde die Erzeugung des Archivs immer bewirken, selbst wenn sich an den Dateien nichts geändert haben sollte (da keine Datei unter dem Namen dieses Zieles existiert).

Die Einführung eines neuen Ziels »archiv«, das von »booksrc.tar.gz« abhängt und keine Kommandos besitzt, bringt das gewünschte Ergebnis:

```
archiv: booksrc.tar.gz

booksrc.tar.gz: $(TEXFILES) $(PICTURES) makefile
	tar czf booksrc.tar.gz $(TEXFILES) $(PICTURES) makefile
```

»make archiv« ruft jetzt implizit »make booksrc.tar.gz« auf.

Das neue Makefile

Unser neues Makefile besitzt nun folgende Gestalt:

```
TEXFILES = ch01.tex ch02.tex ch03.tex ch04.tex \
	ch05.texch06.tex ch07.tex ch08.tex \
	ch09.tex book.tex

PICTURES = picture1.eps picture2.eps

LATEX = /usr/bin/latex
DVIPS = /usr/bin/dvips
DVIPSOPTS = -D 600
RM = /bin/rm -f
TAR = /bin/tar z
PS2PDF = /usr/bin/ps2pdf

book.ps: $(PICTURES) book.dvi
	$(DVIPS) $(DVIPSOPTS) book.dvi

book.dvi: $(TEXFILES)
	$(LATEX) book.tex

booksrc.tar.gz: $(TEXFILES) $(PICTURES) makefile
	$(TAR) cf booksrc.tar.gz $(TEXFILES) $(PICTURES) makefile

archiv: booksrc.tar.gz

book.pdf: book.ps
	$(PS2PDF) book.ps book.pdf

clean:
	$(RM) book.[^ t][^ e][^ x] book.ps *~ booksrc.tar.gz
```

Makefile - Weitere Möglichkeiten

Phony - Ausführung immer erzwingen

Angenommen im aktuellen Verzeichnis existiert eine Datei »clean«... Dann würde ein Aufruf von »make clean« nichts bewirken, da dieses Ziel von nichts abhängig ist und »make« es damit für "up to date" hält.

Um die Abarbeitung eines Zieles immer zu erzwingen, muss diese von einem "Phony"-Target abhängen:

```
.PHONY: clean
```

Dateinamensubstitution

Ist ein Ziel von Dateinamen eines bestimmten Musters abhängig, lassen sich die Abhängigkeiten auch mit Hilfe von Metazeichen ausdrücken:

```
...
booksrc.tar.gz: *.tex *.eps [Mm]akefile
...
```

Allerdings funktioniert eine solche Substitution nicht in Variablendefinitionen. Hier muss auf die Funktion »wildcard« zurückgegriffen werden:

```
...
TEXFILE = $(wildcard *.tex)
...
```

Ausgaben von make unterdrücken

»make« schreibt die Kommandozeile des Makefiles, die aktuell bearbeitet wird, immer auf die Standardausgabe. Um bestimmte Ausgaben zu vermeiden, muss einem solchen Kommando ein »@« vorangestellt werden:

```
...
clean:
	@$(RM) book.[^t][^e][^x] book.ps *~ booksrc.tar.gz
```

Makefile - Die fertige Datei

Die beschriebenen Möglichkeiten zum Umgang mit Makefiles behandeln nur einen Bruchteil des tatsächlichen Funktionsumfangs. Dennoch sollten diese Mechanismen den Systemverwalter befähigen, Einsatzpunkte von »make« für die Verwaltungsaufgaben zu erkennen und entsprechende Steuerdateien zu verfassen.

Mit unserem bisherigen Kenntnissen sollte ein Makefile, das die obigen Anforderungen erfüllt, in etwa so aussehen:

```
TEXFILES = $(wildcard *.tex)
PICTURES = $(wildcard *.eps)
LATEX = /usr/bin/latex
DVIPS = /usr/bin/dvips
DVIPSOPTS = -D 600
RM = /bin/rm -f
TAR = /bin/tar z
PS2PDF = /usr/bin/ps2pdf
ARCHIV = booksrc.tar.gz
ARCHIVFILES = $(TEXFILES) $(PICTURES)
ARCHIVFILES += makefile

book.ps: $(PICTURES) book.dvi
	@$(DVIPS) $(DVIPSOPTS) book.dvi

book.dvi: $(TEXFILES)
	@$(LATEX) book.tex

$(ARCHIV): $(ARCHIVFILES)
	@$(TAR) cf $(ARCHIV) $(ARCHIVFILES)

archiv: $(ARCHIV)
```

```
book.pdf: book.ps
	@$(PS2PDF) book.ps book.pdf

.PHONY: clean

clean:
	@$(RM) book.[^ t][^ e][^ x] book.ps *~ booksrc.tar.gz
```

Anmerkungen: Im Beispiel wurden zwei weitere Varianten der Variablendefinition eingebaut. Zum einen die Verwendung einer bereits definierten Variable bei einer weiteren Definition (»ARCHIVFILES = $(TEXFILES) $(PICTURES)«); zum anderen das Hinzufügen von Elementen zum Inhalt einer Variable (»ARCHIVFILES += makefile«).

Linux - Systemadministration

Überblick

Systemadministration ist eine enorm komplexe und - wenn man für den reibungslosen Betrieb eines von zahlreichen Benutzern frequentierten Systems verantwortlich zeichnet - auch undankbare Aufgabe. Geht alles wie erhofft, so dankt der Benutzer dies niemals dem Admin, klappt etwas nicht, landet der Zorn garantiert beim vermeintlichen Übeltäter.

Eine befriedigende Installation erreicht man nicht von heute auf morgen. »CD-ROM rein« und alle Fragen bejahen führt zwar schnell zu einem lauffähigen Linux. Aber die Arbeit des Administrators beginnt erst jetzt. Benutzer anlegen, Hardware einrichten, zusätzliche Software installieren, Dienste konfigurieren, den X-Server erziehen, Sicherheitslöcher stopfen und und und...

Ohne ein tieferes Grundlagenwissen wird man viele Klippen als solche nicht erkennen und somit nicht umschiffen können. Aus diesem Grund ist das folgende Kapitel gespickt mit vielen Fakten, die nicht unmittelbar die Tippel-Tappel-Tour der Administration beschreiten, aber reichlich Einblicke hinter die Kulissen gewähren. Ein komplettes System ist unter Linux schnell hochgezogen. Der Teufel liegt im Detail und offenbart sich erst im laufenden System. Wo beginnt man die Suche dann nach einem Fehler?

Dreh- und Angelpunkt der nachfolgenden Abschnitte sind alle Belange der Administration eines *Einzelplatz-Linuxsystems*. Natürlich lässt sich eine strikte Trennung bei einem mit dem (und im) Internet groß gewordenen Betriebssystem nicht in allen Fällen aufrecht erhalten, sodass bei Bedarf auf Belange der Netzwerkeinrichtung eingegangen wird, ohne sie über das unbedingte Muss hinaus zu diskutieren.

Die von uns gewählte Reihenfolge der Darlegung sollte unterstreichen, dass jedes der folgenden Themen in sich abgeschlossen ist und somit ohne Kenntnis vorheriger oder nachfolgender Punkte bearbeitet werden kann. Was Sie nachfolgend - bis auf wenige Ausnahmen - vergeblich suchen werden, sind Beschreibungen zur Verwendung distributionsspezifischer Administrationswerkzeuge. Hierzu werden die Handbücher der von Ihnen bevorzugten Distribution sicher detailliertere und aktuellere Anleitungen enthalten.

Ziel des Kapitels

Wir hoffen, dass Sie mit den nachfolgend dargebotenen Anleitungen rasch zu einem funktionierenden Linuxsystem gelangen. Vermutlich werden die Installations- und Administrationshilfen der von Ihnen eingesetzten Distribution Ihnen zahlreiche Handgriffe abnehmen und im Erfolgsfall werden Sie die nachfolgenden, teils komplexen Pfade der Administration kaum beschreiten müssen.

Aus dem teils enormen Fundus an Lösungen für ein und dasselbe Problem haben wir uns auf die Diskussion weniger, verbreiteter und allgemeiner Techniken beschränkt. Jede zu beherrschen, ist nicht notwendig, aber sie zu kennen, hilft, Stärken wie auch Schwächen der einzelnen Varianten zu bewerten und so gezielt für einen Anwendungsfall die optimale Strategie zu finden.

Inhalt des Kapitels

Systemadministration- Der Bootvorgang

Übersicht

Wenn nach dem Einschalten des Rechners endlich der Anmeldebildschirm erscheint, ist nicht nur einige Zeit verstrichen, sondern es sind auch einige komplexe Vorgänge abgelaufen.

Der Schwerpunkt dieses Buches soll bei Linux liegen, dennoch wollen wir Ihnen die Arbeitsweise des BIOS Ihres Rechners und die prinzipiellen Funktionen eines Bootloaders nicht vorenthalten.

Nach dem einführenden Abhandlungen bringen wir etwas Licht in die zahlreichen Ausgaben, die der Kernel während des Bootens auf den Bildschirm schreibt. Leider beschreiten die verschiedenen Distributionen schon in dieser frühen Phase eigene Wege, so dass wir hin und wieder auf sehr spezifische Details eingehen müssen.

Es gibt Situationen, in denen die üblichen Mechanismen versagen. Im Falle des Bootvorganges werden wir deshalb auf die Möglichkeit des Startens von Rechnern eingehen, deren Hardware-Konfiguration dem Kernel vorab nicht bekannt ist. Was es damit auf sich hat, erfahren Sie unter Alternative Konzepte - Ramdisks.

Ein gänzlich anderes Bootvorgehen ist notwendig, falls der eigene Rechner über keinerlei Festplatte verfügt, das System also erst über das Netz geladen werden muss.

BIOS Selbsttest

Nach Einschalten oder Reset des Rechners beginnt dieser mit der Programmausführung an einer festgelegten Adresse (f000:fff0h). An jener Position befindet sich immer das BIOS, das sogleich zu den Initialisierungs- und Testroutinen, dem so genannten Power On Self Test **POST**, springt.

POST arbeitet in zwei Schritten:

- **Test und Initialisierung zentraler Hardware:** hierzu gehören die CPU, BIOS-ROM (Bildung einer Prüfsumme), DMA-Controller, Tastatur, die ersten 64k des RAMs, Interrupt-und Cache-Controller
- **Test und Initialisierung von System-Erweiterungen:** hierzu zählen der RAM über 64k, die Schnittstellen, Disketten- und Festplatten-Controller

Des Weiteren werden nach peripheren ROM-Erweiterungen wie dem Video-ROM gesucht und diese initialisiert. Bei SCSI-Systemen mit eigenem BIOS wird dieses ebenso bearbeitet.

Tritt in dieser Phase ein Fehler auf, wird entweder eine Meldung auf dem Bildschirm ausgegeben oder das BIOS macht mit Hilfe einer Tonfolge auf das Problem aufmerksam. Für bestimmte Fehler sind genaue Signale vorgeschrieben:

Ton	Defekt
Anhaltend (oder kein Ton)	Netzteil
Einmal lang	kein DRAM-Refresh
Einmal lang, einmal kurz	Mainboard
Einmal lang, zweimal kurz	Video-Controller oder Bildschirmspeicher

Dreimal kurz	RAM
Einmal kurz	Laufwerk oder Video-Controller

Sind alle Tests positiv verlaufen, sucht das BIOS auf den Bootgeräten nach einer gültigen Bootsequenz. Bootgeräte können eine Diskette, eine Festplatte, das CD-ROM-Laufwerk und die Netzwerkkarte sein. Welche Bootmedien durchsucht werden und die Reihenfolge, in der das BIOS die Geräte durchsucht, kann im CMOS-Setup eingestellt werden. Der erste gefundene Bootkode wird geladen und gestartet.

Bootloader

Im weiteren Verlauf lädt das BIOS den Mbr (*Master Boot Record*) des ersten eingetragenen Bootmediums. Der Mbr enthält neben der Partitionstabelle mit den Koordinaten der maximal 4 primären (bzw. maximal 3 primären und einer erweiterten) Partitionen ein kleines Programm (446 Bytes), das die Auswertung der Daten der Partitionstabelle übernimmt. Ist eine dieser Partitionen mittels eines **bootable Flags** markiert, wird deren Bootsektor (der erste Sektor dieser Partition) angesprungen und der dortige Kode ausgeführt. Fehlt eine Markierung, so fährt das BIOS mit dem Laden des Mbr vom nächsten Bootmedium fort. Ist dagegen der Bootkode einer »bootable« Partition ungültig, stoppt der Bootvorgang mit einer Fehlermeldung (bei einer Diskette wird bspw. zu deren Wechsel aufgefordert).

Die allgemeine Aufgabe des Bootkodes ist das Laden des Betriebssystems. Systeme, die nur einen solchen intialen Bootkode mit sich bringen, besitzen meist die unangenehme Eigenschaft, dass sie nach der Installation ungefragt das bootable Flag im Mbr auf »ihre« Partition umbiegen und daher das Booten von bereits installierten »Fremdsystemen« verhindern.

Ein Bootkode, der hingegen das Laden mehrerer Betriebssysteme unterstützt, wird als **Bootloader** oder **Bootmanager** bezeichnet. Linux selbst ist zwar nicht auf einen Bootloader angewiesen, mit Ausnahme des Starts von einem Wechselmedium (Diskette, CDROM) wird dennoch auf einen solchen zurückgegriffen. Der Standard-Loader von Linux - Lilo - kann sowohl das kleine Programm im Mbr ersetzen als auch im Bootsektor einer Partition liegen. In ersterem Fall lädt das BIOS direkt den Bootloader; in letzterem Fall muss die den Lilo enthaltene Partition in der Partitionstabelle mit dem »bootable« Flag versehen sein.

Bootloader vermögen oft weit mehr als nur das Laden eines Betriebssystems. So können sie den Start der Systeme mit einem Passwort schützen oder Parameter an das System übergeben, die dessen Arbeit dann beeinflussen.

Bei einer solch umfangreichen Funktionalität ist es leicht nachvollziehbar, dass der gesamte Kode eines Bootloaders nicht in die dafür reservierten 512 Bytes eines Bootsektors passt. Zumal von diesem Speicherplatz weitere 2 Bytes für eine »Magic Number« (AA55; sie markiert den Sektor als gültigen Bootkode) und - im Falle des Mbr - noch 64 Bytes für die Partitionstabelle abzuziehen sind. Deshalb werden heutige Bootloader in zwei Stufen realisiert, wobei die erste Stufe im Bootsektor bzw. im Mbr einzig die Aufgabe hat, die zweite, auf der Festplatte liegende Stufe, in den Hauptspeicher zu laden.

Ein Bootloader kennt nun die Speicherplätze der von ihm verwalteten Betriebssysteme und wird das Auserwählte in den Hauptspeicher laden.

Im Falle von Linux übernimmt mit der Meldung

```
Uncompressing Linux...
```

der Kernel die Kontrolle über den Rechner.

Initialisierung des Kernels

Jeder Kernel besitzt eine Einsprungmarke, an der er mit seiner Arbeit beginnt. Zunächst bewegt sich seine Tätigkeit sehr nahe an der Hardware, er ermittelt aus dem BIOS elementare Parameter und schaltet den Prozessor in den Protected Mode. Die nächsten Schritte betreffen die Initialisierung der Speicherverwaltung (MMU), des eventuell vorhandenen Coprozessors und der Interruptcontroller sowie die Erzeugung einer minimalen Umgebung.

Alles Bisherige realisierten Assembler-Routinen, also Programmteile, die in der Sprache des Prozessors geschrieben sind. Die weiteren Funktionen sind weniger architekturabhängig und deswegen in der Sprache C implementiert.

Im nächsten Schritt erfolgt die Initialisierung aller Kernelteile. Hierzu zählen die virtuelle Speicherverwaltung, die Interruptroutinen, die Zeitgeber, der Scheduler, der für die Zuteilung der CPU an die Prozesse verantwortlich ist. Weiter wird das Dateisystem initialisiert, verschiedene Kernelpuffer angelegt, Netzwerkschnittstellen und die Ressourcen der Interprozesskommunikation...

Was jetzt läuft, wird als **Idle**-Prozess, der Prozess mit der Prozessnummer 0, bezeichnet. Genau genommen ist er der erste Prozess im System, jedoch erledigt er rasch seine Aufgabe und ist dann nicht mehr sichtbar, obwohl er immer dann aktiv wird, wenn eigentlich nichts zu tun ist. Dennoch hat der Prozess mit der Nummer 0 eine wichtige Aufgabe: er ist derjenige, der den »ersten« Prozess ins Leben ruft. Der **init** -Prozess gilt allgemein als erster Prozess im System, da er der Ursprung aller weiteren Prozessfamilien ist. Bei Multiprozessorsystemen übernimmt der Prozess 0 die Initialisierung der weiteren Prozessoren.

init bekennt sich sogleich zu seiner absoluten Alleinherrschaft und sperrt für alle anderen Interessenten (Prozess 0) den Zugang zu den Kernelfunktionen. In den kommenden Schritten darf ihm einfach niemand dazwischen funken. **init** initialisiert weiter. Der Name ist Programm... Worum es sich konkret handelt, hängt von der Konfiguration des Kernels ab. Übliche Punkte sind die (PCI-) Geräte und Sockets. Die ersten Dämonen werden ins Leben gerufen, der **bdflush** zur Synchronisation von Cache und Dateisystem und der **kswapd** zur Verwaltung des Swapspeichers. Dem Kernel werden nun die unterstützten Binärformate und Dateisysteme bekannt gegeben und anschließend wird versucht, das Root-Dateisystem zu mounten (dieses ist im Kernel fest verankert; es kann per Bootoption überschrieben werden). Ist bislang nichts schief gegangen, lässt **init** auch den anderen Prozessen eine Chance, indem es den Kernel wieder frei gibt.

Er selbst fährt mit dem Eröffnen einer Konsole für die Ein- und Ausgaben fort. Gelingt dies nicht, ist die Ausgabe "*Warning: unable to open an initial console.*" die Folge.

Abschließend wird das Programm **init** auf der Platte gesucht. Die Reihenfolge der betrachteten Verzeichnisse ist

1. /sbin/init
2. /etc/init
3. /bin/init

Der rufende Prozess führt den Systemruf **exec()** aus, so dass dessen Programm vom Neuen überlagert wird. Scheitert der erste **exec()**-Aufruf, so erreicht das originale Programm den nächsten Aufruf usw.

Falls kein Programm **init** gefunden wurde, wird abschließend versucht, die Shell **/ bin/ sh** zu starten, um dem Administrator eine Reparatur des Systems zu ermöglichen. Gelingt auch dieses nicht, hält der Kernel mit der Ausschrift "*No init found. Try passing init= option to kernel.*" an.

Der erste Prozess

Bis dato konnte der Administrator nur über Optionen, die dem Kernel auf der Bootzeile angegeben werden, oder durch Erzeugen eines neuen Kernels den Bootvorgang steuern. Mit dem Laden des Programms **init** in die Umgebung des ersten Prozesses beginnt die eigentliche administrative Einflussnahme.

Bis auf wenige Ausnahmen verwenden nahezu alle aktuellen Distributionen das Startverfahren nach System V. Jede Komponente, die während des Systemstarts aktiviert werden soll, wird durch ein eigenes Skript beschrieben. Die verschiedenen Skripte können in geeigneter Reihenfolge abgearbeitet werden, in Abhängigkeit der tatsächlich zum Einsatz kommenden Teilmenge der Skripte gelangt das System in einen unterschiedlichen Zustand, das so genannten Runlevel.

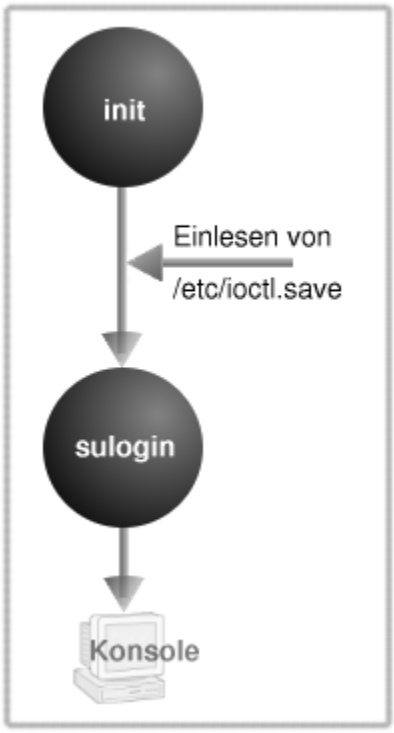


Abbildung 1: Start im Single User Mode

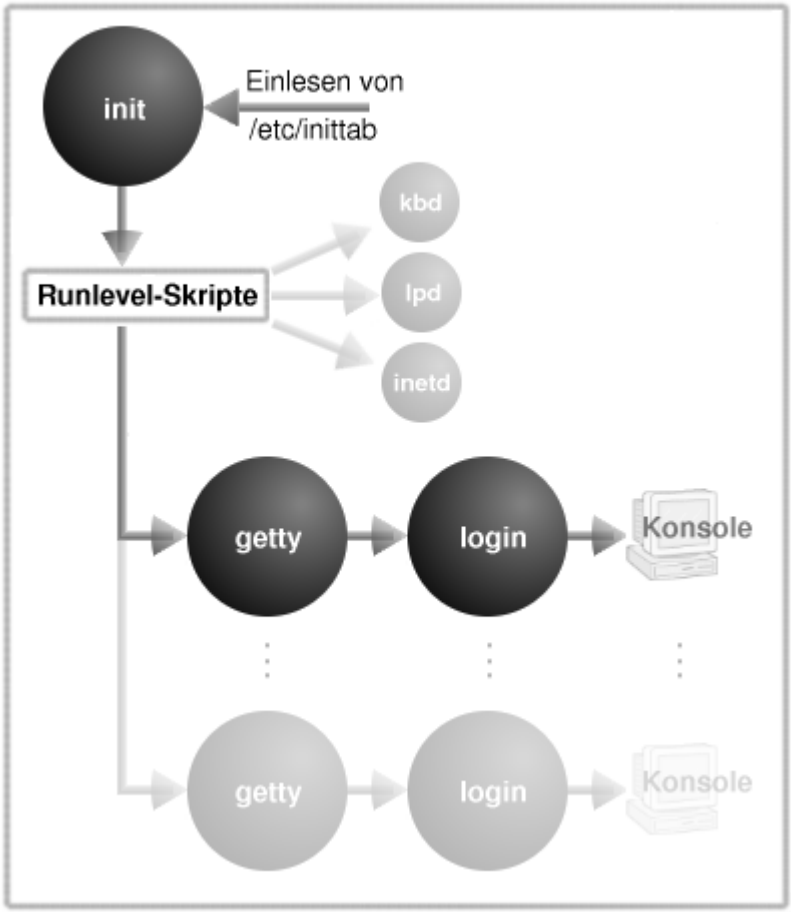


Abbildung 2: Start im Multi User Mode

Wenn **init** mit seiner Arbeit beginnt, sucht es zunächst nach seiner Konfigurationsdatei /etc/inittab. Für den Fall, dass diese nicht existiert oder kein mit **initdefault** beginnender Eintrag enthalten ist, fordert **init** zur Eingabe des zu startenden Runlevels auf.

Bei einer fehlenden Datei /etc/inittab ist nun nur der Start im Single User Modus (Runlevel 1, S bzw. s) zulässig, da

zum Start im Multi User Modus weitere Informationen benötigt werden, welche die Konfigurationsdatei enthält. Im Single User Modus entnimmt **init** der Datei /etc/ioctl.save (sofern vorhanden, sonst gelten Default-Werte und die Datei wird erzeugt) die Einstellungen für das Terminal und startet das Programm **/ sbin/ sulogin**, das mit der bereits eröffneten Konsole "/dev/console" verbunden wird. In diesem Modus kann sich einzig der Systemverwalter anmelden, es laufen auch nur die zwingend benötigten Prozesse, so dass alle Ressourcen dem Administrator zur Verfügung stehen. Der Modus dient vorrangig der Pflege des Systems, so wäre es bspw. fatal, wenn ein Benutzer während der Reparatur des Dateisystems schreibenden Zugriff auf dieses erhielte.

Im Falle des Multi User Modus kommt auf **init** eine Menge weiterer Arbeit zu. In einem nächsten Schritt lässt **init** alle dem jeweiligen Runlevel zugeordneten Skripte von einem Shellskript namens **rc** starten. Das ist der Zeitpunkt, wenn die zahlreichen Meldungen über gestartete Dämonenprozesse über den Bildschirm flimmern. Ist das aktuelle Runlevel erreicht, startet **init** eine Reihe von **getty**-Prozessen, die wiederum auf der ihnen zugedachten Konsole das Kommando **login** ausführen, welches letztlich die Aufforderung zur Anmeldung auf den Bildschirm bringt.

Die Datei / etc/ inittab

Jede Zeile der zentralen Konfigurationsdatei des **init**-Prozesses besitzt folgenden Aufbau:

```
ID:Runlevel:Aktion:Prozess
```

ID

Die ID ist eine fast beliebige, ein- bis vierstellige Zeichenfolge. Jede ID darf nur einmal vergeben sein und f Login-Prozesse ("getty") sollte sie der Nummer des Terminals entsprechen ("1" für "/dev/tty1").

Runlevel

Hier stehen alle Runlevel, für die die nachfolgende Aktion auszuführen ist. Da die Bezeichner eines Runleve aus nur einem Zeichen bestehen, werden diese einfach hintereinander geschrieben. Dieses Feld wird ignori wenn im Feld "Aktion" "sysinit", "boot", oder "bootwait" steht.

Aktion

Hier wird das »Wie« des Prozessstarts angegeben, die möglichen Einträge lauten:

respawn

Sobald der Prozess endet, wird er von **init** erneut gestartet

wait

init wartet nach dem Start des Prozesses auf dessen Terminierung

once

Der Prozess wird nur einmal bei Erreichen des angegebenen Runlevels gestartet

boot

Der Prozess wird nur während des Bootvorganges ausgeführt
Der Prozess wird nur während des Bootvorganges ausgeführt und **init** wartet auf dessen Terminier

off

Der Prozess wird niemals gestartet

ondemand

Der Prozess wird gestartet, wenn das angegebene »ondemand«-Runlevel gerufen wurde. Diese Ru b und c) können verwendet werden, um zusätzliche Dienste zu starten, ohne das aktuelle Runlevel

beenden.

initdefault

Der Eintrag gibt das default-Runlevel an, das Prozessfeld wird gnoriert

sysinit

Falls ein solcher Eintrag existiert, wird der angegebene Prozess vor allen anderen gestartet. Typisch Anwendung ist die Initialisierung von Komponenten, die in allen anderen Runleveln verfügbar sein

powerwait

Der Prozess wird gestartet, wenn der Strom ausfällt (ist nur sinnvoll, wenn eine Notstromversorgur eine Zeitlang den Saft liefert), **init** wartet auf das Prozessende

powerfail

Wie »powerwait«, **init** wartet nicht auf das Prozessende

powerokwait

Wenn der Strom wieder anliegt, führt **init** diesen Prozess aus

powerfailnow

Falls die Notstromversorgung allmählich versagt, wird dieser Prozess noch aktiviert

ctrlaltdel

Das Verhalten beim »Affengriff« ([Ctrl]-[Alt]-[Del])

kbrequest

Wenn auf der Tastatur das vereinbarte Signal auftrat, wird der angegebene Prozess ins Leben deru

Prozess

Das zu startende Skript oder Programm.

Nach all der Theorie verfolgen wir, was **init** tatsächlich der Reihe nach vollführt.

Als erstes durchsucht **init** die Tabelle /etc/inittab nach einem Eintrag, dessen Aktion auf **sysinit** steht. Ein solcher Eintrag dient i.A. zur Erledigung der »Hausaufgaben«, wie dem Überprüfen und Mounten der Dateisysteme, Setzen der Systemzeit, Initialisierung des Swap-Speichers, also Arbeiten, die für jedes Runlevel von Nöten sind. Ein solcher Eintrag ist typisch für die meisten RedHat-basierenden Distributionen:

```
si::sysinit:/etc/rc.d/rc.sysinit
```

Debian nutzt den »sysinit«-Eintrag, um die Skripte des Verzeichnisses **/ etc/ init.d/ rcS.d** zu verarbeiten:

```
si::sysinit:/etc/init.d/rcS
```

Ein gänzlich anderes Verhalten legt SuSE-Linux vor, das die globalen Initialisierungen einem **bootwait**-Eintrag vollziehen lässt:

```
# SuSE <= 7.0
si:I:bootwait:/sbin/init.d/boot
# SuSE >= 7.1
si:I:bootwait:/etc/init.d/boot
```

boot und **bootwait** sind die folgenden Einträge, nach denen **init** die Datei durchforstet. Wie man sein System letztlich konfiguriert, ist Geschmackssache. Verwendet man ein einziges Skript, das die allgemeinen Einstellungen regelt, so spielt es keine Rolle, ob es mit einem **sysinit**-, oder **bootwait**- Eintrag in den Bootprozess eingebunden wird. Bei Verwendung mehrerer Skripte kann mit **sysinit** die Verarbeitung eines Skriptes vor den anderen erzwungen werden. Beide Einträge erfordern das Warten von **init**, bis die Prozesse ihre Tätigkeit abgeschlossen haben.

Der für den Administrator wohl interessanteste Eintrag ist die **initdefault**-Zeile, die Linux anweist, in einem konkreten Runlevel zu starten:

```
id:2:initdefault:
```

Der zweite Eintrag bestimmt dabei das zu aktivierende Level. Entgegen den anderen Zeilen der Datei darf hier nur ein einziges Level angegeben werden. Möchte man permanent ein anderes Runlevel einrichten, so kann dieser Wert von Hand editiert oder mittels (distributionsspezifischer) Werkzeuge gesetzt werden:

- Debian: Es geht doch nichts über die Handarbeit...
- RedHat: linuxconf ➜ Verwaltung ➜ Verschiedene Dienste ➜ Initiale Systemdienste
- SuSE: yast ➜ Administration des Systems ➜ Login-Konfiguration ➜ Login-Oberfläche (ein Wechsel ist nur zwischen Runlevel 2 und 3 möglich)

Für jedes von der Distribution definierte Runlevel muss ein Eintrag in der /etc/inittab vorhanden sein. Welche Distribution welches Runlevel mit welcher Eigenschaft versieht, soll Thema des nachfolgenden Abschnitts sein. Hier betrachten wir nur eine exemplarische Zeile aus einem SuSE-System (Versionen vor 7.1):

```
l0:0:wait:/sbin/init.d/rc 0
```

Die anderen Distributionen unterscheiden sich zumeist einzig in der Lage des Skripts **rc**. Unser Beispieleintrag betrifft das Runlevel 0, jenes, welches i.A. das Anhalten des Systems regelt. Erfolgt ein Wechsel zu diesem Runlevel, so wartet **init** auf das Ende des Skripts »rc« (wait). **rc** wird bei jedem Wechsel des Runlevels gerufen, als Argument wird ihm das neue Level übergeben. Die Einträge für alle weiteren Level sind analog aufgebaut.

Um sinnvoll mit dem System arbeiten zu können, muss **init** noch für die Login-Konsolen Sorge tragen:

```
1:2345:respawn:/sbin/mingetty tty1
```

Wieder sei beispielhaft nur ein Eintrag für einen getty-Prozess auf der ersten Konsole erwähnt. Als Bezeichner (ID) empfiehlt das Manual zu inittab die Nummer des betreffenden Terminals. **respawn** besagt hier, dass nach dem Ableben des Prozesses (/sbin/mingetty) sofort ein Neuer zu starten ist. Das ist der Grund, warum nach einem **logout** eine neue Anmeldeaufforderung erscheint.

Unter Linux wird dem normalen Benutzer häufig das Herunterfahren des Systems über den so genannten Affengriff gestattet. Die Reaktion des Systems auf diese Tastenkombination ([Ctrl]-[Alt]-[Del]) wird in der Datei /etc/inittab festgelegt:

```
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
```

Im Beispiel reagiert Linux nach 4 Sekunden Verzögerung mit einem Reboot.

Nach einer Modifikation der Datei "/etc/inittab" sollte der **init**-Prozess seine Konfigurationsdatei neu einlesen. **telinit q** überredet ihn hierzu.

Die Runlevel

Der Begriff des Runlevels tauchte in den vorangegangenen Abschnitten wiederholt auf, ohne ihn im Detail diskutiert zu haben. Dies soll nun schleunigst nachgeholt werden.

Ein **Runlevel** definiert einen Zustand des Unix-Systems. Unter einem Zustand verstehen wir eine bestimmte Konstellation aktiver Prozesse, die während des Bootvorgangs initiiert wurden. So werden in einem Netzwerk-Runlevel zahlreiche Dämonenprozesse gestartet (z.B. inetd, httpd,...), die in einem Modus ohne Netzwerk nicht notwendig sind und demzufolge in dessen Konstellation nicht erscheinen.

Prinzipiell stehen die Runlevel 0 bis 9, a, b, c und s zur Verfügung. 0 und 6 sind dabei von vornherein mit bestimmten Funktionalitäten versehen, die Verwendung der weiteren Level ist stark distributionsabhängig. Traditionell werden die Runlevel 7-9 nicht verwendet, auch a-c sucht man bei den verbreiteten Distributionen vergeblich. »s« wird häufig als Single User Modus verwendet.

Die hinter einem Runlevel stehende Konstellation ist hochgradig konfigurierbar und es steht in der Verantwortung des Administrators, welche Dienste in welchem Zustand im System residieren. Dennoch stellen alle Distributionen mit der Neuinstallation eine brauchbare Konfiguration bereit. Die Runlevel 0 (Systemhalt) und 6 (Reboot) sollten in jeder Linuxdistribution einheitlich belegt sein. Runlevel 1 wird bei Debian, RedHat und SuSE (erst ab 7.3) ebenso einheitlich als *Single User Mode* konfiguriert. Darüber hinaus unterscheiden sich die Vorgehensweisen der Distributoren.

Debian

Im Runlevel *s* stehen alle Skripte, die einmalig während des Bootvorgangs abgearbeitet werden müssen (es handelt sich also um kein eigenständiges Runlevel). Diese Skripte werden vor dem eigentlich zu startendem Runlevel ausgeführt, nicht jedoch beim Wechsel der Runlevel im laufenden System (telinit).

Die Runlevel 2-5 werden bei Debian identisch gehandhabt. Was konkret in ihnen geschieht, hängt einzig von der installierten Software ab (ist bspw. der Xdm installiert, wird das grafische Login aktiviert).

Redhat

RedHat kennt das Runlevel 2 als *Multiuser ohne Netzwerk*. Im Runlevel 3 kommt die Netzwerkfunktionalität hinzu.

Runlevel 4 wird in der Standardinstallation nicht verwendet.

Runlevel 5 ist die »volle Ausbaustufe«; also *Multiuser mit Netzwerk und grafischem Login*.

SuSE

Ab Version 7.3 entspricht das Startverhalten der SuSE-Distribution dem von RedHat.

In früheren SuSE-Versionen stand Runlevel S für den *Single User Modus*. Runlevel 1 diente als *Multiuser ohne Netzwerk*, 2 als *Multiuser mit Netzwerk* und 3 als *Multiuser mit Netzwerk und grafischem Login*. Runlevel 4 und 5 wurden nicht verwendet (in 7.2. war 5 identisch zu 3).

Init-Skripte

Zahlreiche Skripte werden in verschiedenen Runleveln benötigt. Man denke nur an den Protokollanten von Linux, den syslogd, dessen Dienste eigentlich immer nützlich sein sollten. Um solche "Init"-Skripte nicht mehrfach im System speichern zu müssen, sammelt man alle in einem einzigen Verzeichnis. Dieses ist:

- "/etc/init.d" bei Debian
- "/etc/rc.d/init.d" bei RedHat
- "/sbin/init.d" bei SuSE <= 7.0
- "/etc/init.d" bei SuSE ab 7.1

Die Namensgebung der einzelnen Skripte verrät oftmals deren Zweck (Beispiel aus RedHat-Linux):

```
user@sonne> ls / etc/ rc.d/ init.d
```

```
apmd      functions  inet      linuxconf  network  random   rwhod     snmpd
arpwatch  gpm        keytable  lpd        nfslock  routed   sendmail  syslog
atd       halt       killall   named      pcmcia   rstatd   single    xfs
crond     identd     kudzu     netfs      portmap  rusersd  smb       ypbind
```

Jedes konfigurierte Runlevel ist im Dateisystem durch ein eigenes Verzeichnis repräsentiert, wobei die Namensgebung **rc*x*.d**, *x* steht für das Runlevel, lautet. Auch hier gehen die Distributionen, was die Lage dieser Verzeichnisse betrifft, eigene Wege:

- "/etc/rc*x*.d" bei Debian
- "/etc/rc*x*.d" bei RedHat (ab 7.0 sind dies Links auf gleichnamige Verzeichnisse in /etc/rc.d/)
- "/sbin/init.d/rc*x*.d" bei SuSE <= 7.0
- "/etc/rc.d/rc*x*.d" bei SuSE ab 7.1

Namensgebung der Runlevelskripte

Jedes Skript, das nun in einem Runlevel zu starten ist, erscheint als Link unter zwei verschiedenen Namen im Verzeichnis des Runlevels. Dabei ist die Namensgebung der Verweise verbindlich und in unten stehender Abbildung skizziert.

Abbildung 3: Die Namensgebung der Runlevelskripte

Die Linknamen setzen sich aus drei Komponenten zusammen, die das Skript **rc** bewertet:

- **Der erste Buchstabe** beschreibt, ob es ein Stoppskript (Buchstabe = K) oder ein Startskript (Buchstabe = S) ist
- **Zwei Ziffern** bestimmen die Priorität des Skriptes (00 bis 99). Von den Skripten in einem Runlevel werden alle mit kleinerer Nummer vor den Skripten mit höherer Nummer abgearbeitet, somit wird sicher gestellt, dass ein Dienst, der einen anderen zu seiner Arbeit bedingt, erst nach diesem aktiviert wird. Skripte mit identischer Nummer sind voneinander unabhängig und werden entsprechend ihrer alphabetischen Ordnung betrachtet.
- **Der Rest** ist der »eigentliche« Name und kann beliebig gewählt werden. Normal wird man sich für einen Namen entscheiden, der auf das entsprechende Skript schließen lässt.

Die Arbeitsweise des Skripts rc

Das *Resource Control*-Skript wird immer mit dem zu startenden Runlevel als Argument aufgerufen, also z.B. "rc 3" oder "rc S". **rc** bestimmt zunächst das alte Runlevel (der Schritt entfällt beim Systemstart), wechselt in dessen Verzeichnis und liest alle mit einem **K** beginnenden Links ein. Jedes dieser Skripte wird, entsprechend der durch die Priorität vorgegebenen Reihenfolge, mit dem Argument **stop** gestartet und auf dessen Terminierung gewartet. Nachdem alle Stoppskripte behandelt wurden, wechselt **rc** in das Verzeichnis des neuen Runlevels und verfährt analog mit den dortigen **S**-Links, denen als Argument **start** mitgegeben wird.

Der Aufbau eines Init-Skripts

Bei den Init-Skripten handelt es sich um nichts anderes als (Bash) Shellskripte. Sie können demzufolge ebenso

»von Hand« aufgerufen werden. So ist Root durchaus berechtigt mit:

```
# Beispiel aus RedHat-Linux
root@sonne>  /etc/rc.d/init.d/network stop
root@sonne>  /etc/rc.d/init.d/network start
```

das gesamte Netzwerk zunächst herunterzufahren, um es anschließend erneut zu starten (z.B. nach Änderungen in der Konfiguration). Darüber hinaus kennen die meisten Skripte weitere Argumente:

- **probe** testet, ob seit dem letzten Neustart des Dienstes dessen Konfigurationsdatei(en) modifiziert wurde(n)
- **reload** arbeitet wie "restart"
- **restart** Neustart eines Dienstes (wie "stop" mit anschließendem "start")
- **status** zeigt Statusinformationen an (bei Samba z.B. die aktiven Verbindungen)

Als Beispiel dient ein Skript, das den X-Font-Server **xfs** in einem RedHat-System startet:

```
#!/bin/sh
#
# xfs:       Starts the X Font Server
#
# Version:      @(#) /etc/rc.d/init.d/xfs 1.6
#
# chkconfig: 2345 90 10
# description: Starts and stops the X Font Server at boot time and shutdown.
#
# processname: xfs
# config: /etc/X11/fs/config
# hide: true

# Source function library.
. /etc/rc.d/init.d/functions
# See how we were called.
case "$1" in
  start)
        echo -n "Starting X Font Server: "
        rm -fr /tmp/.font-unix
        daemon xfs -droppriv -daemon -port -1
        touch /var/lock/subsys/xfs
        echo
        ;;
  stop)
        echo -n "Shutting down X Font Server: "
        killproc xfs
        rm -f /var/lock/subsys/xfs
        echo
        ;;
  status)
        status xfs
        ;;
  restart)
        echo -n "Restarting X Font Server. "
        if [ -f /var/lock/subsys/xfs ]; then
            killproc xfs -USR1
        else
            rm -fr /tmp/.font-unix
            daemon xfs -droppriv -daemon -port -1
            touch /var/lock/subsys/xfs
        fi
        echo
        ;;
  *)
        echo "*** Usage: xfs {start|stop|status|restart}"
        exit 1
esac

exit 0
```

Mit etwas Kenntnis zur Shellprogrammierung sollte der Leser in der Lage, eigene Init-Skripte zu verfassen und

somit zusätzliche Dienste den Runleveln hinzuzufügen.

Wechsel des Runlevels

Im laufenden Betrieb ist der Wechsel eines Runlevels ebenso möglich. Denkbar ist, dass Root das System im Single User Modus startet, irgendwelche administrativen Eingriffe vornimmt und nachfolgend Linux in den »normalen« Multi User Modus mit Netzwerk versetzt. Ein solcher »online«-Wechsel ist wesentlich schneller vollzogen als ein Reboot. Der Systemadministrator behilft sich hierzu des Kommandos **telinit** (meist ein Link auf "/sbin/init"), das er mit dem neuen Runlevel als Argument startet:

```
root@sonne> telinit 3
```

Bearbeiten der Runlevelskripte

Das Hinzufügen und Entfernen von Skripten zu einem Runlevel kann in althergebrachter Manier per Hand erfolgen. Etwas komfortabler geht es mit dem Editor **ksysv** vom KDE-Projekt (Abbildung 4), der per Drag&Drop die Verwaltung der Links ermöglicht. Per Dialogboxen arbeitet in RedHat-System **ntsysv**.

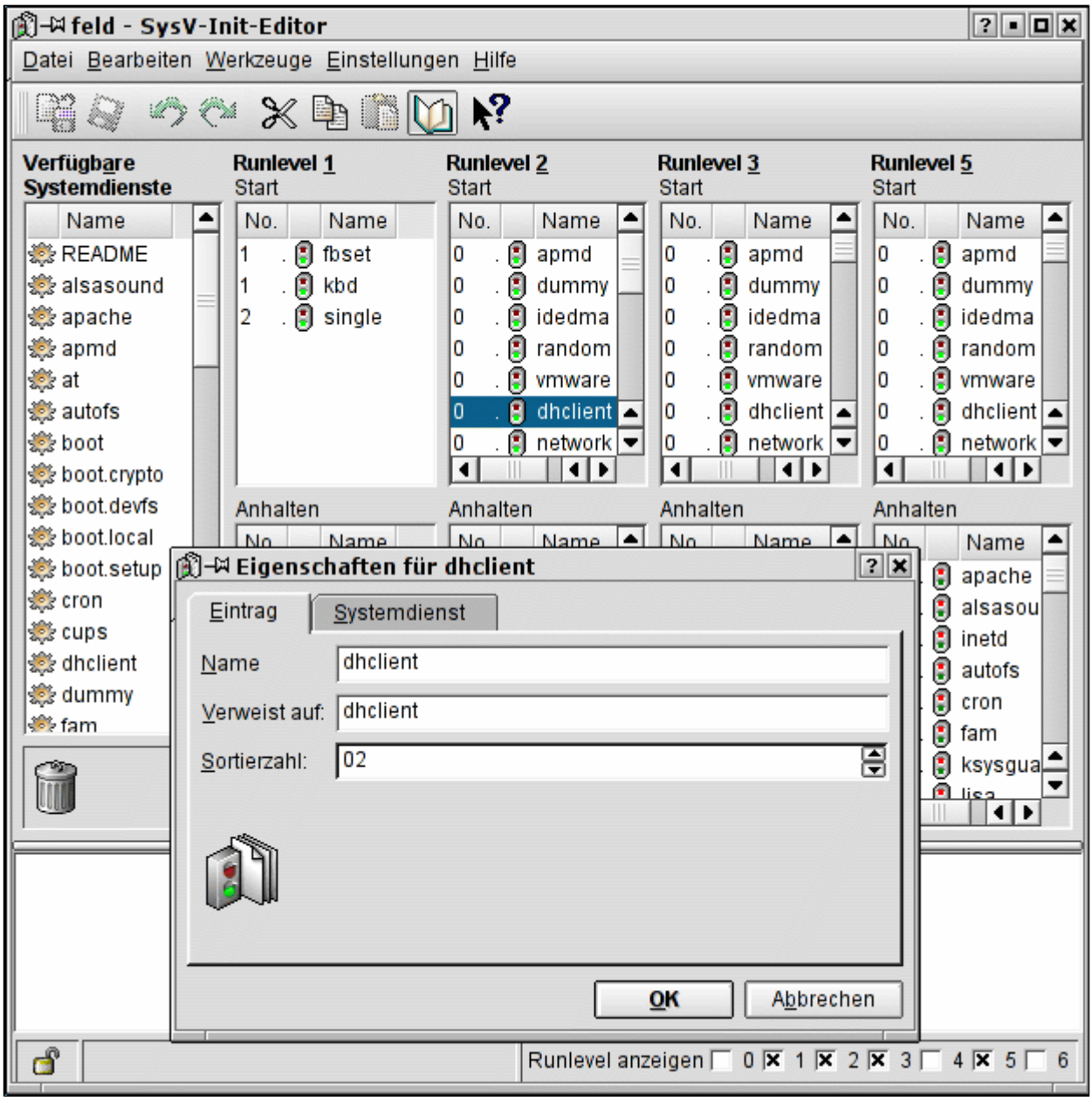


Abbildung 4: Der KDE-Runlevel-Editor

Achtung: Unter SuSE-Linux genügt das alleinige Hinzufügen der Links bei vorgefertigten Diensten nicht, da die SuSE-Skripte intern die Datei /etc/rc.config einlesen und dortige Variablen auswerten. Sie müssen die entsprechende Variable in /etc/rc.config (bspw. »START_XFS« für den X-Font-Server »xfs«) entweder von Hand oder per »Yast1« auf »yes« setzen.

Alternative Konzepte - Ramdisks

Um auf Hardware zugreifen zu können, benötigt Linux entsprechende Treiber. Diese Treiber können entweder direkt im Kernel integriert sein oder als dynamisch ladbare Module vorliegen. Die Menge der fest einkompilierten Treiber ist allerdings beschränkt, da in der Ladephase des Kernels dieser vollständig in den Hauptspeicher passen muss. Zum Zeitpunkt des Ladens (gewöhnlich nimmt dies ein Bootloader vor) befindet sich der Rechner allerdings noch im so genannten *Real Modus*, d.h. der adressierbare Hauptspeicher und damit auch die Größe des Kernels sind begrenzt (aus DOS-Zeiten sind Ihnen sicherlich 640kByte und die Tricks mit EMS und XMS noch ein Begriff).

Um Module verwenden zu können, müssen diese irgendwo im Dateisystem liegen. Der Kernel muss auf das Dateisystem zugreifen können, womit zumindest die Treiber für das entsprechende Dateisystem und für das Gerät, auf dem dieses liegt (Festplatten-Controller, SCSI-Adapter...) fest im Kernel enthalten sein müssen. Nun gibt es aber eine ganze Reihe von Controllern, über die die Festplatten angesprochen werden, und jeder Controller benötigt einen anderen Treiber. Ein universeller Kernel, wie er bspw. bei der Installation erforderlich ist, müsste also alle erdenklichen Treiber enthalten, um auf allen erdenklichen Hardware-Konstellationen laufen zu können. Ein solcher Kernel wäre vermutlich wieder zu groß um in den Hauptspeicher zu passen... womit wir der Lösung des Problems kein Stück näher gerückt wären.

Und hier setzen **initrd** und **Ramdisks** an, die den eigentlichen **Bootvorgang in zwei Schritten** realisieren. Eine Ramdisk belegt einen Teil des Hauptspeichers (RAM) und legt in diesem ein Dateisystem an (DISK). Ein Bootloader legt nun im Speicher hintereinander die Datei »initrd« und den Kernel ab. Der nachfolgend zu startende (minimale) Kernel enthält nun den Treiber, um den Inhalt von »initrd« in eine Ramdisk zu entpacken und diese als sein Root-Dateisystem zu mounten. Die ursprüngliche Datei »initrd« wird aus dem Hauptspeicher entfernt.

In der Ramdisk sollte nun eine Datei /linuxrc existieren, die nun abgearbeitet wird (i.A. beinhaltet diese ausführbare Datei Schritte zum Testen der Hardware und zum Laden der notwendigen Module zum Zugriff auf die erkannten Geräte). Sobald die Datei abgearbeitet wurde, wird das »eigentliche« Dateisystem als Wurzelverzeichnis gemountet. Existiert in diesem das Verzeichnis »/initrd«, wird die Ramdisk dorthin verschoben, anderenfalls wird sie abgehangen.

Des Weiteren wird mit dem Start von /sbin/init wie »üblich« fortgefahren.

Das Erstellen einer Ramdisk

An dieser Stelle können wir das Thema nicht erschöpfend darstellen. Deshalb werden wir die erforderlichen Schritte durchlaufen, um eine funktionsfähige Ramdisk zu erstellen. Diese wird nichts anderes tun, als den Benutzer während des Systemstarts zu einer Eingabe aufzufordern. Das Beispiel wird hoffentlich die Klippen beim Vorgehen verdeutlichen und kann als Basis für eigene Experimente dienen.

Ramdisk erstellen und mounten

Sie benötigen ein Dateisystem. Möglich wären:

Verwendung einer Diskette

```
# Formatieren der Diskette (ext2)
root@sonne> mke2fs -m0 /dev/fd0
# Mounten der Diskette
root@sonne> mount /dev/fd0 /mnt
```

Verwendung des Loopback-Devices (dies muss der Kernel unterstützen!)

```
# Erstellen einer Datei (hier 4M groß)
root@sonne> dd if=/dev/zero of=initrd bs=1k count=4096
```

Verwendung eines Teils des Hauptspeichers über das Device / dev/ ram

```
# Dateisystem mit 4096 1k-Blöcken
root@sonne> mke2fs -m0 / dev/ ram -b 1024 4096
# Mounten
root@sonne> mount / dev/ ram / mnt
```

Anmerkung: Die Größe des Dateisystems hängt natürlich von den darin zu installierenden Dateien (Programme, Bibliotheken) ab. In bestimmten Situationen kann die Kapazität einer Diskette schon zu gering sein. Das nachfolgende Beispiel benötigt z.B. ca. 2M!

Dateien kopieren

Dieser Schritt erfordert die sorgfältigsten Überlegungen und ist Quelle zahlreicher Fehler. Es gilt alle Programme, Bibliotheken, Devices und Konfigurationsdateien zu erzeugen, die von der später auszuführenden Datei »linuxrc« benötigt werden. Greifen Sie auf nur sehr wenige Programme zurück, sollten Sie statisch kompilierte Versionen verwenden, da Sie dann die speicherintensiven Bibliotheken einsparen. Allerdings bedarf es einiger Tests, bei welchen Zusammenstellungen statisch und wo dynamische gelinkte Versionen weniger Ressourcen verzehren. Überzeugen Sie sich bei dynamischen Programmen, dass sämtliche Bibliotheken ebenfalls vorhanden sind. Das Notwendige verrät Ihnen der Linker:

```
# Welche Bibliotheken benötigt die dynamische Version der bash?
root@sonne> ldd bash
      libncurses.so.4 => /lib/libncurses.so.4 (0x4001d000)
      libdl.so.2 => /lib/libdl.so.2 (0x4005c000)
      libc.so.6 => /lib/libc.so.6 (0x4005f000)
      /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Im neuen Dateisystem (das unter »/mnt« gemountet ist) sollten zumindest folgende Verzeichnisse existieren:

```
root@sonne> ls -l / mnt
total 16
drwxr-xr-x   2   root    root      1024 Jun  2 13:57  /mnt/bin
drwxr-xr-x   2   root    root      1024 Jun  2 13:47  /mnt/dev
drwxr-xr-x   2   root    root      1024 May 20 07:43  /mnt/etc
drwxr-xr-x   2   root    root      1024 May 27 07:57  /mnt/lib
drwxr-xr-x   2   root    root     12288 May 27 08:08  /mnt/lost+found
```

In »realen« Ramdisks ist die Verzeichnisstruktur komplexer. Meist existieren die vom Filesystem Hierarchie Standard vorgeschriebenen Verzeichnisse. Für unsere Zwecke reichen die oben benannten Einträge aus.

Unsere Demonstrations-Ramdisk benötigt einzig eine Shell »bash« und das Kommando »echo«. Wir verwenden die dynamischen Varianten der Programme und legen zusätzlich den Link «sh« auf »bash« an (anstelle der »bash« könnte man auch eine der »schlanken« Shells wie die »ash« verwenden):

```
root@sonne> ls -l / mnt/ bin
total 497
-rwxr-xr-x   1   root    root     490932 Nov  9  1999  /mnt/bin/bash
-rwxr-xr-x   1   root    root       7972 Nov 22  1999  /mnt/bin/echo
lrwxrwxrwx   1   root    root          4 Jan 27  2000  /mnt/bin/sh -> bash
```

Als einziges Device benötigen wir /dev/console. Fehlt dieses, würde der Kernel mit *Warning: unable to open an initial console.* seinen Dienst quittieren. Möchten Sie auf Hardware zugreifen, müssen Sie die Devices für diese anlegen. Im Falle von /dev/console geht man wie folgt vor:

```
# Device anlegen mit "mknod", die Argumente bedeuten:
```

```
#       "c" = character
#       "5" = Hauptgerätenummer
#       "1" = Nebengerätenummer:
root@sonne> mknod /mnt/dev/console c 5 1
# Default-Rechte ändern:
root@sonne> chmod 622 /mnt/dev/console
root@sonne> ls -l /mnt/dev
crw--w--w-   1   root    root      5,    1 Sep  1 09:25  /mnt/dev/console
```

Beim Anlegen weiterer Devices entnehmen Sie die notwendigen Haupt- und Nebengerätenummern der Datei »/usr/src/linux/Documentation/devices.txt«. Bei einem blockweise arbeitenden Gerät verwenden Sie anstatt »c« ein »b«.

Da wir dynamische Programme verwenden, muss dem dynamischen Linker/Loader **ld.so** mitgeteilt werden, in welchen Pfaden er die Bibliotheken suchen soll. Die kompilierten Pfade beinhaltet die Datei »/etc/ld.so.cache«, die wir in unser »etc«-Verzeichnis kopieren:

```
root@sonne> ls -l /mnt/etc
-rw-r--r--   1   root    root      43258 Sep  1 09:21  /mnt/etc/ld.so.cache
```

Schließlich benötigen wir noch die Bibliothek des dynamischen Linkers und Loaders »ld-linux.so.*Versionsnummer*« und alle von den Programmen »bash« und «echo« benötigten Bibliotheken. Damit sieht der Inhalt des Verzeichnisses »lib« wie folgt aus:

```
root@sonne> ls -l /mnt/lib
-rwxr-xr-x   1   root    root     360274 Jan 27  1999  /lib/ld-2.1.2.so
lrwxrwxrwx   1   root    root         11 Jan 27  2000  /lib/ld-linux.so.1 -> ld-2.linux.so.2
lrwxrwxrwx   1   root    root         11 Jan 27  2000  /lib/ld-linux.so.2 -> ld-2.1.2.so
lrwxrwxrwx   1   root    root         11 Jan 27  2000  /lib/libc.so.5 -> libc.so.6
-rwxr-xr-x   1   root    root    4223971 Jan 27  1999  /lib/libc.so.6
-rwxr-xr-x   1   root    root      75167 Jan 27  1999  /lib/libdl.so.2
lrwxrwxrwx   1   root    root         11 Jan 27  2000  /lib/libncurses.so -> libncurses.so.4
lrwxrwxrwx   1   root    root         11 Jan 27  2000  /lib/libncurses.so.4 -> libncurses.so.4.2
-rwxr-xr-x   1   root    root     526454 Jan 27  1999  /lib/libncurses.so.4.2
```

Manche Programme erwarten bestimmte Versionen einer Bibliothek. Meist enthalten die neueren Bibliotheken dieselben Schnittstellen wie ihre Vorgänger, so dass es genügt, einen Link unter dem alten Namen anzulegen (dies klappt leider nicht immer!). Inwiefern Sie tatsächlich obige Links benötigen, hängt von den konkreten Versionen der von Ihnen verwendeten Programme ab.

linuxrc

Diese ausführbare Datei befindet sich in der Wurzel des neuen Dateisystems und wird automatisch gestartet. Bei ihr wird es sich zumeist um ein Shellskript handeln, das alle Schritte, die innerhalb der Ramdisk auszuführen sind, beinhaltet. Es handelt sich quasi um die zentrale Steuerdatei; mit ihrer Beendigung fährt der Kernel mit der »normalen« Bootsequenz fort (Mounten des eigentlichen Rootverzeichnisses und Start von »init«). Unser einfaches Beispiel soll einzig einen Text ausgeben und auf eine beliebige Eingabe warten.

```
root@sonne> cat /mnt/linuxrc
#!/bin/sh

echo "Die Datei linuxrc wird ausgefuehrt"
read
# Execute-Rechte setzen:
root@sonne> chmod a+x /mnt/linuxrc
```

Zum Abschluss ist das gemountete Dateisystem abzuhängen und das »Image« in eine Datei (vorteilhaft im Verzeichnis /boot) zu kopieren:

```
root@sonne> umount /mnt
# Kopieren: falls Sie eine Diskette verwendet haben:
root@sonne> dd if=/dev/fd0 of=/boot/initrd
```

```
# Kopieren: falls Sie ein Loopback-Device verwendet haben:
root@sonne> cp initrd / boot/
# Kopieren: falls Sie /dev/ram verwendet haben:
root@sonne> dd if= / dev/ ram bs= 1k count= 4096 of= / boot/ initrd
# Eine Ramdisk sollte freigegeben werden:
root@sonne> freeramdisk / dev/ ram
```

Überzeugen Sie sich noch, dass das Device /dev/initrd existiert:

```
root@sonne> ls -l / dev/ initrd
brw-rw----   1   root     disk     1,  250  Jan 27  2000  /dev/initrd
```

Zu Testzwecken ist es günstig, wenn Sie ein Verzeichnis »/initrd« erzeugen. In dieses wird nach dem Booten die Ramdisk gemountet und kann somit einfach modifiziert werden.

Der Kernel

Eventuell müssen Sie erst einen neuen Kernel generieren, um diesen zur Verwendung einen initialen Ramdisk vorzubereiten. Aktivieren Sie auf jeden Fall die Optionen:

- RAM disk support
- Initial RAM disk (initrd) support

Bootloader

Beim Bootloader sind Sie entweder auf Lilo, LOADLIN oder Chos angewiesen. Alle drei unterstützen die Verwendung von initrd. Tragen Sie in die Konfigurationsdateien des von Ihnen verwendeten Bootmanagers die Zeile "initrd=/boot/initrd" ein und installieren Sie den Bootsektor neu. Das genaue Vorgehen finden Sie im Abschnitt Bootmanager beschrieben.

Alternative Konzepte - Booten übers Netz

Das dritte hier vorgestellte Bootkonzept tangiert zahlreiche Aspekte der Netzwerkadministration. Wir beschränken uns an dieser Stelle auf eine motivierende Einleitung und werden die Schritte zur Konfiguration einzig benennen. Die konkreten Realisierungen zur Bereitstellung notwendiger Dienste wird man im Abschnitt Netzwerk-Dienste finden.

Beim Booten übers Netz besitzen die Clients einzig ein minimales Programm, das in der Lage ist, während des Bootens einen entsprechenden Serverrechner im Netzwerk zu kontaktieren und sein Root-Dateisystem von diesem zu importieren. Das Vorgehen ermöglicht bspw. den kostengünstigen Aufbau eines ganzen Rechner-Pools, denn auf Clientseite kann sowohl auf Festplatten als auch auf Disketten- oder CDROM-Laufwerke verzichtet werden. Mit einem schnellen Netzwerk (100 MBit), mittelprächtigem Prozessor (kleiner Pentium) und relativ bescheidenem Hauptspeicherausbau (32 MB) wird ein Anwender an einem Client kaum einen Performance-Unterschied feststellen (im Gegenzug sollte der Server mit reichlich Leistung aufwarten). Ein nicht unwesentlicher Aspekt ist die nun einfache Wartung der Software, da diese an einem Ort (Server) konzentriert ist und ein Update damit transparent für alle Clients ist.

Der Client

Der Client benötigt zunächst den **Bootkode**, damit er während des Hochfahrens auf einen Server zugreifen kann. Der übliche Weg ist, diesen Kode in einem EPROM (Erasable Programmable Read Only Memory) auf der Netzwerkkarte zu halten. Den Kode selbst erhält man u.a. mit den freien Paketen »Etherboot« (aktuelle Version 4.6.6) oder »Netboot« geliefert, allerdings besteht das Problem des Beschreibens des EPROMs, wofür es eines bestimmten Gerätes bedarf. Alternativ kann zum Booten eine Diskette verwendet werden.

Der Bootkode enthält nun den Treiber für die jeweilige Netzwerkkarte sowie die Unterstützung der Protokolle BOOTP (oder DHCP) und TFTP.

BOOTP dient nun der **Feststellung der IP-Adresse** des lokalen Rechners. Denn da dieser über keinerlei permanenten Speicher verfügt (außer dem EPROM), kann er die ihm zugewiesene IP-Adresse nicht wissen. Zum Glück besitzt jede Ethernet-Karte eine weltweit einmalige Adresse. Ein Client sendet also eine BOOTP-Anfrage per Broadcast (denn die Adresse des Servers kennt er ebenso wenig) ins Netz. Ein BOOTP-Server sollte mit der dem Client zugeordneten IP-Adresse antworten.

Das **Laden des Kernels** erfolgt über eine »abgespeckte« Variante des bekannten FTP-Protokolls, dem »Trivial FTP« (TFTP). Hauptmerkmale gegenüber dem normalen FTP sind eine fehlende Authentifizierung und die Verwendung des UDP-Protokolls (anstatt dem TCP) zur Datenübertragung.

Damit ist die Arbeit für den Bootkode erledigt und er wird die Kontrolle dem soeben heruntergeladenen Kernel übergeben. Dieser sollte die Fähigkeit besitzen, sein Root-Dateisystem via NFS (Network File System) von einem Server zu importieren.

Der Server

Für gewöhnlich wird ein Server alle Aufgaben erledigen, aber das ist kein Muss. Der Server muss folgende Dienste erbringen:

- **BOOTP-Server:** Der Server antwortet auf BOOTP-Anfragen aus dem Netz. Dazu verfügt er über eine Datenbank (/etc/bootptab) mit der Zuordnung von Ethernet- zu IP-Adresse (und ggf. weitere Informationen). Im Falle vom Booten übers Netz enthält die Datenbank auch den Namen des Kernels für den Client.
- **TFTP-Server:** Um dem Client den Kernel zuzuschieben, findet das TFTP-Protokoll Anwendung. Dieser Kernel muss den Treiber für die Netzwerkkarte und die Unterstützung für das Root-Dateisystem via NFS beinhalten (keine Module). Außerdem muss ihm ein bestimmter Header voranstehen, der mit dem Kommando »mknbi-linux« erzeugt wird.
- **NFS-Server:** Der Server muss ein Verzeichnis an den Client exportieren, das diesem als Wurzelverzeichnis dient. Aus Sicherheitsgründen sollte es sich nicht um das Wurzelverzeichnis des Servers handeln, sondern unter /tftpboot/*Client-IP-Adresse* liegen.

Datensicherung

Übersicht

Die regelmäßige Sicherung des Datenbestandes zählt zu den wichtigsten Aufgaben eines Systemverwalters. Ein Datenverlust kann verschiedene Ursachen haben. Sei es durch einen Hardwareausfall, durch einen Softwarefehler, sei der Grund eine Fehlbedienung oder ein Virus oder es zeichnet sich gar eine dubiose Gestalt mit böswilligen Manipulationen dafür verantwortlich. Nach Murphy's Gesetz ereilt einen das Schicksal stets im unpassendsten Augenblick. Glück für denjenigen, der durch eine besonnene Strategie der Datensicherung den Schaden in Grenzen halten konnte.

Die Begriffe »Backup« und »Archiv« verwenden wir im weiteren Verlauf synonym. Allgemein bezeichnen wir als ein A r c h i v einen Container von Daten, der aufbewahrt wird, um genau auf diese Daten später zugreifen zu können. Archive legt man an, um momentan nicht benötigte Daten von der Festplatte zu verbannen oder um sie von einem Rechner auf einen anderen zu kopieren...

Die Motivation zu einem B a c k u p resultiert aus anderen Überlegungen. Hier ist die Absicht, einen bestimmten Systemzustand aufzuzeichnen. Dabei bezieht man sich nicht auf den konkreten Inhalt einer Datei - wie bei Archiven - sondern ordnet eine Datei in die Schublade »wichtig« oder »unwichtig« ein, je nachdem, ob sie von meiner Backupstrategie betroffen sein wird oder nicht.

Grundsätzlich werden zwei Arten des Backups unterschieden. Das v o l l e B a c k u p sichert stets den kompletten Bestand an Daten, während das i n k r e m e n t e l l e B a c k u p nur Daten archiviert, die innerhalb einer bestimmten Periode modifiziert wurden. Dabei steht die Periode meist für den Zeitraum seit der letzten Sicherung.

Ein volles Backup ist wegen des hohen Bedarfs an Zeit (die Sicherung erfolgt meist auf ein Bandmedium) weniger für den täglichen Einsatz geeignet, deswegen entscheidet man sich heute meist für eine Mischform aus voller und inkrementeller Datensicherung. Hierbei wird zu einem Zeitpunkt der komplette Datenbestand gesichert und nachfolgend - in regelmäßigen Abständen - archiviert man nur die modifizierten Daten.

Die tatsächlich gewählte Strategie des inkrementellen Backups hängt stark von der »Statik« der Daten und vom Sicherheitsbedürfnis des Administrators ab. Im einfachsten Fall betrifft das Backup jeweils alle modifizierten Daten seit dem letzten vollen Backup bis hin zum aktuellen Zeitpunkt. Das umgekehrte Verfahren ist die Aufzeichnung der Änderungen seit der letzten inkrementellen Sicherung. Während in ersterem Fall nur die Archive des vollen und letzten inkrementellen Backups aufbewahrt werden müssen, sind beim zweiten Vorgehen das volle Backup und sämtliche inkrementellen Datensicherungsarchive aufzuheben. In der Praxis findet man auch Archivierungsformen zwischen den beiden beschriebenen Extremen (Multilevel-Backups).

Bedenken Sie, dass ein Festplattenfehler Sie auch während der Aufzeichnung eines Backups ereilen kann. Überschreiben Sie daher niemals ihre aktuelle Sicherungskopie.

Bei der Vorstellung der Backup-Werkzeuge beschränken wir uns auf Programme, die der GPL oder einer ähnlichen Lizenz unterstehen. In vielen Fällen lassen sie die Bedienfreundlichkeit kommerzieller Produkte vermissen, jedoch ist es durchaus möglich - die notwendigen Kenntnisse vorausgesetzt - diese in Skripten so zu kapseln, dass sie zum einen die Funktionalität und zum anderen die einfache Benutzbarkeit ausgereifter Werkzeuge erreichen.

Medien für die Datensicherung

Wohin mit den ganzen Daten?

Zunächst ist einfach nachvollziehbar, dass die Speicherkapazität des Backupmediums dem aufkommenden

Datenvolumen entsprechen muss. Ein volles Backup könnte man durchaus auch auf Disketten vornehmen, jedoch wird man des Wechsels sicherlich bald überdrüssig und die Diskette ist selbst schon Quelle des lauernden Datenverlusts. Wer kennt nicht das leidige »Can't read sector xxx.«? Besser geeignet - vor allem bei dem geringeren Datenaufkommen des inkrementellen Backups - sind ZIP-Disketten.

Traditionell sind Magnetbänder (Quarter-Inch- und DAT-Streamer) weit verbreitet. Sowohl Kapazität als auch Zuverlässigkeit sprechen für dieses Medium. Leider lässt sich auf einem Band kein Dateisystem einrichten, so dass die Daten mit Hilfe spezieller Programme sequentiell auf dieses abgelegt werden müssen. Im Falle eines »Restores« muss demnach der gesamte Bandinhalt zurückgespielt werden, da ein wahlfreier Zugriff nicht möglich ist. Ein Magnetband muss vor dem Zugriff zurückgespult werden, hierzu nutzt man das Kommando **mt** mit der Option **rewind**:

```
user@sonne> mt -f <Device> rewind
```

Es ist außerdem möglich, mehr als ein Archiv auf einem Band unterzubringen, immer vorausgesetzt, die Speicherkapazität setzt keine Schranken:

```
# Band zur ersten Dateiende-Markierung spulen
user@sonne> mt -f <Device> fsf 1
# Band zur dritten Dateiende-Markierung spulen
user@sonne> mt -f <Device> fsf 3
# Band zur letzten Dateiende-Markierung spulen
user@sonne> mt -f <Device> eof
```

Waren früher die Medien für einem Streamer die preiswerteste Alternative, so geht heute nichts über eine Festplatte. Dem System eigens für die Datensicherung eine eigene Platte zu spendieren, mag auf den ersten Blick befremdend erscheinen, jedoch erhält man für den Preis eines Streamers eine Menge Festplattenspeicher. Die weiteren Vorteile liegen im wahlfreien und wesentlich zügigerem Zugriff. Allerdings schützt die eingebaute Festplatte nur mäßig vor mutwilliger Sabotage, hier kann eine Datensicherung auf einem anderen Rechner (über das Netz) Abhilfe bringen. Die durch RAID-Systeme realisierbare Datenredundanz schützt nur vor Hardwareausfall einer Festplatte, hat also nichts mit einem Backup zu tun!

Während sich die »normale« CD-ROM bzw. DVD wegen der nur einmaligen Beschreibbarkeit eher als Medium zur reinen Archivierung eignet, kann die CD-RW und vor allem die DVD-RW durchaus für die alltägliche Datensicherung eingesetzt werden. Allerdings erfordern umfangreiche Backups, die die Kapazität einer einzelnen CD-ROM übersteigen, die Anwesenheit des Administrators.

Welche Daten sind Kandidaten?

Es lohnt sich, etwas Zeit in den Plan zu investieren, welche Daten denn überhaupt den Aufwand eines Backups rechtfertigen.

Vielleicht beginnen wir mit Daten, die sichere Kandidaten sind. Da wären:

- Die Daten unter /home. Klar, die Benutzer ziehen sich ja täglich neue Daten aus dem Netz.
- Die Daten unter /etc. Oder haben Sie Interesse, Ihr System komplett neu konfigurieren zu müssen?
- Die Daten unter /var. Hier liegen Protokolle, Mails, Druckaufträge... also Daten, die i.d.R. permanenter Änderung unterliegen.
- Projektverzeichnisse, Datenbanken, ...???

Als Daten, deren Aufzeichnung Sie sich getrost sparen können, sind all jene zu nennen, die ohnehin auf einem anderen Medium vorhanden sind, z.B. die Daten der Linuxdistribution, die auf CD-ROM im Schrank liegen. Ebenso sollten Dateisysteme, die von entfernten Rechnern gemountet wurden, besser von dessen Backup-Strategie erfasst werden.

Aber letztlich ist es die Ermessensfrage des Administrators, welche Daten er für sicherungswürdig erachtet. Eine Neuinstallation von Linux hat ja manchmal auch den Vorteil, etwas Ordnung im System zu schaffen..

Backup mit tar

Der *Tape Archiver* ist das verbreitetste Werkzeug zur Datensicherung in Unix-Systemen. Ursprünglich rein zur Datenübertragung auf Bänder konzipiert, mit deren Hilfe der Datenaustausch zwischen verschiedenen Computern zu Zeiten fehlender Netzwerke ermöglicht werden sollte, hat sich gerade durch die Eignung von Bändern als Backupmedium das Programm quasi zum Standardwerkzeug für das Backup in kleineren Netzwerken etabliert, trotz des Fehlens einer brauchbaren Unterstützung von inkrementellen Backups.

Als Werkzeug zur Archivierung wurde tar im Kapitel Nutzerkommandos bereits vorgestellt. Hier soll nun der Schwerpunkt auf den Einsatz des Kommandos zum Backup von Daten liegen.

Aufruf: tar [OPTION...] [DATEI...]

tar arbeitet auf Dateiebene, kann also beliebige Dateien und Verzeichnisse in eine einzige Zieldatei verpacken. Zieldatei kann dabei alles bedeuten, was unter Unix als Datei betrachtet wird, also ein Gerät, eine normale Datei, eine Pipe...

Die drei wichtigsten Optionen sind **c** zum Erzeugen eines Archivs, **t** zum Betrachten des Archivinhalts und **x** zum Entpacken desselben.

Volles Backup

tar wird alle angegebenen Dateien (oder rekursiv den Inhalt von Verzeichnissen) in ein Archiv verpacken. Per Voreinstellung versucht das Kommando das Archiv auf einen Streamer zu schreiben (/dev/tape oder /dev/rmt0), ist kein solcher im System installiert, schreibt **tar** das Ergebnis auf die Standardausgabe. Um das Archiv in einer Datei zu sichern, ist die Option **-f Datei** zu wählen.

Eine angenehme Eigenschaft ist der Umgang mit »Multivolumes« (Option **-M**). Speichert man z.B. das Archiv auf eine Diskette und reicht deren Speicherkapazität nicht aus, so fordert **tar** automatisch zum Wechsel des Mediums auf:

```
user@sonne> tar -Mcf /dev/fd0h1440 ~/Books/
tar: Removing leading `/' from absolute path names in the archive
Prepare volume #2 for /dev/fd0h1440 and hit return:
Prepare volume #3 for /dev/fd0h1440 and hit return:
Prepare volume #4 for /dev/fd0h1440 and hit return:
```

Im Beispiel schreibt **tar** die Daten auf eine (unformatierte) Diskette, weswegen dem Kommando genau mitgeteilt werden muss, wie groß die Speicherkapazität dieser ist (es ist das entsprechende Device anzugeben). Als erstes entfernt **tar** in allen Pfadnamen den führenden Slash (**-P** forciert die Verwendung absoluter Pfadnamen), so dass das Archiv bei einem späteren Entpacken an beliebiger Stelle im Dateisystem extrahiert werden kann - bei Beibehaltung der Verzeichnisstruktur. Nachfolgend wird der Benutzer zum Wechsel der Disketten aufgefordert. Alternativ zu »-M« kann mit **-L Anzahl** der Medienwechsel nach Archivierung von **Anzahl** Bytes erzwungen werden.

Es ist allerdings furchtbar uneffizient, das Archiv unkomprimiert abzulegen. **tar** kann zum Glück mit mehreren Packern zusammen arbeiten. Mit **-z** wird das Archiv mit gzip komprimiert; **-Z** nutzt das (veraltete) Werkzeug compress und **-j** (alt: -I) den derzeit effektivsten Packalgorithmus des Kommandos bzip2. Wurde ein Archiv einmal gepackt, ist bei allen weiteren Operationen die Option für den jeweiligen Packer anzugeben! Das Packen funktioniert allerdings nicht bei Multilevel-Archiven. Ein Ausweg wäre das »vorab«-Komprimieren einer jeden Datei (z.B. mittels eines Skripts) mit anschließender Archivierung.

Inkrementelles Backup

tar unterstützt nur eine rudimentäre Variante des inkrementellen Backups, in dem Daten zur Archivierung ausgewählt werden können, die »neuer« als ein anzugebendes Datum sind. Das Datum (**-n Datum**) ist hierbei im Format »Jahr-Monat-Tag« anzugeben. Um z.B. alle Dateien zu erfassen, die in den letzten 7 Tagen (ab aktuellem

Zeitpunkt) modifiziert wurden, kann folgender Kommandoaufruf verwendet werden:

```
user@sonne> tar -n $(date -d "now 7 days ago" +%Y-%b-%d) -czf /tmp/backup.tgz ~/Books/
```

Im Beispiel wurde bewusst die etwas verwirrende Kalkulation des Datums mit Hilfe von date gewählt, da dessen Einsatz das Schreiben eines Skripts zum automatischen Erzeugen von inkrementellen Backups vereinfacht.

Überprüfung des Archives

Um sich von einer ordnungsgemäßen Archivierung zu überzeugen, sollte das resultierende Archiv einem Test unterzogen werden. Da der Slash automatisch entfernt wurde, ist zuvor ins Wurzelverzeichnis zu wechseln oder das Arbeitsverzeichnis mit **-C Pfad** zu ändern und (in Bezug auf obiges Beispiel) Folgendes aufzurufen:

```
user@sonne> tar -C/ -d -f /dev/fd0h1440
```

tar überzeugt sich nun, dass alle Dateien des Archivs auch im Dateisystem existieren. Sobald eine Unstimmigkeit festgestellt wird, wird das Kommando den Fehler ausgeben:

```
# Um einen Fehler zu provozieren, wurde eine Datei verschoben
user@sonne> tar -df /dev/fd0h1440
home/user/Books/linuxfibel.pdf: File does not exist
```

Recovery

Zu einem Backup gehört natürlich auch eine Möglichkeit, dieses wieder zurückzuspielen. Hier gelangt die Option **-x** (extract) zum Einsatz. Im Falle von relativen Pfadangaben im Archiv wird die Verzeichnisstruktur unterhalb des aktuellen Verzeichnisses erzeugt. Sie sollten also entweder zuvor ins Zielverzeichnis wechseln, oder dieses mit **-C Pfad** explizit angeben:

```
user@sonne> tar -C/ -x -f /dev/fd0h1440
```

Handelt es sich um ein auf mehrere Medien verteiltes Archiv, muss die Option **-M** angegeben werden. **tar** fordert dann zum Medienwechsel auf.

Möchten Sie nur einzelne Dateien extrahieren, sind deren Namen mit vollständiger Pfadangabe auf der Kommandozeile anzugeben. Hier hilft eventuell die Option **-t**, um den korrekten Namen, so wie er im Archiv gespeichert ist, zu erfahren:

```
user@sonne> tar tf /dev/fd0h1440
home/user/Books/access.htm
home/user/Books/index.htm
home/user/Books/stuff.htm
home/user/Books/test.htm
...
user@sonne> tar -C/ -x -f /dev/fd0h1440 home/user/Books/index.htm
```

Grafische Verpackung: kdat

Mit dem Programm **kdat** steht den KDE-Benutzern eine grafische Oberfläche zur Backupverwaltung mit **tar** zur Verfügung. Leider wird als Backupmedium derzeit nur das Tape unterstützt.

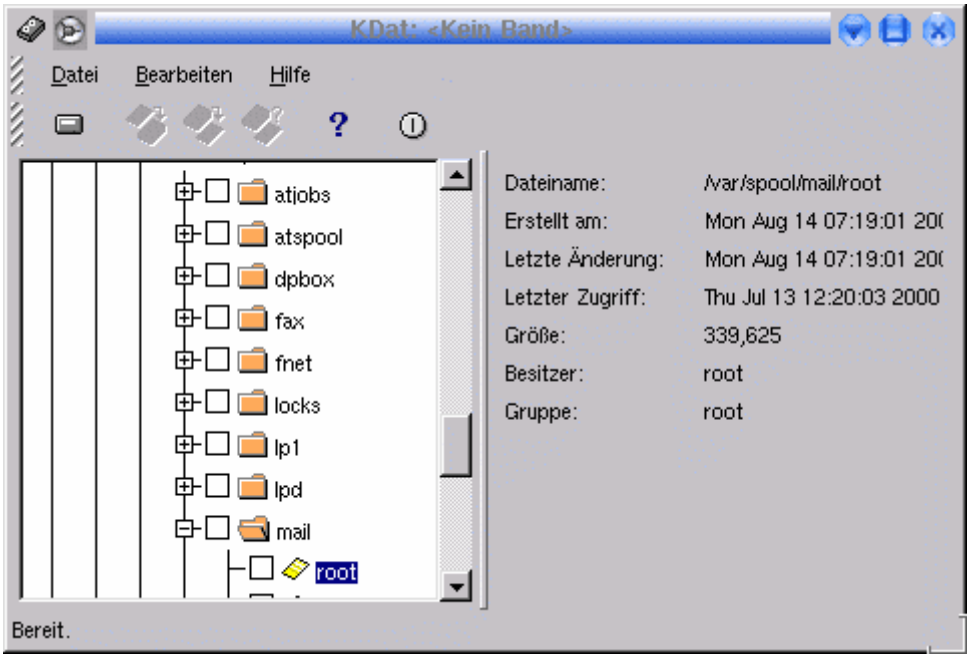


Abbildung 1: Backups mit kdat

Nach dem Start von **kdat** sollte unter dem Menüpunkt »Bearbeiten➔Einstellungen« der Typ des Tapes (Speicherkapazität, Device, Blockgröße) angepasst werden.

Anschließend muss das Tape gemountet werden, wobei **kdat** testet, ob das Band formatiert ist und ggf. zu einer Formatierung auffordert (Vorsicht: eventueller Datenverlust). Das vorgeschlagene »Label« kann übernommen werden, es ist ein eindeutiger Name für das Backup.

Im Hauptfenster ist der Verzeichnisbaum u.a. des Rechners dargestellt. Jedes zu sicherne Verzeichnis bzw. jede Datei ist durch Klick mit der rechten Maustaste zu selektieren. Ein Verzeichnis schließt dabei die enthaltenen Dateien/Verzeichnisse ein, wobei diese wiederum von einer Sicherung ausgeschlossen werden können. Ein markierter Eintrag ist hervorgehoben dargestellt. Über das Sichern-Symbol (oder »Datei➔Sichern«) werden alle selektierten Dateien und Verzeichnisse auf das Tape geschrieben.

Um nicht bei jedem Aufruf die Dateien einzeln markieren zu müssen, können beliebig viele Sicherungsprofile angelegt werden. Jedes Profil enthält eine Liste der zu sichernden Daten.

Backup mit cpio

cpio (*cpio in/out*) besitzt einen ähnlichen Funktionsumfang wie das zuvor beschriebene Kommando **tar**, so dass wohl eher die Vorliebe des Benutzers das eine oder andere Werkzeug favorisiert. Ein Vorteil wäre dennoch zu nennen: **cpio** verzweifelt nicht an beschädigten Archiven und vermag zumindest den unversehrten Teil komplett wieder herzustellen.

Aufruf: cpio MODUS [OPTIONEN]

cpio kennt drei Betriebsarten:

- **copy in** zum Extrahieren von Dateien aus einem Archiv: Option **-i**
- **copy out** zum Erzeugen eines Archivs: Option **-o**
- **pass througth** als Kombination aus den obigen Modi erlaubt das Kopieren ganzer Verzeichnisbäume: Option **-p**

Volles und inkrementelles Backup

Um Dateien in einem cpio-Archiv zu speichern, müssen deren Namen dem Kommando mitgeteilt werden. **cpio**

erwartet allerdings, dass jeder Dateiname auf einer neuen Zeile erscheint. Deshalb füttert man das Kommando am einfachsten über eine Pipe:

```
user@sonne> ls *.txt | cpio -o > /dev/fd0
11 blocks
```

Das Beispiel sichert alle auf ".txt" endenden Dateien des aktuellen Verzeichnisses auf die Diskette. Die Ausgabeumleitung ist wichtig, da sonst die Dateiinhalte auf dem Bildschirm landen würden.

Hiermit kennen Sie nun das Prinzip des Backups mit **cpio**; zur Implementierung eines inkrementellen Backups bietet sich die Nutzung der Möglichkeiten des Kommandos find an, mit dem nach den verschiedensten Kriterien nach Dateien gefahndet werden kann. So archiviert der Aufruf von:

```
user@sonne> find /etc -maxdepth 2 -depth | cpio -o > /dev/fd0
319 blocks
```

alle Dateien aus dem Verzeichnis /etc, wobei alle Unterverzeichnisse, nicht aber die darin befindlichen Unterverzeichnisse berücksichtigt werden. Die Option »-depth« von **find** bewirkt, dass der Name eines Verzeichnisses selbst erst nach den enthaltenden Daten ausgegeben wird. Bei eventuellem Zurückschreiben der Daten, wird nun der häufige Fehler verhindert, dass ein Verzeichnis mit »falschen« Zugriffsrechten erzeugt wurde und anschließend das Übertragen der Daten in dieses scheitert. Durch die hier gewählte Reihenfolge wird mit der ersten Datei aus dem Verzeichnis dieses mit den korrekten Rechten gesetzt (mit Rechten, die das Schreiben der Datei ermöglichen).

Im Sinne eines inkrementellen Backups ist aber sicher die Suche nach Dateien, die innerhalb einer bestimmten Zeitspanne geändert wurden, interessant. Der nächste Aufruf findet alle Dateien des Verzeichnisses /etc, deren Modifikationszeit nicht älter als 5 Tage ist:

```
root@sonne> find /etc -mtime -5 -maxdepth 2 -depth | cpio -o > /dev/fd0
112 blocks
```

Überprüfung des Archives

Um sich eine Liste der Dateien eines Archives zu betrachten, ruft man das Kommando im »copy in«-Modus auf:

```
user@sonne> cpio -itvI /dev/fd0
-rw-r--r--   1 root     root          270 Aug 15 07:07 /etc/mtab
-rw-r--r--   1 root     root         2510 Aug 10 13:49 /etc/logfiles
-rw-r--r--   1 root     root        37911 Aug 15 07:06 /etc/ld.so.cache
-rw-r--r--   1 root     root          271 Aug 10 15:27 /etc/hosts.allow
-rw-r--r--   1 root     root         5762 Aug 10 15:27 /etc/inetd.conf
-rw-r--r--   1 root     root         3349 Aug 14 12:18 /etc/printcap
-rw-------   1 root     root           60 Aug 15 07:06 /etc/ioctl.save
-rw-r--r--   1 root     root         3799 Aug 15 09:10 /etc/XF86Config
-rw-------   1 root     root          512 Aug 15 07:07 /etc/ssh_random_seed
-rw-r--r--   1 root     root         3309 Aug 14 12:18 /etc/printcap.old
-rw-rw----   1 lp       lp           1244 Aug 14 12:05 /etc/apsfilterrc.ljet4
-rw-rw----   1 lp       lp           1261 Aug 14 12:37 /etc/apsfilterrc.lj4dith
-rw-r--r--   1 root     root           88 Aug 15 09:38 /etc/dumpdates
drwxr-xr-x  30 root     root            0 Aug 15 08:28 /etc
112 blocks
```

Die Option **-t** bewirkt die Anzeige des Archivinhalts und verhindert das Entpacken. **-v** bringt die Rechte der Dateien zum Vorschein und **-I Archiv** lässt **cpio** die Daten aus dem Archiv anstatt von der Standardeingabe lesen.

Ein Vergleich der Dateien im Archiv mit den Dateien im Verzeichnisbaum ist nicht direkt möglich. Hier kann nur ein Entpacken der Daten in ein separates Verzeichnis mit anschließendem Dateivergleich (z.B. mit diff) helfen.

Recovery

Schließlich möchte man die Daten auch wieder zurückschreiben können. Hierzu ist **cpio** im »copy-in«-Modus wie folgt zu starten:

```
root@sonne> cpio -id / etc/ mtab < / dev/ fd0
cpio: /etc/mtab not created: newer or same age version exists
112 blocks
```

Im Beispiel stellt **cpio** nur fest, dass die gewünschte Datei nicht älter ist als die im Archiv enthaltene Version. Also tut das Kommando nichts. **-d** erzwingt das Anlegen von Verzeichnissen, falls diese nicht schon existieren, die nachfolgende Angabe von zu rekonstruierenden Dateien kann entfallen, dann werden alle Dateien aus dem Archiv extrahiert.

Backup mit dump und restore

tar und **cpio** enthalten wenige Eigenschaften, die man sich von einem guten Backup-Werkzeug wünscht. Zwar sind beide effektiv zum Erzeugen voller Backups geeignet, jedoch besitzen sie von Haus aus keinerlei Unterstützung inkrementeller Datensicherungsstrategien. Allerdings passen sie mit ihrer Philosophie ideal in das Kommandokonzept von Unix, wo jedes Standardkommando einen »sinnvollen« Funktionsumfang mit sich bringen sollte und alle komplexeren Aufgaben durch Kombination der Grundbausteine realisiert werden. Ein Backup wird auch heute häufig auf Basis von Shellskripten und unter Verwendung der bereits vorgestellten Kommandos implementiert.

dump hingegen verdient die Bezeichnung eines Backup-Werkzeuges. Es vermag sowohl mit vollen als auch mit inkrementellen Backups umzugehen, wobei letztere in bis zu 9 Abstufungen vorgenommen werden können.

Multilevel-Backups

Ein Multilevel-Backup bezeichnet eine Strategie, wobei ein volles Backup mit inkrementellen Backups kombiniert wird. Im einfachsten Fall wird einmal in einem Monat ein volles Backup des Datenbestandes vorgenommen und anschließend werden täglich nur die modifizierten Dateien gesichert. Dem vollen Backup wird hierbei das Level 0 zugeordnet und die weiteren Sicherungen erhalten das Level 1. Nach Ablauf des Monats wiederholt sich der Vorgang.

Eine Verallgemeinerung führt nun beliebig viele Level ein, wobei bei einem Backup des Levels *x* alle geänderten Daten im Zeitraum seit der letzten Sicherung desselben Levels berücksichtigt werden.

Der **Sinn solcher Level** ist die Einsparung von Speichermedien bei gleichzeitiger Verlängerung der Backup-Historie. Möchten Sie bspw. die Daten über den Zeitraum dreier Monate aufbewahren, so benötigen Sie bei Realisierung mittels zweier Level ca. 92 Bänder (1 Band für das volle Backup und für jeden Tag ein weiteres). Ein anderes Vorgehen wäre ein anfängliches volles Backup (Level 0, 1 Band), jeden Monat ein Backup Level 1 (2 Bänder, der 3. Monat wird vom kommenden vollen Backup erfasst), jede Woche ein Backup Level 2 (4 Bänder) und schließlich ein viertes Level zur Speicherung der täglichen Daten (6 Bänder). Mit 13 Bänder können Sie also jederzeit den Datenbestand auf den Stand des letzten Tages bringen.

Die Verwendung von dump

Aufruf: dump [OPTIONEN] [DATEISYSTEM]

dump vermag derzeit nur mit dem Datensystem ext2 zusammen zu arbeiten. Ein zu sicherndes Dateisystem kann in der Datei /etc/fstab markiert werden, so dass bei einem Aufruf von **dump** über den gesamten Verzeichnisbaum nur diese Dateisysteme berücksichtigt werden.

dump versucht in der Voreinstellung das Gerät »/dev/st0« zu öffnen. Existiert das Device nicht, fordert das Kommando zur Eingabe eines anderen Ausgabegerätes auf. Mit der Option **-f Datei** kann die Sicherung in eine beliebige Datei umgelenkt werden.

Dem Kommando ist natürlich mitzuteilen, was zu sichern ist. Hier kann entweder ein Verzeichnis oder das Device

angegeben werden. Des Weiteren muss das Backup-Level (0-9) benannt werden. **dump** selbst ist in der Lage, die Zeiten der letzten Sicherung eines Levels zu notieren und anhand derer zu entscheiden, ob ein erneutes Backup dieses Levels überhaupt notwendig ist. Hierzu schreibt das Kommando, wird es mit der Option **-u** aufgerufen, die Zeiten in die Datei **/ etc/ dumpdates**. Existiert keine Datei, sollte zuvor eine leere von Hand erzeugt werden.

Volles Backup

Um nun ein volles Backup des Verzeichnisses »/etc/rc.d« auf Diskette zu sichern, ist Folgendes einzugeben:

```
root@sonne> dump -0 -u -f / dev/ fd0 / etc/ rc.d/
  DUMP: Date of this level 0 dump: Tue Aug 15 09:38:13 2000
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/hda5 (/) to /dev/fd0
  DUMP: Label: none
  DUMP: mapping (Pass I) [regular files]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 652 tape blocks on 0.02 tape(s).
  DUMP: Volume 1 started at: Tue Aug 15 09:38:13 2000
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular files]
  DUMP: DUMP: 725 tape blocks on 1 volumes(s)
  DUMP: finished in less than a second
  DUMP: Volume 1 completed at: Tue Aug 15 09:38:13 2000
  DUMP: level 0 dump on Tue Aug 15 09:38:13 2000
  DUMP: DUMP: Date of this level 0 dump: Tue Aug 15 09:38:13 2000
  DUMP: DUMP: Date this dump completed: Tue Aug 15 09:38:13 2000
  DUMP: DUMP: Average transfer rate: 0 KB/s
  DUMP: Closing /dev/fd0
  DUMP: DUMP IS DONE
```

dump fordert selbständig zum Mediumwechsel auf, wenn dessen Speicherkapazität nicht mehr genügt.

dump hat in obigen Beispiel die letzte Sicherung des zugrunde liegenden Dateisystems in der Datei »/etc/dumpdates« vermerkt. Die Informationen sind dort im Klartext enthalten und können somit editiert werden, um z.B. temporär eine andere Backupstrategie zu wählen. Mit der Option **-W** zeigt **dump** die gesicherten Dateisysteme mit Datum und letztem Backuplevel an und gibt zusätzlich noch Empfehlungen, welches Dateisystem eine erneute Sicherung vertragen könnte:

```
root@sonne> dump -W
Last dump(s) done (Dump '>' file systems):
  /dev/hda5     (     /) Last dump: Level 1, Date Tue Aug 15 13:38
```

Inkrementelles Backup

Um nun eine Level-2-Archivierung desselben Dateisystems vorzunehmen, gibt der Administrator folgende Zeile ein:

```
root@sonne> dump -2 -u -f / dev/ fd0 / etc/ rc.d/
```

Die eigentliche Aufgabe beim Backup ist es nun, die verschiedenen Level-Sicherungen in sinnvollen Zeiträumen anzuordnen, z.B.

- Level-0-Backup einmal in 3 Monaten
- Level-1-Backup monatlich
- Level-2-Backup wöchentlich
- Level-3-Backup täglich

Es bietet sich an, ein solches Schema per cron-Job automatisch zu realisieren. Die Aufgabe des Administrators reduziert sich damit auf das rechtzeitige Wechseln der Bänder.

Überprüfung des Archives mit restore

Um einen Vergleich der im Archiv vorhandenen mit den installierten Dateien vorzunehmen, dient das Kommando **restore** in Verbindung mit der Option **-C**. Wird kein weiteres Argument angegeben, versucht **restore** das Archiv von /dev/st0 zu lesen. Wir verweisen es deshalb auf eine andere Datei (Option **-f Datei**):

```
root@sonne> restore -C -f / dev/ fd0
Dump   date: Tue Aug 15 15:38:38 2000
Dumped from: the epoch
Level 0 dump of / on sonne:/dev/hda5 (dir /etc/rc.d)
Label: none
filesys = /
./lost+found: (inode 11) not found on tape
./usr: (inode 2049) not found on tape
...
```

Die angeblich fehlenden Dateien sind auf einen Link im archivierten Verzeichnis zurückzuführen und können ignoriert werden.

Recovery mit restore

Zum Recovery der Daten wird im einfachsten Fall ein Archiv mittels **recover** und der Option **-r** zurückgeschrieben. Beachten Sie, falls Sie alle Daten durch ihre Kopie aus den Archiven ersetzen wollen, alle Archive in der Reihenfolge ihrer Aufzeichnung einzuspielen, d.h. Sie spielen zunächst das volle Backup (Level 0) ein, dann alle Backups des Levels 1 usw.

```
root@sonne> restore -r -f / dev/ fd0
```

Eine Besonderheit von **recover** ist der interaktive Modus (Option **-i**), der die Selektion einzelner Dateien aus einem Archiv ermöglicht.

```
root@sonne> restore -i -f / dev/ fd0
Verify tape and initialize maps
Tape block size is 32
Dump   date: Tue Aug 15 15:38:38 2000
Dumped from: the epoch
Level 0 dump of / on dev17:/dev/hda5 (dir tmp)
Label: none
Extract directories from tape
Initialize symbol table.
restore > help
Available commands are:
        ls [arg] - list directory
        cd arg - change directory
        pwd - print current directory
        add [arg] - add `arg' to list of files to be extracted
        delete [arg] - delete `arg' from list of files to be extracted
        extract - extract requested files
        setmodes - set modes of requested directories
        quit - immediately exit program
        what - list dump header information
        verbose - toggle verbose flag (useful with ``ls'')
        help or `?' - print this list
If no `arg' is supplied, the current directory is used
restore > ls
.:
   2 ./      2 ../   6145 tmp/

restore > verbose
verbose mode off
restore > ls
.:
```

```
tmp/

restore > quit
```

Die Kommandos **ls**, **cd** und **pwd** besitzen gleiche Bedeutung wie in der Shell, wobei als Argument für **ls** nur Dateinamen mit enthaltenen Metazeichen jedoch keine Optionen zulässig sind.

Um nun eine Liste der zu extrahierenden Dateien zu erstellen, werden Dateien mit **add** hinzugefügt und mittels **delete** entfernt. In der Ausgabe von **ls** erscheint eine Markierung vor den in der Liste enthaltenen Dateien.

Mit **extract** werden schließlich die markierten Dateien aus dem Archiv ins System eingespielt, wobei **recover** zunächst zur Eingabe der Volumenummer auffordert (also das Band, auf dem sich die Datei befindet). Ein Beispieldialog könnte wie folgt aussehen:

```
restore > add boot*
restore > ls boot*
*bootsec.lin
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
Mount tape volume 1
Enter ``none'' if there are no more tapes
otherwise enter tape name (default: /dev/fd0)
resync restore, skipped 64 blocks
set owner/mode for '.'? [yn] n
restore >
```

Backup mit Taper

Von den vorgestellten Backup-Werkzeugen besitzt **Taper** die freundlichste Benutzerschnittstelle. Es unterstützt die volle Palette an Backupmedien (Tapes, Festplatte, SCSI, Floppy). Wesentlich ist die zusätzliche Speicherung des Inhatlsverzeichnisses eines Archivs, das die Suche vor allem in großen Datenbeständen enorm beschleunigt.

Start von Taper

Der erste Aufruf erfordert die Angabe des Backup-Mediums. **Taper** speichert die Angaben, so dass bei folgenden Anwendungen der Kommandoaufruf ohne Argumente genügt:

```
# Verwendung einer Diskette
root@sonne> taper -T removable -b / dev/ fd0

# Verwendung eines Tapes
root@sonne> taper -T tape -b / dev/ tape

# Verwendung einer Partition einer SCSI-Platte
root@sonne> taper -T scsi -b / dev/ sdb3

# Verwendung einer Datei
root@sonne> taper -T file -b / home/ backup/ my_backup
```

Nach dem Start befinden wir uns im Hauptmenü.

Backup mit Taper

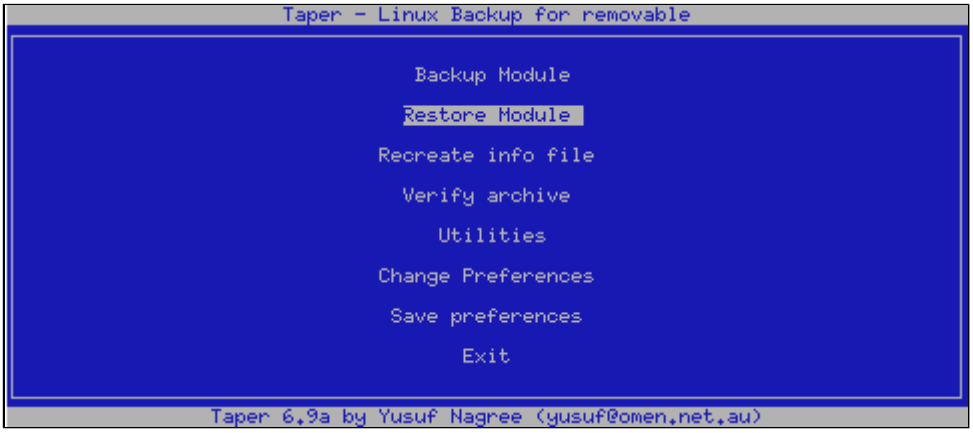


Abbildung 2: Taper Startmenü

Backup-Module: Zunächst sind der Titel des Backups und des Volumes anzugeben bzw. die vorgeschlagenen Werte eines vorangegangenen Taperlaufes zu bestätigen. Bei einem existierenden Backuparchiv besteht die Wahl, diesem Daten hinzuzuführen oder das bestehende Archiv zu überschreiben.

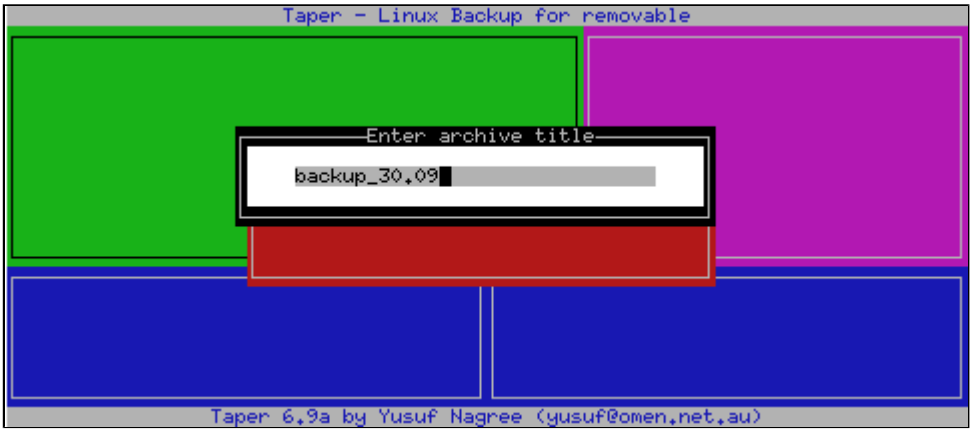


Abbildung 3: Anlegen eines neuen Backuparchivs

Das folgende Fenster ist in vier Bereiche aufgeteilt, zwischen denen mit **[Tab]** gewechselt werden kann. **Links oben** befindet sich die Verzeichnisansicht. Mit den Cursortasten kann die Selektion verändert werden. Ein **[Enter]** auf einem Verzeichniseintrag wechselt in dieses.

Um einen selektierten Eintrag zum Archiv hinzuzufügen, ist **[i]** zu drücken. Handelt es sich um ein Verzeichnis, werden alle enthaltenen Dateien/Unterverzeichnisse automatisch zum Archiv hinzugefügt. Um einzelne Dateien/Verzeichnisse vom Archiv zu verbannen, sind diese zu selektieren und **[u]** einzugeben. Bei indirekt zum Backup markierten Einträgen (die durch Auswahl des übergeordneten Verzeichnisses selektiert wurden) ist zum Ausschluss dieser **[e]** (anstatt **[u]**) zu einzugeben.

Nähere Informationen zu einer selektierten Datei / einem Verzeichnis erhält man mittels **[d]**.

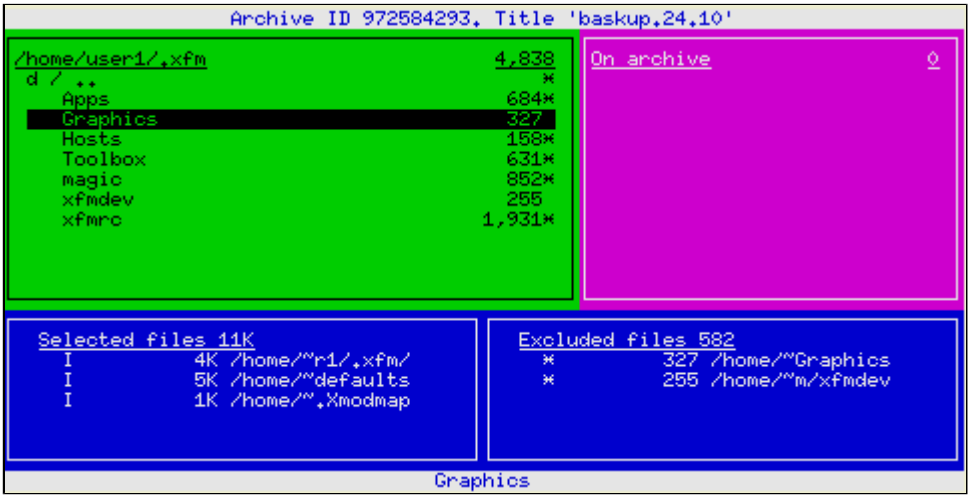


Abbildung 4: Selektion der zu sichernden Daten

Im Fenster **rechts oben** wird der aktuelle Inhalt des Archivs (Dateien, die zuvor archiviert wurden) angezeigt. **Links unten** stehen die zum Backup selektierten und **rechts unten** die vom Backup ausgeschlossenen Dateien. Um solch eine explizit ausgeschlossenen Datei wieder für die Archivierung vorzusehen, ist in das Fenster zu wechseln (**[Tab]**) und die Datei aus der Auswahl zu entfernen (**[u]**). Durch Betätigen von **[h]**, **[?]** oder einer beliebigen, nicht mit einer Funktion versehenen Taste erhält man eine Hilfe zu Taper:

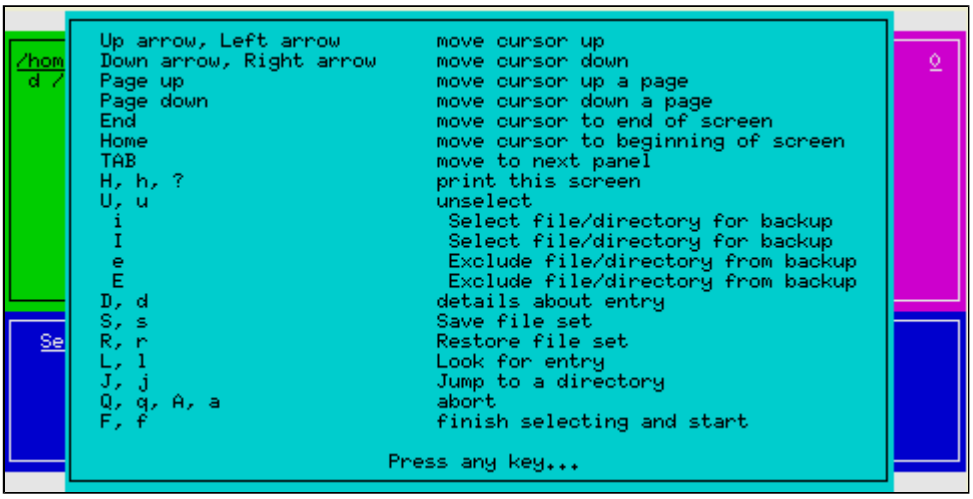


Abbildung 5: Kurzhilfe zu Taper

Das Erzeugen des Backups wird mit Eingabe von **[f]** begonnen. Ist die Kapazität eines Backupmediums erschöpft, fordert Taper zum Wechsel dessen auf. Nach Abschluss des Vorgangs erscheint ein Statusbericht auf dem Bildschirm:

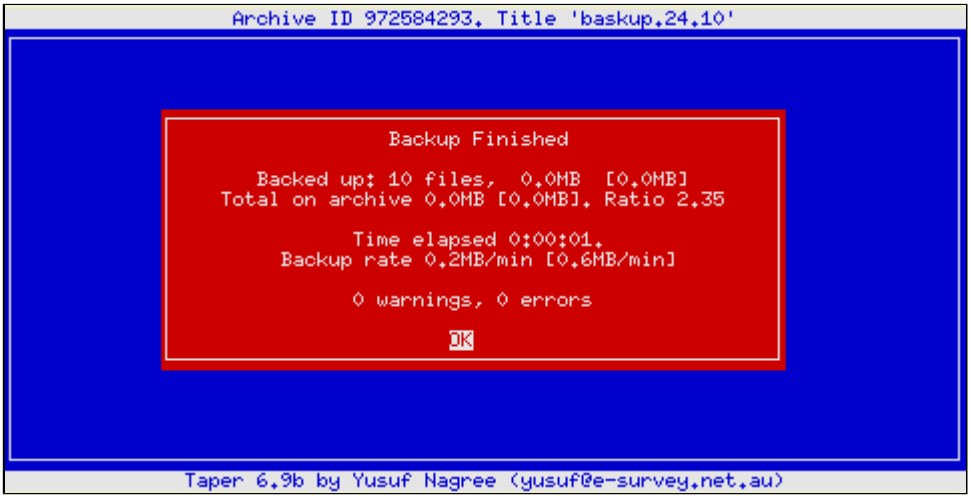


Abbildung 6: Statusbericht zu einem Backup

Alle getroffenen Änderungen können durch Eingabe von **[q]** verworfen werden.

Backup-Modi

Der default-Modus von Taper ist ein inkrementelles Backup. D.h. Taper schaut vor dem Archivieren der selektierten Dateien nach, ob diese im verwendeten Archiv bereits existiert. Die dortige Datei wird nur ersetzt, falls sie älteren Datums ist, als der selektierte Eintrag. Ebenso werden Dateien, die 0 Bytes enthalten, ignoriert.

Beim vollen Backup hingegen, wird eine Datei prinzipiell ersetzt, unabhängig davon, ob die Version im Archiv ggf. gleichen oder gar neuerem Datums ist oder ob ihr Inhalt leer ist.

Das voreingestellte Backup-Verhalten kann entweder per Kommandozeilenoption (inkrementelles Backup mittels **--incremental-on (+u)**; volles Backup mittels **--incremental-off (-u)**) oder in den Preferences (Change preferences ➜ Backup preferences 2) eingestellt werden.

```
# Start mit erzwungenem vollen Backup
root@sonne> taper -T removable -b / dev/ fd0 --incremental-off
```

Die Dateien lassen sich ebenso **gepackt** im Archiv ablegen. Taper kennt hierzu drei Typen von Kompressionen. **Typ 1** verwendet das Unix-Kommando **gzip** (wer Taper aus dem Quellen selbst übersetzt, kann in »defaults.h« auch ein anderes Kompressionsprogramm eintragen). **Typ 2** ist ein Taper-eigenes compress-ähnliches Verfahren und **Typ 3** ist »gzip« nachempfunden. Die internen Verfahren arbeiten schneller, liefern aber i.A. schlechtere Packraten. Der Typ der Komprimierung kann wiederum in den Preferences eingestellt (Change preferences ➜ Backup preferences 1) oder per Kommandozeile (**--compress-typ** ***Typnummer*** bzw. **-c** ***Typnummer***) mitgegeben werden.

```
# Archivierung ohne Komprimierung
root@sonne> taper -T removable -b / dev/ fd0 --compress-type 0
```

Restore mit Taper

Im Hauptmenü ist der Eintrag **Restore Module** zu wählen.

Taper präsentiert eine Liste aller Archive. Das zu rekonstruierende ist zu selektieren und die Wahl mit **[Enter]** zu bestätigen. Taper fragt nun nach dem Verzeichnis, in das die Daten aus dem Backup eingespielt werden sollen. Ist das Ziel dasselbe Verzeichnis, aus dem die Daten des Backups gewonnen wurden, kann auf die Eingabe verzichtet werden.

Das Restore-Fenster des Programmes ist wiederum in 4 Bereiche unterteilt. **Links oben** sind alle Dateien aus dem Archiv dargestellt. **Rechts oben** wird das komplette Archiv dargestellt, also einschließlich der vom Backup

ausgeschlossenen Daten (in runden Klammern). **Links unten** stehen alle zur Wiederherstellung vorgesehenen Dateien; das rechte untere Fenster bleibt leer.

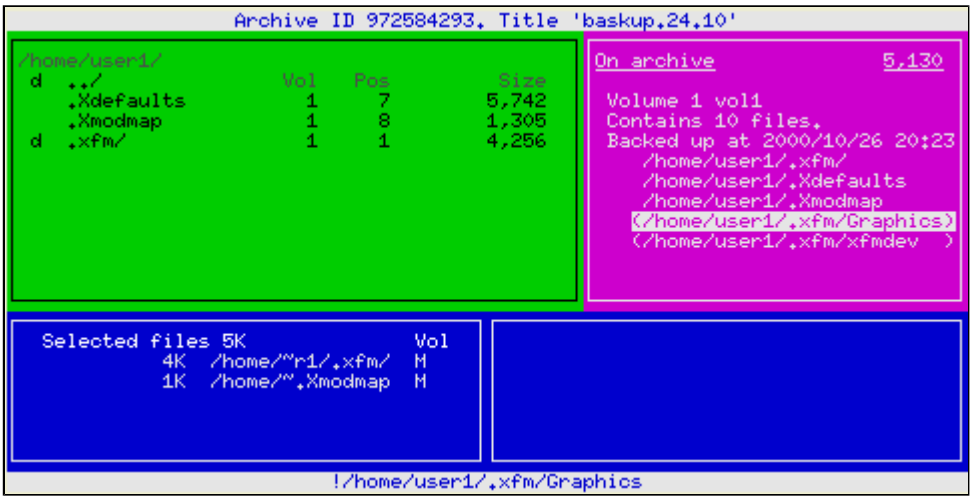


Abbildung 7: Restore mit Taper

Die Auswahl der Dateien erfolgt analog zum Backup-Modul: Mit **[i]** wählen Sie eine Datei/Verzeichnis aus (ein Verzeichnis schließt wiederum die enthaltenen Dateien/Verzeichnisse ein); ein **[u]** verwirft einen zuvor selektierten Eintrag. Mit **[f]** wird die Wiederherstellung begonnen bzw. mittels **[q]** verworfen.

Low Level Backup mit dd

Was tun, wenn das Dateisystem zerstört ist, ein aktuelles Backup aber nicht zur Verfügung steht? Bevor die Flinte ist Korn geworfen wird, kann ein Reparaturversuch des Dateisystems nicht schaden. Doch Vorsicht! Wir bewegen uns auf einem sehr unsicherem Terrain, eine unbeholfene Aktion und all die Daten sind unwiderruflich pfutsch...

Da sollte man sich doch besser zuvor ein Backup vom zerstörten System erstellen, um im Falle eines Falles weitere Versuche unternehmen zu können. Aber welches der Backup-Kommandos ist schon in der Lage, zerstörte Dateien aufzuzeichen? Keines, hier kommt **dd** zum Einsatz, das einfach ab einem bestimmten Startpunkt eine bestimmte Menge »roher« Daten liest und diese 1:1 in eine Zieldatei oder auf ein Zielgerät überträgt. Diese Eigenschaft erlaubt es auch, Dateien beliebiger Dateisysteme zu sichern, selbst wenn Linux diese gar nicht lesen kann...

Aufruf: dd if=QUELLE of=ZIEL [OPTIONEN]

QUELLE und ZIEL können hierbei sowohl ein Device als auch eine Datei sein. Werden keine weiteren Optionen angegeben, so werden Daten im Umfang von QUELLE kopiert. Handelt es sich bei QUELLE um eine Partition, wird deren gesamter Inhalt kopiert:

```
root@sonne> dd if=/dev/hda of=/dev/hdc
```

Im Beispiel wird die gesamte erste IDE-Festplatte (/dev/hda) des Systems auf die dritte (/dev/hdc) kopiert. Es sollte jedem bewusst sein, dass der alte Inhalt der dritten Festplatte damit überschrieben wird. Auch sollte diese über die gleiche Kapazität wie die erste Platte verfügen (sonst muss man sich die Anzahl der kopierten Bytes merken). Um die Daten später zurückzuspielen, vertauscht man die Angaben von QUELLE und ZIEL.

Ist das Ziel einer Kopieraktion eine Datei, könnte bei Kernel-Versionen < 2.4 die Beschränkung der Dateigröße von 2 GB unser Vorhaben zunichte machen, in einem solchen Fall muss die QUELLE auf mehrere Zieldateien aufgeteilt werden. Hierzu benötigt **dd** mehrere Optionen. Mit **bs=BYTES** muss die Anzahl Bytes, die in einem Schritt zu lesen oder schreiben sind, angegeben werden. Wieviele Schritte getätigt werden sollen, legt die Option **count=ANZAHL** fest. Um bspw. den Masterbootsektor (Mbr) der ersten Festplatte in eine Datei zu schreiben, könnte man folgenden Aufruf verwenden:

```
root@sonne> dd if=/dev/hda of=/tmp/mbr.save bs=512 count=1
```

Um jetzt den Superblock (1 k groß) der ersten Partition zu sichern, müssen sowohl Mbr als auch der 512 Bytes lange Bootsektor (also zwei Blöcke) übersprungen werden. Hierzu verwendet man die Option **skip= ANZAHL**.

```
root@sonne> dd if=/dev/hda of=/dev/superblock.save bs=512 count=2 skip=2
```

Das Beispiel demonstriert zwei Sachen: Zum einen, wie ein Abbild auf mehrere Dateien verteilt werden kann und zum anderen, dass die Sache verdammt kompliziert ist. Deswegen wird man **dd** niemals als Werkzeug für ein regelmäßiges Backup einsetzen, sondern nur in außergewöhnlichen Situationen, wie eingangs beschrieben.

Amanda

AMANDA (Advanced Maryland Automatic Network Disk Archiver) ist ein auf den Kommandos dump bzw. tar basierendes Backup-System, das die Datensicherung in einem Netzwerk unterstützt. Hierzu wird ein Rechner als Backup-Server eingerichtet, der neben verschiedenen Unix-Clients über den Samba-Service auch die Daten eines Windows-Rechner zu verwalten vermag.

Installation

Die Aktualität der Programmversionen sollte bei jeder modernen Distribution gewährleistet sein. Als Programme müssen installiert sein:

- GNU tar
- Samba
- Perl
- GNU readline
- GNU awk
- gnuplot

Bei den meisten Distributionen wird man um die Erzeugung der Programme aus den Quelltexten nicht umhin kommen, deshalb soll das Vorgehen hier kurz skizziert werden (im Basisverzeichnis der Quellen finden Sie die Dateien README und INSTALL mit weitergehenden Informationen).

Im Quellverzeichnis befindet sich das Skript **configure**, das die Steuerdateien für den folgenden Kompilierungsvorgang erzeugt. Einige Optionen für configure sind zwingend erforderlich:

--with-user= ***Amanda_Benutzer***

Amanda läuft unter dieser Benutzerkennung.

--with-group= ***Amanda_Gruppe***

Amanda läuft unter dieser Gruppenkennung.

--prefix= ***Verzeichnis***

Amanda wird unterhalb dieses Basisverzeichnisses installiert (Voreinstellung ist /usr/local).

Unterhalb des Basisverzeichnisses für die Amandainstallation werden die Server-Programme in ein Verzeichnis »sbin«, die Client-Programme unter »libexec«, die dynamischen Bibliotheken unter »lib« usw. eingespielt. Jeder dieser voreingestellten Pfade lässt sich über eine Option separat modifizieren. Rufen Sie **configure --help** auf, um eine vollständige Liste der Optionen zu erhalten.

Im Zusammenhang mit durch Firewalls geschützten Systemen kann die Konfigurationsoption **--portrange=** ***TCP-Port1*****[,*****TCP-Port2*****...]** erforderlich werden, um den Datentransfer auf die genannten TCP-Ports zu beschränken. Eine Firewall darf aber niemals den UDP-Port 10080 sperren, der dem Server zu Anfragen an die Clients dient.

Der nachfolgende Aufruf sollte für zahlreiche Fälle genügen:

```
root@sonne> ./configure --prefix=/usr \
      --libexecdir=/usr/lib/amanda \
      --sysconfdir=/etc \
      --localstatedir=/var/lib \
      --with-user=amanda --with-group=disk
```

Der Übersetzungsvorgang und die anschließende Installation werden durch zwei make-Aufrufe eingeleitet:

```
root@sonne> make; make install
```

Anmerkung: Die nachfolgende Beschreibung geht von obigen Angaben aus, d.h. für alle nicht genannten configure-Optionen gelten die Default-Einstellungen. Dies gilt insbesondere für den Namen der verwendeten Konfiguration (Voreinstellung »**DailySet1**«; überschreibbar durch die Option **--with-config=...**), der serverseitig als Verzeichnisname dient.

Der Backup-Server

Zunächst sollten Sie die **Verzeichnisse anlegen**, die der Server nachfolgend für Status- und temporäre Dateien verwendet. Dies sind im Einzelnen

```
root@sonne> mkdir -p /etc/amanda/DailySet1
root@sonne> mkdir -p /var/amanda/DailySet1
root@sonne> mkdir -p /dump/amanda
root@sonne> mkdir -p /var/lib/amanda/gnutar-lists
root@sonne> chown -R amanda /var/lib/amanda
```

Die Namen der beiden ersten Verzeichnisse korrespondieren mit der Bezeichnung der Konfiguration. »DailySet1« ist die Voreinstellung, falls Sie bei der Kompilierung nichts anderes angegeben haben. Unter /etc landen die Setup-Daten; in /var werden Protokolldateien und Datenbanken abgelegt.

/dump/amanda dient als Zwischenlager. Hier sammelt der Server zunächst die zu archivierenden Daten. Erst wenn sie komplett sind, schreibt er sie aufs Band.

Informationen zur Aktualität der Archive hält der Server in einer Datei **/etc/amandates**, die noch erzeugt werden sollte:

```
root@sonne> touch /etc/amandates
```

Ein Backup macht erst wirklich Sinn, wenn ein solches regelmäßig vorgenommen wird. So ist ein Eintrag in den Terminkalender dringend zu empfehlen:

```
root@sonne> su - amanda
amanda@sonne> crontab -e
0  16 * * 1-5   /usr/sbin/amcheck -m DailySet1
45  0 * * 2-6   /usr/sbin/amdump DailySet1
```

Da es sich bei Amanda um einen Netzwerkdienst handelt, werden einige Einträge in Konfigurationsdateien notwendig. Tragen Sie die folgenden Zeilen in die **/etc/services** ein:

```
root@sonne> grep amanda /etc/services
amanda          10080/udp
amandaidx        10082/tcp
amidxtape        10083/tcp
```

Beachten Sie, dass der UDP-Port zwingend 10080 sein muss. Die TCP-Ports können ggf. angepasst werden.

amandaidx und **amidxtape** sind Dienste, die den Zugriff auf die Kataloge und das Backup-Bandlaufwerk ermöglichen. **amanda** ist der Backup-Client. Da sicher auch auf dem Server eine Datensicherung erwünscht ist,

sollte dieser Eintrag nicht fehlen. Damit sie bei Bedarf aktiviert werden, sind Einträge in der Datei /etc/inetd.conf (bei Verwendung von inetd) oder in /etc/xinetd.conf (xinetd) erforderlich:

```
root@sonne> grep -i amanda /etc/inetd.conf
# Amanda Backup-Client-Service
amanda      dgram   udp wait    amanda /usr/lib/amanda/amandad amandad
# Amanda Katalog-Service
amandaidx   stream  tcp nowait  amanda /usr/lib/amanda/amindexd amindexd
# Amanda Tape-Service
amidxtape   stream  tcp nowait  amanda /usr/lib/amanda/amidxtaped amidxtaped
```

Der Eintrag der fünften Spalte entspricht dabei dem Namen, der mittels »configure --with-user=...« vereinbart wurde. Auch müssten Sie eventuell die Pfade der Programme (Spalte 6) korrigieren, falls Sie bei der Konfiguration andere Einstellungen vorgenommen haben.

Im folgenden Schritt muss die Konfigurationsdatei des Servers angelegt werden. Die den Quellen beiliegende Beispieldatei kann als Ausgangspunkt dienen:

```
root@sonne> cp $AMANDA_SRC_DIR/examples/amanda.conf /etc/amanda/DailySet1
```

Editieren Sie die Datei *DailySet1* und passen folgende Einträge an Ihre Gegebenheiten an:

org

Berichte, die Amanda per Mail versendet, enthalten diesen Text als "Subject".

mailto

Die Empfängeradresse für Berichte.

dumpuser

dump sollte unter derselben Benutzerkennung laufen wie Amanda, deshalb sollte hier der mit "configure --withuser=..." vereinbarte Name stehen.

dumpcycle

Der Zyklus für ein volles Backup. Die Zeit sollte in Konfigurationen mit vielen Clients nicht zu gering gewählt werden; zumindest muss das Backup vollständig auf die Bänder gespielt werden können, bevor ein neuer Zyklus startet. Allerdings bedingt ein langer Backupzyklus entsprechend viele Bänder, da mehr Sicherungsd aufgehoben werden müssen. Typisch sind hier Werte zwischen 1 Woche und 3 Monaten.

runspercycle

Die Anzahl Durchläufe pro Zyklus. Wie viele inkrementelle Backups werden also innerhalb der Zeitspanne ei vollen Backups vorgenommen.

tapecycle

Der Bandzyklus ist die minimale Anzahl Bänder, die Amanda benötigt. Er errechnet sich aus dem Wert »runspercycle« mal »runtapes« und sollte niemals kleiner gewählt werden, da Amanda ansonsten die Bänd des letzten vollen Backups überschreibt. Stellt man Amanda mehr Bänder zur Verfügung, verfügt man über längere Backup-History (über das letzte volle Backup hinaus), da Amanda die Bänder zyklisch beschreibt.

runtapes

Anzahl der Bänder, die für einen Durchlauf maximal verwendet werden. Genügt der Speicherplatz nicht, bri Amanda mit einem Fehler ab.

tapedev

tapetype

Der Bandtyp.

netusage

Bandbreite, die Amanda maximal im Netzwerk verwenden darf.

labelstr

Ein regulärer Ausdruck. Alle Bänder, deren Namen mit dem Ausdruck überein stimmen, gelten als reserviert Die Bandnamen lauten DailySet1-*xx*, wobei »DailySet1« der gewählte Name der Konfiguration und *xx* eine ist.

Der entsprechende Auszug der Konfigurationsdatei könnte so ausschauen:

```
root@sonne> cat / etc/ amanda/ DailySet1/ amanda.conf
#
# amanda.conf - Eine minimale Konfiguration
#
org          "DailySet1"
mailto       "amanda"
dumpuser     "amanda"
netusage     600 Kbps
dumpcycle    4 weeks
runtapes     1
runspercycle 20
tapecycle    25 tapes
tapedev      "/dev/nst0"
tapetype     HP-DDS3
labelstr     "^ DailySet1-[0-9][0-9]*$"
...
```

Nachfolgend sollten Sie ein erstes Band für die Verwendung mit Amanda vorbereiten:

```
root@sonne> su - amanda
amanda@sonne> amlabel DailySet1 DailySet1-01
rewinding, reading label, not an amanda tape
rewinding, writing label DailySet1-01, done.
```

Entsprechend des Eintrags **tapecycle** in /etc/amanda/DailySet1/amanda.conf müssen Sie obigen Schritt für die weiteren Bänder wiederholen. Erhöhen Sie einzig die Nummerierung »01« fortlaufend.

Die Backup-Clients

Für jeden Client, der auf einem vom Server abweichenden Rechner installiert wird, müssen Sie zunächst die Einträge in /etc/services und /etc/inetd.conf vornehmen. Gehen Sie wie beim Server vor mit der Ausnahme, dass in den Dateien einzig die mit »amanda« beginnenden Zeilen aufzunehmen sind.

Legen Sie auf den Clients den **Benutzer amanda** und die **Gruppe disk** an (bzw. die Namen, die Sie bei der Konfiguration gewählt haben). Der Server muss sich nun ohne Angabe eines Passworts auf dem Client einloggen dürfen. Um dies zu konfigurieren, können Sie entweder im Heimatverzeichnis des Benutzers eine Datei **.rhosts** (Voreinstellung) oder, falls Sie »configure --amandahosts« wählten, **.amandahosts** anlegen:

```
amanda@client> cat .rhosts
sonne.galaxis.de   amanda
sonne.galaxis.de   root
sonne.galaxis.de
```

Beide Dateien dürfen nur den Eigentümer zum Lesen und Schreiben berechtigen!

Das wär's schon auf Seiten des Clients. Nun muss nur noch auf dem Server eine Datei mit der Beschreibung der zu sichernden Verzeichisse oder Partitionen angelegt werden.

Ein Eintrag der Datei **disklist** besitzt immer folgende Gestalt:

```
<Clientname>  <Verzeichnis>  <Backuptyp>
```

Als **Clientname** wird der volle Rechnername empfohlen. **Verzeichnis** kann eben ein solches sein oder aber der Name eines Devices (/dev/hda...).

Mit dem **Backuptyp** soll an dieser Stelle nur auf Standardtypen eingegangen werden. Der Eintrag selbst beschreibt, mit welchen Mitteln das Backup erzeugt werden soll. In der Datei **amanda.conf** lassen sich eigene Typen deklarieren, die mehrere Standardtypen zusammenfassen (ein Beispiel folgt im Anschluss). Die wichtigsten Typen sind:

comprate *Wert1* [, *Wert2*]

Die Kompressionsrate spiegelt die erwartete Verkleinerung des zu sichernden Verzeichnisses wieder. Wert1 steht für die Kompressionsrate des vollen Backups und Wert2 für das Inkrementelle. Fehlt Wert2, wird er a Wert1 gesetzt. Ein Wert von »0.5« besagt, dass die Archivdatei in etwa halb so groß wird wie das Original.

compress *Modus*

Als Modi sind möglich:

- **none** keine Komprimierung
- **client best** Komprimierung findet auf dem Client statt bei Verwendung des besten Algorithmus
- **client fast** Komprimierung findet auf dem Client statt bei Verwendung des schnellsten Algorithmus
- **server best** Komprimierung findet auf dem Server statt bei Verwendung des besten Algorithmus
- **server fast** Komprimierung findet auf dem Server statt bei Verwendung des schnellsten Algorithmus

Ohne explizite Modusangabe gilt "client fast".

exclude

Die dem Eintrag folgende Liste (mit Wildcards) enthält Dateien/Verzeichnisse, die vom Backup ausgeschlos werden. Mit **list < Dateiname>** kann ebenso eine Datei angegeben werden, die die auszuschließenden Dateien und Verzeichnisse enthält.

holdingdisk [yes| no]

Aus Effizienzgründen wird bei realen Konfigurationen nicht unmittelbar auf das Band geschrieben, sondern Daten landen in einem temporären Verzeichnis (die »Holding disk«), um die Clients nicht durch das langsa Bandlaufwerk auszubremsen. In der Voreinstellung ist die Option aktiviert (»yes«), mit »no« kann sie abgeschaltet werden.

index

Zum Archiv wird zusätzlich ein Inhaltsverzeichnis erzeugt.

program [DUMP| GNUTAR]

Mit welchem Programm soll das Backup vorgenommen werden? Voreinstellung ist DUMP.

record [yes| no]

Soll das Backup in der Datei /etc/dumpdates vermerkt werden? Voreinstellung ist ja.

Ein benutzerdefinierter Backuptyp wird in der Datei **amanda.conf** wie folgt vereinbart:

```
...
define dumptype >root-tar {
    global
    program "GNUTAR"
    comment "root partitions dumped with tar"
    compress none
    index
    exclude list "/usr/lib/amanda/exclude.gtar"
    priority low
}
...
```

Eine einfache Datei **disklist** sieht somit wie folgt aus:

```
amanda@sonne> cat / etc/ amanda/ DailySet1/ disklist
sonne.galaxis.de      /home   root-tar
```

Erster Test

Ein erster Test soll den korrekten Verlauf der bisherigen Installation beweisen. Als Root wird zunächst der Server begutachtet:

```
root@sonne> amcheck DailySet1
Amanda Tape Server Host Check
-----------------------------
/dumps/amanda: 2531432 KB disk space available, that's plenty.
NOTE: skipping tape-writable test.
Tape DailySet1-01 label ok.
Server check took 0.052 seconds.

Amanda Backup Client Hosts Check
--------------------------------
Client check: 1 host checked in 0.108 seconds, 0 problems found.

(brought to you by Amanda 2.4.1p1)
```

"0 problems found" ist keine schlechte Antwort...

Vorläufig stoßen wir ein Backup von Hand an. Später, wenn die Installation perfekt ist, sollte ein Cronjob die Aufgabe erledigen:

```
root@sonne> su - amanda
amanda@sonne> amdump DailySet1
```

Eine Mail sollte dem festgelegten Report-Empfänger bald ins Haus stehen. Mit etwas Glück sieht sie ähnlich der folgenden aus (stark gekürzt):

```
From amanda@sonne.galaxis.de  Sun Jan 21 02:49:35 2001
...

These dumps were to tape DailySet1-01.
Tonight's dumps should go onto 1 tape: a new tape.

FAILURE AND StrANGE DUMP SUMMARY:
  sonne   /home lev 0 StrANGE

STATISTICS:
                          Total       Full      Daily
                        --------   --------  --------
```

```
Dump Time (hrs:min)        0:42       0:41       0:00   (0:02 start, 0:00 idle)
Output Size (meg)         718.2      718.2        0.0
Avg Compressed Size (%)    28.9       28.9         --
Tape Used (%)               6.0        6.0        0.0
Filesystems Dumped            1          1          0
Avg Dump Rate (k/s)       300.5      300.5         --
Avg Tp Write Rate (k/s)   300.4      300.4         --

...

NOTES:
  planner: Adding new disk sonne:/u01.
  taper: tape DailySet1-01 kb 735456 fm 1 [OK]

DUMP SUMMARY:
                                DUMPER STATS                  TAPER STATS
HOSTNAME  DISK          L  ORIG-KB   OUT-KB COMP%  MMM:SS   KB/s  MMM:SS   KB/s
--------------------------- ---------------------------------------- --------------
sonne     /u01          0  2542390   735424  28.9   40:48  300.5   40:48  300.4

(brought to you by Amanda version 2.4.1p1)
```

Der nächste Schritt provoziert einen Fehler, da das Band die Datenmenge nicht aufnehmen kann:

```
amanda@sonne> /usr/sbin/amcheck -m DailySet1
```

In der Mail äußert sich der Fehler wie folgt (Ausschnitt):

```
Amanda Tape Server Host Check
-----------------------------
/dumps/amanda: 2523880 KB disk space available, that's plenty.
ERROR: cannot overwrite active tape DailySet1-01.
       (expecting a new tape)
NOTE: skipping tape-writable test.
Server check took 33.920 seconds.
```

Es genügt nun, ein neues Band für die Nutzung durch Amanda vorzubereiten. Amanda wird später an der Stelle fortfahren, an der das letzte Schreiben wegen des Bandendes stoppte (ein neues Band wurde zuvor eingelegt):

```
amanda@sonne> /usr/sbin/amlabel DailySet1 DailySet1-02
rewinding, reading label, not an amanda tape
rewinding, writing label DailySet1-02, done.

amanda@sonne> /usr/sbin/amcheck -m DailySet1
```

Die nächste uns erreichende Mail sollte wieder einen Erfolg verkünden.

Weitere wichtige Kommandos für Amanda

Für die tägliche Arbeit bringt das Amanda-Paket eine Reihe von Hilfsprogrammen mit, deren Zweck knapp erläutert sein soll:

amcheckdb <Konfiguration>

Konsistenzprüfung der Datenbank auf den Bändern, die in der angegebenen Amanda-Konfiguration (in obig Beispielen *DailySet1* genannt) aufgelistet sind

amoverview [-config <Konfiguration>]

Zeigt eine Statistik der von Amanda täglich in Abhängigkeit vom Backup-Level ("0" entspricht einem vollen Backup) abgearbeiteten Rechner und Dateisysteme (die voreingestellte Konfiguration ist *DailySet1*)

amcleanup < Konfiguration>

Generiert einen Mail-Bericht und setzt die Datenbank nach einem Fehler zurück. Alternativ zum manuellen Aufruf kann das Programm in einem der Bootskripte aufgerufen werden, um eventuelle Fehler automatisch zurück zu setzen (trat kein Fehler auf, tut dieses Programm auch nichts)

amverify < Konfiguration> < Slot>

Stellt durch Wiedereinlesen des Bandes sicher, dass es durch **amrestore** bearbeitet werden kann (nur für tar formatierte Backups)

amtape < Konfiguration> Kommando

Benutzer-Schnittstelle für die manuelle Ansteuerung von Bandwechslern unter Amanda

amadmin < Konfiguration> Kommando

Das Kommando für verschiedenste administrative Schritte wie bspw. zum Ermitteln der notwendigen Bände Wiederherstellung eines Dateisystems oder das beschleunigte Starten eines vollen Backups für einzelne Platten...

amrmtape < Konfiguration> < Label>

Löscht Bänder aus der Bänderliste und aus der Amanda Datenbank. (sinnvoll bei einem fehlgeschlagenen Backup oder bei fehlerhaften bzw. auszutauschenden Bändern)

amstatus < Konfiguration>

Gibt den Status eines laufenden Backups bzw. des dazu laufenden"amdump-Prozesses" wieder

Restauration - oder der Fall der Fälle

Für das Wiederherstellen von Daten mit Amanda gibt es drei prinzipielle Wege. Welcher beschritten werden muss, hängt von den in der Praxis vorkommenden Eventualitäten ab:

1. Dateien oder Verzeichnisse sind "plötzlich" verschwunden bzw. beschädigt und müssen wiederhergestellt werden
2. Eine Festplatte musste ersetzt werden, weil sie nach dem letzten Power-off nicht wieder angelaufen ist. Sie sollte nun komplett in den inhaltlichen Zustand der alten Platte versetzt werden
3. Die Platte mit dem kompletten System - und vielleicht gar die komplette Amanda-Installation - haben sich verabschiedet und sollen nun von einem Not-System wieder restauriert werden. Dies geschieht dann i.A. ohne Amanda

Wiederherstellen von Daten mit Amanda

Zum Wiederherstellen einzelner Dateien, Verzeichnisse oder kompletter Festplatten dient das Kommando **amrestore**:

```
amrestore [ -r | -c ] [ -p ] [ -h ] tapedevice [ hostname
        [ diskname [ datestamp [ hostname [ diskname [ datestamp
        ... ]]]]]]
```

Der Einsatz der Optionen wirkt sich dabei hierarchisch von oben aus, d.h. wenn der Plattenname (»diskname«) nicht angegeben wird, werden alle Backups für den vorangestellten Rechner (»hostname«) zurück gespielt. Ist kein Zeitstempel (»datestamp«) gesetzt, werden alles Backups für das zugehörige Rechner-Platten-Paar zurück gespielt. Wurde keiner der Parameter angegeben, finden alle zur Verfügung stehenden Backups bei der Wiederherstellung Verwendung.

Während des Zurückholens der Daten vom Band mit **amrecover** werden diese nicht direkt auf die Datenträger an

ihre ursprünglichen Plätze (die sie zum Zeitpunkt des Backups inne hatten) zurück gespielt. Stattdessen werden einzig die Archive vom Band zurück gelesen, welche mittels **amflush** auf dieses Band geschrieben wurden. Dabei werden Dateien erzeugt, deren Namen folgender Konvention entsprechen:

```
# hostname.diskname.datestamp.dumplevel
sonne._u01.20010204.3
```

Eine alternative Realisierung gestattet die Verwendung der Option **-p** (von "Pipe"), wodurch die Inhalte der Archive unmittelbar an ihre ursprünglichen Positionen entpackt werden.

Die Option **-c** (»compression«) legt die Archivdateien komprimiert auf der Festplatte ab, unabhängig davon, ob sie auf dem Band komprimiert vorliegen oder nicht.

Mittels der Option **-r** (»raw«) interpretiert **amrestore** die Daten auf dem Band als 1:1-Kopien der gesicherten Daten und schreibt sie unverändert zurück.

-h bringt einzig die Header der Archive auf dem Band zum Vorschein. So kann bspw. die Archivierungsmethode ermittelt werden, mit der ein Archiv angelegt wurde.

Ein Beispiel:

Aus einem unerfindlichen Grund ist uns der Überblick über die Inhalte unserer Backup-Bänder abhanden gekommen. Leider zwingt ein Datenfehler zur Restauration einiger Dateien, doch wo liegen sie?

Finden sich in unseren grauen Zellen noch diffusive Ahnungen ob des richtigen Backups, sollten wir das vermutete Band einlegen und **amrestore** mit möglichst detaillierten Parametern zu Rechnername, Plattenname und Zeitstempel ausstatten. Ist der Schleier der Erinnerung gar zu undurchsichtig, hilft ein Versuch mit bloßer Angabe des Rechnernamens:

```
amanda@sonne> amrestore /dev/nst0 sonne
```

Nach einiger Zeit der Suche auf dem Band sollte eine Dateiliste erscheinen:

```
amanda@sonne> ls /u01/recover
.                   sonne._u01.20010204.3  sonne._u01.20010210.3
..                  sonne._u01.20010205.3  sonne._u01.20010211.3
sonne._u01.20010131.3  sonne._u01.20010206.3  sonne._u01.20010203.3
sonne._u01.20010208.3
```

Welches Format sich hinter den Dateien verbirgt, offenbart ein Aufruf des Kommandos **file**:

```
amanda@sonne> file /u01/recover/*
sonne._u01.20010131.3: GNU tar archive
sonne._u01.20010203.3: GNU tar archive
sonne._u01.20010204.3: GNU tar archive
sonne._u01.20010205.3: GNU tar archive
sonne._u01.20010206.3: GNU tar archive
sonne._u01.20010208.3: GNU tar archive
sonne._u01.20010210.3: GNU tar archive
sonne._u01.20010211.3: GNU tar archive
```

Da es sich im Beispiel um tar-Archive handelt, hilft **tar**, um in den Archiven nach den zu restaurierenden Dateien zu suchen:

```
amanda@sonne> tar -tvf - /u01/recover/sonne._u01.20010131.3
...
```

Als Hilfe zum Wiederherstellen von Dateien und Verzeichnissen empfiehlt sich das Kommando **amrecover**. Dieses Programm bietet eine Benutzerschnittstelle, die auf Indexdaten zugreift, welche für Dumptypen mit eingeschalteter

Index-Option beim Backup erstellt wurden (Option **index** bei **define dumptype...** aus /etc/amanda.conf). Die Schnittstelle erlaubt eine bequeme Navigation durch die verschiedenen Versionen von gesicherten Dateien und Verzeichnissen. Zu restauriende Einträge lassen sich selektieren und der Vorgang der Wiederherstellung aus dem Programm heraus anstossen.

Wiederherstellen von Daten ohne Amanda

Ein Verlust wichtiger Daten bei gleichzeitiger Zerstörung der Amanda-Installation könnte man als Supergau der Datensicherung betrachten. Was nun?

Ein Blick hinter die Kulissen lohnt sich. Unsere Kulissen sind die Bänder mit den Sicherungsarchiven, die hoffentlich unbeschädigt in Griffweite stehen. **Amanda** hatte die verwendeten Bänder vorab formatiert und genau dieses Format machen wir uns zu Nutze, um an die begehrten Daten zu gelangen.

Die **erste Datei** liegt im Klartext vor (plain text) und ist das so genannte Volume Label. Sie enthält die *Volume Serial Number* (VSN) und den Zeitstempel der Formatierung durch Amanda.

Unmittelbar dem Volume Label folgt eine Liste von Dateien. Bekannt ist nun, dass jede dieser Dateien in Blöcken zu je 32 kByte abgelegt ist, wobei der erste Block (Header) von Amanda gespeicherte Statusinformationen umfasst.

Einem Header folgen unmittelbar die Blöcke mit den Daten der Datei. Daran schließt sich der Header der folgenden Datei an, dem wiederum die Datenblöcke folgen. Eine Restauration ohne **amrestore** läuft somit nach folgendem Schema ab:

1. Rücksetzen des Bandes
2. Einlesen der ersten Datei (Volume Label)
3. Solange noch Blöcke auf dem Band sind:
    1. Lies den nächsten Block (Header einer Datei)
    2. Lies alle Blöcke, die zu dieser Datei gehören
    3. Tue etwas mit den Daten (bspw. ins System einspielen)

Anhand eines Beispiels sollen die einzelnen Schritte begangen werden:

1. Rücksetzen des Bandes:

```
amanda@sonne> mt rewind
```

2. Auslesen der Datei mit dem Volume Label:

```
# Spulen an den Anfang der nächsten (also ersten) Datei:
amanda@sonne> mt fsf NN
# Im ersten Block (Amanda-Header) steht, mit welchem Programm das Archiv erzeugt wurde:
amanda@sonne> dd if=/dev/nst0 bs=32k
AMANDA: FILE 20010131 sonne /u01 lev 3 comp .gz program /bin/gtar To restore, position tape at start of file and run:
bs=32k skip=1 | /usr/bin/gzip -dc | bin/gtar -f... -
1+0 records in
1+0 records out
```

3. Dann die Images:

```
# Wir nutzen den Aufruf, den Amanda im Header speicherte (siehe oben):
amanda@sonne> dd if= bs=32k skip=1 | /usr/bin/gzip -dc | bin/gtar -fv -
```

Nach diesem Schema lassen sich fortlaufend die Images vom Band ziehen und ins aktuelle Verzeichnis entpacken.

Dateisysteme

Übersicht

Jede Computeranwendung wird irgendwelche Informationen speichern wollen, seien es permanente Daten, die ein Programm als Ausgabe liefert oder aber temporäre Dateien, die nicht komplett in den Adressraum eines Prozesses passen. Beim erneuten Start des Programms wird dieses seine abgespeicherten Informationen schnell wiederfinden wollen. Damit haben wir nur zwei wesentliche Aspekte eines Dateisystems tangiert...

Dateisystemimplementierungen gibt es zuhauf und Linux kennt die meisten. Wie das möglich ist, ist u.a. Gegenstand des Abschnitts.

Dateisysteme müssen vor ihrer Verwendung angelegt werden. Fehler beim Speichern, z.B. durch Rechnerabsturz, bleiben nicht aus. Also gehören auch Kommandos zur Überprüfung und Reparatur von Dateisystemen zum Umfang eines jeden Betriebssystems.

Das virtuelle Dateisystem

So ziemlich jedes neue Betriebssystem, das jemals jemand in die Welt gesetzt hatte, brachte eigene Vorstellungen vom Aussehen seines Dateisystems mit. Gemeinsam ist ihnen die Absicht, eine große Menge an Informationen permanent speichern zu können, so dass auch noch mehrere Prozesse gleichzeitig auf diese zugreifen können. Dateien sind die Container, in denen die Informationen abgelegt sind und Festplatten, Disketten, ... sind das Medium der dauerhaften Speicherung.

Aber damit hören die Gemeinsamkeiten der Dateisysteme auch schon auf.

Aus Sicht des Benutzers unterscheiden sich vor allem:

- die Möglichkeiten zur Benennung von Dateinamen; man denke nur an die 8.3-Regel bei MSDOS
- der Schutz der Daten (Attribute, Zugriffsrechte...)
- die maximal mögliche Dateigröße
- definierte Operationen (Öffnen, Schließen, Erzeugen, ..., Wahlfreier Zugriff)

Aus Sicht des Betriebssystems kommen noch die Details der Implementierung hinzu:

- Wie wird freier Speicher verwaltet?
- Welche Blockgröße wird verwendet?

Die meisten Betriebssysteme umgehen das Problem mit dem Vogel-Strauß-Algorithmus: sie unterstützen neben ihrem eigenen Dateisystem bestenfalls marktstrategisch wichtige Fremdsysteme.

Nicht so Linux, das so ziemlich mit jedem Dateisystem zu Rande kommt, und selbst den Zugriff auf Partitionen von Windows-Systemen ermöglicht.

Wie wurde das realisiert?

Indem eine Schnittstelle zwischen Betriebssystem und den unterschiedlichen Dateisystemen geschaffen wurden. Es

liegt nun in der Verantwortung dieses **Virtuellen Dateisystems**, wie es die Systemrufe des Kernels in die spezifischen Aktionen eines konkreten Dateisystems umsetzt.

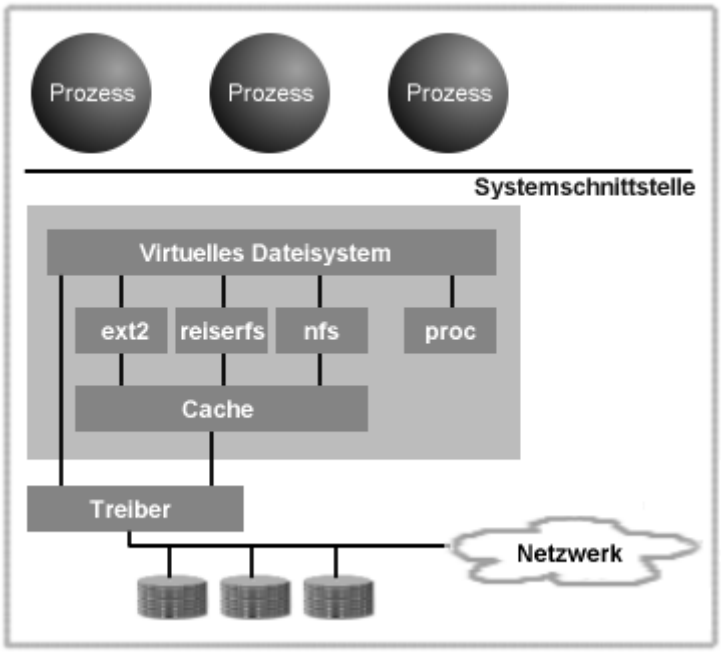


Abbildung 1: Prinzip des Virtuellen Dateisystems

Unterstützte Dateisysteme

Die Unterstützung der Dateisysteme ist Aufgabe des Kernels, d.h. jedes Dateisystem, auf das Ihr System zugreifen soll, muss in den Kernel einkompiliert oder als Modul realisiert sein. Um schnell zu entscheiden, wie die momentane Unterstützung ausschaut, können Sie einen Blick in die Datei »/proc/filesystems« werfen:

```
user@sonne> cat / proc/ filesystems
nodev   sockfs
nodev   tmpfs
nodev   shm
nodev   pipefs
nodev   proc
        ext2
        vfat
        iso9660
nodev   nfs
nodev   devpts
        reiserfs
```

Liegt der Treiber für ein Dateisystem als Modul vor, so erscheint es erst in obiger Datei, wenn sein Modul geladen wurde. Sehen Sie im Verzeichnis »/lib/modules/<kernel_version>/kernel/fs« nach, welche Dateisystemtreiber durch Module realisiert sind (seit Kernel 2.4.0 sind die Module jedes Dateisystems in einem eigenen Unterverzeichnis enthalten; bei älteren Kerneln sind alle Module in einem Verzeichnis »/lib/modules/<kernel_version>/fs« organisiert):

```
user@sonne> ls / lib/ modules/ 2.4.16/ kernel/ fs/
adfs      binfmt_aout.o coda    hpfs  nls     reiserfs udf
affs      binfmt_coff.o cramfs  jffs  ntfs    romfs    ufs
autofs    binfmt_misc.o efs     ncpfs qnx4    smbfs    umsdos
autofs4   binfmt_xout.o hfs     nfsd  ramfs   sysv
```

Beim Zugriff auf ein solches Dateisystem sollte der Kernel (exakt: der kerneld [Kernel 2.0.x] bzw. kmod [Kernel >= 2.2]) das entsprechende Modul automatisch laden. Näheres zum Kernel und Modulen erfahren Sie im Kapitel

Kernel.

Einige der interessanteren Dateisysteme fasst die folgende Tabelle zusammen:

bfs

UnixWare Boot Filesystem

devfs

Device File System; virtuelles Dateisystem zur Verwaltung der Geräte, das die (noch) üblichen Gerätedateie des Verzeichnisses /dev ersetzen wird.

ext

Der Vorgänger des ext2, kaum mehr in Gebrauch

ext2

Das Standarddateisystem von Linux

ext3

Weiterentwicklung von ext2 mit zusätzlichen Funktionen eines Journaling File Systems

iso9660

Alle CD-ROMs speichern ihre Daten unter diesem Format

hpfs

OS-2 (nur lesend)

loopback

Mounten einer einzelnen Datei als Dateisystem

minix

Minix ist der eigentliche Vorgänger von Linux, sein Dateisystem wird gern noch für (Installations-) Disketter eingesetzt

msdos

DOS, wichtig für den Zugriff auf DOS-formatierte Disketten

ncpfs

Novell Netware

nfs

Network File System

ntfs

WindowsNT (2000, XP), Schreiben ist auf dieses Dateisystem zwar möglich, jedoch kann dies leicht zu irreparablen Schäden führen

ramdisk

SMB

(Server Message Block) NT, Windows für Workgroups; zum Zugriff auf von einem Windows-Rechner freigegebene Verzeichnisse

swap

Swap-Partitionen oder -Dateien

SystemV

Verschiedene Unixe

proc

Prozessverwaltung

reiserfs

Eines der ersten Journaling File Systeme für Linux

umsdos

DOS-Dateisystem unter Linux verwenden, hier wird vor allem ein Mapping der unterschiedlichen Dateinamensformate und -rechte vorgenommen

vfat

Windows95, sowohl in der 16 bit als auch 32 bit Variante

Auf jedes Dateisystem im Detail einzugehen, würde Bände füllen, weswegen wir nur das Linux-Dateisystem **ext2**, die beiden Vertreter der Journaling Filesysteme **ReiserFS** und **ext3** sowie das Device-Dateisystem **devfs** genauer vorstellen möchten. Im Kapitel Kernel erfahren Sie Einzelheiten zum Prozessdateisystem.

Das Extended File System Version 2 (ext2)

ext2 galt über Jahre hinweg als das Standard-Dateisystem für Linux. Erst mit Aufkommen der Journaling Dateisysteme etablierten sich ernsthafte Alternativen. Doch erforderten Unzulänglichkeiten in Kernelversionen der 2.2er Serie und im Bootmanager Lilo < Version 21.6, dass zumindest der Kernel selbst und die Dateien des Lilo-Bootmanagers zwingend auf einem **ext2-Dateisystem** platziert wurden. Aktuellen Kernel- und Lilo-Versionen sind derartige Einschränkungen fremd.

Dass das **ext2** gegenüber anderen Implementierungen zahlreiche Vorteile mit sich bringt, werden Sie in diesem Abschnitt kennen lernen.

ext2 hält sich in seinem Aufbau stark an die bewährte Architektur, die allen UNIX-Dateisystemen gemein ist. Beginnen wir deshalb mit dem Allgemeinen.

Das Prinzip des Inodes ist allen Unix-Dateisystemen gleich

Generell werden die Verwaltungsinformationen von den eigentlichen Daten getrennt gespeichert. Die **Verwaltungsdaten** einer Datei, also alle Merkmale wie Rechte, Zugriffszeiten, Größe usw., werden in so genannten **Inodes** gehalten. Die einzige wichtige Information, die nicht im Inode steht, ist der Name der Datei. Der selbst steht in einem Verzeichnis und verweist auf den zugehörigen Inode. Ein Verzeichnis wiederum ist auch nur eine Datei. Es wird durch einen Inode repräsentiert und die Daten der Verzeichnisdatei sind die enthaltenen Verzeichniseinträge.

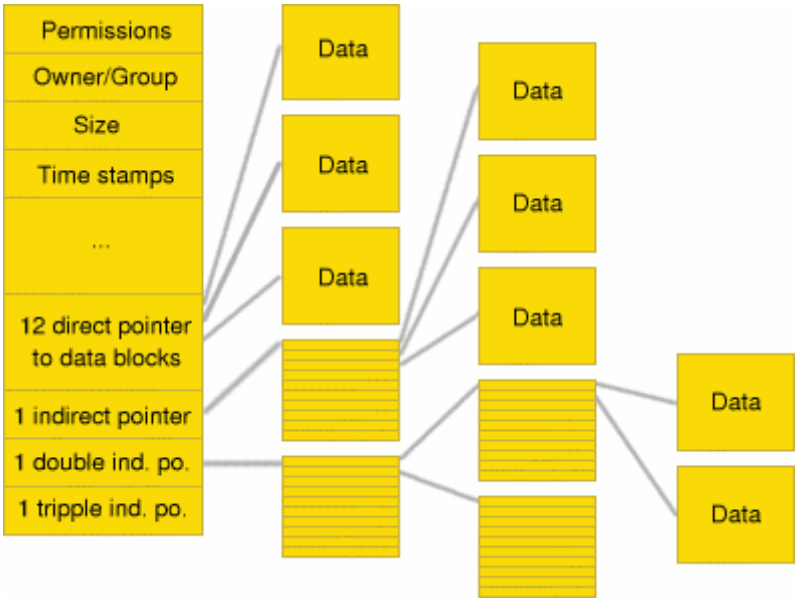


Abbildung 2: Speicherung einer Datei im Inode

Genau 60 Bytes stünden im Inode zur Speicherung von Daten zur Verfügung. Allerdings werden einzig die Daten symbolischer Links direkt im Inode gehalten, für alle anderen Dateitypen gelten die letzten 15 Einträge im Inode als 32-Bit Zeiger und verweisen auf Datenblöcke, die nun die eigentlichen Daten aufnehmen. Wie viele Zeigereinträge gültig sind, wird anhand der im Inode gespeicherten Anzahl belegter Datenblöcke berechnet.

12 direkte Zeiger auf Datenblöcke sind in einem Inode vorgesehen. Genügt der Speicherplatz für eine Datei noch immer nicht aus, referenziert der nächste Eintrag im Inode einen Datenblock, dessen Inhalt nun als Zeiger auf die eigentlichen Datenblöcke betrachtet wird; man spricht von einem **einfach indirekten Zeiger**. Ist die Datei weiterhin unersättlich, kommt der nächste Eintrag ins Spiel. Er verweist auf einen Datenblock, dessen Inhalt Zeiger auf Datenblöcke sind, deren Inhalte letztlich Zeiger auf die Daten der Datei sind. Alles klar? Es handelt sich um einen **zweifach indirekten Zeiger**. Und manche Dateien können einfach nicht genug Speicherplatz bekommen, also ist der nächste (und letzte) Eintrag im Inode ein Zeiger auf... [ich erspare mir die Ausführungen]. Es ist ein **dreifach indirekter Zeiger.**

Während die **Größe eines Inodes** im Falle des **ext2** konstant 128 Byte beträgt, kann die **Datenblockgröße** beim Einrichten des Dateisystems festgelegt werden. Zulässig sind 1k, 2k und 4k (wird meist bevorzugt; höhere Blockgrößen sind (derzeit) nicht zulässig, da die Pagegröße 4k beträgt).

Der Einfluss der Datenblockgröße auf die Speicherkapazität

Eine einfache Berechnung soll zeigen, wie die Größe eines Datenblocks die maximal mögliche Größe einer Datei beschränkt.

Unsere Blockgröße sei »1024 Bytes«, dann lassen sich mit den 12 direkten Zeigern des Inodes maximal »12*1024 Bytes» (12k) Daten speichern. Der einfach indirekte Zeiger verweist auf einen Datenblock, der maximal »1024/4« Zeiger (ein Zeiger belegt 4 Bytes) aufnehmen kann, also erreicht dieser »1024*256 Bytes« (256k) Daten. Mit derselben Überlegung erhält man für den zweifach indirekten Zeiger »1024*256*256 Bytes« (65536k) Daten und der dreifache schafft gar »1024*256*256*256 Bytes« (16GByte) Daten.

Um die machbare Dateigröße zu erhalten, sind die vier Werte zu addieren. Die nachfolgende Tabelle enthält die (theoretischen) Werte für die verschiedenen Datenblockgrößen:

Blockgröße (kByte)	Maximale Dateigröße (GByte)
1	16
2	256
4	4100

Wie so oft klafft zwischen Theorie und Praxis ein riesiges Loch, so dass tatsächlich nur Daten bis zu einer Größe von 2 GByte (Linux Kernel 2.2.x) gehandhabt werden können. Diese Beschränkung liegt in der Handhabung der Dateigröße durch den Kernel begründet, der diese nur mit 32 Bit speichert, wobei das höchstwertige Bit auch noch als Vorzeichen betrachtet wird. Im 2.4er Kernel wird mit 64 Bit gearbeitet, so dass er den »Schwarzen Peter der Begrenzung« den Dateisystemimplementierungen in die Schuhe schiebt. Um das »ext2« für diese Dateigröße einsetzen zu können, wird ab Kernel 2.4 ein bislang nicht verwendeter Eintrag im Inode zusätzlich zur Speicherung der Dateigröße verwendet, so dass die notwendigen 64 Bit - bei Wahrung der bisherigen Strukturen - zur Verfügung stehen.

Der Datenblock ist im Sinne der Dateiverwaltung die kleinste zu adressierende Einheit. Als Konsequenz ist der physische Speicherplatzverbrauch einer Datei im Mittel größer ist als ihr tatsächlicher Bedarf. Ist eine Datei bspw. 1500 Bytes groß, so benötigt sie bei einer 1k Blockgröße 2k Speicherplatz, bei 2k Blockgröße 2k Speicherplatz, bei 4k Blockgröße belegt sie jedoch 4k Speicherplatz, d.h. im letzten von einer Datei belegten Block geht im Mittel die Hälfte des Speicherplatzes als »Verschnitt« verloren. Die Logik legt nahe, dass die 1k-Blockgröße die beste Wahl sei.

Die Vermutung kann man bestätigen, für den Fall, dass man überwiegend sehr kleine Dateien speichern will. Bei großen Dateien jedoch beschleunigt die 4k-Blockgröße den Lesezugriff enorm, da sicher gestellt ist, dass umfangreiche Teile der Datei hintereinander auf der Festplatte liegen und der Schreib-Lese-Kopf diese »in einem Ritt« besuchen kann. **Hohe Blockgrößen minimieren den Effekt der Fragmentierung.**

Welcher Inode oder Block ist frei?

Wird eine neue Datei erzeugt, so muss ein freier Inode für sie gefunden und reserviert werden. Ebenso reserviert **ext2** eine Reihe möglichst zusammen hängender Blocknummern, um die Daten aufnehmen zu können. Vermutlich werden viel zu viele Blöcke als belegt markiert. Aber solange die Datei nicht geschlossen wurde, ist ihr tatsächlicher Speicherbedarf ungewiss. Wird die Bearbeitung einer neuen Datei beendet, gibt **ext2** die nicht benötigten Blöcke wieder frei, verbraucht die Datei alle Einheiten, liegen diese nun in einem Stück auf der Platte. Dieselbe »vorausschauende« Markierung von Blöcken kommt beim Anfügen von Daten an eine existierende Datei zur Anwendung, so dass der Grad der Fragmentierung einer Datei von Haus aus in akzeptablen Grenzen gehalten wird.

Um einen freien Inode oder Block aufzuspüren, könnte sich **ext2** durch die Inodes bzw. Blöcke »durchhangeln«, bis endlich ein unbelegter Eintrag gefunden wurde. Jedoch wäre diese Art der Suche furchtbar ineffizient. **ext2** verwendet deshalb **Bitmaps**, die bitweise kodieren, welche Blöcke bzw. Inodes frei und welche belegt sind. Es existieren jeweils eine Bitmap für Blöcke und eine für die Inodes. Eine »0« des Bits *x* kennzeichnet den Block (Inode) mit der Nummer *x* als frei, eine »1« steht für einen verwendeten Block (Inode).

Eine solche Bitmap belegt genau einen Block, d.h. bei 1k Blöcken verwaltet eine Bitmap 8192 Inodes bzw. Datenblöcke. Mit 4k Blockgröße können 32768 Einheiten angesprochen werden, was wiederum die mögliche Größe einer Datei auf 128 MByte beschränken würde.

Das Konzept der Gruppen

ext2 organisiert das Dateisystem in Gruppen. Jede Gruppe beinhaltet eine Kopie des Superblocks, eine Liste aller Gruppendeskriptoren, die Bitmaps für die enthaltenen Inodes und Datenblöcke und die Listen derselbigen.

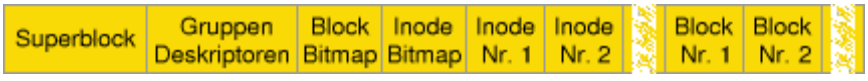


Abbildung 3: Aufbau eines ext2-Gruppe

Der **Superblock** einer jeden Gruppe ist die exakte Kopie des Superblocks des Dateisystems und beinhaltet Informationen zur Blockgröße, der Anzahl der Inodes und Datenblöcke, den Status des Dateisystems, Anzahl der Mountvorgänge seit der letzten Überprüfung usw. Alle Informationen lassen sich mit dem Kommando **dumpe2fs** gewinnen:

```
root@sonne> dumpe2fs / dev/ hda4
Filesystem volume name:   <none>
```

```
Last mounted on:          /mnt/boot
Filesystem UUID:          99c0ce4a-9e7d-11d3-86d7-b832bfd4e1f7
Filesystem magic number:  0xEF53
Filesystem revision #:    0 (original)
Filesystem features:      (none)
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              8016
Block count:              16033
Reserved block count:     801
Free blocks:              13365
Free inodes:              7983
First block:              1
Block size:               1024
Fragment size:            1024
Blocks per group:         8192
Fragments per group:      8192
Inodes per group:         4008
Inode blocks per group:   501
Last mount time:          Mon Jul  3 07:07:46 2000
Last write time:          Mon Jul  3 07:07:51 2000
Mount count:              17
Maximum mount count:      100
Last checked:             Fri Jun  9 07:06:36 2000
Check interval:           15552000 (6 months)
Next check after:         Wed Dec  6 06:06:36 2000
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)


Group 0: (Blocks 1 -- 8192)
  Block bitmap at 3 (+2), Inode bitmap at 4 (+3)
  Inode table at 5 (+4)
  6044 free blocks, 3981 free inodes, 2 directories
  Free blocks: 519-525, 1410-1416, 1826-1834, 2155-2158, 2174-2178, 2181-8192
  Free inodes: 20, 29-4008
Group 1: (Blocks 8193 -- 16032)
  Block bitmap at 8195 (+2), Inode bitmap at 8196 (+3)
  Inode table at 8197 (+4)
  7321 free blocks, 4002 free inodes, 1 directories
  Free blocks: 8712-16032
  Free inodes: 4015-8016
```

Gruppendeskriptoren beinhalten Informationen zur relativen Lage der Bitmaps und der Inode-Tabelle innerhalb einer Gruppe. Da die Deskriptoren aller Gruppen in jeder Gruppe gespeichert sind, kann bei Beschädigung einer Gruppe diese anhand der Informationen restauriert werden (zumindest solange kein mechanischer Defekt des Speicherbereichs vorliegt).

Die Nummerierung der Inodes und Blöcke ist innerhalb eines Dateisystems eindeutig. Ist der letzte Inode der Gruppe *x* der Inode Nr. *y*, so ist der erste Inode der Gruppe *x+1* der Inode Nr. *y+1*.

Und der Sinn der Gruppe? Durch die gewählte Struktur wird sicher gestellt, dass sich die Daten einer Datei nahe bei ihrem Inode befinden und die Dateien eines Verzeichnisses nahe der Verwaltungsstruktur des Verzeichnisses liegen. Letztlich verhilft die Anordnung zu einem schnelleren Zugriff auf die Festplatte.

Die äußere Struktur des ext2

Abbildung 4: Aufbau des ext2

Das Dateisystem **ext2** stellt sich als eine Aneinanderreihung gleich großer Gruppen dar, denen der in jedem

Dateisystem vorhandene und im Aufbau standardisierte Bootsektor voransteht. Die Anzahl der Gruppen wird dabei durch die Faktoren der Partitionsgröße einerseits und durch die gewählte Blockgröße und Inode-Anzahl andererseits bestimmt.

Da die Speicherkapazität einer einzelnen Gruppe auf maximal 128 MByte begrenzt ist, werden die Daten sehr großer Dateien auf mehrere Gruppen verteilt.

Journaling Dateisysteme

Die Idee

In regelmäßigen Abständen, und prinzipiell zum ungünstigsten Zeitpunkt (weil man gerade mal wieder unter Zeitdruck steht), wartet Linux mit einem verzögerten Systemstart auf:

```
/dev/hda1 has reached maximal mount count, check forced...
```

Das Linuxdateisystem **ext2** hält es mal wieder für angebracht, seinen Inhalt einer gewissenhaften Überprüfung zu unterziehen. Je nach Größe der Partition und Leistungsfähigkeit des Prozessors kann die Revision wenige Minuten bis hin zu einigen Stunden konsumieren. Solch geplanter Aktivität lässt sich mit kleineren Manipulationen zuvor kommen. Was aber nach einem Absturz oder einem hängen gebliebenen »not clean - Filesystem state«, nur weil Sie den Ausschalter zwei Sekunden zu früh bemühten? Jetzt hilft nur Geduld ...irgendwann wird der Test schon fertig werden...

Für den privaten Rechner ist das Verhalten zwar lästig, aber ohne Folgen. Was aber, wenn die Datenbank des Servers für einige Zeit ausfällt, nur weil dieser nach einem Absturz die Platten scannt? Hunderte Clients warten vergebens auf die angeforderten Daten? Hier sollte die Überführung des Dateisystems in einen konsistenten Zustand höchstens ein paar Sekündchen dauern.

Journal - Protokollierung des Dateizugriffs

ext2 ist ziemlich faul, was die Buchführung seiner Aktivitäten betrifft. Wird ein **ext2**-Dateisystem »gemountet«, vermerkt es nur, dass es in Benutzung ist. »Not clean« lautet der Status des Systems. Folgt später dann ein »umount«, wechselt der Status zurück auf »clean«. Wenn das saubere Abhängen - aus welchem Grund auch immer - scheiterte, muss **ext2** beim folgenden Systemstart alle Dateien überprüfen, um eventuellen Ungereimtheiten auf die Schliche zu kommen.

Hier setzt die Idee des **Journal Dateisystems** an. In ihm werden alle momentan **bearbeiteten Dateien protokolliert**, so dass im Falle eines Systemcrashs beim Wiederanlauf einzig diese Dateien auf Inkonsistenzen hin zu überprüfen sind. Eine weitere Maßnahme ist die Einführung eines transaktionsbasierten Modells, wonach eine Datei solange ihre Gültigkeit behält, bis die Bearbeitung einer neueren Version vollständig abgeschlossen wurde. Somit wird die fehlerträchtige Zeitspanne auf den Moment des Abspeicherns reduziert.

Die verschiedenen Implementierungen

Journaling Dateisysteme bewähren sich schon seit einigen Jahren im praktischen Einsatz. Nur eine Implementierung unter Linux ließ lange auf sich warten. Derzeit existieren vier Implementierungen:

- ext3 (Nachfolger des ext2)
- ReiserFS
- jfs (IBM)
- xfs (SGI)

Dem Umstand, ersteres einsatzbereites System mit einem Journal gewesen zu sein, hat das ReiserFS seine weite Verbreitung unter Linux zu verdanken. Dieses Dateisystem und die Erweiterung des *ext2*, das ext3 werden Sie im weiteren Text kennen lernen.

Die Entwicklungen von IBM und SGI sind zwar unterdessen ebenso verfügbar, werden aber (derzeit) kaum eingesetzt. Über die folgenden Sätze zu den maximal zulässigen Dateigrößen der einzelnen Journaling Dateisysteme hinaus, werden Sie keinen weiteren Hinweise zu diesen beiden Vertretern im Buch finden.

ext3, das sich an den Maßgaben des **ext2** orientiert und den selben Beschränkungen unterliegt, ermöglicht somit maximal 4 TByte große Dateien. Gar mit stolzen 4 Petabytes ($4*10^{15}$ Bytes) für einen einzelne Datei wartet IBM's **jfs** auf. Noch ganz andere Dimensionen schwebten den Entwicklern bei SGI vor. Ihr **xfs** soll immerhin 9 Exabytes ($9*10^{18}$ Bytes) in eine Datei stopfen können. Und ReiserFS wird in zukünftigen Versionen 10^{25} Bytes handhaben können! Inwiefern die Angaben einen praktischen Nutzen haben, sei dahingestellt...

Das ReiserFS

Das Prinzip des ReiserFS

Das ReiserFS speichert die Daten in einem **B-Baum** (ein »Balanced«-Baum ist eine Verallgemeinerung der bekannten Binärbäume). Ein Element des Baumes wird als **Knoten** bezeichnet. Ein Knoten ist dabei als **Wurzel** ausgezeichnet, von dem aus alle weiteren Knoten erreichbar sind. Alle Knoten unterhalb eines bestimmten Knotens bilden einen Teilbaum und die maximale Anzahl von Knoten, die zu einem **Blatt** (unterster Knoten) führen, bezeichnet man als die **Höhe des Teilbaums**. Unterscheidet sich die minimale Anzahl von Knoten auf dem Weg von der Wurzel eines jeden Teilbaums hin zu einem Blatt von der maximalen Anzahl auf einem beliebigen anderen Weg höchstens um 1, so spricht man von einem **balancierten** Baum.

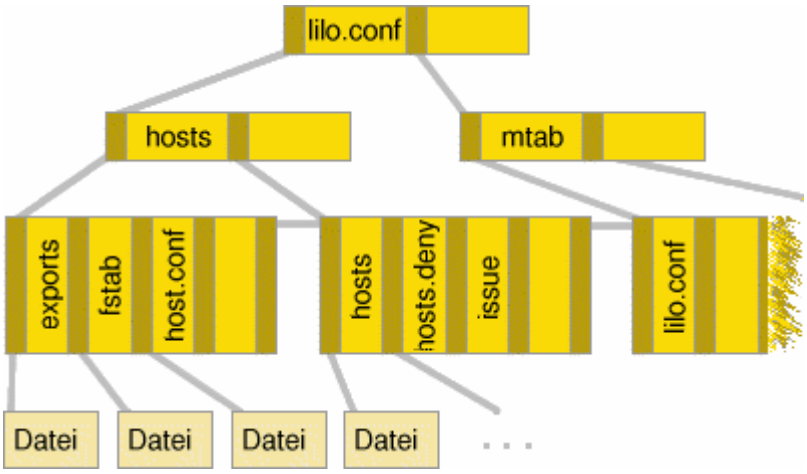


Abbildung 5: Die Baumstruktur eines ReiserFS

Ein Knoten des Baumes enthält nun die Suchschlüssel. Im Falle des ReiserFS werden die Dateinamen als Schlüssel verwandt. Innerhalb eines Knotens sind die enthaltenen Schlüssel aufsteigend sortiert und für jeden Schlüsselwert gilt, dass im Teilbaum, der durch den linken Zeiger referenziert wird, nur Schlüsselwerte gespeichert sind, die kleiner als dieser sind und im rechts angeordneten Zweig sind alle Schlüssel gleich oder größer.

Im Jargon des ReiserFS werden drei **Typen von Knoten** unterschieden:

- Der **unformatierte Knoten** enthält letztlich die Daten zu einer Datei und korrespondiert mit dem Datenblock im **ext2** (der Typ ist in der Abbildung nicht dargestellt)
- Der **Blattknoten** (»formatierter Knoten«) enthält die Einträge. Ein Eintrag kann ein »direkter Eintrag«, ein »indirekter Eintrag«, ein »Verzeichniseintrag« oder ein »Statuseintrag sein« (in der Abbildung ist der Typ in der untersten Reihe dargestellt)
- Der **interne Knoten** dient letztlich zum schnellen Auffinden eines Eintrags und enthält die geordneten Schlüssel (Zur Veranschaulichung wurden in der Abbildung 5 die Zeiger auf die Teilbäume direkt neben den Schlüsselwerten dargestellt. Tatsächlich speichert das ReiserFS zunächst eine Liste der (n) Schlüssel und nachfolgend die Liste der (n+1) Zeiger.)

Beispiel: Um die Datei hosts.deny aufzufinden, geht der Suchalgorithmus wie folgt vor: Die gesuchte Datei wird mit dem ersten Schlüsselwert aus dem Wurzelknoten verglichen. Ist er kleiner, wird im Baum, der über den links stehenden Zeiger referenziert wird, weiter gesucht. Ist der Wert gleich, so wird jeweils dem rechten Zeiger gefolgt.

Ist er größer, so wird, falls vorhanden, der nächste Schlüssel im Knoten betrachtet und das Verfahren wiederholt sich. Im obigen Beispiel ist der Schlüsselwert »hosts.deny« kleiner als »lilo.conf«, also wird im linken Teilbaum gesucht. Der erste Wert dort »hosts« ist kleiner, also wird, da kein weiterer Wert im Knoten enthalten ist, rechts weiter gegangen. Im letzten Level der internen Knoten tritt beim zweiten Vergleich eine Übereinstimmung ein. Über den korrespondierenden Zeiger kann auf die Informationen der Datei zugegriffen werden.

Das **Einfügen und Entfernen von Schlüsseln** ist eine komplexe Angelegenheit und Gegenstand der theoretischen Informatik. Dennoch möchte ich versuchen, das Prinzip knapp zu präsentieren.

Beginnen wir mit dem Anlegen einer neuen Datei (ein Verzeichnis ist auch nur eine Datei...). Der Algorithmus beginnt wieder mit dem Vergleich (Kriterium ist der neue Dateiname) in der Wurzel und folgt den Zeigern bis zu der Position im untersten internen Knoten, der die neue Datei aufnehmen müsste. Ist im Knoten noch genügend Speicherplatz zur Aufnahme des neuen Schlüssels vorhanden, so wird dieser hier eingefügt. Dabei wird die alphabetische Anordnung beibehalten (durch Verschieben größerer Schlüssel nach rechts). Jetzt wird es irgendwann geschehen, dass der Belegungsgrad eines Knotens einen bestimmten Wert übersteigt (der Faktor hängt von der Konfiguration ab). Jetzt wird der Knoten zweigeteilt, wobei jede Hälfte die gleiche Anzahl Schlüssel erhält. Der neue »linke« Knoten (mit den kleineren Schlüsselwerten) ist schon im übergeordneten Knoten enthalten, nur der erste Schlüsselwert des neuen »rechten« Knotens muss noch im »Vater« aufgenommen werden. Also wiederholt sich dort das Spielchen, eventuell muss auch der Vaterknoten gesplittet werden. Sollte gar der Wurzelknoten überlaufen, so wird auch er geteilt und eine neue Wurzel erzeugt.

Der umgekehrte Weg wird beim Löschen eines Eintrages gegangen. Wird im untersten Knoten der Eintrag entfernt, muss zunächst überprüft werden, ob der Schlüssel nicht in den übergeordneten Knoten verwendet wird. Falls dem so ist, ist der dortige Schlüssel durch einen geeigneten anderen zu ersetzen. Weiterhin wird der Grad der Belegung mit den benachbarten Knoten verglichen und ggf. eine Vereinigung dieser veranlasst. Dies zieht weite Kreise, da nun im übergeordneten Knoten ein Schlüssel zu viel enthalten ist und dieses korrigiert werden muss.

Die Speicherung der Informationen zu Dateien und Verzeichnissen findet in den **Blattknoten** statt. In der aktuellen Konfiguration existiert der Statuseintrag nicht eigenständig, sondern ist dem Datei- bzw. Verzeichniseintrag zugeordnet.
Ein **Verzeichniseintrag** beinhaltet neben den Statusinformationen alle Schlüssel zu den enthaltenen Dateien. In einem **indirekten Eintrag** stehen Zeiger auf unformatierte Knoten, die die Daten zu einer Datei beinhalten. Im Unterschied zum **ext2** sind diese Blöcke immer voll belegt, da das Ende einer Datei in einem **direkten Eintrag** gespeichert ist (es sei denn, die Datei endet exakt auf einer Blockgrenze).

Mapping der Einträge

Dem ReiserFS voran steht ein Informationsblock, der die notwendigen Angaben zum Auffinden des Journals und des Rootknotens enthält. Wenn ein Zugriff auf eine Datei erfolgt, wird diese zunächst in einer Hashtabelle gesucht. Ist die Blocknummer zu dieser Datei dort nicht enthalten, beginnt die Suche immer im Rootblock des Dateisystems. **dumpreiserfs** bringt die Strukturen des Dateisystems ans Licht:

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

<----------DUMPREISERFS, 2000---------->
ReiserFS version 3.5.23
0x3:0x7's super block in block 16
======================
Reiserfs version 0
Block count 787176
Blocksize 4096
Free blocks 454227
Busy blocks (skipped 16, bitmaps - 25, journal blocks - 8193
1 super blocks, 324714 data blocks
Root block 11598
Journal block (first?) 18
Journal dev 0
Journal orig size 8192
Filesystem state ERROR
Tree height 4
```

Das Ext3-Dateisystem

Bei der enormen Verbreitung des ext2-Dateisystems war es nur eine logische Konsequenz, dieses um die modernen Fähigkeiten eines Journaling Dateisystems nachzurüsten. Mit **ext3**, das ab Version 2.4.16 offiziell in den Kernel Einzug hielt, steht ein Dateisystem zur Verfügung, das zu 100% kompatibel mit allen Werkzeugen rund um das Dateisystem *ext2* ist. Jedes ext2-Dateisystem kann ohne Risiko in ein ext3-System konvertiert werden (und umgekehrt), sodass Sie selbst bestehenden Dateisysteme ohne Neuinstallation die Journaling-Fähigkeit verleihen können.

Das Prinzip

Journaling meint im Allgemeinen die Protokollierung der Änderungen im Dateisystem. Im Speziellen können jedoch Änderungen an den Metadaten (Informationen zu einer Datei) und den Dateien (Inhalt der Dateien) gesondert behandelt werden. Das **ext3**-Dateisystem unterstützt deshalb gleich **drei Journaling-Modi**, die sich aus den Kombinationen dieser Handhabungen ergeben:

journal

Änderungen an Metadaten und Dateien werden protokolliert. Dies ist die sicherste aber auch langsamte Journaling-Methode des *ext3*.

ordered

Änderungen an Metadaten werden protokolliert. Geänderte Dateien, deren Metadaten dies betrifft, werden zuvor gesichert (flush). Dies ist die Voreinstellung des *ext3*. Im Unterschied zu »journal« schließt dieser Mo den Verlust von Änderungen in Datei nicht aus, garantiert jedoch zu jedem Zeitpunkt die Konsistenz des Dateisystems.

writeback

Nur die Änderungen an Metadaten werden protokolliert. Änderungen an zugehörigen Dateien werden nur n den Standard-Verfahren auf die Platte geschrieben (also normalerweise aller 30 Sekunden). Ist ist durchaus inkonsistender Zustand möglich. Falls bspw. eine Datei neue Datenblöcke anfordert, werden diese in den Metadaten unverzüglich vermerkt. Die Blöcke selbst werden jedoch erst beim folgenden Sync-Zyklus mit sinnvollen Werten beschrieben.

Wenn Sie sich Vorteile davon versprechen, einen anderen als den Default-Journaling-Modus zu verwenden, dann müssen Sie nur beim Mounten des Dateisystems den gewünschten Modus angeben:

```
root@sonne> mount -t ext3 -o data=writeback /dev/hdb9 /mnt
```

Spezielle Dateisysteme

Logical Volume Manager

Wozu?

Wie oft haben Sie Linux bereits neu installiert und im Nachhinein dennoch fest gestellt, dass die von Ihnen gewählte Plattenpartitionierung doch nicht so optimal war? Sprengen die Daten nun die eine Partition?

Mit ein paar Handgriffen könnten Sie die Daten via Links auf andere Partitionen auslagern oder aber Sie könnten die Platte umpartitionieren. Eine Neuinstallation ist Dank geeigneter Programme nicht zwangsläufig notwendig.

Aber was, wenn diese Maßnahmen Ihren öffentlichen Webserver betreffen? Können Sie die notwendigen Wartungs- und Stillstandzeiten den Besuchern Ihrer Webseite zumuten?

LVM (Logical Volume Manager) könnte die Lösung solcher Probleme sein!

Wie?

Der Zugriff auf die Partitionen einer Festplatte erfolgt in herkömmlicher Weise (also ohne LVM) über die Blockgerätedateien (/dev/hda*x*, /dev/hdb*x*,... , /dev/sda*x*,...). LVM greift nun nicht mehr direkt auf diese Gerätedateien zu, sondern verwendet eigene Treiber, sodass eine Entkopplung der Gerätedateien vom physikalischen Speichermedium erfolgt.

Die Architektur des Logical Volume Manager lässt sich somit ein drei Ebenen unterteilen. Die unterste Ebene bilden die physikalischen Speichermedien (Physical Volumes). Ein oder mehrere dieser Volumes gruppieren in der mittleren Ebene die virtuellen Platten (Volume Groups). Wiederum einer oder mehrerer dieser Volume Groups stehen in der oberen Ebene als virtuelle Partitionen zur Verfügung (Logical Volumes). Der Zugriff auf ein solche virtuelle Partition erfolgt über eine Blockgerätedatei.

Mit Physical und Logical Extents tauchen im Umgang mit LVM zwei weitere Begriffe auf, deren Erklärung die folgenden Abschnitte versuchen.

Physical Volume (PV)

Ein Physical Volume umfasst die eigentliche Festplatte bzw. eine einzelne Partition auf dieser. In einem LVM System werden Sie nur bei der Adminstration mit dieser Ebene konfrontiert; bei der Anwendung übernimmt Management von LVM die Zuordnung von Verzeichnissen und Dateien zu einer konkreten Festplatte.

Physical Extent (PE)

Ein Physical Volume (PV) wird nochmals in kleinere Einheiten unterteilt. Bei diesen Physical Extents (PE) ha es sich um Bereiche zusammenhängender Sektoren; je nach Kapazität des Physical Volumes können diese Bereiche in ihrer Größe zwischen 8KB und 16GB (Voreinstellung 32 MB) variieren. Die Existenz von Physical Extents ist ein Implementierungskompromiss, um nicht jeden einzelnen Sektor in den Verwaltungsstrukture halten zu müssen. Bei einigen anderen Betriebssystemen finden Sie hierzu auch den Begriff der Physical Partition (PP).

Volume Group (VG)

Verlassen wir nun die physikalische (unterste) Ebene und wenden wir uns einer abstrakteren, einer logische Ebene zu. Die Volume Group ist ein Zusammenschluss einer oder mehrerer Physical Volumes. Mit anderen Worten, eine oder viele Festplatten oder Partitionen werden zusammengefasst zu einer großen administrati Einheit, eben einer Volume Group (VG). Von nun an kann Ihnen völlig egal sein, wieviele Festplatten Sie ha ob noch Platten hinzu kommen oder ausgebaut werden, Sie arbeiten mit einer Volume Group.

Logical Volume (LV)

Im übertragenen Sinne Können Sie sich in LVM das Volume Group (VG) als Festplatte vorstellen. Um nicht r der kompletten Volume Group (VG) arbeiten zu müssen, kann diese in Logical Volumes (LV) unterteilt werd (das Pendant zu den Partitionen). Auf solchen logischen Einheiten werden letztlich die Dateisysteme angele

Logical Extent (LE)

Ein Logical Extents (LE) ist die Repräsentation des Physical Extents (PE) im Adressraum eines Logical Volum (VL). Logical und z7ugehöriges Physical Extent besitzen demzufolge dieselbe Größe.

Einrichten des LVM-Systems

Vorbereitungen

I.d.R. werden Sie Festplatten als Speichermedium für Ihr LVM-System verwenden. Die vorgesehenen Partitionen müssen Sie auf den Typ »0x8e« (Linux LVM) setzen, damit sie für LVM nutzbar werden. Das Setzen kann u.a. mit fdisk geschehen.

```
root@sonne> fdisk -l
```

```
Platte /dev/hda: 40.0 GByte, 40020664320 Byte
255 Köpfe, 63 Sektoren/Spuren, 4865 Zylinder
Einheiten = Zylinder von 16065 * 512 = 8225280 Bytes

   Gerät boot.  Anfang     Ende    Blöcke  Id  Dateisystemtyp
/dev/hda1   *        1      957   7687071   8e  Linux LVM
/dev/hda2          958     4865  31391010    5  Erweiterte
/dev/hda5          958      990    265041   82  Linux Swap
/dev/hda6          991     2296  10490413+  8e  Linux LVM
```

Anmerkung: Wenn Sie die gesamte Festplatte als ein einziges Physical Volume betreiben, ist deren Formatierung mittels fdisk nicht zwingend erforderlich. Allerdings würde eine solche Partition weder unter /proc/partitions erscheinen noch bei »fdisk -l« angezeigt werden.

Hinweis: Zur Demonstration der weiteren Schritte wählen wir ein anderes Vorgehen. Anstatt ein LVM-System auf Festplatten(partitionen) einzurichten, legen wir uns einige hinreichend große Dateien an, verwenden diese als Loopback-Dateien und setzen darauf LVM auf. Somit bleibt das Risiko, irgendetwas an unserer Linuxinstallation zu ruinieren, relativ gering.

Folgende Kommandos richten acht Loopback-Dateien zu je 100MB ein:

```
root@sonne> for ((i=0; i<8; i++)); do
> dd if=/dev/zero of=/tmp/lvm_loop$i bs=1024 count=100k
> losetup /dev/loop$i /tmp/lvm_loop$i
> done
102400+0 Records ein
102400+0 Records aus
...
```

Wenn Sie die Loopback-Dateien nicht mehr benötigen, müssen Sie vor derem Löschen die Verbindung zur Gerätedatei lösen:

```
root@sonne> losetup -d /dev/loop0
```

Einrichten der Physical Volumes

```
pvcreate [Optionen] PhysicalVolume [PhysicalVolume]
```

Mit **pvcreate** bereiten Sie eine Partition (alternativ die gesamte Festplatte oder eine Loopback-Datei) zur Verwendung durch LVM vor. Die möglichen Optionen dienen im Wesentlichen der Ausgabesteuerung und sind nur in seltenen Fällen sinnvoll.

In unserem Beispiel initialisieren wir die sechs mit den Loopback-Dateien verbundenen Gerätedateien:

```
root@sonne> pvcreate /dev/loop{0,1,2,3,4,5}
pvcreate -- physical volume "/dev/loop0" successfully created
pvcreate -- physical volume "/dev/loop1" successfully created
...
root@sonne> pvdisplay /dev/loop0
pvdisplay -- "/dev/loop0" is a new physical volume of 100 MB
```

Das Kommando **pvdisplay** zeigt im Moment noch nicht allzu viele Informationen, da die Verwaltungsstruktur erst bei der Integration eines Physical Volumes in eine Volume Group mit sinnvollen Werten belegt wird (vergleiche nächster Schritt).

Einrichten einer Volume Group

```
vgcreate [-l max-LV] [-p max-PV] [-s PE-Größe] VG-Name PhysicalVolume [PhysicalVolume]
```

Eine oder mehrere der Physical Volume werden mittels **vgcreate** zu einer Physical Group zusammengefasst. Wichtig ist die Angabe des eindeutigen Namens für die Volume Group, der als Verzeichnisname unterhalb von »/dev« erscheint. In diesem Verzeichnis landen später die Informationen zu den aufsetzenden Logical Volumes.

Von den weiteren Optionen des Kommandos kommen dreien eine gewisse Bedeutung zu. Mit »-l max-LV« bzw. »-p max-PV« können Sie die Grenzen für die maximal zulässige Anzahl enthaltener Logical bzw. Physical Volumes herunter setzen. In der Voreinstellung betragen beide Werte 256, was für viele Anwendungen zu hoch gegriffen ist und letztlich durch Verwaltungsstrukturen nur Plattenplatz verschwendet. Beachten Sie aber, dass eine Reduzierung unumkehrbar ist und nur durch Löschen und Neuanlegen der Volume Group korrigiert werden kann. Die Größe eines Physical Extents können Sie mit »-s PE-Größe« anpassen (zwischen 8KB und 16GB in Zweierpotenzen). Da maximal ca. 64k Extents in einem Logical Volume verwaltet werden können, beeinflusst dieser Wert letztlich die Speicherkapazität (max. 2TB).

Für unser Beispiel fassen wir sechs Physical Extents zu einer Volume Group zusammen, wobei wir die Defaultwerte beibehalten:

```
root@sonne> vgcreate vg_demo0 / dev/ loop{ 0,1,2,3,4,5}
vgcreate -- INFO: using default physical extent size 32 MB
vgcreate -- INFO: maximum logical volume size is 2 Terabyte
vgcreate -- doing automatic backup of volume group "vg_demo0"
vgcreate -- volume group "vg_demo0" successfully created and activated
```

vgcreate belegt das Volume Group Descriptor Area der eingeschlossenen Physical Volumes mit konkreten Werten:

```
root@sonne> pvdisplay / dev/ loop0
--- Physical volume ---
PV Name               /dev/loop0
VG Name               vg_demo0
PV Size               100 MB [204800 secs] / NOT usable 32.19 MB [LVM: 128 KB]
PV#                   1
PV Status             NOT available
Allocatable           yes
Cur LV                0
PE Size (KByte)       32768
Total PE              2
Free PE               2
Allocated PE          0
PV UUID               2gOYAB-bXBc-uEdx-SUs4-24aW-I6Th-pHS7pu
```

Mit **vgdisplay** bringen Sie die Informationen der gesamten Volume Group zum Vorschein:

```
root@sonne> vgdisplay
--- Volume group ---
VG Name               vg_demo0
VG Access             read/write
VG Status             available/resizable
VG #                  0
MAX LV                256
Cur LV                0
Open LV               0
MAX LV Size           2 TB
Max PV                256
Cur PV                6
Act PV                6
VG Size               384 MB
PE Size               32 MB
Total PE              12
Alloc PE / Size       0 / 0
Free  PE / Size       12 / 384 MB
VG UUID               QZIbE7-jmoQ-CkTD-S3Qb-mMvX-6OEG-0Pt9W0
```

Wenn Sie die Kapazitäten der verwendeten sechs Loopback-Dateien aufsummieren, erhalten Sie 600MB. Tatsächlich sind in unserer Beispiel-Volume-Group ganze 384 MB zur Datenaufnahme verfügbar. Woher dieser

Diskrepanz? Ganz einfach... wir haben zu kleine Größen für die Physical Volumes angesetzt, sodass unverhältnismäßig viele Verwaltungsdaten anfallen. Pro Physical Volume werden ca. 32 MB für die Verwaltung benötigt. Bleiben noch runde 68MB übrig, von denen wegen der gewählten Physical-Extent-Größe von 32 MB allerdings auch nur 64 MB nutzbar sind. Bei realem Einsatz von LVM sollten Sie also die Physical Volumes wesentlich größer dimensionieren.

Einrichten eines Logical Volumes

```
lvcreate [-n LV-Name] -L LV-Größe VG-Name
```

Die zuvor erzeugte Volume Group präsentiert sich quasi als unformatierte Festplatte. Das Anlegen der Logical Volumes auf dieser kommt einer Partitionierung gleich. Neben dem Namen der zu verwendenden Volume Group ist die Angabe der Größe des Logical Volume mittels »-L LV-Größe« erforderlich. Optional können Sie den Namen für dieses Logical Volume festlegen (»-n LV-Name«). Verzichten sie darauf, wird automatisch ein Name generiert.

Jedes Logical Volume erscheint unter seinem Namen im Verzeichnis der betreffenden Volume Group (im Beispiel unterhalb von »/dev/vg_demo0«).

Zur Demonstration unterteilen wir die Volume Group in zwei Logical Volume mit 200 bzw. 184 MB Kapazität:

```
root@sonne> lvcreate -L200 -n lv_demo0 vg_demo0
lvcreate -- rounding size up to physical extent boundary
lvcreate -- doing automatic backup of "vg_demo0"
lvcreate -- logical volume "/dev/vg_demo0/lv_demo0" successfully created
root@sonne> lvcreate -L184 -n lv_demo1 vg_demo0
lvcreate -- rounding size up to physical extent boundary
lvcreate -- only 5 free physical extents in volume group "vg_demo0"
```

Offensichtlich ging beim Anlegen des zweiten Logical Volumes etwas schief. Warum? Weil für die ersten angeforderten 200 MB die Reservierung von sieben Physical Extents notwendig wurden und damit nur noch fünf Extents (160 MB) übrig bleiben. Wir nutzen also die verbliebenen Extents für das zweite Volume:

```
root@sonne> lvcreate -L160 -n lv_demo1 vg_demo0
lvcreate -- doing automatic backup of "vg_demo0"
lvcreate -- logical volume "/dev/vg_demo0/lv_demo1" successfully created
```

Mit dem Kommando **lvdisplay** holen Sie Informationen zu einem Logical Volume hervor:

```
root@sonne> lvdisplay / dev/ vg_demo0/ lv_demo0
--- Logical volume ---
LV Name                /dev/vg_demo0/lv_demo0
VG Name                vg_demo0
LV Write Access        read/write
LV Status              available
LV #                   1
# open                 0
LV Size                224 MB
Current LE             7
Allocated LE           7
Allocation             next free
Read ahead sectors     1024
Block device           58:0
```

Verwendung der Logical Volumes

Ihre neu erzeugten Logical Volumes können Sie analog zu Partitionen der Festplatten benutzen. Nachdem Sie auf ihnen ein Dateisystem erzeugt und dieses gemountet haben, können Sie wie gewohnt Ihre Dateien darauf ablegen. LVM kümmert sich für Sie um die Verteilung der Daten auf die vorhandenen Physical Volumes.

Mit folgenden Schritten formatieren wir unsere beiden Logical Volumes mit einem ext3-Dateisystem:

```
root@sonne> mkfs -t ext3 /dev/vg_demo0/lv_demo0
mke2fs 1.28 (31-Aug-2002)
Filesystem label=
OS type: Linux
...
root@sonne> mkfs -t ext3 /dev/vg_demo0/lv_demo1
...
```

Als Ziel des Mountens erzeugen wir uns zwei Verzeichnisse unterhalb von /mnt und hängen die Logical Volumes dort ein:

```
root@sonne> mkdir /mnt/lvm_demo{0,1}
root@sonne> mount /dev/vg_demo0/lv_demo0 /mnt/lvm_demo0
root@sonne> mount /dev/vg_demo0/lv_demo1 /mnt/lvm_demo1
root@sonne> df -h | grep -B 1 lvm
/dev/vg_demo0/lv_demo0
                      217M  4,1M  202M   2% /mnt/lvm_demo0
/dev/vg_demo0/lv_demo1
                      155M  4,1M  143M   3% /mnt/lvm_demo1
```

Vergrößern/Verkleinern eines LVM-Systems

Seine Stärken offenbart LVM erst, wenn die Kapazität des aktuellen Systems erweitert werden muss. Wenn Sie Plug&Play-fähige Hardware einsetzen (bspw. SCSI-Platten), dann ist nicht einmal eine Unterbrechung des laufenden Systems erforderlich. Vorrausgesetzt natürlich, es existieren Werkzeuge, die das aufgesetzte Dateisystem »on the fly« erweitern können.

Angenommen, Sie spendieren ihrem System eine neue Platte, so sollten Sie diese zunächst per fdisk partitionieren und die Partitionen mittels **pvcreate** als Physical Volumes initialisieren. Gehen Sie also genau vor, wie zu Beginn des Abschnitts beschrieben.

Anschließend verwenden Sie das Kommando **vgextend**, um die neuen Physical Volumes in die Volume Group zu integrieren.

```
vgextend VG-Name PhysicalVolume [PhysicalVolume]
```

In unserem Beispiel verwenden wir eine weitere Loopback-Datei, um die Volume Group zu vergrößern:

```
user@sonne> dd if=/dev/zero of=/tmp/lvm_loop6 bs=1024 count=200k
204800+0 Records ein
204800+0 Records aus
root@sonne> losetup /dev/loop6 /tmp/lvm_loop6
root@sonne> pvcreate /dev/loop6
pvcreate -- physical volume "/dev/loop6" successfully created

root@sonne> vgextend vg_demo0 /dev/loop6
vgextend -- INFO: maximum logical volume size is 2 Terabyte
vgextend -- doing automatic backup of volume group "vg_demo0"
vgextend -- volume group "vg_demo0" successfully extended
```

In der erweiterten Volume Group könnten Sie nun ein neues Logical Volume anlegen oder aber ein bestehendes mit Hilfe des Kommandos **lvextend** erweitern.

```
lvextend -L [+]LV-Größe VG-Name [PhysicalVolume]
```

Als neue Größe können Sie sowohl den absoluten Wert als auch den Betrag der Erweiterung angeben. In der Volume Group müssen genügend freie Physical Extents vorhanden sein. Mit der Angabe der Namens eines Physical Volumes können Sie explizit Einfluss auf die Verwendung dieses nehmen. Ohne die Angabe wird das erste freie gwählt. Das Physical Volume muss in der Volume Group vorhanden sein.

```
root@sonne> lvextend -L+160M /dev/vg_demo0/lv_demo0
vextend -- extending logical volume "/dev/vg_demo0/lv_demo0" to 384 MB
lvextend -- doing automatic backup of volume group "vg_demo0"
lvextend -- logical volume "/dev/vg_demo0/lv_demo0" successfully extended
```

Zum Abschluss müssen Sie das Dateisystem der betreffenden Logical Volume vergrößzlig;ern. Im Falle von **ext2/ ext3** könnten Sie **resize2fs** auf dem nicht-gemounteten (!) Dateisystem anwenden.

Anstatt die beiden letztgenannten Schritte separat auszuführen, bringt das LVM-Paket das Kommando **e2fsadm** mit, das die Schritte **lvextend** und **resize2fs** zusammenfasst.

Zum Verkleinern eines LVM-Systems müssen Sie in umgekehrter Reihenfolge vorgehen, d.h. Sie müssen zunächst das aufgesetzte Dateisystem, dann das Logical Volume mittels **lvreduce** und anschließend die Volume Group mittels **vgreduce** verkleinern. Im Falle eines ext2/ext3-Dateisystem kann die Schritte der Dateisystemreduzierung und von **lvreduce** wiederum **e2fsadm** übernehmen.

Entfernen eines LVM-Systems

Zum Entfernen der einzelnen Volumes benötigen Sie folgende drei Befehle:

```
lvremove LogicalVolume [LogicalVolume]
vgchange [Optionen ]VG-Name [VG-Name]
vgremove VG-Name [VG-Name]
```

Die Physical Volumes müssen nicht explizit entfernt werden; verwenden Sie die zugrunde liegenden Partitionen einfach nach Bedarf.

Zum Löschen des Logical Volumes darf dieses nicht gemountet sein:

```
root@sonne> lvremove /dev/vg_demo0/lv_demo0
lvremove -- can't remove open logical volume "/dev/vg_demo0/lv_demo0"
root@sonne> umount /mnt/lvm_demo0 /mnt/lvm_demo1
root@sonne> lvremove /dev/vg_demo0/lv_demo0 /dev/vg_demo0/lv_demo1
lvremove -- do you really want to remove "/dev/vg_demo0/lv_demo0"? [y/n]: y
lvremove -- doing automatic backup of volume group "vg_demo0"
lvremove -- logical volume "/dev/vg_demo0/lv_demo0" successfully removed
lvremove -- do you really want to remove "/dev/vg_demo0/lv_demo1"? [y/n]: y
lvremove -- doing automatic backup of volume group "vg_demo0"
lvremove -- logical volume "/dev/vg_demo0/lv_demo1" successfully removed
```

Sollten Sie jetzt versuchen, im nächsten Schritt die Volume Group zu löschen, wird Ihr Bemühen mit einer Fehlermeldung quittiert werden:

```
root@sonne> vgremove vg_demo0
vgremove -- ERROR: can't remove active volume group "vg_demo0"
```

Eine Volume Group kann nur entfernt werden, wenn sie keine Logical Volumes mehr enthät und - der Grund für obiges Scheitern - sie sich im inaktiven Zustand befindet. Zum Ändern des Zustands benötigen Sie das Kommando **vgchange**:

```
root@sonne> vgchange -a n vg_demo0
vgchange -- volume group "vg_demo0" successfully deactivated
```

Ohne Angabe des Volumen-Group-Names wirkt das Kommando auf alle Volume Groups. Zur Aktivierung einer Volume Group dient die Option »-a y«.

Jetzt sollte das Entfernen der Volume Group gelingen:

```
root@sonne> vgremove vg_demo0
```

```
vgremove -- volume group "vg_demo0" successfully removed
```

Verschlüsselte Dateisysteme

Dem Sinn oder Unsinn, der Theorie und Praxis der Verschlüsselung von Daten haben wir ein eigenes Kapitel Sicherheit gewidmet. Wie gehen an dieser Stelle nicht explizit darauf ein, warum ein verschlüsseltes Dateisystem für Sie in Frage käme. Wie konzentrieren uns hier einzig auf das Wie der Konfiguration.

Einrichtung

Aus Lizenzrechtlichen oder Sicherheitsgründen enthalten aktuelle Kernel oft nur wenige kryptografischen Algorithmen. Theoretisch werden alle in der Manpage zu *lopsetup* genannten Algorithmen unterstützt. Der SuSE-Kernel bringt allerdings einzig den **Twofish**-Algorithmus mit sich, weshalb wir im weiteren Text einzig diesen betrachten.

Der **Twofish**-Algorithmus liegt den SuSE-Kerneln als Modul bei. Falls noch nicht bereits geschehen, muss dieses vorab geladen werden:

```
root@sonne> lsmod | grep loop_fish2 || modprobe loop_fish2
```

Kann das Modul nicht gefunden werden, kommen Sie um einen neuen Kernel nicht umhin. Abbildung 6 zeigt die auszuwählenden Optionen aus dem Menü *Block devices*.

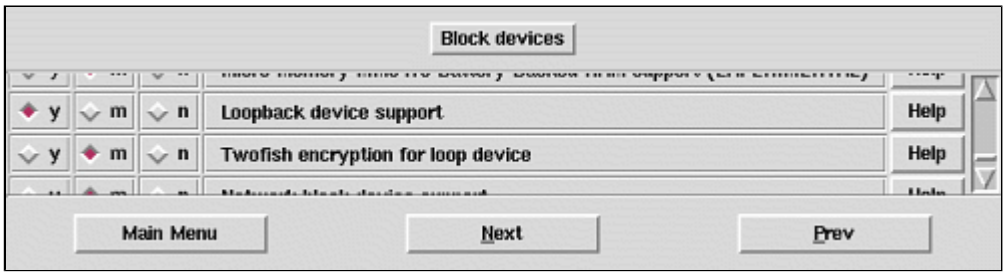


Abbildung 6: Kerneloptionen für die twofish-Verschlüsselung

Aus der Abbildung 6 ist ersichtlich, dass die Verschlüsselung von Dateisystemen nur über das Loopback-Device funktioniert. Dies stellt aber keine Einschränkung dar, sondern ermöglicht sogar die Verwendung einer Datei anstatt einer Partition als Dateisystem. Wie demonstrieren das weitere Vorgehen daher anhand einer Loopback-Datei, die folgender Befehl erstellt:

```
root@sonne> dd if=/dev/zero of=/kryptofs bs=1024 count=1024
1024+0 Records ein
1024+0 Records aus
```

Der einzige Unterschied zur Verwendung unverschlüsseltere Loopback-Dateien ist nun die Angabe des zu verwendenden Verschlüsselungsalgorithmus beim Aktivieren des Loopback-Devices:

```
root@sonne> losetup -e twofish /dev/loop1 /kryptofs
Password:
```

Sie werden zur Eingabe eines Passworts aufgefordert. Wählen Sie dieses gewissenhaft aus und vor allem... vergessen Sie es niemals. Sie haben weder eine Möglichkeit das Passwort zu ändern noch an die Daten zu gelangen, sollte Ihnen das Passwort abhanden gekommen sein.

Vor der Verwendung der Loopback-Datei müssen Sie dieses mit dem Dateisystem Ihrer Wahl formatieren:

```
root@sonne> mke2fs /dev/loop1
mke2fs 1.28 (31-Aug-2002)
```

```
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
5016 inodes, 20000 blocks
1000 blocks (5.00%) reserved for the super user
First data block=1
3 block groups
8192 blocks per group, 8192 fragments per group
1672 inodes per group
Superblock backups stored on blocks:
        8193

Writing inode tables: done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

Um das Loopback-Device freizugeben, sollten Sie zum Abschluss folgenden Befehl ausführen:

```
root@sonne> losetup -d / dev/ loop1
```

Verwenden des verschlüsselten Dateisystems

Zum Verwenden des verschlüsselten Dateisystems müssen Sie dieses an eine beliebige Position des Root-Dateisystems mounten. Prinzipiell kann dies bereits beim Booten des Systems erfolgen. Jedoch ist hiervon abzuraten, da das Mounten stets die Eingabe des Passwortes erfordert und der Bootvorgang solange unterbrochen wird.

Ohne Eintrag in die Datei /etc/fstab bleibt das Mounten dem Systemverwalter vorbehalten. Der korrekte Aufruf müsste folgendermaßen aussehen:

```
root@sonne> mkdir -p / mnt/ kryptofs
root@sonne> mount -t ext2 -o loop,encryption= twofish / kryptofs / mnt/ kryptofs
Password:
```

Um allen Benutzern das Mounten zu ermöglichen, insofern Sie Kenntnis vom Passwort haben, muss die Datei /etc/fstab um folgende Zeile ergänzt werden:

```
root@sonne> vi / etc/ fstab
...
/kryptofs   /mnt/kryptofs       ext2    loop,encryption=twofish,noauto,user   0  0
```

RAM-Dateisysteme (am Beispiel tmpfs)

Wozu?

Wie der Name schon andeutet, handelt es sich um Dateisysteme, die im RAM-Speicher liegen. Stark verbreitet sind solche RAM-Dateisysteme bspw. bei so genannten Embedded Systemen, die beim Start ein äußerst schlankes Betriebssystem zumeist aus einem ROM-Speicher in eine RAM-Disk entpacken und diese RAM-Disk als ihr Dateisystem verwenden. Auch aktuelle Linuxkernel werden häufig so konfiguriert, dass sie zunächst auf einem RAM-Dateisystem (Initial-Ramdisk) starten, dort gewisse Systeminitialisierungen vornehmen und erst anschließend auf die Dateisysteme der Festplatten zugreifen. Im Zusammenhang mit letzerem Verfahren finden Sie im Abschnitt Systemadministration, Booten eine ausführliche Einführung.

Ein weiterer Aspekt ist der enorme Geschwindigkeitsvorteil, den ein RAM-Dateisystem gegenüber Dateisystemen auf Festplatten besitzt. Bspw. arbeiten zahlreiche Programme mit temporären Dateien, die sie zumeist im Verzeichnis »/tmp/« anlegen. Da das Verzeichnis /tmp nach Filesystem-Hierarchie-Standard ohnehin keine Daten über einen Systemstart hinweg sichern muss, ist dieses ein geeigneter Kandidat für die Realisierung in einem RAM-

Dateisystem.

RAM-Dateisysteme existieren in mehreren Variationen. Jede besitzt ihre Stärken und Schwächen und damit ihren besonderen Einsatzbereich. Wir verfolgen im weiteren Text speziell das **tmpfs**. Im Unterschied zu den meisten RAM-Dateisystem kann **tmpfs** sowohl RAM als auch den Swap-Speicher zur Datenablage verwenden, deshalb finden Sie auch oft den Begriff des *Virtuellen Speicher Dateisystems*.

Wie?

Der Linuxkernel ist für die Verwaltung des virtuellen Speichers allein verantwortlich. D.h. **tmpfs** tut letztlich nichts anderes, als Seiten des virtuellen Speichers vom Kernel anzufordern. Das Dateisystem selbst hat weder Kenntnis noch Einfluss darauf, ob eine Seite aktuell im RAM oder im Swap liegt. Je nach Speicherauslastung wird die Speicherverwaltung des Kernels Seiten bei Bedarf in den RAM ein- oder aus dem RAM auslagern.

Das **tmpfs** selbst ist als Bestandteil des Virtuellen-Speicher-Subsystems realisiert und ist damit im Unterschied zu den sonstigen RAM-Dateisystemen kein Blockgerät, d.h. es erscheint weder als Gerätedatei im Verzeichnis /dev noch muss explizit ein Dateisystem auf diesem erzeugt werden.

Daraus erwächst letztlich ein enormer Vorteil: Die Grösz lig;e des Dateisystem passt sich dynamisch den Erfordernissen an. Sie ist intitial relativ klein und wächst und sinkt mit dem Speichern und Löschen von Dateien. Beachten sollten Sie allerdings, dass RAM und Swap entsprechend großzügig dimensioniert sein sollten.

Einrichtung

Sie benötigen einen Kernel mit Unterstützung für das **tmpfs**. Schauen Sie in der Datei »/proc/filesystems« nach, ob ein Eintrag *nodev tmpfs* existiert. Wenn nicht, kommen Sie um eine Kernelübersetzung nicht umhin. Wichtig ist der in Abbildung 7 zu sehende aktivierte Eintrag *Virtual memory file system support (former shm fs)*:

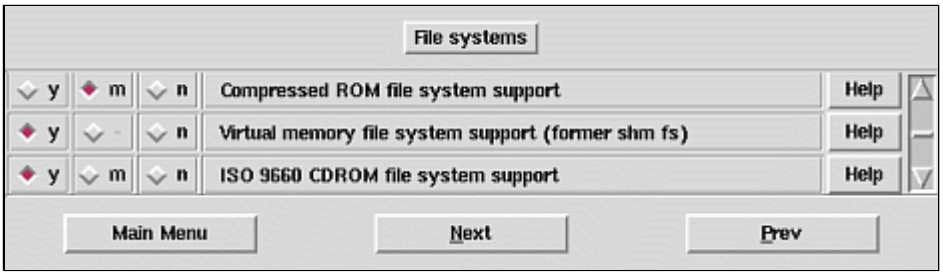


Abbildung 7: Kerneloption für das Virtuelle Speicher Dateisystem

Zum Erzeugen genügt ein simpler Mount-Aufruf mit einem existierenden Mountpunkt:

```
root@sonne> test -d /mnt/tmpfs || mkdir /mnt/tmpfs
root@sonne> mount tmpfs -t tmpfs /mnt/tmpfs
```

So einfach obiges Verfahren ist, so gefährlich ist es auch. Die Größe des Dateisystems könnte nun wachsen bis der Virtuelle Speicher erschöpft ist. Tritt dieser Fall jedoch ein, ist der Kernel gezwungen, Speicher freizuräumen, er wird einen speicherintensiven Prozess auswählen und beenden. Vermutlich werden Sie mit dem System nicht mehr vernünftig arbeiten können...

Deshalb sollten Sie dem Dateisystem eine Obergrenze setzen. Wie hoch diese ist, hängt letztlich von der Auslastung Ihres Systems ab. Beobachten Sie eine Zeit lang die Entwicklung der Speicherbedarfs und verwenden Sie für **tmpfs** höchstens so viel, wie bei maximaler Auslastung (Summe aus RAM und Swap) noch übrig blieb. Geben Sie den ermittelten Wert beim Mounten des Dateisystems an und es wird niemals diese Grenze überschreiten:

```
root@sonne> mount tmpfs /mnt/tmpfs -t tmpfs -o size=128M
```

Im Beispiel verwendeten wir ein neues Verzeichnis als Mountpunkt. Hätten wir /tmp benutzt, hätten wir vermutlich

einigen Programmen ihrer temporären Dateien beraubt. Sie hätten es uns sicher übel genommen. Um **tmpfs** für /tmp zu nutzen, müssen Sie dieses noch während des Bootens mounten. Am eifachsten ist ein Eintrag in die Datei /etc/fstab:

```
root@sonne> grep tmpfs / etc/ fstab
tmpfs  /tmp   tempfs  size=128m  0 0
```

Das Device-Dateisystem

Wie es vor dem Device-Dateisystem war...

Dieser Fall dürfte in den meisten aktuellen Distributionen noch Gang und Gäbe sein, da aufgrund des experimentellen Status der Device-Dateisystem-Entwicklung und gewisser Hürden bei der Migration auf den Einsatz (noch) verzichtet wird.

Sie müssen nur einmal einen Blick in das Verzeichnis »/dev« werfen, um ein grobes Gefühl vom Wirrwarr mit den Gerätetreibern zu erlangen. Typischer Weise werden sie mehrere hundert Dateien vorfinden, die die Hardwaretreiber im Dateisystem repräsentieren. Programme werden i.d.R. mit Hilfe dieser Datei mit den Geräten kommunizieren, anstatt die relativ komplexe Schnittstelle der Hardware direkt anzusprechen.

Jede diese Gerätedateien verfügt über zwei Nummern, die *Major Number* und die *Minor Number*. Wenn ein Treiber sich beim Kernel registriert, dann teilt er diesem eine Nummer mit, die den Treiber eindeutig identifiziert. Die *Major Number* der Datei deckt sich mit dieser Nummer des Treibers. Ein Treiber wiederum vermag mitunter mehrere Geräte kontrollieren. Welches aktuell zu verwenden ist, besagt letztlich die *Minor Number* der Datei. Beachten Sie, dass für zeichen- und blockweise arbeitende Geräte trotz identischer Nummern unterschiedliche Treiber verwendet werden. *Major Number* und *Minor Number* adressieren letztlich einen Satz von Dateioperationen, über die mit der Hardwarekomponente kommuniziert werden kann.

Warum existieren derer so viele? Weil die aktuellen Linuxkernel halt so viel unterschiedliche Hardware unterstützen. Und jedes neue Gerät, für das ein neuer Linuxtreiber geschrieben werden sollte, wird sich mit einer neuen Datei repräsentieren, insofern der Treiber es zum offiziellen Bestandteil des Kernels schafft.

Sind wirklich alle Dateien notwendig? Eigentlich nein. Aber die Distributoren haben letztlich keine Kenntnis, welche Hardware auf dem installierten System vorhanden ist. Deshalb ist es für sie am einfachsten, alle bekannten Gerätedateien mit Hilfe des Programmes »MAKEDEV« vorab anzulegen. Prinzipiell wäre es denkbar mittels eines Programmes nach nicht verfügbaren Treibern zu scannen und die zugehörigen Gerätedateien zu entfernen. Jedoch würde damit der nachträgliche Einbau von Hardware oder auch nur das nachträgliche Laden von Modulen erschwert, da in solchen Situationen (bspw. beim Anstecken eines USB-Sticks) die Gerätedateien zu erzeugen wären.

Die Identifizierung von Treibern über Major und Minor Numbers hat noch weitere Schwächen. Ein akutes Problem ist die beschränkte Anzahl der Nummern (8 Bit), sodass bei der Fülle heutiger Treiber die 128 Major und 128 Minor Numbers kaum mehr ausreichen. Für SCSI-Geräte führte das schon dazu, dass mehrere Major Numbers für diese reserviert werden mussten. Theoretisch wäre eine Erweiterung auf 16-Bit Nummern denkbar, was aber zu extrem großen Datenstrukturen im Kernel führen würde, um Einsprungspunkte für alle Treiber zu verwalten (derzeit geht das über eine Tabelle mit 128 Einträgen; bei 16 Bit müsste diese Tabelle 32768 Einträge umfassen).

Wie es mit dem Device-Dateisystem ist...

Zunächst einmal handelt es sich um ein eigenes Dateisystem. Es existiert, sobald dieses in den Kernel einkompiliert wurde. Es kann an beliebiger Position ins Dateisystem eingehangen werden; üblich ist jedoch »/dev« als Mountpunkt. Selbst wenn das Device-Dateisystem nicht gemountet ist, existiert es mitsamt seiner Einträge, sodass die Treiber stets verfügbar sind.

Der Inhalt des Device-Dateisystems wird nicht auf Festplatte gespeichert, d.h. bei einem Neustart des Systems muss der Kernel die Hardware selbstätig erkennen und initialisieren. In einer frühen Phase des Starts wird der Kernel hierzu die Initialisierungsroutinen der fest einkompilierten Treiber aufrufen. Über Kommandozeilenoptionen des Kernels ist das Verhalten dieser Routinen sogar beeinflussbar. Der jeweilige Treiber registriert sich über seinen Namen beim Kernel. Im Falle von Hardwaretreibern wird die Registrierung scheitern, falls das Gerät nicht

vorhanden ist. Ist sie erfolgreich, erzeugt die Registrierungsroutine unter dem Namen des Treibers einen Eintrag im Device-Dateisystem und vermerkt, dass diese Datei zum soeben im Kernel registrierten Treiber gehört. Die zugehörigen Dateioperationen stehen nun direkt im Dateisystemeintrag anstatt in einer kernelinternen Tabelle.

Dieser Registrierung erfolgt analog beim Laden von Modulen. Beim Entfernen von fest einkompolierten oder als Modul realisierten Treibern verschwinden auch die Einträge aus dem Devivce-Dateisystem.

Wenn nun eine Anwendung auf einen Eintrag des Device-Dateisystems zugreift, dann könnte es sein, dass dieser nicht existiert, weil der zugehörige Treiber als Modul realisiert und noch nicht geladen wurde. In diesem Fall veranlasst das Device-Dateisystem automatisch das Laden der relevanten Module und der angeforderte Eintrag wird erzeugt (hierzu muss der **devfsd** aktiv und entsprechend konfiguriert sein; siehe weiterer Text).

Die aktuelle Implementierung des Device-Dateisystems umfasst einen Device Mangement Daemon **devfsd**, der weitere Aufnahmen wahrnimmt. Diesem Daemon können bei Ereignissen wie bspw. der Registrierung eines Treibers Nachrichten gesandt werden, die ihn zu bestimmten Aktionen veranlassen. So vermag er die Zugriffsrechte der Einträge im Device-Dateisystem ebenso zu ändern, wie Programme aufzurufen, um bspw. ein Dateisystem zu mounten, wenn eine Diskette ins Laufwerk eingelegt wurde.

Da bei weitem noch nicht alle Treiber auf Verwendung des neuen Device-Dateisystems umgestellt wurden, ist es zweckmäßig, den **devfsd** zu starten, der automatisch benötigte Gerätedateien nach altem Schema im Device-Dateisystem erstellt. Hierzu muss dieses jedoch gemountet sein. Alternativ könnten Sie solche Einträge auch per Hand erstellen, falls Sie nichts besseres zu tun haben...

Umstieg auf das Device-Dateisystem

Warnung: Der Umstieg ist alles andere als trivial und kann Ihr System im schlimmsten Fall unbrauchbar machen.

Kernelunterstützung

Wie anfangs angedeutet, haben die aktuellen Distributionen in ihren Kerneln die Unterstützung für das Device-Dateisystem deaktiviert. Sie kommen um die Konfiguration eines neuen Kernels nicht umhin.

Hierzu muss zunächst unter *Code maturity level options* die Option *Prompt for development and/or incomplete code/drivers* aktiviert werden, um die notwendigen Auswahlen freizuschalten.

Im Menü *File systems* muss mindestens die Option */dev file system support (EXPERIMENTAL)* angeschaltet werden (Abbildung 8). Das Dateisystem automatisch vom Kernel mounten zu lassen (*Automatically mount at boot*) muss nicht immer eine gute Idee sein. Nämlich dann nicht, wenn Sie die Rechte an Einträgen ändern und diese einen Reboot überdauern sollen. Den Mountvorgang können Sie ebenso gut in einem Startskripte oder gar in der /etc/fstab unterbringen.
Die Option für erweiterte Debug-Ausgaben sollten Sie nicht aktivieren, es sei denn Sie mögen etwas Aktivität auf der Konsole.

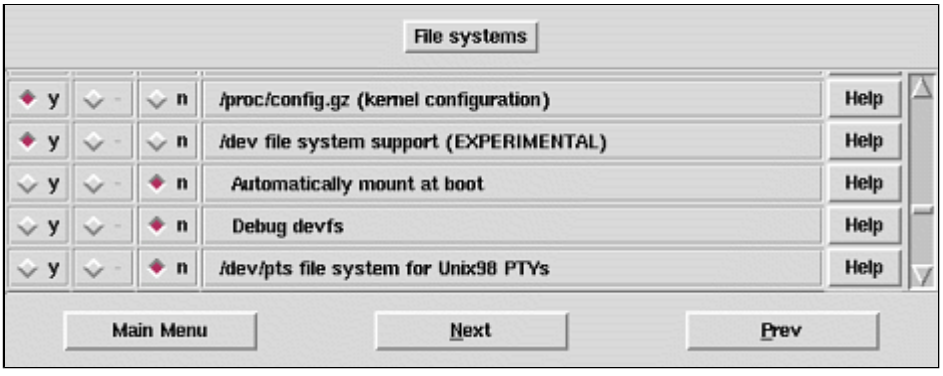


Abbildung 8: Kerneloption für das Device-Dateisystem

Da das Device-Dateisystem die Pseudoterminals (/dev/pts) selbst verwaltet, sollten sie die Option */dev/pts file system for Unix98 PTYs* entweder deaktivieren oder einfach dessen Mounten verhindern, indem Sie den Eintrag

aus /etc/fstab entfernen.

```
root@sonne> grep devpts / etc/ fstab
#devpts      /dev/pts      devpts   mode=0620,gid=5      0 0
```

Änderungen in /etc/securetty

Da die Zugangskontrolle fü den Systemverwalter *root* heute meist über die Pluggable Authentication Modules realisiert ist und PAM ein Problem mit symbolischen Links hat, muss die Datei /etc/securetty um Einträge ergänzt werden, die den direkten Zugriff auf die Gerätedateien ermöglichen (anstatt über Links):

```
root@sonne> vi / etc/ securetty
#
# This file contains the device names of tty lines (one per line,
# without leading /dev/) on which root is allowed to login.
#
tty1
tty2
...
tty6
# for devfs:
vc/1
vc/2
...
vc/6
```

Start des Device Mangement Daemons

Das weitere Vorgehen ist ziemlich trickreich und funktioniert erst mit Kerneln ab Version 2.4.0.

Wir nehmen an, dass sich mit hoher Wahrscheinlichkeit in Ihrem System noch ältere Treiber tummeln, auf deren Verwendung Sie angewiesen sind. Da für etliche Gerätedateien die Rechte anzupassen sind, um sie für den »normalen« Benutzer verfügbar zu machen, wären Sie sicherlich nicht ganz unglücklich, wenn solche Änderungen einen Neustart des Systems überdauern würden. D.h. nichts anderes als das die Modifikationen auf die Festplatte gespeichert werden müssen. Doch wohin?

Nach /dev! Dieses Verzeichnis existiert ja weiterhin, es wird nur durch das Mounten von **devfs** unsichtbar. Aber mit der *Bind-Eigenschaft* neuerer Versionen des Virtuellen Dateisystems kann der Zugang zum »alten« Inhalt bewahrt werden.

Der Trick ist nun, dass wir den Zugriff auf den alten Inhalt erhalten, indem wir das Verzeichnis /dev *vor dem Mounten des Device-Dateisystems* an eine andere Position im Dateisystem via *bind* mounten. Erst anschließend wird das Device-Dateisystem auf /dev eingehangen und der Device Mangement Daemon gestartet. Der Daemon wird nun seine Änerderungen permanent an der neuen Mountposition speichern.

Die soeben beschriebenen Schritte sollten so früh wie möglich während des Bootprozesses vorgenommen werden. Welches Skript dafür in Frage kommt, differiert bei den verschiedenen Distributionen. Schauen Sie hierzu in der Datei /etc/inittab nach (bei aktuellen SuSE-Distributionen ist es die Datei /etc/rc.d/boot). Ergänzen Sie das Skript um folgende Einträge:

```
# Beispiel gilt nur für SuSE 8.2!
root@sonne> vi / etc/ rc.d/ boot
# ...ziemlich zu Beginn des Skripts
mkdir -p /dev-state
mount --bind /dev /dev-state
mount -t devfs none /dev
devfsd /dev
...
```

Konfiguration des Device Management Daemons

Die Konfiguration erfolgt in der Datei »/etc/devfsd.conf«. Die im Paket ausgelieferte Version sollte alle relevanten

Einträge umfassen, nur sind sie zum Teil auskommentiert.

Eine Zeile der Konfigurationsdatei umfasst stets drei Spalten:

```
Ereignis    Device    Aktion
```

Die reservierten Bezeichner für **Ereignis** sind zumeist selbsterklärend. Mögliche Werte sind *REGISTER*, *UNREGISTER*, *CREATE*, *DELETE*, *CHANGE*, *RESTORE* und *LOOKUP*. Das Ereignis *LOOKUP* tritt ein, wenn irgendein Prozess einen Eintrag aus dem Device-Dateisystem öffnet.

Die zweite Spalte benennt die Devices, für die die Zeile relevant ist. Hier dürfen auch Reguläre Ausdrücke Verwendung finden.

In Spalte drei steht die Aktion, die beim Eintreten des Ereignissen für die angegebenen Devices auszuführen ist. Eingeleitet wird eine Aktion durch ein Schlüsselwort. Für verschiedene Ereignisse sind verschiedene Schlüsselworte zulässig. Manche Schlüsselworte erfordern weitere Argumente, bspw. irgendwelche Kommandoaufrufe. Wegen der Fülle der erlaubten Einträge verzichten wir auf eine detaillierte Vorstellung und verweisen auf die folgenden Beispiele.

Die folgenden Fragmente aus der Datei sind Empfehlungen. Wir erlätern ihren Sinn jeweils am Anschluss.

```
REGISTER        .*              MKOLDCOMPAT
UNREGISTER      .*              RMOLDCOMPAT
REGISTER        .*              MKNEWCOMPAT
UNREGISTER      .*              RMNEWCOMPAT
```

Mit diesen Anweisungen wird zu jedem Treiber, der sich am Device-Dateisystem registriert, automatisch vom **devfsd** eine Gerätedatei nach alten Stil erzeugt (also bspw. /dev/hda falls sich der IDE-Treiber registriert). Beim Entfernen eines Treibers wird diese Gerätedatei ebenfalls gelöscht. Diese Zeilen helfen letztlich bei der Migration zum Device-Dateisystem, da Sie am Anfang kaum wissen können, welche der installierten Programme die alten Gerätedateien benötigen. So sollten sie wie gewohnt arbeiten.

```
LOOKUP          .*              MODLOAD
```

Mit diesem Eintrag wird der Device Management Daemon automatisch die notwendigen Module laden, wenn ein Prozess eine Datei des Device-Dateisystems öffnet und der betreffende Treiber noch nicht registriert ist. Intern wird »modprobe <Eintrag>« gerufen. Dieser Eintrag ist der Name der Gerätedatei, die der Prozess versucht zu finden. Damit ein Modul gefunden werden kann, muss ein entsprechender Alias in einer Datei **/etc/modules.devfs** existieren. Die Syntax der Datei entspricht exakt der der Datei /etc/modules.conf. Angenommen, ein Prozess versucht /dev/cdrom/cdrom0 zu öffnen, dann sollte die Datei »modules.devfs« einen Eintrag analog zu diesem enthalten:

```
user@sonne> grep '/dev/cdrom/cdrom0' /etc/modules.devfs
alias   /dev/cdrom/cdrom0   <mein_cdrom_modul>
```

Sie könnten obige Zeile auch direkt in die Datei /etc/modules.conf aufnehmen, allerdings könnte das Probleme geben, falls Sie Ihr System einmal ohne das Device-Dateisystem starten. Die Datei »modules.devfs« bindet »modules.conf« ohnehin mittels des Include-Befehls ein, sodass für **devfsd**-spezifische Ergänzungen »modules.devfs« die sauberere Lösung ist.

```
REGISTER        ^pt[sy]/.*      IGNORE
CHANGE          ^pt[sy]/.*      IGNORE
REGISTER        .*              COPY    /dev-state/$devname $devpath
CHANGE          .*              COPY    $devpath /dev-state/$devname
CREATE          .*
```

Die Gruppenverwaltung

Übersicht

Berechtigungen beim Zugriff auf Dateien (Verzeichnisse, Geräte usw. sind bekanntlich nur spezielle Varianten von Dateien) werden unter Unix in drei »Ebenen« vergeben. Da wäre der Besitzer einer Datei, der letztlich selbst über die Rechte des Zugriffs entscheidet. Diesen Befugnissen für eine einzelne Person stehen die »Weltrechte« gegenüber, die bestimmen, was alle »Nicht-Besitzer« mit der Datei anstellen dürfen. Natürlich existieren genügend Szenarien, in denen Rechte weder exklusiv einer Person zugeordnet, noch jedermann gewährt werden sollten. Genau diese Beschränkung der Rechte auf einen bestimmten Personenkreis kann über Gruppen geregelt werden, indem die betreffende Datei einer konkreten Gruppe gehört, deren Mitglieder die befugten Personen umfassen.

Die Zugehörigkeit von Benutzern zu einer Gruppe kann auf mehreren Wegen erfolgen. Zum einen wird durch den Gruppeneintrag in der Datei /etc/passwd die Default-Gruppe eines Benutzers bestimmt. Legt dieser Benutzer z.B. eine neue Datei an, wird diese zunächst dieser Gruppe gehören. Soll ein Benutzer mehreren Gruppen angehören, so ist er in die Liste der Gruppenmitglieder der Datei /etc/group aufzunehmen. Ein Benutzer kann dann für die Dauer der aktuellen Sitzung eine dieser Gruppen zu seiner Default-Gruppe ernennen. Und schließlich besteht die Möglichkeit, eine Gruppe durch ein Passwort zu schützen. In diesem Fall darf jeder Mitglied der Gruppe werden, der das Passwort kennt.

Die Datei / etc/ group

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

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

Die Einträge bedeuten:

Gruppenname

Der Name der Gruppe; auch hier ist die Beschränkung auf Kleinbuchstaben und maximal 8 Zeichen üblich (nicht notwendig).

Passwort

Benutzer können bei Kenntnis des Passwortes die Gruppe wechseln, auch wenn sie kein Mitglied der Grupp sind; es gelten die gleichen Aussagen wie für die Passwörter der /etc/passwd.

Gruppennummer

Nichtnegative Zahl < 64000. Gruppennummern < 100 sind für Systemzwecke reserviert und sollten nicht verwendet werden.

Mitgliederliste

Durch Komma getrennte Liste der Nutzerkennzeichen.

Der zweite und vierte Eintrag können entfallen, d.h. die Gruppe ist nicht durch ein Passwort geschützt bzw. in die Gruppe kann kein Benutzer wechseln, der diese nicht als Default-Gruppe hat.

Die Passwortabfrage entfällt für Gruppenmitglieder. Passwörter werden bei Verwendung des Shadow-Passwort-Systems (dieses verwenden alle aktuellen Distributionen) in der Datei /etc/gshadow gespeichert.

Die Datei / etc/ gshadow

Anstatt Gruppenpasswörter und -mitglieder in der für alle lesbaren Datei /etc/group zu speichern, werden bei Verwendung von Shadow-Passwort-Systemen diese in der nur für Root lesbaren Datei »/etc/gshadow« gehalten. Der Aufbau eines Eintrages besitzt folgendes Format:

```
Gruppenname:Passwort:Gruppenverwalter:Mitgliederliste
```

Die Einträge bedeuten:

Gruppenname

Wie in /etc/group

Passwort

Das verschlüsselte Passwort; meist wird auf eine solche Möglichkeit verzichtet. Zum Setzen des Gruppenpasswortes bedient sich der Gruppenverwalter des Kommandos **gpasswd**.

Gruppenverwalter

Nutzerkennzeichen der/des Benutzer(s), den/die der Systemverwalter zu Administratoren für diese Gruppe ernannt hat. Der hier angeführte Nutzer darf andere Nutzer zur Gruppe hinzufügen und deren Einträge auch wieder entfernen, sowie das Gruppenpasswort ändern.

Mitgliederliste

Durch Komma getrennte Liste der Nutzerkennzeichen.

Konvertieren der Einträge

Anmerkung: Die nachfolgend beschriebenen Programme liegen nicht jeder Distribution bei. Ob sie in Ihrer Version der Shadow-Utilities vorliegen, erkennen Sie anhand der Ausgabe von:

```
user@sonne> rpm -ql shadow
```

Bevorzugen Sie die manuelle Bearbeitung der Konfigurationsdateien, so können Sie im Falle der Gruppenverwaltung auf zwei Kommandos zurückgreifen, die eine automatische Konvertierung der Dateien /etc/group und /etc/gshadow in das jeweils andere Format vornehmen. Ein solche Umwandlung kann in mehreren Situationen erforderlich werden:

- Sie bearbeiten die /etc/group per Hand und möchten daraus die /etc/gshadow automatisch generieren lassen
- Sie stellen Ihr Passwortsystem nachträglich vom traditionellen Mechanismus auf das Shadow-System um
- Sie wollen das Shadow-System entfernen

Das Kommando **grpconv** erzeugt aus einer vorhandenen Datei /etc/group und ggf. /etc/gshadow eine neue Datei /etc/gshadow, indem es Passwörter und Mitgliederlisten aus der Gruppen- in die Shadowdatei überträgt:

```
root@sonne> grpconv
```

grpunconv schreibt die Passwörter und Mitgliederlisten aus der /etc/gshadow in die Datei /etc/group zurück und löscht anschließend die Shadowdatei.

```
root@sonne> grpunconv
```

Eventuell wird eine Konsistenzprüfung der beiden Dateien nötig. Das Kommando **grpck** gibt enthaltene Unstimmigkeiten aus und fordert ggf. zu Korrekturen auf:

```
root@sonne> grpck
group uucp: no user uucpfaxroot
delete member `uucpfaxroot'? n
grpck: no changes
```

Anlegen von Gruppen

Das Anlegen neuer Gruppen kann prinzipiell auf drei Wegen erfolgen:

- mittels reiner Handarbeit
- mit Hilfe der allgemeinen Werkzeuge
- durch Verwendung distributionseigener Verwaltungstools

Die Handarbeit

1. Der Systemverwalter bearbeitet die Datei /etc/group. Als Editor ist das Kommando **vigr** zu empfehlen (falls installiert), da sich das Kommando selbsttätig um die Dateisperre kümmert. Welcher Editor sich hinter dem Aufruf verbirgt, kann durch Setzen der Shellvariablen "$VISUAL" bzw. "$EDITOR" (Auswertung in dieser Reihenfolge) gesteuert werden. Erst wenn beide Variablen nicht gesetzt sind, wird auf den vi zurück gegriffen.

   ```
   root@sonne> vigr
   ...
   fibel::102:
   newgroup::103:
   ~
   ~
   ~
   -- INSERT --                    46,14      Bot
   ```

   Achten Sie auf die Eindeutigkeit von Gruppenname und -nummer; die beiden anderen Felder sollten frei bleiben.

2. Nun ist die Datei /etc/gshadow anzupassen. Falls vorhanden, kann das Kommando **grpconv** verwendet werden, das automatisch eine Angleichung der Einträge vornimmt. Ist grpconv nicht installiert, muss die Datei von Hand bearbeitet werden. **vigr -s** öffnet die Datei /etc/gshadow.

   ```
   root@sonne> vigr -s
   ...
   fibel::root:user
   newgroup::root:
   ~
   ~
   ~
   -- INSERT --                    46,14      Bot
   ```

   In die dritten Spalte ist der Gruppenverwalter einzutragen, die 4. Spalte kann eine kommaseparierte Liste der Gruppenmitglieder beinhalten.

3. Falls erforderlich, ist das Gruppenpasswort zu setzen:

   ```
   root@sonne> gpasswd newgroup
   Changing the password for group users
   New Password:
   Re-enter new password:
   ```

4. Die Gruppenmitglieder sind anzulegen (siehe Gruppenmitglieder).

Allgemeine Werkzeuge

Auf Systemen mit traditioneller Passwort-Verwaltung heißt das benötigte Kommando **addgroup**. Seine Bedienung ist analog dem nachfolgend diskutiertem **groupadd**, das auf Systemen mit Shadow-Passwort-Verwaltung eingesetzt wird. Letzteres Kommando nimmt automatisch die notwendigen Änderungen in der Datei /etc/gshadow vor.

Um eine neue Gruppe "newgroup" mit der Gruppennummer 111 anzulegen, gibt Root Folgendes ein:

```
root@sonne> groupadd -g 111 newgroup
```

Wird auf die Angabe einer Gruppennummer verzichtet, bekommt "newgroup" die nächste freie Gruppennummer zugewiesen. "groupadd" beendet seine Arbeit mit einer Fehlermeldung, falls eine schon verwendete Gruppennummer als Argument übergeben wurde. Ist eine doppelte Vergabe von GID's erwünscht, muss das dem Kommando mit der Option "-o" mitgeteilt werden:

```
root@sonne> groupadd -g 0 -o admins
```

Distributionseigene Werkzeuge

Die distributionseigenen Werkzeuge verpacken den Kommandozeilenaufruf in einen grafischen Dialog. Ihr Vorteil liegt oft in der Kopplung mehrerer Kommando in einer Maske, so ist es möglich, während des Anlegens einer Gruppe das Passwort zu setzen und gleichzeitig die Liste der Mitglieder anzulegen. Der Nachteil der Tools ist ihre meist auf nur ein System beschränkte Verfügbarkeit.

RedHat-basierende Distributionen administrieren die Gruppen über **userconf**.

SuSE hat die Gruppenverwaltung in **Yast** integriert (Administration des Systems → Gruppenverwaltung; Abbildung 1).

Abbildung 1: Gruppenverwaltung im Yast

Löschen von Gruppen

Die Handarbeit

Die Handarbeit läuft auf das manuelle Löschen der entsprechenden Einträge der Dateien /etc/group und /etc/gshadow hinaus.

1. Entfernen Sie den Gruppeneintrag aus der Datei /etc/group. Falls installiert, nutzen sie zum Bearbeiten das Kommando **vigr**.
2. Entfernen Sie den Gruppeneintrag aus der Datei /etc/gshadow. Dies kann durch Konvertierung mit Hilfe des Aufrufes von **grpconv** geschehen, oder durch Editieren der Datei (**vigr -s**). Falls beide Kommandos nicht auf dem System verfügbar sind, hilft jeder andere Editor.

3. Vergewissern Sie sich, dass kein Nutzer die entfernte Gruppe als Default-Gruppe in der Datei /etc/passwd eingetragen hat. Passen Sie solche Eintrage ggf. an (durch eine vorhandene Gruppe ersetzen).
4. Eventuell sollten Sie Dateien im System, die dieser Gruppe gehören, einer anderen Gruppe zuordnen (Siehe Systemverwaltung → Zugriffsrechte → Besitzer ändern) bzw. solche Dateien entfernen.

Allgemeine Werkzeuge

In Nicht-Shadow-Passwort-Systemen ist **delgroup** das Kommando. Dessen Bedienung erfolgt analog zum Kommando **groupdel**. Beide Kommandos erwarten als Argument einzig den Namen der zu löschenden Gruppe. "groupdel" entfernt zusätzlich den Eintrag aus der Shadow-Gruppendatei.

```
root@sonne> groupdel fibel
```

Distributionseigene Werkzeuge

Das Entfernen kann in den Tools **userconf** von RedHat bzw. **Yast** (Administration des Systems → Gruppenverwaltung) bei SuSE erfolgen.

Gruppenmitglieder

Die Handarbeit

In einem System mit Shadow-Passwort-Verwaltung werden die Nutzerkennzeichen der Mitglieder einfach in das vierte Feld des betreffenden Gruppeneintrags der Datei /etc/gshadow eingetragen. Die einzelnen Einträge sind durch Kommas voneinander zu trennen. Zum Bearbeiten der Datei sollte, falls installiert, das Kommando **vigr -s** verwendet werden.
Bei Systemen ohne Shadow-Passwort-Verwaltung wird die Nutzerliste in das vierte Feld des Eintrages der Datei /etc/gshadow geschrieben. Dort können die Nutzer zwar auch bei Shadow-Systemen stehen, doch ist diese Liste dann für jeden lesbar!

Allgemeine Werkzeuge

Zum Verwalten von Gruppenmitgliedern dient das Kommando **gpasswd**. Sowohl Root als auch von ihm ernannte Gruppenverwalter können mit Hilfe von "gpasswd"

Mitglieder hinzufügen

```
root@sonne> gpasswd -a user fibel
Adding user user to group fibel
```

Mitglieder entfernen

```
root@sonne> gpasswd -d user fibel
Removing user user from group fibel
```

Das Gruppenpasswort ändern

```
root@sonne> gpasswd fibel
Changing the password for group fibel
New Password:
Re-enter new password:
```

Das Gruppenpasswort löschen

```
root@sonne> gpasswd -r fibel
```

Alle Gruppenmitglieder entfernen

```
root@sonne> gpasswd -R fibel
```

Distributionseigene Werkzeuge

Ein Eintrag von Nutzern kann durch Editieren der Gruppeneinträge mit den Tools **userconf** von RedHat bzw. **Yast** (Administration des Systems → Gruppenverwaltung) bei SuSE erfolgen.

Gruppenwechsel

Der Wechsel einer Gruppe ist notwendig, falls:

- Der Nutzer kein ständiges Mitglied der Gruppe ist (dann muss er das Gruppenpasswort kennen)
- Alle neu anzulegenden Dateien automatisch der neuen Gruppe zugeordnet werden sollen
- Kommandos mit der Berechtigung einer anderen Gruppe ausgeführt werden sollen

Der Gruppenwechsel erfolgt mit dem Kommando **newgrp**:

Aufruf: newgrp [-] [group]

Das "-" als Argument bewirkt eine erneute Initialisierung aller Umgebungsvariablen inklusive einem eventuellen Verzeichniswechsel. Wird dem Kommando kein Gruppenname angegeben, wird in die Default-Gruppe (aus /etc/passwd) gewechselt. Ist ein Nutzer nicht zum Wechsel berechtigt, wird er zur Eingabe des Passwortes aufgefordert.

```
user@sonne> id
uid=500(user) gid=100(users) Gruppen=100(users),102(fibel)

user@sonne> newgrp fibel
user@sonne> id
uid=500(user) gid=102(fibel) Gruppen=100(users),102(fibel)

user@sonne> newgrp root
Password:
```

Um die effektive Gruppe nur während der Ausführung eines einzelnen Kommandos zu wechseln, kann auf das Kommando **sg** zurückgegriffen werden:

Aufruf: sg [-] [group [[-c] command]]

Die Angabe des Gruppennamens ist zwingend. Ein Aufruf ohne Kommando funktioniert zwar, bewirkt aber praktisch nichts.

```
user@sonne> sg - root ifconfig
Password:
```

Der Gruppenwechsel mittels newgrp kann vom (Gruppen)Administrator unterbunden werden:

```
root@sonne> gpasswd -R fibel
```

In die Gruppe "fibel" kann nun niemand mehr wechseln (alle Mitglieder mit Ausnahme von "root" wurden aus der Liste entfernt.).

Gruppenverwalter

Die Verwaltung aller Gruppen einer einzigen Person aufzubürden, kann diese schnell überfordern. Warum sollte die Mitglieder einer Gruppe, die gemeinsam an einem Programmierprojekt arbeiten, nicht auch der Projektleiter administrieren?

Das Einrichten und Entfernen von Gruppen wird immer Root anlasten, aber die Verwaltung dieser kann er delegieren:

```
root@sonne> gpasswd -A user fibel
```

Der Nutzer "user" ist nun berechtigt, Nutzer zur Gruppe "fibel" hinzuzufügen bzw. jene aus dieser Gruppe zu entfernen. Außerdem kann der Gruppenverwalter das Passwort einer Gruppe entfernen, so dass kein Nicht-Mitglied sich der Gruppe zuordnen kann:

```
user@sonne> gpasswd -r fibel
```

Root kann die Liste der Gruppenverwalter wieder löschen:

```
root@sonne> gpasswd -A "" fibel
```

Integration von Hardware

Übersicht

Das Dilemma mit Linux und Hardware ist, dass kaum ein Hersteller sich bislang um die Entwicklung und Auslieferung von Treibern scherte. Ja schlimmer noch, man verwehrte vielfach den freiwilligen Entwicklern Einblick in die interne Struktur. Und ohne Kenntnis der Parameter, ist ein brauchbarer Treiber nur schwerlich zu programmieren...

Auf lange Sicht ist Besserung in Sicht, die wenigsten Hardware-Lieferanten verschließen vollends die Augen vor dem wachsenden Linuxmarkt. Viele unterstützen tatkräftig die Open Source Gemeinde; andere drehen noch die Däumchen. Aber: »Wer zu spät kommt, den bestraft das Leben!«. Gute Hardware, die sich zum Zusammenspiel mit Linux überreden lässt, gibt es unterdessen zur Genüge. Notfalls sollte man vor dem Kauf einmal im Internet nach dem Status des gepriesenen Stücks Ausschau halten.

Das Wort **Treiber** machte schon die Runde. Hierbei handelt es sich um Software, die die Ansteuerung einer Hardwarekomponente regelt. Um einen Treiber verwenden zu können, muss der Linuxkernel für diesen vorbereitet sein. Entweder ist der Treiber fester Bestandteil des Kernels oder er liegt als so genanntes Modul bereit, um im Falle seines Einsatzes dynamisch den Kernel erweitern zu können. Auch im Fall des Modules muss der Kernel die Voraussetzungen erfüllen, um mit diesem zu Rande zu kommen. Eventuell ist also ein neuer Kernel zu generieren. Den Umgang mit Modulen erläutert der gleichnamige Abschnitt.

CD-Laufwerk & Brenner

ATAPI-CDROM

Im offiziellen Sprachgebrauch des *ANSI* (American National Standards Institute) bezeichnet man *IDE* (Integrated Drive Electronics) als *ATA* (Advanced Technology Attachment). *ATAPI* (ATA Packet Interface) ist also nichts anderes als ein Protokoll, das zum Anschluss von Massenspeichergeräten an einen IDE-Bus dient.

ATAPI-CDROM-Laufwerke - und aktuelle Nicht-SCSI-Modelle sind fast immer solche - gliedern sich analog zu Festplatten im System ein. Nach dem Einbau eines solchen Laufwerks ist schon an den Ausgaben des BIOS ersichtlich, ob das Gerät ordnungsgemäß erkannt wurde. Merken müssen Sie sich nur, ob das Laufwerk am ersten oder zweiten IDE-Kontroller hängt und ob es dort als Master oder als Slave fungiert. Beide Informationen sollten im BIOS Ihres Rechners zu finden sein.

Nach dem Start von Linux sollten Sie im Verzeichnis **/ dev** einen Link **cdrom** auf das entsprechende Device, an dem Ihr CDROM-Laufwerk hängt, anlegen. Es gibt einige Programme, die diesen Namen erwarten und außerdem merkt sich dieser leichter, als es der physische Devicename tut.

```
root@sonne> ln -s / dev/ hdd / dev/ cdrom
```

Falls das CDROM-Laufwerk nicht als Slave am 2.Controller (/dev/hdd) installiert ist, dann ersetzen Sie in obiger Kommandozeile /dev/hdd durch:

/ dev/ hda Master am 1. Kontroller

/ dev/ hdb Slave am 1. Kontroller

/ dev/ hdc Master am 2. Kontroller

Vor dem Zugriff auf die Daten, muss eine CD noch gemountet werden. Anzumerken ist noch, dass im Kernel die Unterstützung für »IDE/ATAPI CDROM« und das Dateisystem ISO9660 aktiviert sein muss; dies sollte aber in allen Standardkerneln der Fall sein.

SCSI-CDROM-Laufwerke

SCSI (Small Computer Systems Interface) erhitzt die Gemüter so mancher Computer-Enthusiasten. Wer es hat, belächelt diejenigen, die »IDE-Kinderkram« einsetzen. Im Vergleich zu IDE-Realisierungen sind SCSI-Komponenten allerdings extrem teuer und die Zeiten, da ein SCSI-Gerät entscheidend flotter die Daten übertrug, gehören spätestens seit UltraDMA der Vergangenheit an.

SCSI kann dennoch durchaus nützlich sein. Nämlich dann, wenn CD-Brenner, CDROM und mindestens drei Festplatten ins System sollen. Mit IDE-Geräten ist dies nicht ohne Weiteres zu realisieren, da heutige Mainboards nur zwei Kontroller mit je zwei Anschlüssen mit sich bringen. Über spezielle Kernelparameter kann Linux allerdings bis zu vier Kontroller ansprechen, aber wer eine solche Konfiguration benötigt, sollte dann doch besser gleich zu SCSI greifen.

Um auf SCSI-CDROM-Laufwerke zuzugreifen, muss der Kernel ein paar mehr Fähigkeiten enthalten. Und diese sind selten in den Standardkerneln integriert. Modellieren Sie also einen Kernel mit SCSI-Support, der darüber hinaus den Treiber für Ihren SCSI-Adapter, für die SCSI-CDROM-Unterstützung und das ISO9660-Dateisystem umfasst.

Wenn Sie nur ein SCSI-CDROM in Ihrem System installiert haben, erreichen Sie dieses über das Device **/ dev/ scd0**. Bei mehreren hängt die Namensvergabe zunächst vom SCSI-Bus ab, an dem das Laufwerk hängt. Jenes am ersten Bus reiht sich vor einem CDROM-LW am zweiten Bus ein usw. Sind mehrere Laufwerke an einem einzigen Bus angeschlossen, so entscheidet die SCSI-ID (am Gerät einstellbar), welches Laufwerk unter /dev/sdc0, welches unter /dev/scd1 usw. angesprochen wird. Sie sollten zur Vereinfachung wiederum einen Link unter dem Namen /dev/cdrom auf das Device des ersten CDROMs legen.

Weitere Laufwerkstypen

So genannte propriätere Laufwerke, wie sie vor einigen Jahren gern im Paket mit Soundkarten vertrieben wurden, findet man im Handel zum Glück nicht mehr. Dennoch sind derartige Exoten in dem einen oder anderen - zumeist betagteren - Rechner noch installiert. Jetzt muss man nicht gleich resigniert zum Schraubendreher greifen, denn die Chancen stehen günstig, dass ein solches Laufwerk wohlwollend von Linux behandelt wird.

In fast allen Fällen ist jedoch ein neuer Kernel - oder zumindest ein entsprechendes Modul - von Nöten. Weil es sich wirklich um seltene Hardwarekomponenten handelt, fehlt die Unterstützung solcher Laufwerke quasi in jedem Standardkernel. Basteln Sie sich in dem Fall einen neuen Kernel zusammen und wählen Sie die Option, die Ihrem Laufwerktyp am ehesten entspricht. Das Device, hinter dem sich später Ihr Laufwerk versteckt, trägt meist einen Namen, der in abkürzender Manier auf den Hersteller hinweist (/dev/aztcd (Aztek), /dev/sbpcd (Creative Labs, Soundblaster), /dev/mcd (Mitsumi), /dev/sonycd (Sony)...).

Nicht nur CDROM-Laufwerke werden häufig auch über die parallele Schnittstelle betrieben. Auch in dieser Situation wird es ohne einen neukompilierten Kernel nicht gehen. Dieser sollte den Treiber für Parallelport-IDE-Geräte als Modul ("paride") enthalten. Die Devicenamen sind /dev/pcd0, /dev/pcd1,...

Allgemeines zu CD-Brennern

Abgesehen von der Fähigkeit zum Brennen von CDs unterscheiden sich solche Geräte nicht von herkömmlichen Nur-Lese-Laufwerken. Nach einer Installation gemäß dem Vorgehen von CDROM-Laufwerken sollte das Lesen vom Laufwerk bereits funktionieren.

Um das Schreiben des Images vor dem eigentlichen Brennvorgang zu simulieren, ist es günstig, den Treiber für das Loopback-Device in den Kernel zu integrieren (wahlweise als Modul).

An dieser Stelle sollen nur die Programme genannt werden, die zum Brennen und für vorbereitende Maßnahmen verwendet werden können.

Um eine Daten-CD zu brennen, muss ein entsprechendes CD-Image erzeugt werden. Hierzu dient das Programm **mkisofs**. Eine Alternative ist **mkhybrid**. Die Tracks einer AudioCD vermögen die Programme **cdparanoia** oder **cdda2wav** auszulesen. Während die gelesenen Daten des ersteren Tools unverändert auf eine CD gebrannt werden können, muss das Ausgabeformat von **cdda2wav** zunächst gewandelt werden. Ein Programm, das verschiedene Audioformate in die richtige Form bringt, ist **sox**. Formate von VideoCDs vermag das Werkzeug **mkvcdfs** zu erzeugen.

Den eigentlichen Brennvorgang können Sie mit einem der Programme **cdrecord** oder **cdrdao** vornehmen.

Grafische Verpackungen für **mkisofs** und **cdrecord** bieten u.a **xcdroast** (X), **krabber**, **keasycd** (KDE) und **gcombust** (Gnome).

SCSI-CD-Brenner

Gehen Sie zur Installation analog vor, wie bei einem SCSI-CDROM-Laufwerk. Lässt sich das Gerät nachfolgend wie ein gewöhnliches CDROM-LW ansprechen, sind bereits alle Vorarbeiten geleistet.

IDE-CD-Brenner

Alle weiter oben genannten Brennprogramme setzen einen SCSI-CD-Brenner voraus. Leider hat sich noch kein Entwickler die Bürde auferlegt, die Linuxgemeinde mit Treibern für IDE-Brenner zu beglücken. Vielleicht auch deshalb, weil ein solcher Treiber niemals wirklich notwendig wurde, um auch mit einem IDE-Gerät die CD zu toasten. Denn schon seit geraumer Zeit gehört eine SCSI-Emulation zum Funktionsumfang des Kernels (ab dem Kernel 2.6 gehört ein ATAPI-CDROM-Brenner-Treiber direkt zum Kernel).

Also macht sich (in den meisten Fällen) wieder einmal eine Kernelübersetzung notwendig. Achten Sie auf die Aktivierung folgender Punkte (wenn nicht anders angegeben, *kann* eine Option als Modul realisiert werden):

Enhanced IDE/MFM/RLL disc/cdrom...	**Aktivieren, kein Modul**
IDE/ATAPI CDROM support	**Aktivieren als Modul**
SCSI emulation support	**Aktivieren als Modul**

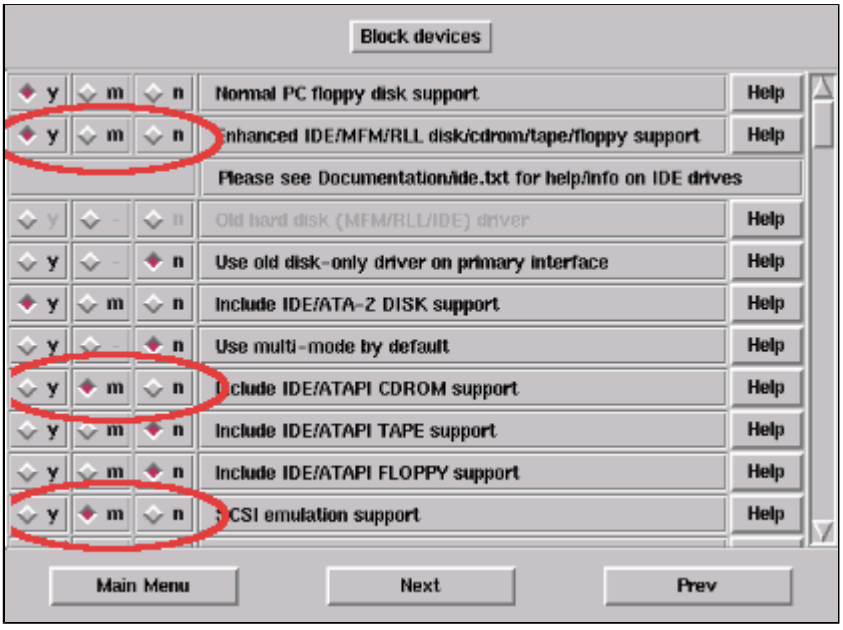


Abbildung 1: Kerneleinstellungen: SCSI-Emulation für IDE-Brenner

SCSI support	Aktivieren
SCSI CDROM support	Aktivieren
Enable vendor-specific extensions	Aktivieren
SCSI generic support	Aktivieren
ISO 9660 CDROM filesystem support	Aktivieren (nicht abgebildet)
Microsoft Joliet cdrom extensions support	Optional (nicht abgebildet)

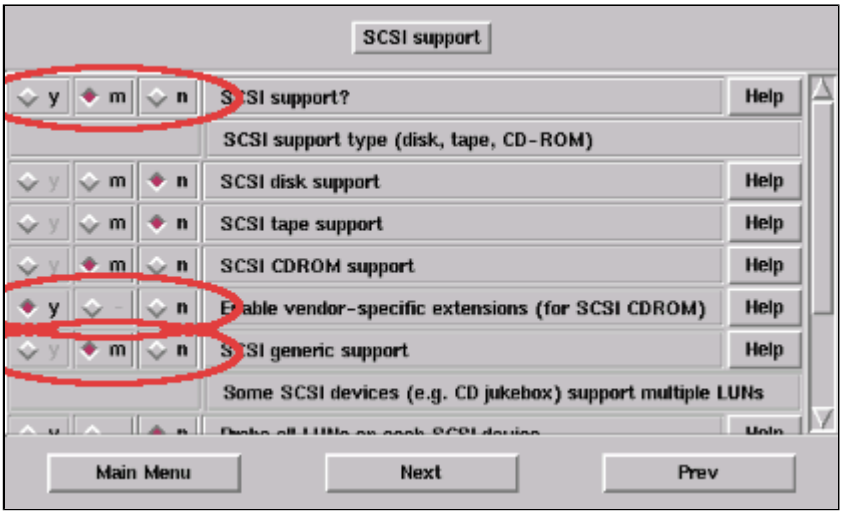


Abbildung 2: Kerneleinstellungen: SCSI-Optionen für IDE-Brenner

In einem IDE-System müssen die Treiber für IDE fest in den Kernel eingebunden werden. Das würde nun dazu führen, dass beim Booten der CD-Brenner schon als IDE-Gerät initialisiert werden würde, noch bevor die SCSI-Emulation überhaupt zum Zuge kommt. Um dies zu verhindern, ist zum Zeitpunkt des Bootens dem Kernel dies mitzuteilen. Automatisieren lässt sich dies durch einen entsprechenden Eintrag in die Datei des Bootmanagers (im Beispiel handelt es sich um den Lilo):

```
root@sonne> vi /etc/lilo.conf
...
image = /boot/bzImage
root = /dev/hda6
label = kernel2_4_2
append = "vga=normal hdc=ide-scsi hdd=ide-scsi"
...
root@sonne> lilo
added kernel2_4_2
...
```

Genau jene Geräte, die nicht vom IDE-Treiber angesprochen werden sollen, müssen durch einen append-Eintrag »<Gerät>=ide-scsi« gekennzeichnet werden. Im Beispiel handelt es sich um beide Geräte am 2. IDE-Kontroller; ein CD-ROM und ein Brenner. Ein CDROM-Laufwerk ist nicht zwingend erforderlich, da ein Brenner ebenso die Daten auslesen, das Images zunächst auf die Platte schreiben und anschließend auf den Rohling brennen kann. Für ein direktes Kopieren muss das CDROM-Laufwerk ebenso als SCSI-Gerät vorhanden sein!

Je nach Kernelkonfiguration müssen Sie nun die Module laden:

cdrom	CDROM-Unterstützung
sr_mod	SCSI-CDROM
sg	SCSI generic support
ide-scsi	SCSI-Emulation

cdrom muss vor **sr_mod** geladen werden; die restliche Reihenfolge ist unerheblich.

Der ganze Ladevorgang reduziert sich auf einen Aufruf von **modprobe ide-scsi**, falls die

Datei **/ etc/ modules.conf** um folgende Einträge ergänzt wird:

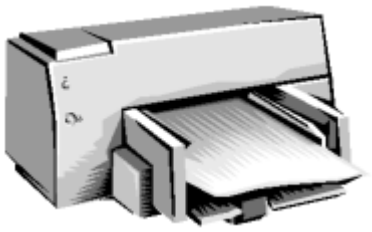
```
root@sonne> vi / etc/ modules.conf
...
pre-install sg modprobe ide-scsi
pre-install sr_mod modprobe ide-scsi
```

Falls der anschließende Aufruf von »modprobe« erfolgreich verlief, sollten CD-Brenner und - falls vorhanden - das CDROM-Laufwerk im Prozessdateisystem auftauchen:

```
root@sonne> modprobe ide-scsi
root@sonne> cat / proc/ scsi/ scsi
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor:         Model: ATAPI CDROM      Rev: 130P
  Type:   CD-ROM                      ANSI SCSI revision: 02
Host: scsi0 Channel: 00 Id: 01 Lun: 00
  Vendor: TEAC    Model: CD-W54E          Rev: 1.1B
  Type:   CD-ROM                      ANSI SCSI revision: 02
```

Integrieren Sie den modprobe-Aufruf in eines der System-Startskripte, so sollte beim folgenden Booten des Rechners eine Meldung in der Art von »hdd: ... enabling SCSI emulation« erscheinen. Zudem sollten »scsi0 : SCSI host adapter emulation for IDE ATAPI devices« und eine den konkreten Brenner beschreibende Zeile zu sehen sein.

Drucker



Selten offenbart Linux seine Abstammung von Unix so deutlich wie in der Handhabung von Druckern. Unix, das man mit gutem Gewissen Analogien zu Netzwerkbetriebssystemen bescheinigen kann, ist von Haus aus darauf getrimmt, Ressourcen im Netzwerk zu teilen. So arbeitet das Drucksystem, das in portierter Form auch auf Linux heimisch wurde, transparent im Netzwerk. Das Problem des Einrichtens eines Druckers reduziert sich auf die lokale Anbindung am Rechner, der Schritt zum Netzwerkdrucker ist trivial.

Reduziert suggeriert Einfachheit, doch mit der einen Ausnahme von Postscript-Druckern (Postscript ist eine von Adobe eingeführte Seitenbeschreibungssprache) ist die Konfiguration alles andere als ein Kinderspiel. Doch Entwarnung! Fast jede Distribution beinhaltet Administrationshilfen, die das Typenrad gängiger Druckermodelle mit geringem Aufwand in Schwung versetzen.

Nach wie vor verwaltet per Voreinstellung in nahezu allen Distributionen der **Line Printer Daemon** (Lpd) die Druckaufträge im System. Dessen Entstehung datiert allerdings schon um einige Jahrzehnte zurück. Anno dazumal waren Typenraddrucker das Nonplusultra, später wurden für die gehobenen Ansprüche die Postscript-Drucker verfügbar. Kein Wunder also, dass der Lpd von Haus aus nur mit diesen gut zu Rande kommt.

Der Line Printer Daemon ist nur eines der so genannten Spooling-Systeme, die unterdessen für Linux verfügbar sind. »Spooling« deshalb, weil die Druckaufträge nicht direkt von einer Anwendung zum Drucker gelangen, sondern in einer Druckerwarteschlange zwischengespeichert werden. Für jeden Drucker im System - dabei spielt es keine Rolle, ob dieser am lokalen Rechner oder irgendwo im Netzwerk angeschlossen ist - existiert mindestens eine Warteschlange. Üblich sind sogar mehrere Warteschlangen für jeden lokalen Drucker, da häufig ein bestimmter Filter einer Warteschlange zugeordnet wird. Aufgabe eines solchen »Eingabefilters« ist die Aufbereitung der zu druckenden Daten in ein für den Drucker verständliches Format. Derartige Filter existieren zuhauf, der bekannteste und verbreiteste dürfte »Apsfilter« (wird nicht bei CUPS verwendet) sein, der Dateitypen automatisch erkennt und seinerseits einen konkreten Filter aufruft. Für die Wandlung des Postscript-Formats, was die meisten Anwendungen, die das Drucken vorsehen, erzeugen, ist »Ghostscript« der relevante Filter. Er soll uns noch beschäftigen.

Als Alternative zum **Line Printer Daemon** kam in einer neu gefassten Implementierung später der **LPRpn** hinzu (LPRng wird hier nicht betrachtet werden), der ähnliche Eigenschaften wie der Lpd aufweist, allerdings auch die meisten seiner Schwächen. Ein gänzlich anderes Drucksystem ist **PDQ** (Print, don't queue), das ohne ein Spooling System auskommt. Auch dieser Variante werden wir nicht weiter folgen. Neben dem Lpd richten wir unser

Augenmerk auf **CUPS** (Common Unix Printer System), das 1999 unter die GPL gestellt wurde und damit eine umfassende Alternative zum LPD darstellt. Vermutlich wird CUPS den Lpd eines Tages komplett verdrängen.

Der Druckeranschluss

Bevor Sie hier weiter lesen, sollten Sie im Handbuch Ihres Druckers nachsehen, ob es sich um einen so genannten GDI-Drucker handelt. In diesem Fall werden Sie mit den üblichen Methoden keine Druckergebnisse erzielen können. Wir gehen auf diese Modelle auch nicht weiter ein.

Die wohl meisten Drucker heutiger Bauart werden entweder an einer parallelen Schnittstelle des Rechners angeschlossen oder an einem USB-Port. Seltener findet man noch Modelle, die an der seriellen Schnittstelle betrieben werden.

Je nach verwendetem Druckertyp muss der Kernel die entsprechenden Treiber parat halten. Parallele und serielle Schnittstellen sind in den meisten Konfigurationen frei geschaltet, sodass eine Kernel-Neuübersetzung höchstens zur Anbindung von USB-Druckern in Frage kommt. Letzteres Vorgehen ist im Abschnitt zu USB-Unterstützung beschrieben.

Ist der entsprechende Treiber erst einmal geladen, kann ein erster Test nicht schaden. Ein rudimentäres Verfahren ist das direkte Senden eines Textes an die Schnittstelle:

```
root@sonne> echo -e "Hello\ f" > /dev/lp0
```

Im Beispiel sollte ein Drucker an der ersten parallelen Schnittstelle loslegen. Es muss allerdings nicht zwingend einen Fehler bedeuten, wenn der Drucker sich nicht rührt. Denn einige Modelle sind nicht in der Lage, einfachen ASCII-Text aufs Papier zu bringen. In der nachfolgenden Abhandlung zu Ghostscript erfahren Sie, wie Sie Druckausgaben für derartige Drucker generieren können.

Fehlersuche

Angenommen es handelt sich um eine Fehlkonfiguration. Dann geht es an die Fehlersuche. Erster Anlaufpunkt sind die Meldungen des Bootvorgangs, die Sie mit dem Kommando **dmesg** nochmals auf den Bildschirm bannen. Von Interesse sind Zeilen, die einen Hinweis auf die parallele Schnittstelle geben (für serielle Drucker gelten natürlich die seriellen Schnittstellen; für USB-Drucker sind die jeweiligen Module interessant):

```
user@sonne> dmesg | egrep 'lp|parport'
parport0: PC-style at 0x378 [SPP]
lp0: using parport0 (polling).
```

Die wichtigste Information enthält die mit **lp** beginnende Zeile. Im Beispiel ist »lp0« auf »parport0« gebunden. »parport0« selbst verwendet als IO-Adresse 0x378 und ist auf Polling eingestellt. Polling bedeutet, dass die Schnittstelle regelmäßig vom Kernel getestet wird, ob neue Daten anliegen. Ein solches Verfahren ist natürlich weniger effizient als die Steuerung via Interrupts. Falls die Schnittstelle Interrupts unterstützt, sollte sie darauf umgestellt werden:

```
# Falls lp0
root@sonne> echo "7" > /proc/parport/0/irq
```

Der eingestellte Interrupt muss mit den Einstellungen im BIOS übereinstimmen! Um zurück zum Polling-Verfahren zu wechseln, schreiben Sie »none« in obige Datei.

/proc/parport existiert bei Ihnen nicht? Dann verwenden Sie eventuell noch einen alten Kernel, der die Verwendung einer parallelen Schnittstelle durch mehrere Geräte noch nicht unterstützt. In den Bootmeldungen hätte dann etwa folgende Zeile gestanden:

```
user@sonne> dmesg | egrep 'lp|parport'
lp0: at 0x378 (polling).
```

Den Interruptbetrieb für eine solche Schnittstelle aktivieren Sie mit **tunelp**:

```
# Falls lp0
root@sonne> tunelp /dev/lp0 -i 7
```

Wieder ist wichtig, dass Sie keinen von den BIOS-Einstellungen abweichenden Interrupt einschalten. Zum Deaktivieren verwenden Sie als Interruptnummer die »0«.

Falls aber in den Bootmeldungen keinerlei Hinweise auf die parallelen Schnittstellen auftauchen, so wurde entweder das entsprechende Treibermodul noch nicht geladen oder im Kernel fehlt die Unterstützung. In letzteren Fall müssen Sie einen neuen Kernel übersetzen. Die für die parallele Schnittstelle notwendigen Optionen (günstig als Modul) nennen sich:

- Parallel port support
- PC-style Hardware
- Parallel printer support
- PLIP (parallel support) **nicht oder höchstens als Modul!**

Und wenn Sie schon beim Kernelbau sind, so sollten Sie die Aktivierung von **TCP/IP** nicht vergessen, da der Lpd diese benötigt.

Bevor Sie sich an den Kernel wagen, sollten Sie prüfen, ob nicht vielleicht die Module bereits vorhanden sind. Wenn ja, dann liegen sie im Verzeichnis **/lib/modules/*Kernelversion*/misc/** und heißen:

```
lp.o
parport.o
parport_pc.o
parport_probe.o
```

Bei einem älteren Kernel existiert einzig »lp.o«. Bei korrekter Beschreibung in der Datei module.conf genügt ein Aufruf von:

```
root@sonne> modprobe lp
```

In der Ausgabe von **lsmod** sollten alle 4 oben genannten Module erscheinen. Klappt es nicht, können die Module auch von Hand geladen werden, wobei die Reihenfolge wichtig ist:

```
root@sonne> insmod parport
Using /lib/modules/2.2.16-22/misc/parport.o
root@sonne> insmod parport_probe
Using /lib/modules/2.2.16-22/misc/parport_probe.o
root@sonne> insmod parport_pc
Using /lib/modules/2.2.16-22/misc/parport_pc.o
root@sonne> insmod lp Using /lib/modules/2.2.16-22/misc/lp.o
```

Führen obige Versuche nicht zum Erfolg, so können Sie die Treiber mit konkreten Vorgaben für Interrupt und IO-Adresse laden (weiter gehende Informationen enthält der Abschnitt Kernel-Module). Und bevor Sie verzweifeln... - prüfen Sie das Kabel (nicht jedes ist geeignet!) und die Verbindungen.

Das Sprachentalent Ghostscript

Ghostscript kommt immer dann ins Spiel, wenn es gilt, Postscript-Dokumente auf nicht Postscript-fähigen Druckern auszudrucken. Praktisch gibt es kaum eine verbreitete Anwendung, die nicht ihre Druckausgaben im Format Postscript erzeugt.

Die Parameter von Ghostscript sind verwirrend vielgestaltig und sollen uns hier nur insofern interessieren, wie sie zum Verständnis der Arbeitsweise des Filterprogramms erforderlich sind. Zum letzten Feinschliff kommen Sie ohnehin nicht umhin, die ausführliche Dokumentation bez. des für Ihren Drucker zuständigen Ghostscript-Treibers

zu wälzen (aktuelle Distributionen bringen zum Glück brauchbare Voreinstellungen mit, sodass Sie vermutlich nicht mit Ghostscript direkt in Berührung kommen).

Der direkte Umgang mit Ghostscript ist zumindest bei der Fehlersuche während der Druckerinstallation nützlich. Generieren Sie hiermit das für einen konkreten Drucker bestimmte Ausgabeformat und senden die so aufbereiteten Daten zur Druckerschnittstelle, so können Sie zumindest den Drucker als Fehlerquelle ausschließen, wenn ein Ausdruck nicht gelingt (es sei denn, der Drucker ist defekt). Auch die Korrektur von Papierformaten, Ausrichtung, Auflösung usw. ist schnell mit Ghostscript getestet.

Gs interaktiv

Das zuständige Programm **gs** kann sowohl interaktiv verwendet als auch über die Kommandozeile gesteuert werden:

```
user@sonne> gs
GNU Ghostscript 5.50 (2000-2-13)
Copyright (C) 1998 Aladdin Enterprises, Menlo Park, CA. All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
GS> (/usr/share/ghostscript/5.50/examples/tiger.ps) run
>>showpage, press <return> to continue<<[Enter]
GS> quit
user@sonne>
```

Wird **gs** ohne Angabe des Ausgabegerätes aufgerufen, verwendet das Programm das Standard-Ausgabegerät. Dabei handelt es sich um das erste Device der Liste, die der Aufruf von **gs -h** hervorbringt (im Beispiel ein X-Fenster; zur Konsolenausgabe muss ein entsprechender Ghostscript mit SVGA-Unterstützung installiert werden).

gs interpretiert der Reihe nach die per Kommandozeile übergebenen Postscript- oder PDF-Dateien und erzeugt die Ausgabe des eingestellten Devices. Wurde keine Datei angegeben oder wurden alle Dateien bearbeitet, geht **gs** in den interaktiven Modus über. In obigem Anwendungsbeispiel wird das manuelle Laden einer Datei und das abschließende Beenden des Interpreters vollzogen. Das Ausgabefenster zeigt Abbildung 3.

Abbildung 3: Ausgabefenster des Programms »gs«

Auswahl des Treibers

Um **gs** zu veranlassen, das für Ihren Drucker verständliche Ausgabeformat zu erzeugen, müssen sie den

gewünschten Treiber angeben. Schnellste Möglichkeit geht über die Kommandozeilenoption »-sDEVICE= *Devicename*«.

Und welcher Devicename? Die möglichen Namen, es wurde bereits erwähnt, erhalten Sie durch Aufruf von »gs -h«. In etlichen Fällen deuten die Kürzel den unterstützten Drucker an, aber eben nicht immer. Suchen Sie in den Dateien Ihrer Ghostscript-Installation nach »Devices.htm« (manchmal auch »Devices.txt«) bzw. »catalog.devices«, die ausführlichere Listen der durch einen Treiber unterstützten Drucker enthalten. Oft können Sie für einunddenselben Drucker aus mehreren Treibern wählen. Hier sind Versuche angesagt, welcher Treiber die besten Resultate erzeugt.

Um bspw. die obige Datei »tiger.ps« für einen Epson-Drucker aufzubereiten, ist folgende Kommandozeile aufzurufen:

```
user@sonne> gs -sDEVICE=epson /usr/share/ghostscript/5.50/examples/tiger.ps -c quit
GNU Ghostscript 5.50 (2000-2-13)
Copyright (C) 1998 Aladdin Enterprises, Menlo Park, CA. All rights reserved.
This software comes with NO WARRANTY: see the file COPYING for details.
»showpage, press <return> to continue«
```

»-c quit« am Ende bewirkt, dass **gs** sich sofort beendet, nachdem die erfolgte Bearbeitung mit [Enter] bestätigt wurde. Die Ausgabe hat **gs** in eine temporäre Datei im Verzeichnis /tmp geschrieben. Natürlich ist das nicht das von uns erwünschte Verhalten. Deshalb sollte die Ausgabe per Option »-sOutputFile= *Datei*« umgeleitet werden. Um auch die Statusmeldung von **gs** zu unterdrücken, nutzen wir die Option »-q«. Das Warten auf Bestätigung der Auftragsabarbeitung verhindert »-dNOPAUSE«. Somit lautet die komplette Zeile zum Erzeugen wie folgt:

```
user@sonne> gs -q -dNOPAUSE -sDEVICE=epson -
sOutputFile=test.epson /usr/share/ghostscript/5.50/examples/tiger.ps -c quit
```

Zum Test sollten Sie als Root die Datei nun an die Schnittstelle des Druckers senden (das Beispiel geht wiederum von einem Drucker an der ersten parallelen Schnittstelle aus):

```
root@sonne> cat test.epson > /dev/lp0
```

Drucken mit dem Lpd

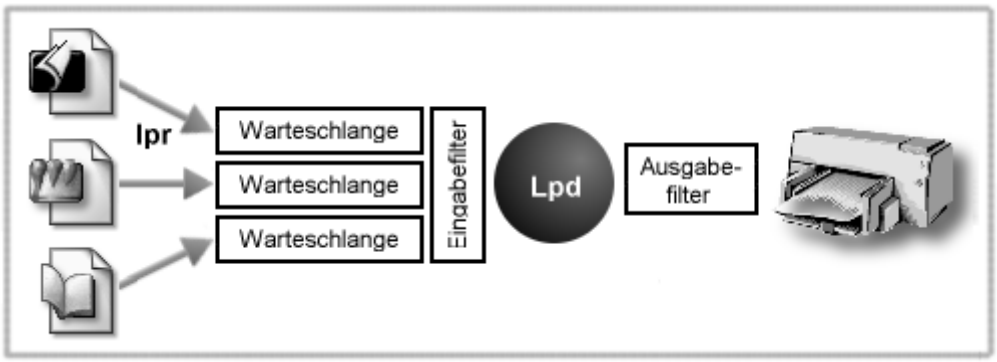


Abbildung 4: Arbeitsweise des Lpd

Der **Line Printer Daemon** wird i.d.R. während des Bootvorgangs gestartet. Seine Aufgabe ist die Verwaltung der Druckaufträge. Einmalig während des Starts bezieht der Lpd seine Konfigurationsdaten aus der Datei /etc/printcap. Diese Datei enthält die Beschreibungen sämtlicher Druckerwarteschlagen. Initial durchsucht der Daemon alle Spoolverzeichnisse, ob diese unbearbeitete Druckaufträge enthalten. Wird ein Auftrag entdeckt, erzeugt der Lpd einen Kindprozess, übergibt diesem diesen Auftrag und begibt sich in den Wartezustand, um erneute Aufträge entgegen nehmen zu können. Der neue Lpd-Prozess bereitet die zu druckende Datei geeignet auf - wie, das steht ebenso in den Definitionen von /etc/printcap - und sendet die Datei zum Drucker. Nach Auftragserledigung beendet sich der Kindprozess von selbst. In Abbildung 4 ist das Verfahren stark vereinfacht skizziert.

Der Druckerdaemon selbst gönnt den Spoolverzeichnissen nach getaner Initialisierung aus eigenem Antrieb heraus

keinen Blick mehr. Er überwacht (falls konfiguriert) den TCP-Port 512 auf neue Aufträge aus dem Netzwerk bzw. den Socket /dev/printer auf Benachrichtung über lokale Auftragseingänge.

Das Erzeugen von Druckaufträgen geschieht mit dem Kommando lpr. Selbst hinter den Druckknöpfen in den grafischen Anwendungen verbirgt sich nichts anderes als ein Aufruf von lpr mit geeigneten Parametern. Lpr schaut nun selbst in der Datei /etc/printcap nach, ob der entweder auf der Kommandozeile angegebe oder der voreingestellte Drucker ($PRINTER) existiert und ermittelt aus dem zugeordneten Eintrag das richtige Spoolverzeichnis. Lpr kopiert (oder verlinkt [Option -s]) die zu druckende Datei zusammen mit einer Statusdatei in das Spoolverzeichnis. Die Statusdatei enthält u.a. die UID, GID des den Auftrag initiierenden Benutzers sowie ggf. Drucksteueroptionen, die per Kommandozeilenoptionen angegeben wurden. Nach den Vorarbeiten informiert Lpr den Daemon über den Socket /dev/printer, dass Arbeit auf ihn wartet.

Die Kommandos **lpr** (Drucken), **lprm** (Druckauftrag löschen), **lpq** (Druckerwarteschlange anzeigen) und **lpc** (Druckersteuerung) sind im Abschnitt Nutzerkommandos, Dateien und Verzeichnisse beschrieben.

Die Datei »/etc/printcap«

Die Datei »/etc/printcap« ist das Herz der Lpd-Konfiguration. Jede Zeile, die nicht mit dem Kommantarzeichen # beginnt, beschreibt einen Drucker im System. Der erste Eintrag einer Zeile muss mindestens einen Namen für den definierten Drucker enthalten, die weiteren Parameter lassen sich, jeweils in Doppelpunkte eingeschlossen, in beliebiger Reihenfolge anordnen. Um zu lange Zeilen zu vermeiden, können diese durch einen abschließenden Backslash \ umgebrochen werden:

```
# Durch den abschließenden Backslash werden die 3 Zeilen zu einer zusammengefasst
lp:\
  :lp=/dev/lp1:\
  :sd=/var/spool/lp:
```

Zuweisungen von Zeichenketten an Variablen werden mittels des Gleichheitszeichens vorgenommen (bspw: lp=/dev/lp1). Für nummerische Werte hingegen dient # als Zuweisungsoperator (bspw: mx#0).

Bevor wir konkrete Beispieleinträge diskutieren, fasst folgende Tabelle alle gebräuchlichen Parameter zusammen (insbesondere verzichten wir auf die Erläuterung etlicher Parameter zur Konfiguration einer seriellen Schnittstelle). Ein dem Namen folgendes »=« steht für einen Zeichenkettenparameter; ein »#« kennzeichnet einen nummerischen Parameter. Ein Parameter ohne Wert steht für sich allein:

af=

Name einer Datei mit Abrechnungsdaten zum Druckauftrag. Den Inhalt kann ggf. ein Eingabefilter füllen.

br#

Ein Muss-Parameter, falls der Drucker an einer seriellen Schnittstelle hängt. Dient der Konfiguration der Baudrate (Datenübertragungsgeschwindigkeit).

ff=

Zeichenkette, die den Drucker zu einem Seitenvorschub veranlasst; Voreinstellung ist \f.

fo

Ein Seitenvorschub wird vor jedem neuen Ausdruck vorgenommen.

hl

Die Titelseite wird nach dem Druckauftrag ausgegeben (Voreinstellung ist zuvor).

lf=

Name der Logdatei, in die Fehlermeldungen geschrieben werden. Fehlt die Angabe, werden Fehler auf /dev/console gemeldet; anderenfalls muss die Datei existieren und vom Lpd beschrieben werden könne

(also Eigentümer und Gruppe auf »lp«).

lo=

Name der Lockdatei, die einen in Arbeit befindlichen Auftrag kennzeichnet.

lp=

Dieser erforderliche Parameter gibt das Device an, an dem der Drucker angeschlossen ist. Bezieht sich der Eintrag auf einen Netzwerkdrucker (vergleiche rm=, so muss dieser Parameter leer (aber vorhanden) sein!

if=

Gibt den zu verwendenden Eingabefilter an. Siehe nachfolgende Diskussion zu apsfilter.

mx#

Maximale Größe einer Datei im Spoolverzeichnis. mx#0 hebt jede Beschränkung auf.

of=

Name des Ausgabefilters. Ein solcher Filter kann bspw. zum Erzeugen einer Statusseite, die vor jedem Druc ausgegeben wird verwendet werden (Rechnung). Ist of belegt, if aber nicht, so wird of einmalig bei Start d Lpd angewendet (also nur für den ersten Druckauftrag).

pc#

Preis einer Seite in 1/100 Cent.

pl#

Anzahl Zeilen pro Seite (nur für Drucker, die den Zeichenmodus beherrschen)

pw#

Anzahl Zeichen pro Zeile (nur für Drucker, die den Zeichenmodus beherrschen)

px#

Anzahl Pixel in x-Richtung (für Bitmap-Grafiken)

py#

Anzahl Pixel in y-Richtung (für Bitmap-Grafiken)

rg=

Nur Benutzer der angegebenen Gruppe dürfen diesen Drucker benutzen.

rm=

Name oder IP-Adresse des entfernten Rechners, an dem der Netzwerkdrucker angeschlossen ist (lp= muss dem Fall frei bleiben!).

rp=

Name des Druckers an einem entfernten Rechner, die Voreinstellung ist der Standarddrucker »lp«.

rs

Den Drucker darf ein Benutzer nur verwenden, wenn er auf dem Rechner, an dem der Drucker angeschloss

ist, über einen Zugang verfügt.

rw

Öffnet das Druckerdevice zum Lesen und Schreiben. Nur sinnvoll, wenn der Drucker bspw. Statusmeldunge zurück senden kann.

sd=

Der Name des Spoolverzeichnisses. Dieses muss angegeben werden, falls es sich nicht um /var/spool/lpd handelt!

sh

Unterdrückt die Ausgabe einer Titelseite vor jedem Druckauftrag.

sc

Verhindert den Ausdruck mehrerer Kopien eines Auftrags.

In der Manual Page zu **printcap** sind weitere Parameter beschrieben, die als Filter für bestimmte Dateitypen dienen. Sie sollen uns nicht interessieren.

Nachfolgend sollen kurze Beispiele die Syntax der printcap-Einträge erläutern. Den Anfang macht eine einfache Konfiguration für einen lokalen Drucker. Dass dieser als Standarddrucker dient, zeigt der Name »lp«, unter dem der Drucker u.a. im System bekannt ist (»lp« sollte genau einmal in der printcap als Druckername auftauchen!). Die optionalen weiteren Namen »brother«, »postscript« sind typische Aliasse, die zumeist auf die Fähigkeiten oder das konkrete Modell hinweisen. Alle Druckernamen werden durch ein | voneinander getrennt angegeben und jeder Druckername darf nur einmal in der printcap-Datei vergeben werden. Einer der Druckernamen muss mit dem Namen des Spoolverzeichnisses übereinstimmen.

Der lokale Drucker im Beispiel ist an der ersten parallelen Schnittstelle angeschlossen, sein Spoolverzeichnis ist /var/spool/lp/brother, worin ebenso die Protokoll- (lf=) und Abrechnungsdaten (af=) gespeichert werden. Zu druckende Daten werden durch ein Skript »/var/spool/lp/brother/mein_filter« aufbereitet.

```
# Lokaler Drucker mit mehreren Namen »lp«, »brother«, »postscript« an /dev/lp1 (LPT1)
lp|brother|postscript:\
  :lp=/dev/lp0:\
  :sd=/var/spool/lpd/brother:\
  :lf=/var/spool/lpd/brother/log:\
  :af=/var/spool/lpd/brother/acct:\
  :if=/var/spool/lpd/brother/mein_filter:\
  :mx#0:\
  :sh:
```

Der nächste Eintrag beschreibt einen Netzwerkdrucker. Am entfernten Rechner wird der Standarddrucker (lp) angesprochen. Lokal wird nur die Protokollierung und Abrechnung vorgenommen; auf eine lokale Aufbereitung der Daten durch einen Eingabefilter wird verzichtet. Das ist Sache des entfernten Drucksystems.

```
# Remote-Drucker am Rechner 192.168.100.200, Warteschlange lp
lp-remote:\
  :lp=:\
  :rm=192.168.100.200:\
  :rp=lp:\
  :lf=/var/spool/lp-remote/log:\
  :af=/var/spool/lp-remote/acct:\
  :if=:\
  :mx#0:\
  :sh:
```

Der Eintrag des letzten hier diskutierten Beispiels stammt aus einem typischen Setup mit »apsfilter«, dem am häufigsten eingesetzen Eingangsfilter für Lpd-basierte Druckdienste unter Linux. Bis auf die Namensgebung

besteht kein Unterschied zum zuvor beschriebenen lokalen Drucker. Dass die Namensgebung einen tieferen Sinn unterliegt, erfahren Sie im anschließenden Abschnitt zu »apsfilter«.

```
# Ein von apsfilter erzeugter Eintrag zur automatischen Konvertierung von ASCII-Dateien ins Druckerformat Laserjet 4
ascii|lp|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:\
  :mx#0:\
  :sh:sf:
```

Apsfilter - Beispiel eines Eingabefilters

Apsfilter ist der bekannteste Vertreter der so genannten »magischen Filter«. Ein solcher Filter versucht das Format der Eingabedateien anhand typischer Muster zu erkennen und präpariert die Ausgabe gemäß seiner Erkenntnisse.

So manche Distribution nimmt Ihnen das Einrichten des Drucksystems - und damit der benötigten Filter - ab; wir betrachten hier dennoch das manuelle Vorgehen zur Konfiguration des Apsfilters (die in den Abbildungen gezeigte SuSE-Version unterscheidet sich nur unwesentlich vom Original).

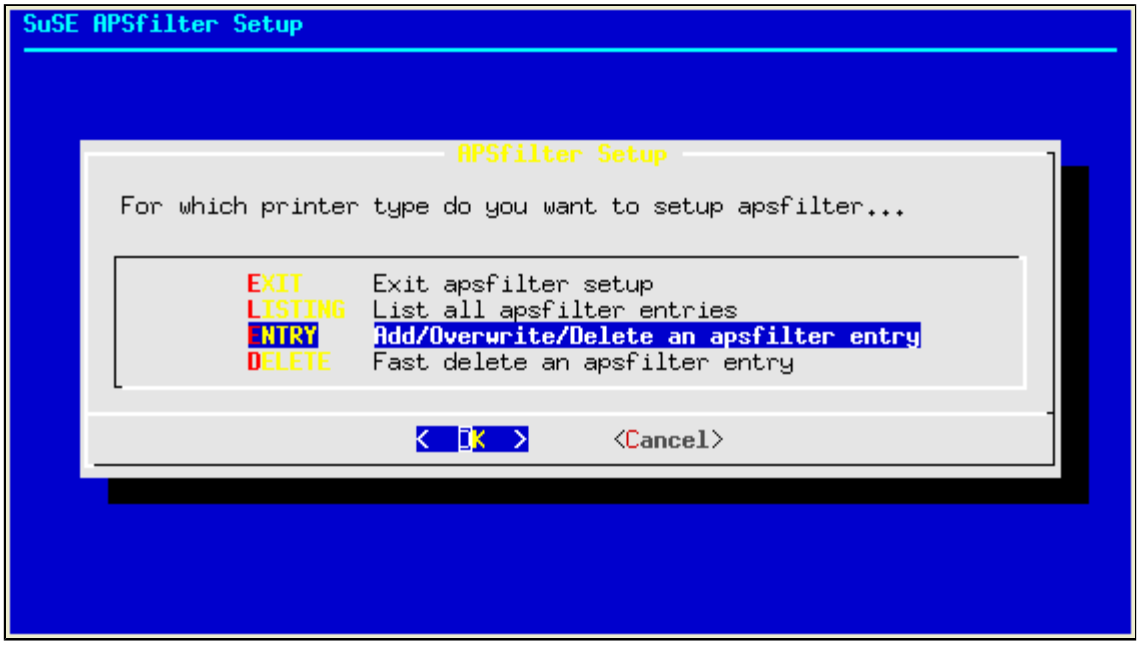


Abbildung 5: Hauptmenü des Apsfilter-Setups

Im Paket finden Sie eine Datei *SETUP* (je nach Installationsverzeichnis in /var/lib/apsfilter oder /usr/lib/apsfilter). Dieses Dialog basierte Programm ist letztlich ein komfortables Frontend zum Einrichten der printcap-Datei.

Wenn Sie *SETUP* als Root starten, gelangen Sie nach einem Begrüßungsbildschirm in das in Abbildung 5 gezeigte Hauptmenü:

```
root@sonne> /var/lib/apsfilter/SETUP
```

Beginnen Sie nun mit dem Erzeugen der notwendigen printcap-Einträge, indem Sie über den Menüeintrag »ENtrY« und darin »DEVICE« anspringen. Je nach Druckertyp und Schnittstelle markieren Sie das entsprechende Gerät (»PARALLEL« bzw. »SERIAL«). Mit »PREFILTER« können Sie den erzeugten printcap-Eintrag als Vorverarbeitungsstufe einsetzen, anschließend wird das Ergebnis zur weiteren Bearbeitung/Ausdruck an die hier eingetragene Warteschlange weiter gereicht. »REMOTE« dient zum Einrichten eines Netzwerkdruckers.

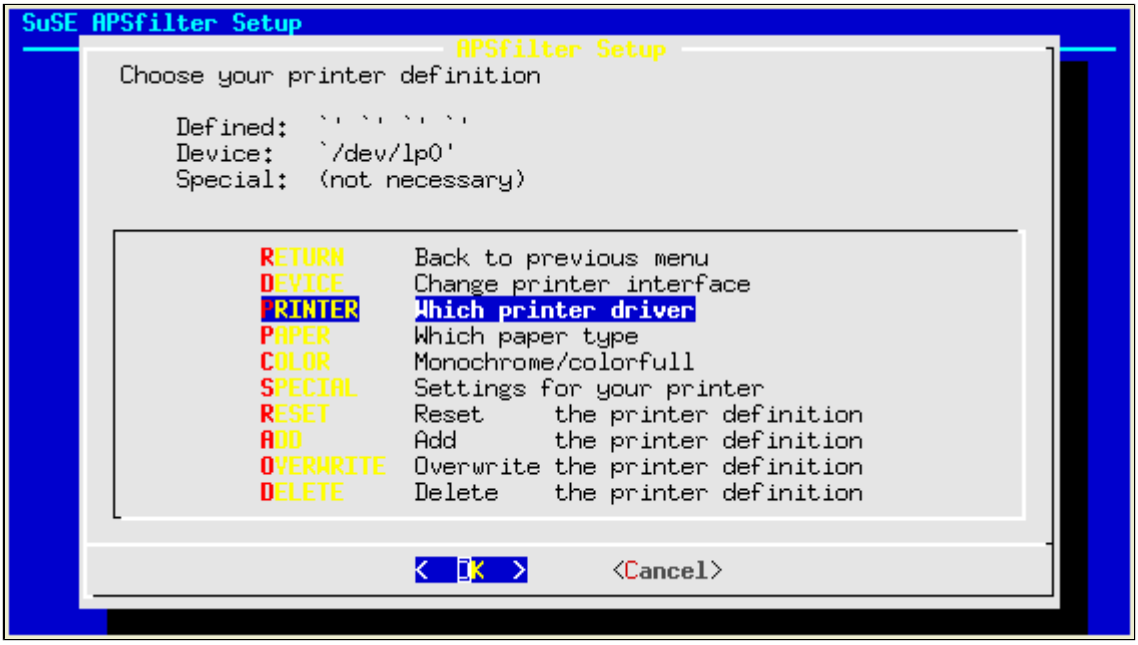


Abbildung 6: Einrichten eines neuen Druckers

Nach Device-Auswahl öffnet sich die in Abbildung 6 gezeigte Maske. Der nächste Schritt wird die Festlegung des zu verwendenden (Ghostscript-)Druckertreibers betreffen. Unter »PRINTER« finden Sie hierzu eine Liste von Druckertypen. Postscript- und HP-Deskjet-Drucker sind extra aufgeführt, die meisten anderen Drucker finden Sie unter »OTHER«. Informationen zu den Druckertreibern erhalten Sie unter dem gleichnamigen Menüeintrag »INFORMATION« (siehe auch Anmerkungen zu Ghostscript weiter oben im Abschnitt); die eigentliche Auswahl geschieht in »COMMIT«, worauf Sie zur Angabe der Auflösung (in dpi) aufgefordert werden.

Die Angabe eines Treibernames unter »FREEDEF« ist erforderlich, wenn der Treiber aus einer neueren Ghostscript-Version stammt und der Name **apsfilter** noch nicht bekannt ist (apsfilter liegt i.d.R. die zurzeit der Kompilierung aktuellste Ghostscript-Version zu Grunde).

Des Weiteren müssen Sie die Papierdimension (PAPER) angeben und kennzeichen, ob der Drucker farbig anzusteuern ist (COLOR).

Weitere Menüpunkte müssen i.d.R. nicht bearbeitet werden, mit »ADD« sind die aktuellen Einstellen noch zu bestätigen. Es erscheint eine Ausgabe der erzeugten Warteschlangen analog zur Abbildung 7.

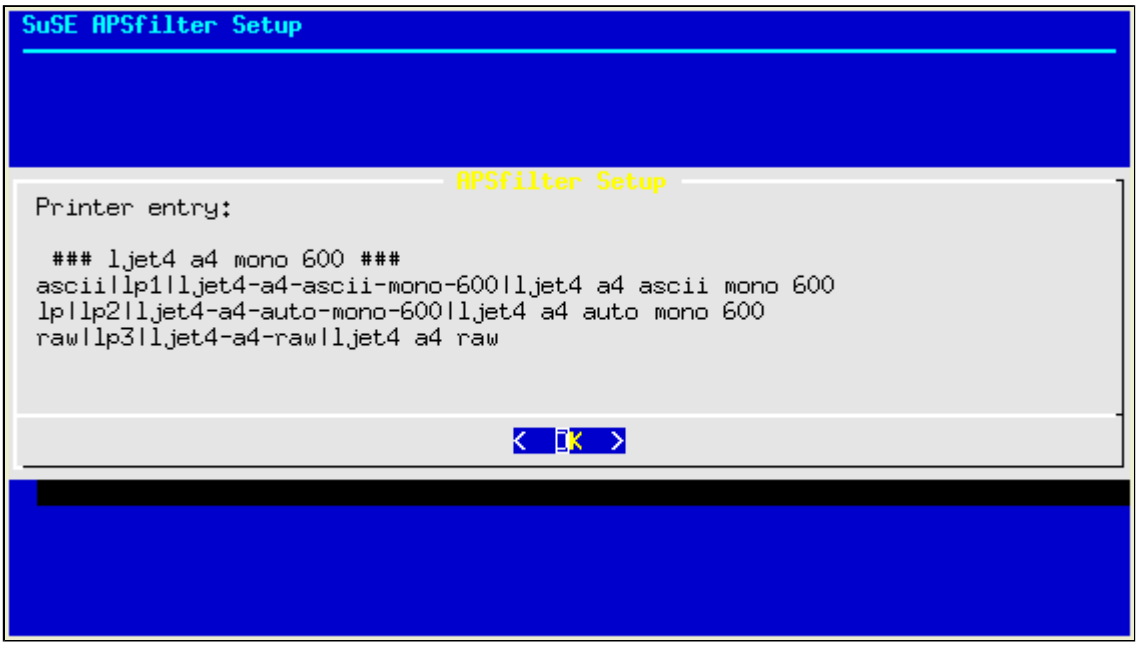


Abbildung 7: Liste der Druckerwarteschlangen

Anhand der Benennung der von **apsfilter** erzeugten Druckerwarteschlangen erkennt der Filter später die zu vollziehenden Schritte zur Aufbereitung eines Druckauftrags. Sowohl Informationen zum Treiber (»ljet4«, vergleiche Abbildung 7), zum Papierformat (»a4«), zur »Art« der Druckdaten (»ascii« - Daten werden als reiner Text interpretiert; »auto« - apsfilter versucht die Art der Daten zu erkennen; »raw« - Daten liegen in einem für den Drucker verständlichen Format vor) sind im Namen kodiert. Ggf. steuern weitere Daten wie Farbansteuerung (im Beispiel »mono«) und Auflösung (»600«) die Filterung.

Mit den bisherigen Schritten sollte ein Probeausdruck zumindest ein einigermaßen brauchbares Ergebnis zu Papier bringen. Die Feinabstimmungen der Arbeit des Ghostscript-Treibers erfolgen global in der Datei **/ etc/ apsfilterc** bzw. treiberspezifisch in **/ etc/ apsfilterrc.*Treibername***. Definitionen aus der treiberspezifischen Konfigurationsdatei überschreiben die zuvor geltenden globalen Einstellungen, sodass Eingriffe nur in dieser Datei erfolgen sollten.

Genau hier deligieren manche Distributionen die Konfiguration an weitere Dateien, sodass obige Aussagen nur den originalen **apsfilter** betreffen. Ohne weiter ins Detail zu gehen, sollten Sie bei der Optimierung der Druckausgabe zunächst wie unter Das Sprachentalent Ghostscript beschrieben mit den Ghostscript-Parametern experimentieren. Erhalten Sie eine brauchbare Einstellung, so setzen Sie die verwendeten Parameter in die Variable(n) »GS_FEATURES« aus **/ etc/ apsfilterrc.*Treibername*** ein. **apsfilter** sollte Ghostscript dann stets mit diesen Argumenten aufrufen.

Ein eigener Filter

Anhand eines einfachen Filtes sollen abschließend die notwendigen Schritte zur Erzeugung eines eigenen printcap-Eintrags vollzogen werden. Unser Filter soll nichts weiter tun, als Postscript-Dateien mittels **psnup** für den zweiseitigen Druck vorzubereiten und die weitere Aufbereitung Ghostscript zu überlassen.

Im ersten Schritt erzeugen wir den Eintrag, wobei als Druckname »ps2side« dient. Beachten Sie, dass genau dieser Name als Name des Spoolverzeichnisses erneut erscheint!

```
ps2side:\
  :lp=/dev/lp0:\
  :sd=/var/spool/lpd/ps2side:\
  :lf=/var/spool/lpd/ps2side/log:\
  :af=/var/spool/lpd/ps2side/acct:\
  :if=/var/spool/lpd/ps2side/demofilter:\
  :mx#0:\
  :sh:
```

Des Weiteren sind die Verzeichnisse und die Protokolldatei zu erzeugen. Vergessen Sie nicht, den Lpd zum Eigentümer zu ernennen!

```
root@sonne> mkdir / var/ spool/ lpd/ ps2side
root@sonne> touch / var/ spool/ lpd/ ps2side/ log
root@sonne> chown -R lp.lp / var/ spool/ lpd/ ps2side
```

Im zu schreibenden Skript verzichten wir der Einfachheit halber auf einen Test des Dateityps. Über eine Pipe senden wir die Daten zunächst durch **psnup** und anschließend in Ghostscript. Wenn Sie im Manual zu gs nachschlagen, werden Sie auch die Erklärung finden, was »-sOutputFile=-« zu bedeuten hat. Das Minus steht hier für die Standardausgabe als Ziel der aufbereiteten Daten. Und woher bezieht gs seine Eingabe? Von der Standardeingabe (die mit der Pipe verbunden ist), benannt durch das Minus am Zeilenende.

```
root@sonne> vi / var/ spool/ lpd/ ps2side/ demofilter
#!bin/sh

PAPER=a4
DRIVER=ljet4
RESOLUTION=600x600

psnup -2 | gs -q -dQUIET -DNOPAUSE -sDEVICE=$DRIVER -r$RESOLUTION -sPAPERSIZE=$PAPER -sOutputFile=- -
```

Nun sind nur noch Eigentümer und Rechte des Filterskripts zu setzen und der Lpd neu zu starten:

```
root@sonne> chmod +x /var/spool/lpd/ps2side/demofilter
root@sonne> chown lp.lp /var/spool/lpd/ps2side/demofilter
root@sonne> /etc/init.d/lpd restart
```

Eventuell liegt bei Ihrer Distribution das Startskript des Lpd an anderer Stelle oder existiert nicht? Dann sollten Sie den Daemon per Hand neu starten (killall lpd; lpd). Ein nachfolgender Test wird hoffentlich die Tigergrafik in halber Größe zu Papier bringen:

```
user@sonne> lpr -Pps2side /usr/share/ghostscript/5.50/examples/tiger.ps
```

Erfolgte kein Ausdruck, so forschen Sie in den Dateien /var/log/messages und /var/spool/lpd/ps2side/log nach den Ursachen.

Das Beispiel eines eigenen Filters wurde bewusst einfach gehalten. Im Prinzip arbeiten alle Filterskripte - auch apsfilter - nach diesem Schema, die magischen Vertreter testen nur zu Beginn des Skripts die ankommenden Daten und beschreiten im weiteren Verlauf abhängig vom erkannten Typ alternative Schritte.

Drucken mit CUPS

Der von Berkeley stammende **Lpd** ist nur eine von vielen Drucklösungen in Unix-Systemen. Gewiss erlangte er nicht zuletzt durch Linux eine gewisse Bedeutung, dennoch existier(t)en neben ihm weitere abgeleitete Realisierungen (bspw. LprNG) als auch gänzlich anders geartete Drucksysteme (System V). Womit diese Systeme allesamt perfekt zurande kommen, sind textbasierte oder Postscriptdrucker. Was sie gar nicht oder nur über Umwege untrestützen, sind all die anderen Drucker, die vor allem bei Privatanwender die Druckaufgaben übernehmen. Für Druckerhersteller war der Unixmarkt zunächst kaum interessant, war die Treiberentwicklung ob der zahlreichen Systeme doch unverhältnismäßig aufwändig und der erreichte Markt recht klein. Erst Linux schickte sich an, ein Konsumentensystem mit entsprechendem Bedarf an »Konsumentendruckern« zu werden. Dennoch stellte der komplizierte Filtermechanismus gewisse Anforderungen an die Treiberentwickler, sodass die Produzenten den Aufwand zumeist scheuten.

Das *Common Unix Printing System* versucht, basierend auf dem *Internet Printing Protocol* (IPP), eine für alle Unix-Systeme einheitliches Drucksystem zu etablieren.

Die Arbeitsweise von Cups

Wenn Sie Cups auf Ihrem Rechner installieren, dann finden Sie die gebräuchlichsten von LPD-System bekannten Kommandos wieder. Cups bringt sein eigene **lpr**-Implementierung mit sowie seine eigenen Druckertreiber. Da Cups für die Seitenbeschreibung analog zum LPD auf Postscript zurückgreift, ist der Einsatz von Cups anstelle des LPD für Programme und Anwender letztlich unsichtbar.

Cups enthält Druckertreiber für die meisten Parallelport-, USB- und Serieller-Port-Drucker. CUPS informiert die im lokalen Netzwerk angeschlossenen Rechnern via Broadcast (bei entsprechender Konfiguration) über den/die lokal verfügbaren Drucker, sodass Netzwerkdrucker von entfernten Systemen dynamisch ermittelt werden können.

Des Weiteren bietet CUPS einen Mechanismus, um mehrere Drucker zu einer Gruppe (Klasse) zu vereinigen. Nach außen hin erscheint eine solche Klasse gegenüber Benutzern und Programmen als einzelner Drucker. Auf welchem Drucker eine Klasse ein Auftrag letztlich ausgeführt wird, entscheidet CUPS intern anhand Benutzerrechte und Verfügbarkeit.

Konfiguration von CUPS

Dieser Abschnitt wird bearbeitet...

Festplatte

Verwechslungsgefahr

Eigentlich ist nichts einfacher, als eine neue Festplatte dem System zu spendieren. Ob IDE oder SCSI, sofern Linux mit den Kontrollern umgehen kann, kann nach dem Einbau über das jeweilige Device auf die Platte zugegriffen werden.

Gliedert sich die neue Platte »hinter« den bereits im System befindlichen ein, sind Komplikationen nahezu ausgeschlossen. »Hinter« steht hier für die Reihenfolge der Platten im BIOS. In einem solchen Fall hängt die Festplatte an einem bislang nicht verwendeten Device und Linux wird keine Einsprüche erheben.

Da das neue Stück aber meist mit einer höheren Leistung aufwartet, möchte man nur allzu gern der Platte einen vorderen Platz einräumen. Wer vorschnell die Platten am Kontroller austauscht, wird beim nächsten Booten schmerzlich an seinen Fehler erinnert:

```
L
```

L? Die Ausgabe stammt vom Linux Loader, der sich beschwert, dass seine Daten nicht mehr dort sind, wo er sie erwartet. Lilo hat sich bei seiner Installation gemerkt, auf welcher Festplatte (und dort im Verzeichnis /boot) die von ihm benötigten Dateien stehen. Doch die Festplatte selbst ist unter dem ursprünglichen Device nicht mehr vorhanden, da sich durch die neue Platte die ganze Anordnung verschoben hat.

Denkbar ist auch, dass Sie die Festplatte mit dem Verzeichnis /boot in ihrer relativen Lage gar nicht verschoben haben, vielleicht ja aber die Platte mit der Rootpartition? Dann wird der Kernel mit Sicherheit Einwände hervorbringen:

```
No init found. Try passing init= option to kernel.
```

Die Beispiele sollen verdeutlichen, dass etwas Sorgfalt beim Einbau neuer Festplatten walten sollte. Werden die Daten des Bootmanagers verschoben, kann der Schaden nur anschließend aus einem Rettungssystem heraus behoben werden. Dazu nehmen wir an, dass bislang nur eine Festplatte (als Master) im System existierte und diese zukünftig als Slave am zweiten Kontroller arbeiten soll.

Nach dem Einbau der neuen Platte müssen Sie ein Linux-Rettungssystem starten. Mounten Sie anschließend das Root-Dateisystem (befand es sich vor dem Einbau unter /dev/hda2, so heißt das Device nun /dev/hdb2):

```
root@rettungssystem> mount / dev/ hdb2 / mnt
```

Weisen Sie Ihr Rettungssystem an, das Verzeichnis /mnt als neues Root-Verzeichnis zu verwenden:

```
root@rettungssystem> chroot / mnt
```

Bearbeiten Sie die Datei /etc/fstab und ändern Sie alle Vorkommen von /dev/hda in /dev/hdb. Modifizieren Sie ebenfalls die **boot**-Zeile der Datei /etc/lilo.conf in »boot=/dev/hdb«. Nehmen Sie analoge Änderungen in allen mit **root**, **other** oder **table** beginnenden Zeilen vor. Starten Sie abschließend Lilo neu:

```
root@rettungssystem> lilo
```

Wenn Sie nun neu booten, sollte Lilo wie gewohnt arbeiten und Ihr Linux problemlos starten.

Die neue Platte nutzen

Die neue Platte allein nützt wenig, wenn die Daten noch immer auf der alten landen.

Jetzt speichert man unter Unix Daten nicht bewusst in eine Partition, sondern in ein Verzeichnis. Die einfachste Variante zur Nutzung wäre, auf der neuen Partition ein Linux -Dateisystem (ext2 oder ReiserFS) anzulegen und dieses unter einem neuen Verzeichnisnamen zu mounten:

```
root@sonne> mkdir /Neues_Verzeichnis
root@sonne> mount -t ext2 /dev/hda1 /Neues_Verzeichnis
```

Jede Datei, die unterhalb von /Neues_Verzeichnis abgelegt wird, landet somit in der ersten Partition auf der neuen Platte (im Beispiel ist diese /dev/hda).

Leider hilft das Vorgehen nichts, wenn eine existierende Partition aus allen Nähten zu platzen droht. Eine solche existierende Partition muss nachträglich verteilt werden. Hierzu sollten Sie zuerst nachschauen, welches Verzeichnis zu welchem Prozentsatz die betreffende Partition belegt:

```
root@sonne> du -sxh /* 2>/dev/null
5.0M    /bin
2.5M    /boot
264k    /dev
5.4M    /etc
160M    /home
25M     /lib
147M    /local
16k     /lost+found
12k     /mnt
10M     /opt
0       /proc
117M    /root
5.7M    /sbin
23k     /tmp
936M    /usr
18M     /var
```

Überprüfen Sie anhand der Ausgabe von **df**, welches der aufgeführten Verzeichnisse ggf. bereits auf einer anderen Partition liegt. Des Weiteren kommen für die Auslagerung hauptsächlich die Verzeichnisse in Frage, die tatsächlich eine größere Menge Plattenplatz benötigen. Wir demonstrieren das weitere Vorgehen anhand des Verzeichnisses **/home**. Dieses soll samt Inhalt in die Partition /dev/hda1 verschoben werden.

Als Erstes muss die neue Partition gemountet werden. Es bietet sich dafür das Verzeichnis /mnt an:

```
root@sonne> mount -t ext2 /dev/hda1 /mnt
```

Zum Kopieren der Dateien bieten sich die Kommandos tar oder cp an:

```
root@sonne> cp -dRp /home/* /mnt
```

Der Inhalt von /home wird nachfolgend gelöscht und /dev/hda1 auf /home gemountet:

```
root@sonne> rm -rf /home/*
root@sonne> umount /mnt
root@sonne> mount -t ext2 /dev/hda1 /home
```

Damit zukünftig /home auch nach dem Booten zur Verfügung steht, muss ein entsprechender Eintrag in /etc/fstab ergänzt werden:

```
root@sonne> vi /etc/fstab
...
/dev/hda1    /home    ext2    defaults   1  2
...
```

Prinzipiell lassen sich nach diesem Schema beliebige Verzeichnisse auslagern. **Vorsicht** ist bei **/boot** geboten, da dieses Verzeichnis den Kernel und die Daten des Bootmanagers enthält. Hier muss anschließend der Bootloader neu konfiguriert und installiert werden. Aus einem laufenden System heraus dürfen Sie keine Verzeichnisse löschen, die dynamische Bibliotheken enthalten (**/lib**, **/usr/lib**...), da somit alle dynamisch gelinkten Programme (und damit Ihr System) unverzüglich unbrauchbar werden. Solche Eingriffe in das Dateisystem sind nur aus einem

Rettungssystem heraus möglich.

Um **mehrere Verzeichnisse** in eine neue Partition umzusiedeln, muss diese zunächst auf ein **neues Verzeichnis** gemountet werden:

```
root@sonne> mkdir / Neues_Verzeichnis
root@sonne> mount -t ext2 / dev/ hda1 / Neues_Verzeichnis
```

Die zu verschiebenden Verzeichnisse sind nach /Neues_Verzeichnis zu kopieren. Als Beispiele sollen /home und /usr/X11R6 dienen:

```
root@sonne> cp -dRp / home / Neues_Verzeichnis
root@sonne> cp -dRp / usr/ X11R6 / Neues_Verzeichnis
```

Die alten Verzeichnisse werden verschoben und als Links auf die Kopien neu erzeugt:

```
root@sonne> mv / home / home.save
root@sonne> ln -s / Neues_Verzeichnis/ home / home
root@sonne> mv / usr/ X11R6 / usr/ X11R6.save
root@sonne> ln -s / Neues_Verzeichnis/ X11R6 / usr/ X11R6
```

Wenn Sie sich vom fehlerfreien Kopieren überzeugt haben, können Sie die Sicherungskopien (home.save, X11R6.save) löschen.

Als weitere Option lässt sich auch ein komplettes Linuxsystem auf die neue Platte verschieben. Die Zielpartition muss groß genug sein, um die gesamten Daten aufnehmen zu können. Ebenso muss auf der Partition ein Linux-Dateisystem eingerichtet worden sein. Nach dem Mounten werden die Dateien beginnend im Wurzelverzeichnis rekursiv in das gemountete Verzeichnis kopiert. Wichtig ist, dass Sie das gemountete Verzeichnis selbst vom Kopieren ausschließen. Vergessen Sie dieses, beginnen Sie eine (Endlos-)Schleife, die erst abbricht, wenn kein Platz mehr auf der Zielpartition verfügbar ist.

```
root@sonne> mount -t ext2 / dev/ hda / mnt
root@sonne> shopt -s extglob
root@sonne> cp -dRp / !(mnt)* / mnt
root@sonne> shopt -u extglob
```

Die obige Befehlsfolge klappt nur in der Bash, da sie bei gesetzter Shelloption **extglob** die Auswertung des Ausdrucks !(...) unterstützt. Nach dem Kopieren müssen Sie das Dateisystem unter /mnt zur neuen Wurzel ernennen:

```
root@sonne> chroot / mnt
```

Wie bereits anfangs beschrieben, sollten Sie die Dateien /etc/fstab und /etc/lilo.conf (bzw. die Konfigurationsdatei des von Ihnen bevorzugten Bootloaders) anpassen und anschließend den Bootmanager neu installieren.

Da Bewegungen von Verzeichnissen, die Systemdateien enthalten, immer mit gewissen Risiken verbunden sind (Wer behauptet von sich, keine Fehler zu machen?), sollten Sie ein vorheriges Backup in Betracht ziehen!

ISDN

Maus

Modem

Ein **Mo**dulator wandelt die Bitfolge eines Computers in analoge Signale um, die über eine herkömmliche Telefonleitung übertragen werden können. Um solche "Töne" auf der Empfängerseite wieder in die für den Computer verständliche Form zu transferieren, ist ein **Dem**odulator erforderlich. Das **MoDem** beinhaltet beide Elemente. Die Digital-nach-Analog-Wandlung (und umgekehrt) kann nach mehreren Techniken erfolgen. Eine

Amplitudenmodulation verwendet die Lautstärke als Kodeträger. Die **Frequenzmodulation** macht sich die Abweichung einer Frequenz von der Trägerfrequenz zu eigen. Eine **Phasenmodulation** baut in die sinusförmige Signalwelle Sprünge ein, die wiederum das kodierte Signal repräsentieren. Um die hohen Datenraten zu erzielen, wird in der Praxis die Amplitudenmodulation mit der Frequenzmodulation gekoppelt (**Quadratur-Amplitudenmodulation**); auch komprimieren die meisten Modems intern den Datenstrom.

Obwohl immer wieder die Bezeichnung des **ISDN-Modems** zu lesen ist, handelt es sich bei solchen Karten nicht in jedem Fall um Modems. **ISDN-Modems** unterstützen neben digitalen zusätzlich analoge Verbindungen, während häufig auch Karten, die nur digitale Verbindungen handhaben können, (fälschlich) als ISDN-Modems bezeichnet werden. ISDN-Karten und Modems ist gemeinsam, dass sie die Datenübertragung über die Telef

Integration von Software

Übersicht

Während der Installation eines Linuxsystems erfolgte eine mehr oder minder detaillierte Selektion zu installierender Software. Der Bedarf der nachträglichen (De)Installation verschiedener Pakete wird irgendwann erwachsen, sei es, um die zu klein gewordene Festplatte von unnötigem Ballast zu befreien oder ein Programm durch eine verbesserte Nachfolgeversion zu ersetzen. Vielleicht lag das gute Stück Software gar nicht »meiner« Distribution bei? Dann muss es nachträglich ins System eingespielt werden.

I.A. ist es nicht erforderlich, jede Version seiner favorisierten Distribution zu erwerben. Solange diese nicht an ihrer Basis schrauben (bspw. eine neue Version der glibc verwenden), schont die alleinige Aktualisierung der häufig verwendeten Programme nicht nur den Geldbeutel, sondern schließt auch den möglichen Ärger aus, den eine Neuinstallation (bzw. Update) so mit sich bringt. »*Never change a running system*« lautet die Maxime des Profis!

Programme unter Unix bestehen nur in seltenen Fällen aus einer einzelnen Datei. Zum Paket zählen zumeist noch Dokumentationen, Bibliotheken, Konfigurationsdateien u.v.m. Als Struktur eines solchen Pakets haben sich im wesentlichen drei Formate etabliert:

- Das alteingesessene Tar-Archiv (auch gepackt)
- Das verbreitete Rpm-Format (bspw. RedHat, SuSE)
- Das von Debian verwendete Deb-Format
- Das (hier nicht betrachtete) Slp-Format von Stampede

Bei rpm und deb handelt es sich im Unterschied zu tar-Archiven um Paket-Manager, d.h., diese Formate kombinieren die reine Archivierung zusammen gehöriger Paketkomponenten mit Methoden zur sauberen (De) Installation und Aktualisierung. Sie helfen ungemein, die Konsistenz der installierten Software zu wahren.

Kompilierte Versionen zu aktualisierter Software bieten die Großen der Branche meist wenige Tage nach deren Freigabe auf ihren Servern ein. Das Herunterladen und Installieren solcher Pakete im »richtigen« Format mit den distributionseigenen Werkzeugen birgt selten Probleme in sich und sollte stets bevorzugt werden. Nicht jede Distribution jedoch misst einem Programm die gleiche Aufmerksamkeit zu. Teils existieren vorkompilierte Pakete nur in einem Format, das ausgerechnet »meine« Distribution nicht unterstützt! In einem solchen Fall kann ein kleines Perl-Skript namens alien Wunder wirken, konvertiert es so manches Paket doch anstandslos in ein Fremdformat.

Wer aus dem vollen Fundus des reichhaltigen Angebots an freier Software schöpfen will, der wird vielfach den Tar-Archiven (*.tar, *.tar.gz, *.tgz) begegnen. Der Vorteil des Formats liegt in seiner Unabhängigkeit von jeglicher Dateisystemstruktur. Der gravierende Nachteil gründet sich auf die zumeist fehlende Routine zur sauberen (De) Installation solcher Programme.

Das Tar-Archiv ist auch die bevorzugte Verpackung für Quellcode-Pakete. Hieraus ein lauffähiges Programm zu kompilieren, kann spielend leicht als auch frustrierend bis unmöglich sein. Der Neuling auf dem Gebiet der Programmierung wird einem Fehler hilflos gegenüber stehen. Die Prinzipien, nach denen sich eine Ursache des Fehlers eingrenzen lässt, möchten wir daher erläutern.

Source-Pakete

Open Source lebt von der Mitarbeit Freiwilliger. Natürlich ist in erster Linie der Programmierer gefragt, aber die

Beihilfe eines jeden in der Testphase garantiert das Aufspüren möglichst vieler Fehler (»Irgend jemand entdeckt den Fehler. Ein anderer wird ihn beheben können.«). Aus diesem Grund gehen die meisten Projekte bereits in einer sehr frühen Entwicklungsphase an die Öffentlichkeit und bieten Schnappschüsse des aktuellsten Standes auf ihren Servern an. Eine als Beta-Version gekennzeichnete Software muss nicht zwingend fehlerbehaftet sein. Oftmals fehlen noch wesentliche Teile, die die Entwickler für notwendig erachten, der eine oder andere Anwender aber nicht benötigt. Warum sollte man das Programm nicht doch schon nutzen?

Derartige Entwicklerversionen werden nahezu ausschließlich als Quelltext angeboten. Was tut man nun, nachdem das Paket auf die lokale Platte geladen wurde?

Das verbreitete Format für Quellcodepakete ist das Tar- Archiv (*T*ape *Ar*chiver). Typische Endungen solcher Dateien lauten ».tar« bzw. im Fall komprimierter Daten ».tar.gz« oder ».tgz«. In neuerer Zeit finden Sie vermehrt ».tar.bz2«, was auf eine Komprimierung mit dem effektiven Burrows-Wheeler-Algorithmus hindeutet. Auch in den Source-Rpm-Paketen (».src.rpm«), in welcher Form vielfach die Quellen fertiger Projekte verbreitet werden, stecken letztlich Tar-Archive, mit denen analog zur folgenden Beschreibung zu verfahren ist.

Beginnend mit dem Entpacken des Quellpakets betrachten wir die notwendigen Schritte zu einem lauffähigen Programm anhand von **Galeon**, einem auf Mozilla basierenden Browser-Projekt. Im Januar 2001 war die Version 09pre3 aktuell.

Entpacken der Quellen

```
user@sonne> tar xzf galeon-0.9pre3.tar.gz
user@sonne> cd galeon-0.9pre3
user@sonne> ls
ABOUT-NLS       INSTALL      TODO          configure       install-sh     po
AUTHORS         Makefile.am  acconfig.h    configure.in    intl           src
COPYING         Makefile.in  aclocal.m4    galeon.desktop  macros         stamp-h.in
COPYING.README  NEWS         anim          galeon.gnorba   missing        ui
ChangeLog       README       autogen.sh    galeon.spec     mkinstalldirs
FAQ             THANKS       config.h.in   galeon.spec.in  myportal.css
```

Eine Reihe der enthaltenen Dateien sind typisch für Quellcodepakete (nahezu jedes aktuelle Projekt verwendet autoconf/automake, womit die Existenz bestimmter Dateien zwingend gegeben ist). **INSTALL** sollten Sie auf jeden Fall lesen, beinhaltet die Datei doch eine detaillierte Beschreibung der notwendigen Schritte bis zur Installation der kompilierten Programme. **README** beschreibt zumeist die Fähigkeiten des Programms und seine Anwendung. In **NEWS** erfahren Sie die wesentlichen Änderungen der Version gegenüber seinen Vorgängern und **TODO** gibt einen Ausblick auf die zukünftige Richtung. Falls Probleme bei den weiteren Schritten auftauchen, sollten Sie einen Blick in **FAQ** (»Häufig gestellte Fragen«) werfen, bevor Sie sich Hilfe suchend an die Projektmitglieder wenden.

Erzeugen des Makefiles

Befindet sich im Stammverzeichnis des Quellpakets keine Datei mit dem Namen »Makefile« (bzw. »makefile«), so muss diese zunächst erstellt werden. Hierfür existieren (fast) immer Shellskripte namens **configure** (manchmal auch kurz **config** oder **config.sh**). Starten Sie dieses:

```
user@sonne> ./ configure
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/ginstall -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal... found
checking for working autoconf... found
checking for working automake... found
checking for working autoheader... found
checking for working makeinfo... found
checking for gnome-config... /opt/gnome/bin/gnome-config
...
checking for Mozilla... no
configure: error:
      *** Mozilla 0.7 is required
```

```
*** A package for 0.7 is available here:
*** http://people.redhat.com/blizzard/software/.
```

Sinn solcher **configure**-Skripte ist das Erzeugen eines auf die lokalen Gegebenheiten zugeschnittenen Makefiles. Was konkret passiert, hängt natürlich vom Verfasser des Skripts ab (i.d.R. greift der Verfasser hierzu auf vorgefertigte Makros des autoconf/automake-Pakets zurück). Typische Punkte sind:

- Test, ob vom Programm benötigte Software (Programme, Bibliotheken,...) installiert ist
- Prüfen der Hardware (bspw. um festzustellen, für welchen Prozessortyp das Programm zu optimieren ist)
- Prüfen der zum Kompilieren notwendigen Software (bspw. wird der Compiler getestet, welche Optionen er kennt)

Unser Beispiel deckt eine der häufigsten Ursachen für einen gescheiterten »configure«-Aufruf auf: Das Fehlen eines Pakets (hier Mozilla der Version 0.7). Abhilfe sollte die nachträgliche Installation des erforderlichen Pakets bringen.

Nicht selten jedoch stottert »configure« über fehlende Dateien, die aber definitiv installiert sind. »configure« selbst sucht allerdings nur in Standardpfaden; wird es dort nicht fündig, ist die Fehlermeldung die Konsequenz. Spätestens hier sollten Sie die Option **--help** konsultieren. Eventuell findet sich eine Option, mit der »configure« der rechte Fleck gewiesen werden kann.

```
user@sonne> ./ configure --help
Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
  --cache-file=FILE       cache test results in FILE
  --help                  print this message
  --no-create             do not create output files
  --quiet, --silent       do not print `checking...' messages
  --version               print the version of autoconf that created configure
Directory and file names:
  --prefix=PREFIX         install architecture-independent files in PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in EPREFIX
                          [same as prefix]
  --bindir=DIR            user executables in DIR [EPREFIX/bin]
...
```

Dem (Shell-)Programmierer bleibt als letzte Option noch die direkte Suche des Kodeabschnitts, der den Fehler verursachte. Ein hartes Setzen der dortigen Variablen kann über so manche Klippe helfen, erfordert jedoch tiefe Kenntnisse der Programmierung.

Wenn »configure« mit der Systemumgebung zufrieden ist, wird es abschließend ein Makefile erzeugen:

```
user@sonne> ./ configure
...
creating Makefile
user@sonne>
```

Übersetzen

Bei dem im vorigen Schritt erzeugten Makefile handelt es sich um eine Steuerdatei, die die weiteren Schritte zur Kompilierung automatisiert. I.d.R. beinhaltet sie eine Reihe so genannter »Ziele« (targets). Jedes dieser Targets führt zu einem anderen Kontrollfluss. Detailliertere Informationen finden Sie im Kapitel Unix-Werkzeuge.

Das erste im Makefile formulierte Ziel ist das Default-Target, d.h. wird make ohne den Namen eines Ziels aufgerufen, beginnt es mit der Ausführung dieses ersten Targets. Die Übersetzung des Paketes beginnt somit i.A. durch den Aufruf von make:

```
user@sonne> make
```

```
cd /home/user/galeon-0.9pre3/src
...
```

Die zahlreichen Ausgaben während des »Builds« protokollieren die Tätigkeit von make. Schritte des Kompilierens finden (bei C- und C++-Programmen) in den mit »gcc ...« bzw. »g++...« beginnenden Zeilen statt. Interessant sind die Ausgaben wohl eher für den Programmierer. Bewahren Sie die Ruhe, wenn »Warnings« über den Bildschirm rauschen. Die Compiler weisen hier auf mögliche Fallstricke des Programms hin, die (zumeist;-) vom Programmierer beabsichtigt sind.

Ein Abbruch des Vorgangs mit einer »Error«-Meldung ist hingegen ein schwerwiegender Fehler. Bei der Fülle der Ursachen würde eine detaillierte Diskussion dieser einen einführenden Programmierkurs bedingen, aber zu einigen Fehlerklassen lassen sich prinzipielle Ansatzpunkte zur Fehlereingrenzung angeben:

Datei oder Verzeichnis nicht gefunden

```
In file included from foobar.c:2:
/opt/gnome/include/gnome.h:12: gnomesupport.h: Datei oder Verzeichnis nicht gefunden
```

Eine typische Ausgabe für eine fehlende Datei oder eine Datei, die nicht gefunden werden kann (»gnomesupport.h« im Beispiel). Hier sollen Sie zunächst feststellen, ob die Datei im System existiert: »locate *Dateiname*«. Falls nicht, so durchsuchen Sie die Installations-CDs bzw. im Web nach dem Paket, das diese Datei enthält, und installieren es nach.

Ist die Datei allerdings vorhanden, so bieten sich im Falle einer Header-Datei (Endung ».h«) drei **Lösungswege** an:

Setzen der Compiler-Umgebungsvariablen

Die Variable »C_INCLUDE_PATH« (im Falle des C-Compilers [gcc, cc, egcs]) bzw. »CPLUS_INCLUDE_PATH« (C++-Compiler [g++, c++]) wird mit dem Pfad zur fehlenden Datei belegt:

```
user@sonne> locate gnomesupport.h
/opt/gnome/lib/gnome-libs/include/gnomesupport.h
user@sonne> export C_INCLUDE_PATH=$(C_INCLUDE_PATH):/opt/gnome/lib/gnome-libs/include
```

Anpassen des Makefiles

Im Makefile kann nach dem den Fehler verursachenden Compileraufruf gesucht werden. Am Ende der Zeile ist -I*Pfad_zur_Datei* hinzuzufügen (diese Möglichkeit bleibt wohl eher dem Profi vorbehalten).

Link in ein Standard-Include-Verzeichnis

Es kann ein Link auf die vermisste Datei in einem der Standard-Include-Verzeichnisse (bspw. »/usr/include«) gesetzt werden. Bei dieser »unsauberen« Methode empfiehlt es sich, den Link nach erfolgreicher Kompilierung des Pakets wieder zu entfernen.

```
root@sonne> ln -s /opt/gnome/lib/gnome-libs/include/gnomesupport.h /usr/include/
```

Eine Bibliothek wurde nicht gefunden

```
/usr/i486-suse-linux/bin/ld: cannot find -lgnome
collect2: ld returned 1 exit status
```

Der Linker kann eine Bibliothek nicht lokalisieren. »-lgnome« ist eine Compileranweisung, die Bibliothek »libgnome.a« hinzuzulinken (Beachten Sie die Namensgebung: Als Compileroption wird vom Dateinamen der Bibliothek sowohl das Prefix »lib« als auch das Suffix ».a« weggelassen!). Falls die alleinige Installation des die Bibliothek enthaltenden Pakets das Problem nicht beheben kann, sollte die Variable »LIBRARY_PATH« gesetzt und

exportiert werden (mehrere Pfade lassen sich durch Doppelpunkt getrennt angeben):

```
user@sonne> locate libgnome.a
/opt/gnome/lib/libgnome.a
user@sonne> export LIBRARY_PATH=$(LIBRARY_PATH):/opt/gnome/lib
```

Undefined Reference...

```
/opt/gnome/lib/libgnome.a(gnome-config.o): In function `_gnome_config_set_vector':
gnome-config.o(.text+0x2630): undefined reference to `g_free'/
```

Derartige Meldungen können so ziemlich alles bedeuten. Für 90% der Fälle zeichnet eine falsche Bibliotheksversion dafür verantwortlich; eine Aktualisierung kann notwendig werden. Wer über etwas Programmiererfahrung verfügt, könnte vorerst das Kommando **nm** bemühen und die Bibliothek suchen, die das fehlende Symbol (im Beispiel »g_free«) definiert. Existiert sie, so kann die Compilerzeile im Makefile um die Anweisung des Linkens dieser Bibliothek ergänzt werden. Weitere Hinweise zur Problematik von Bibliotheken und Symbolen finden Sie unter Bibliotheken und im Abschnitt zur Bashprogrammierung (Komplexe Anwendungen).

Sie merken schon, dass die Behandlung von Compiler-Fehlern besser dem Experten vorbehalten sein sollte. Mit Ausnahme der nicht gefundenen Bibliothek erfordert eine saubere Lösung stets den Eingriff in die Quellen des Pakets (zumindest des Makefiles). Ich möchte es bei diesem kleinen Ausflug in die Trickkiste der Programmierer belassen...

Installation

Die genaue Anweisung kann der Datei **INSTALL** aus dem Quellverzeichnis entnommen werden. Der Ort der Installation lässt sich häufig beim einführenden »configure« per Option variieren, der Vorgang selbst wird üblicherweise im Makefile realisiert:

```
root@sonne> make install
```

Weniger verbreitet ist die Beigabe eines Skripts **install** oder **install.sh**. Falls vorhanden, finden Sie dieses im Basisverzeichnis des Pakets oder unterhalb des dortigen ./bin-Verzeichnisses.

Deinstallation

Nur wenige Projekte bringen eine eigene Routine zum Entfernen der ins System eingespielten Dateien mit. Denn fast ebenso wie die Dokumentation scheut der Entwickler derartige Nebensächlichkeiten, sodass automatische »Hilfsroutinen« - falls überhaupt - erst bei Abschluss eines Projekts zu diesem hinzugefügt werden.

Für den Fall, dass eine Deinstallation per Skript vorgesehen ist, finden Sie im Basisverzeichnis des Quellpakets ein Skript **uninstall** bzw. **uninstall.sh**, in manch anderen Fällen beinhaltet das Makefile ein »Ziel« zur Deinstallation:

```
# Nur selten bietet ein Makefile dieses...
root@sonne> make uninstall
```

So sehr der Anwender die Möglichkeit der automatischen Deinstallation auch missen mag; die Entwickler haben teils bewusst auf deren Implementierung verzichtet. Die Problematik steckt in der fehlenden Datenbasis, die die Zugehörigkeiten und Abhängigkeiten der einzelnen Dateien protokolliert. Stellen Sie sich vor, was passiert, wenn Sie Bibliotheken deinstallieren, die von einer Reihe anderer Anwendungen gefordert werden. Diese Anwendungen würden nicht mehr funktionieren!

Wünschenswert wäre, wenn per Makefile aus den kompilierten Quellen ein Rpm-oder Deb-Paket erzeugt werden würde. Dass dies mit relativ vertretbarem Aufwand möglich ist, soll in späteren Abschnitten demonstriert werden. Vielleicht inspiriert dies den Leser, die Installation solcher »handerzeugten« Pakete über den Bau eines eigenen RPM-Pakets zu realisieren...

Patch

Wenn Entwickler eines Projekts es für sinnvoll erachten, eine neue Version des Quellkodes zu veröffentlichen, so ist der Weg des geringsten Aufwands das Schnüren eines aktualisierten Komplettpakets.

Bei geringem Datenaufkommen ist das Verfahren durchaus legitim, aber bei mehrere Megabytes umfassenden Quellkode-Dateien wird so mancher freiwillige Tester das langwierige und kostspielige Herunterladen der letzten Version scheuen. Effizient wäre, ein Paket nur einmalig komplett zu übertragen und nachfolgend einzig die Änderungen zwischen den Versionen einzupflegen. Ein solches Verfahren wird als **Patchen** (»korrigieren«) bezeichnet und die Datei, die die Unterschiede zwischen zwei Versionen beinhaltet, nennt sich **Patch**.

Erstellen eines Patches

Nichts schult das Verständnis für die Funktionsweise eines Programms eindrucksvoller als ein Beispiel. Deshalb soll dem »Patchen« eine kurze Abhandlung über das Erstellen der benötigten Patchdateien voran stehen. Sicher wird nur der Programmierer in die Verlegenheit gelangen, Patches zu erstellen. Deswegen werden wir uns auf das minimal erforderliche Wissen beschränken.

Ausgangspunkt sei eine einfache Textdatei:

```
user@sonne> cat ErsteVersion.txt
Ein einfaches Beispiel,
um die Arbeitsweise von "patch"
zu demonstrieren.
```

Aufgrund der immensen Wichtigkeit der Aussagen des Inhalts der Datei kopieren sie sich eine Menge Leute. Der Autor indess modifiziert den Text in einer seiner kreativen Phasen:

```
user@sonne> cat ZweiteVersion.txt
Dies ist ein einfaches Beispiel,
um die Arbeitsweise von "patch"
zu demonstrieren.
```

Allen Eignern der Kopie der ersten Version die neue Datei zukommen zu lassen ist bei solch kleinen »Projekten« praktikabel. Aber ebenso genügt es, die Änderungen kund zu geben. Und hierbei gelangt das Kommando diff zum Einsatz:

```
user@sonne> diff -u ErsteVersion.txt ZweiteVersion.txt
--- ErsteVersion.txt    Wed Apr 25 21:06:20 2001
+++ ZweiteVersion.txt   Wed Apr 25 21:06:42 2001
@@ -1,3 +1,3 @@
-Ein einfaches Beispiel,
+Dies ist ein einfaches Beispiel,
um die Arbeitsweise von "patch"
zu demonstrieren.
```

Zugegeben... das Beispiel ist schlecht gewählt, da die Ausgabe von »diff« den Umfang der »neuen« Datei bereits überschreitet. Aber bekanntlich heiligt der Zweck die Mittel.

Die Ausgabe von **diff** bedarf einer Aufbereitung, um **patch** die notwendigen Informationen zukommen zu lassen. Im Beispiel wählten wir das **unified** Ausgabeformat (Option **-u**), das eine »leserliche« Form des **context** Formats (Option **-c**) darstellt. Genau diesen »Kontext« benötigt später **patch**, um die Änderungen an den richtigen Positionen der Zieldatei einpflegen zu können.

Mit den vorgestellten Optionen werden neben den Unterschieden auch die ersten drei unveränderten Zeilen in die Ausgabe von **diff** übernommen. Sie dienen als »Anhaltspunkt« für patch, um auch dann noch »sinngemäß korrekte« Korrekturen zu ermöglichen, wenn Zeilen in das Original eingefügt oder daraus entfernt wurden. Genügt der Kontext nicht, um eine eindeutige Abbildung zu erzielen, kann er mittels **-U *n*** (unified) bzw. **-C *n*** auf **n** Zeilen gesetzt werden.

Nun bestehen große Projekte nicht nur aus einer Datei. Und für jede einen eigenen Patch zu erstellen, würde das Interesse am Patchen schnell schwinden lassen. Aus diesem Grund kann **diff** auch auf Verzeichnisse angesetzt werden. Ist nur eine der beiden Angaben in der Kommandozeile ein Verzeichnis, so wird der Name der anderen Datei in diesem Verzeichnis gesucht. Ist **diff** fündig, vergleicht es diese beiden Dateien.

Handelt es sich bei beiden Argumenten um Verzeichnisse, vergleicht **diff** alle Dateien mit identischen Namen und erzeugt eine gemeinsame Ausgabe. Dateien, die nur in einem Verzeichnis vorhanden sind, erscheinen als »Only in *Verzeichnis*: *Datei*« in der Ausgabe. Hier kann die Option **--new-file** hilfreich sein, um »neue« Dateien (die aus dem zweiten Verzeichnis) vollständig in die Ausgabe einfließen zu lassen (**diff** arbeitet so, als wäre die Datei vorhanden aber leer). Schließlich werden mit der Option **-r** auch Unterverzeichnisse rekursiv betrachtet.

Um endlich zum Erstellen des notwendigen Patches zu gelangen... leiten wir die Ausgabe von **diff** in eine Datei:

```
user@sonne> diff -u ErsteVersion.txt ZweiteVersion.txt > patch_ErsteVersion
```

Bei umfangreichen Patchdateien lohnt sich das Komprimieren derselben.

```
user@sonne> gzip patch_ErsteVersion
user@sonne> ls
patch_ErsteVersion.gz
```

Einspielen eines Patches

Die Aktualisierung der Originaldatei anhand des Patches gestaltet sich für unser Beispiel äußerst einfach. Die Datei mit dem Patch ist zu entpacken und dem Programm **patch** über die Standardeingabe zuzuführen:

```
user@sonne> ls
ErsteVersion.txt
user@sonne> gunzip -c patch_ErsteVersion.gz | patch
patching file ErsteVersion.txt
```

Wichtig ist, dass der Kommandoaufruf aus dem Verzeichnis mit der Originaldatei heraus erfolgt. Sonst würde **patch** die zu »patchende« Datei nicht finden.

Patchdateien, die rekursiv über Verzeichnisse erzeugt wurden, enthalten anstatt des reinen Dateinamens den kompletten Pfad zu jeder zu »patchenden« Datei. Stimmen nun die Verzeichnisstrukturen beim Patch-Erzeuger und Patch-Anwender nicht überein, so wird »patch« die zu korrigierende Datei nicht finden und reagiert leicht verärgert:

```
user@sonne> pwd
/home/user/sonstwo
user@sonne> gunzip -c ../linuxfibel-0.6.0-0.7.0.patch.gz | patch
can't find file to patch at input line 4
Perhaps you should have used the -p or --strip option?
The text leading up to this was:
--------------------------
|diff -u -r --new-file fibel_old/access.htm linuxfibel/access.htm
|--- fibel_old/access.htm       Sun May 6 17:14:57 2001
|+++ linuxfibel/access.htm      Fri Jun 29 15:08:22 2001
--------------------------
File to patch:
```

Die erste Option, die Ihnen bleibt, wäre nun tatsächlich, den Zugriffspfad zur betreffenden Datei an der Eingabeaufforderung anzugeben. Bei wenigen Dateien ein legitimes Vorgehen; bei umfangreichen Patches artet die Methode nur zu schnell in echte Arbeit aus. Die Alternative, die lokale Verzeichnisstruktur dem des Patch-Erzeugers anzugleichen, ist ein gangbarer, aber ebenso unbequemer Weg. Besser, wir bemühen die Option **-p*Anzahl***, die »patch« veranlasst, die angegebene Anzahl von Verzeichnisebenen aus den Pfadangaben in der Patchdatei zu entfernen. Aus einer Angabe von **-p1** in obigem Beispiel resultiert, dass sowohl die Dateinamen »fibel_old/access.htm« als auch »linuxfibel/access.htm« als »access.htm« angenommen werden. Befinden Sie sich

also im »Wurzelverzeichnis« des zu patchenden Zweigs, ist »patch« nun in der Lage, alle Dateien zu finden.

Patchen ist genau das, was dieser Abschnitt vermuten lässt: Es ist schwierig und fehleranfällig. Aus diesem Grund sollte eine Sicherung der originalen Daten zum allgemeinen Vorgehen zählen; es genügt die Option **-b** (Backup), um die Arbeit von »patch« persönlich erledigen zu lassen. Für jede Datei, die nachfolgend verändert wird, wird zuvor eine Kopie als »*Dateiname*.orig« erzeugt (Suffix per Option änderbar). Im Fehlerfall - und bei etwas Kenntnis der Shellprogrammierung - richtet ein kleiner Einzeiler den Schaden:

```
user@sonne> for i in `find . -name "*.orig"`; do mv $i ${i%*.orig}; done
```

Nicht jeder Fehler ist so schwerwiegend, als dass der ganze Patch gleich über den Haufen geworfen werden müsste. Manchmal betrifft es nur einzelne Dateien, die das Kommando nicht korrekt zu behandeln weiß. Gerade wer gern mit neuesten Kernelpatches experimentiert, wird einmal Patches einspielen, die sich mit den Änderungen eines früher angewandten Patches nicht vertragen. Kann »patch« Korrekturen nur zum Teil eindeutig in eine Datei einarbeiten, so legt es alle Bestandteile des Patches, mit denen es nichts anzufangen weiß, in eine Datei »*Dateiname*.rej« ab. Eventuell kann dieser »Rest« manuell an die richtigen Positionen geschrieben werden...

Linuxfibel-Patch

Exemplarisch soll das Erzeugen und Einspielen eines Patches anhand der Linuxfibel demonstriert werden. Hierbei liegen die alten Quellen im Verzeichnis »fibel_old « und die neue Version befindet sich unterhalb von »linuxfibel«. Für »diff« verwenden wir die Optionen -u zum Erzeugen eines Kontexts, -r zum rekursiven Absteigen in die Unterverzeichnisse und --new-file zur Übernahme neuer Dateien in den Patch. Da es sich bei der Ausgabe von »diff« um Text handelt, lohnt sich das anschließende Komprimieren enorm:

```
user@sonne> diff -u -r --new-file fibel_old linuxfibel > linuxfibel-0.6.0-0.7.0.patch
user@sonne> ls -l linuxfibel-0.6.0-0.7.0.patch
-rw-r--r--   1 user     users      2141291 Jul  5 16:44 linuxfibel-0.6.0-0.7.0.patch
user@sonne> gzip -9 linuxfibel-0.6.0-0.7.0.patch
user@sonne> ls -l linuxfibel-0.6.0-0.7.0.patch
-rw-r--r--   1 user     users       363441 Jul  5 16:49 linuxfibel-0.6.0-0.7.0.patch
```

Für das Einspielen des Patches nehmen wir nachfolgend an, dass sich die Dateien unterhalb von »/usr/share/doc/linuxfibel« und der Patch im Heimatverzeichnis von »user« befinden. Der folgende Aufruf aktualisiert die Dateien:

```
# vermutlich darf nur Root in /usr/share/doc/linuxfibel schreiben...
root@sonne> cd / usr/ share/ doc/ linuxfibel
root@sonne> gunzip -c ~user/ linuxfibel-0.6.0-0.7.0.patch.gz |  patch -p1
patching file access.htm
patching file allekapitel.htm
patching file anxious.htm
patching file archiv.htm
...
```

Tar-Archive

Seit dem Durchbruch der Paketformate Rpm und Deb hat die Bereitstellung kompilierter Software in Form von Tape Archiven stark abgenommen; die Gründe sind u.a. in der Einleitung zum Abschnitt des RedHat Package Managers zu lesen. Von den namhaften Distributionen ist *Slackware* als einzige diesem historischen Format treu geblieben.

Sollten Sie in die Zwangslage geraten, ein solches Paket in ihrem (Nicht-Slackware-) System zu installieren, dann vergewissern Sie sich auf jeden Fall, dass die Verzeichnisstruktur im Paket auch relative Pfadangaben verwendet:

```
root@sonne> tar tzf SoftwarePaket.bin.tgz
./SoftwarePaket/README
./SoftwarePaket/INSTALL
./SoftwarePaket/bin/proggy
```

```
...
```

Nutzt das Paket absolute Pfadangaben (mit dem Backslash beginnend), so verwenden Sie sicherheitshalber die Option **-C / Installationspfad**, um die Dateien des Pakets nicht gleich ohne vorherigen Test direkt im Dateisystem zu verstreuen. Gerade hier handeln Sie sich eine Menge Probleme ein, wenn das Paket nicht auf die Verzeichnisstruktur Ihrer Distribution abgestimmt wurde.

Nach der Installation (Option **-x** anstatt **-t** bei tar verwenden) sollten Sie zunächst das Programm in Augenschein nehmen, ob es Ihren Ansprüchen gerecht wird. Ist es erst einmal im Dateisystem installiert, ist das Entfernen aller zum Paket gehörigen Dateien doch vielfach recht aufwändig.

Pakete mit enthaltenen relativen Pfadangaben beinhalten häufig ein Skript zur Installation desselben, typische Namen lauten »install« oder »install.sh«. Trotz solcher Installationshilfen bleibt das Entfernen schwierig, da jegliche Protokollierung zur Installation ausblieb. Sicherer ist die Konvertierung des Paketformats nach *.rpm oder *.deb mit Hilfe des Kommandos alien und anschließender Verwendung eines »richtigen« Paketverwaltungswerkzeugs.

Bei *Slackware* - und ebenso in älteren Versionen nahezu aller weiteren Distributionen - finden Sie noch die Programme **installpkg** und **removepkg**, die - der Name spricht für sich - die Installation/Deinstallation vereinfachen. Beide Werkzeuge vermerken die Paketzugehörigkeit in Listen, sodass zumindest das saubere Entfernen eines Pakets gewährleistet ist. Und dennoch fehlt diesen Programmen die Mächtigkeit eines modernen Paketmanagers, um auch Abhängigkeiten zwischen verschiedenen Paketen auflösen zu können.

Ebenfalls in die Werkzeuggruppe zur Verwaltung von "tgz"-Paketen gehörend, ist **pkgtool**, das letztlich nur eine augenfällige Verpackung für *installpkg* und *removepkg* ist:

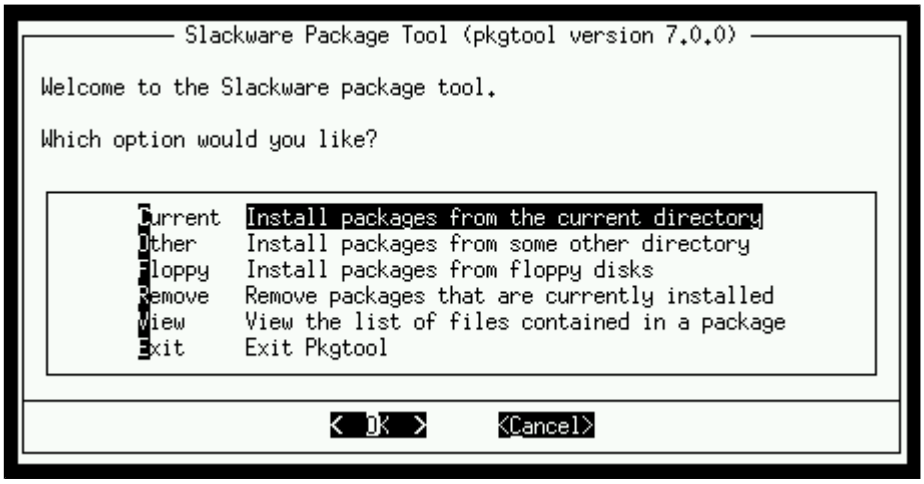


Abbildung 1: pgktool von Slackware

Nicht zuletzt Sicherheitslücken, die diesen Programmen anhaften, führten zu ihrem heutigen Schattendasein...

Rpm-Pakete

Management von Softwarepaketen

Sowohl bei Quellkode als auch bei fertig kompilierten Programmen, die in Form von Tape Archiven verbreitet werden, entpuppt sich die Verträglichkeit mit bereits installierten Komponenten als Quelle häufigen Ärgers. Quellkodepakete schwächen das Problem ab, indem zumeist ein beiliegendes Skript zwingend abzuarbeiten ist (»configure«), sodass die Kompilierung und Installation erst gelingt, wenn gewisse Bedingungen erfüllt sind. Bei Binärpaketen offenbart sich die Lauffähigkeit meist erst nach der Installation. Hat ein solches Paket seine Dateien erst einmal in den Verzeichnissen des Systems verstreut, gleicht dessen saubere Entfernung oft einem Hürdenlauf...

Den ersten Ansatz, durch eine Verwaltung der Dateilisten an zentraler Stelle den Problemen zu entgegnen,

präsentierte »Slackware« mit dem schon genannten »pkgtool«. Bei Bezug auf die Pakete einer einzigen Distribution und einer einzigen Version (der von Slackware) war die saubere (De)Installation zwar gegeben, jedoch fehlten Mechanismen zur Versionsverwaltung, die erst ein Update - und damit eine der wesentlichen Anforderungen an eine Paketverwaltung - ermöglichten. Erst »RedHat« schuf mit dem RedHat Package Manager **rpm** eine ausgefeilte Lösung.

Rpm selbst ist ein Programm, das mannigfaltige Aufgaben wahr nimmt, u.a.:

- Verwaltung der Informationen zu allen installierten Paketen in Datenbanken (im Berkeley-DB-Format); diese liegen unter »/var/lib/rpm«
- Installation, Update und Deinstallation von Paketen (diese müssen in einem für **rpm** verständlichen Format vorliegen)
- Erzeugen von Rpm-Paketen
- Abfrage der Informationen von (nicht) installierten Paketen
- Überprüfung/Zertifizierung von Paketen

Versionsabfrage

Sämtliche Abfragen werden durch die Option **-q** (query) eingeleitet:

```
rpm -q [Optionen]
```

Abfragen können sich auf Einträge aus der RPM-Datenbank oder auf RPM-Pakete beziehen oder, anders ausgedrückt, auf installierte bzw. nicht installierte Pakete. Wichtig ist die Angabe der Quelle, insofern sich die Abfrage **nicht** auf ein installiertes Paket bezieht!

Eine Auskunft über alle in der Datenbank erfassten RPM-Pakete samt ihren Versionsnummern veranlasst die Option **-a**:

```
user@sonne> rpm -qa
aaa_base-2000.7.24-4
aaa_dir-2000.7.29-0
aaa_skel-2000.7.16-1
at-3.1.8-225
...
```

Eine solche Anfrage ist bei der Suche nach einem installierten Paket sinnvoll, falls der genaue Name des Pakets unbekannt ist.

Häufiger interessiert man sich für die Version einer installierten Komponente. Hierzu ist der »Query-Option« **-q** der Paketname nach zu stellen. Im Falle des gezielten Bezugs auf eine konkrete Version (wenn mehrere parallel installiert sind) kann der Name um Versions- und/oder Revisionsnummer ergänzt werden. Zulässige Anfragen bez. des Pakets »cpio« sind bspw:

```
user@sonne> rpm -q cpio          # Paketname
cpio-2.4.2-237
user@sonne> rpm -q cpio-2.4.2    # Paketname + Version
cpio-2.4.2-237
user@sonne> rpm -q cpio-2.4.2-237  # Paketname + Version + Revision
cpio-2.4.2-237
```

Bezieht sich eine Anfrage auf eine RPM-Datei, so ist neben deren Namen die Option **-p** zu verwenden:

```
user@sonne> rpm -q linuxfibel_basis* rpm  # -p fehlt!
Paket linuxfibel_basis-0.6-0.i386.rpm ist nicht installiert
user@sonne> rpm -qp linuxfibel_basis* rpm
linuxfibel_basis-0.6-0
```

Beachten Sie, dass sich der Name einer Rpm-Datei nicht zwingend mit dem Namen des Pakets decken muss!

Detailliertere Auskunft

Mit der reinen Versionsabfrage sind die Möglichkeiten noch lange nicht erschöpft. Aus Anwendersicht sind zunächst die Zugehörigkeiten von Dateien zu Paketen von Interesse. So setzt bspw. die Suche nach der Dokumentation zu einem Paket günstiger Weise bei einem Blick auf die Dateiliste an:

```
user@sonne> rpm -qld ext2fs
/usr/share/doc/packages/ext2fs/RELEASE-NOTES
/usr/share/info/libext2fs.info.gz
/usr/share/man/man1/chattr.1.gz
/usr/share/man/man1/lsattr.1.gz
/usr/share/man/man1/uuidgen.1.gz
/usr/share/man/man8/badblocks.8.gz
/usr/share/man/man8/debugfs.8.gz
/usr/share/man/man8/dumpe2fs.8.gz
/usr/share/man/man8/e2fsck.8.gz
/usr/share/man/man8/e2label.8.gz
/usr/share/man/man8/fsck.8.gz
/usr/share/man/man8/mke2fs.8.gz
/usr/share/man/man8/mklost+found.8.gz
/usr/share/man/man8/tune2fs.8.gz
```

Die Option **-l** zeigt alle zu einem Paket gehörigen Dateien mit vollständigem Pfad an; **-d** beschränkt die Ausgabe auf die zugehörigen Dokumentationsdateien.

Obiges Vorgehen bedingt allerdings die Kenntnis des Paketnamens, zu dem eine Datei gehört. Auch für diese Anfrage sieht **rpm** eine Option vor (**-f**), wobei die komplette Pfadangabe zur Datei erforderlich ist:

```
user@sonne> rpm -qf /usr/bin/chattr
ext2fs-1.18-125
```

Der Vollständigkeit halber soll an dieser Stelle die Abfrage einer »SPEC«-Datei Erwähnung finden. Diese enthalten letztlich die Informationen zum Erzeugen eines Rpm-Pakets (ihr Aufbau wird uns im Rahmen des Paketbaus beschäftigen). Eine Anfrage liefert den Namen des Pakets, das durch diese SPEC-Datei erstellt werden würde:

```
user@sonne> rpm -q --specfile linuxfibel_basis.spec
linuxfibel_basis-0.6-0
```

Tiefer in den Gründen des Pakets forscht die Option **-i**, die neben den bereits genannten Informationen Weiteres zum Inhalt offenbart:

```
user@sonne> rpm -qi ext2fs
Name        : ext2fs                       Relocations: (not relocateable)
Version     : 1.18                             Vendor: SuSE GmbH, Nuernberg, Germany
Release     : 125                       Build Date: Sam 29 Jul 2000 16:33:15 CEST
Install date: Die 27 Mär 2001 19:47:57 CEST      Build Host: Euklid.suse.de
Group       : System Environment/Base       Source RPM: ext2fs-1.18-125.src.rpm
Size        : 519223                          License: Remy Card, Theodore Ts'o
Packager    : feedback@suse.de
Summary     : Utilities for the second extended file system
Description :
Utilities needed to create and maintain ext2 filesystems under Linux.
Included in this package are: chattr, lsattr, mke2fs, mklost+found,
tune2fs, e2fsck, and badblocks.

Authors:
--------
    Remy Card <card@masi.ibp.fr>
    Theodore Ts'o <tytso@mit.edu>

SuSE series: a
```

Um den Status der zu einem Paket gehörigen Dateien zu erfahren, bedienen Sie sich der Option **-s**:

```
user@sonne> rpm -qfs /sbin/fsck
normal      /lib/libcom_err.so.2
normal      /lib/libcom_err.so.2.0
normal      /lib/libe2p.so.2
normal      /lib/libe2p.so.2.3
...
```

Die Angaben entsprechen der Information in der Datenbank und decken sich nicht zwangsläufig mit den Gegebenheiten im Dateisystem. So besagt »normal«, dass - aus Sicht der Datenbank - die Datei so vorhanden ist, wie sie mit dem Paket installiert wurde. Ein »not installed« wird sichtbar, wenn ein Paket nur teilweise installiert wurde (gewisse »exclude«-Optionen); »replaced« erscheint bei Dateien, die durch eine Version aus einem Fremdpaket ersetzt wurden.

Abfrage von Abhängigkeiten

Es macht i.A. selten Sinn, Programme zu installieren, während von diesen benötigte Bibliotheken, Programme oder Interpreter fehlen. Sie würden ja doch nicht laufen. Das Rpm-Paket beinhaltet daher die Informationen, welche Pakete in welcher Version zuvor installiert sein müssen.

Rpm wertet u.a. während der Installation und Deinstallation die Bedingungen für ein Paket intern aus, sodass sich die manuelle Abfrage oft erübrigt. In manchen Situationen finden sich allerdings Abhängigkeiten, die sich nicht automatisch auflösen lassen, bspw. wenn Paket *A* Fähigkeiten eines Pakets *B* benötigt, Paket *B* bedingt aber die vorherige Installation von Paket *A*. Natürlich vermag man solche Pakete auch unter Missachtung der Abhängigkeiten installieren. Günstig ist dennoch, durch vorherige Abfrage sicher zu stellen, dass im Nachhinein eine konsistente Installation vorliegt.

```
user@sonne> rpm -q --requires rsync
/bin/sh
/usr/bin/perl
ld-linux.so.2
libc.so.6
libc.so.6(GLIBC_2.0)
libc.so.6(GLIBC_2.1)
```

Wenn ein Paket bestimmte Anforderungen stellt, so sollte auch ein Paket existieren, das sie erfüllt. Der logische Schritt zur Verifizierung, ob sich ein Paket zur Installation eignet, ist somit der Test, ob die notwendige Software auf dem System existiert. Bei einer sauber geführten **locatedb** sollte auch ein Aufruf von »locate <Datei>« die Lösung offenbaren. Sicher ist jedoch die Anfrage an die Rpm-Datenbank:

```
user@sonne> rpm -q --whatprovides /bin/sh
bash-2.04-30
user@sonne> rpm -q --whatprovides /usr/bin/perl
perl-5.005_03-182
```

Umgangssprachlich ließe sich letztere Abfrage als »*Welches Paket bietet diese Fähigkeit?*« interpretieren. Die Frage nach »*Welche Fähigkeiten bietet dieses Paket an?*«, würde eine **--provide**-Anfrage beantworten:

```
user@sonne> rpm -q --provides rsync
rsync
```

Und letztlich ist auch die Fragestellung »*Welche Pakete benötigen diese Fähigkeit?*« legitim, bspw. um festzustellen, ob ein Paket gefahrlos gelöscht werden kann:

```
user@sonne> rpm -q --whatrequires /usr/bin/perl
aaa_base-2000.7.24-4
groff-1.16-26
lilo-21-132
```

```
ps-2000.7.16-4
rpm-3.0.4-68
texinfo-4.0-112
util-2.10m-41
...
```

Natürlich beschränken sich die Abhängigkeiten nicht allein auf den Paketnamen, sondern sie berücksichtigen ebenso Versionen, denn neuere Software bedingt zumeist auch neuere Bibliotheken...

Alles in Ordnung?

Wer die durch eine Distribution vorgegebenen Pfade zur Administration verlässt, der wird hier und da durch manuelle Eingriffe ins System nicht zuletzt die Belange der Rpm-Paketverwaltung berühren. Einfachstes Beispiel hierfür ist die Aktualisierung eines Pakets, das wiederum nur als Quellkode vorlag und somit unter Umgehung von **rpm** ins System gelangte. Oder aber bewusst »gelockerte« Rechte, um minder-privilegierten Benutzern das Ausführen mancher Programme zu ermöglichen.

»Das Genie beherrscht das Chaos«. Doch da die Wenigsten aus unserer Mitte den Status eines Genies genießen und Dokumentation »per Voreinstellung« stiefmütterlich vernachlässigt wird, wird der Überblick über all die Änderungen bald verloren gehen. Kein Problem... bis zum Auftreten eines Problems. Dann ist es an der Suche, welche Änderung das Übel verbockt haben könnte.

Damit derartige Integritätstests möglich werden, enthalten Rpm-Pakete mehrere Informationen. Eine Anfrage mit der Option **--dump** ist zwar eher zum internen Gebrauch von **rpm** gedacht, schult aber den Sinn für die nachfolgend vorgestellten Tests:

```
user@sonne> rpm -q --dump ext2fs | head
/lib/libcom_err.so.2 17 964881186 0120777 root root 0 0 0 libcom_err.so.2.0
/lib/libcom_err.so.2.0 8133 964881186 c682f94b908a606d053430e2a8078669 0100755 root root 0 0 0 X
/lib/libe2p.so.2 13 964881187 0120777 root root 0 0 0 libe2p.so.2.3
/lib/libe2p.so.2.3 17466 964881187 709a01c3824feccf4b3b35090211902f 0100755 root root 0 0 0 X
/lib/libext2fs.so.2 16 964881187 0120777 root root 0 0 12296 libext2fs.so.2.4
/lib/libext2fs.so.2.4 83724 964881187 3ea426260c3f6c02b629bec7ba636ae6 0100755 root root 0 0 41476 X
/lib/libss.so.2 12 964881186 0120777 root root 0 0 0 libss.so.2.0
/lib/libss.so.2.0 22186 964881186 203ab396745ad6160e318c024734da3d 0100755 root root 0 0 0 X
/lib/libuuid.so.1 14 964881187 0120777 root root 0 0 0 libuuid.so.1.2
/lib/libuuid.so.1.2 11889 964881187 2447a19ed4091e1a22541f1fa2d2e8cb 0100755 root root 0 0 3405 X
```

Von links nach rechts bedeuten die einzelnen Felder:

- Dateiname inklusive Pfad
- Dateigröße in Bytes
- Zeit der letzten Modifikation
- MD5-Prüfsumme (fehlt bei symbolischen Links)
- Zugriffsrechte und Dateityp
- Eigentümer
- Besitzende Gruppe
- Handelt es sich um eine Konfigurationsdatei? 1 - ja; 0 - nein
- Handelt es sich um eine Dokumentationsdatei? 1 - ja; 0 - nein
- Ist die Datei im Speicher verschiebbar (relocatable)? 1 - ja; 0 - nein
- Bei einem symbolischen Link der Name der Zieldatei; »X« sonst

Aus Sicht des Administrators sind einzig die Änderungen relevant, die sich zwischen den Informationen in der Datenbank und der tatsächlichen Installation ergeben und hierfür bietet sich die Option **-V** an. Alle Dateien zu einem angegebenen Paket (bzw. zu allen installierten Paketen bei Kombination mit **-a**) werden einer Reihe von Tests unterzogen, wobei nur Dateien in der Ausgabe erscheinen, die mindestens einen der Tests nicht bestanden haben.

```
user@sonne> rpm -V xf86
......G.   /usr/X11R6/bin
```

```
S.5....T    /usr/X11R6/bin/SuperProbe
......GT    /usr/X11R6/bin/Xmark
S.5...GT    /usr/X11R6/bin/appres
S.5...GT    /usr/X11R6/bin/atobm
...
```

Der Punkt steht hierbei für einen bestandenen Test, die Buchstaben - bzw. die eine Ziffer - kodiert den Grund für ein Scheitern:

- **S** Abweichende Dateigröße
- **M** Die Rechte oder der Dateityp wurden verändert
- **5** Die MD5-Prüfsumme stimmt nicht
- **D** Eine Gerätedatei hat sich verändert
- **L** Ein symbolischer Link zeigt auf eine andere Datei
- **U** Der Eigentümer hat sich geändert
- **G** Die besitzende Gruppe hat sich geändert
- **T** Datei besitzt andere Modifikationszeit

Konfigurationsdateien sind zusätzlich durch ein **c** vor dem Dateinamen gekennzeichnet, sodass schnell entschieden werden kann, ob eine Abweichung von den Informationen aus der Datenbank gewollt ist. Vermisste Dateien enthalten anstatt der Testdaten ein »missing«.

Rpm-Dateien, die aus unbekannter Quelle stammen, sollten im Zweifelsfall auf ihre Unversehrheit hin untersucht werden. Ein **checksig** stellt durch Überprüfung der MD5-Prüfsumme die Originalität eines Pakets sicher:

```
user@sonne> rpm --checksig linuxfibel_basis-0.6.i386.rpm
linuxfibel_basis-0.6-0.i386.rpm: md5 OK
```

Reparatur installierter Pakete

»Wo gearbeitet wird, fallen auch Späne.«, erklärt ein bekanntes Sprichwort. Und wer hin und wieder am System manipuliert, wird auch Verluste verzeichnen. Irgend wann geht irgend etwas mit Gewissheit schief. Wenn dann gar nichts mehr geht, beginnt man besser von vorn, also mit der Neuinstallation der »verbastelten« Pakete. Eine der Installationsoptionen (siehe nachfolgenden Abschnitt) in Verbindung mit **--replacepkgs** erweisen sich als schlagkräftiges Argument:

```
root@sonne> rpm -U --replacepkgs linuxfibel_basis-0.5.i386.rpm
```

Für den verbreiteten Fall, dass ein sorgloser Umgang mit den Dateirechten die Integrität des Systems in Frage stellt, lassen sich auch diese auf den »Originalzustand« zurück versetzen. Verwenden Sie hierzu die Option **--setperms**, um die Rechte eines Pakets zu restaurieren:

```
root@sonne> rpm --setperms linuxfibel_basis-0.5.i386.rpm
```

Um Eigentümer und Gruppenzugehörigkeit wieder herzustellen, ist **--setugids** die rechte Option.

Quasi den »Supergau« bedeutet eine beschädigte RPM-Datenbank. Da sich nahezu die gesamte Abfragefunktionalität von **rpm** auf diese bezieht, ist ohne Datenbasis die gesamte Verwaltung nutzlos. **rpm** ist daher selbst in der Lage, eine bestehende Datenbank neu zu befüllen bzw. eine neue zu erzeugen:

```
# Erzeugen einer neuen Datenbank
root@sonne> rpm --initdb
# Aktualisieren einer bestehenden Datenbank
root@sonne> rpm --rebuilddb
```

In beiden Fällen kann per **--dbpath** ein zu /var/lib/rpm abweichende Pfad zur Datenbank angegeben werden.

Installation von RPM-Paketen

```
rpm -i [Optionen] <Paket>
rpm -U [Optionen] <Paket>
rpm -F [Optionen] <Paket>
```

Die Installation eines Pakets umfasst dessen Erstinstallation als auch die Aktualisierung bereits existierender Pakete. In letzterem Fall wird das alte Paket zuvor implizit aus dem System entfernt.

Die Optionen zum Installieren sind **-i** und zur Aktualisierung **-U** (update). Da letztere Option im Falle, dass das Paket noch gar nicht installiert ist, wie **-i** arbeitet, sollte ihr Einsatz bevorzugt werden.

Bei der erstmaligen Installation ist die Wirkung beider Aufrufe identisch:

```
root@sonne> rpm -i linuxfibel_basis-0.6.i386.rpm
# oder...
root@sonne> rpm -U linuxfibel_basis-0.6.i386.rpm
```

Existiert das Paket hingegen bereits, scheitert **-i**:

```
# linuxfibel_basis wurde bereits installiert...
root@sonne> rpm -i linuxfibel_basis-0.6.i386.rpm
Fehler...
```

Bez. der Aktualisierung von Paketen besteht jedoch oft der Wunsch, einzig bereits installierte Pakete zu ersetzen, ohne dieses ggf. zu installieren. In einem solchen Fall muss die Option **-F** (freshen) anstatt **-U** verwendet werden.

Werkzeuge, die die Paketverwaltung in grafische Masken oder Skripte packen, präsentieren den Installationsvorgang häufig durch einen fort schreitenden Balken. Rpm trägt dafür selbst Sorge, wenn eine der Installationsoptionen (-i|-U|-F) mit **-h** (hash) kombiniert wird. Der Zusatz von **-v** schreibt den Paketnamen vor den Balken:

```
root@sonne> rpm -Uhv linuxfibel_basis-0.6.i386.rpm
linuxfibel_basis ########################################
```

Ohne eine Installation zu vollziehen, arbeitet die Option **--test**. Sie kann zur Kontrolle heran gezogen werden, um von vorn herein Konflikte auszuschließen.

Mitunter erweist sich erst in der Praxis, ob eine neuere Version einer Software das hält, was man sich von ihr verspricht. Dass sie mitunter mehr schadet als nutzt, ist leider nur allzu oft zu verzeichnen. Der Wunsch, auf die alte Version zurückzustellen, wird erwachsen. Es sollte einleuchten, dass **-U** oder **-F** eine »Aktualisierung« auf eine ältere Version abweisen und der Weg über eine vorherige Deinstallation ist dann doch unnötig weit. Einfacher ist der Verwendung der Option **--oldpackage**:

```
root@sonne> rpm -U --oldpackage linuxfibel_basis-0.5.i386.rpm
```

Im Zusammenhang mit der Abfrage von Paketabhängigkeiten wurde die Problematik wechselseitiger Bedingungen von Paketen bereits erörtert. Es findet sich mitunter keine Installationsreihenfolge bei mehreren voneinander abhängigen Paketen, sodass obige Mechanismen jeden Versuch einer Installation oder Aktualisierung abweisen. Per **--nodeps** lässt sich die Überprüfung durch **rpm** deaktivieren.

Eine weitere explizite Aufforderung an **rpm** wird nötig, wenn ein Paket Dateien eines anderen Pakets entfernt, dessen Dateien in der RPM-Datenbank erfasst sind. **--replacefiles** zwingt **rpm** dieselben durch die »neue« Version zu ersetzen.

Die »brutalste« Option, die **--replacefiles**, **--replacepkg** und **--oldpackage** in sich vereint, ist **--force**. Kombinieren Sie diese gar mit **--nodeps**, verdammen Sie **rpm** zum unbedingten Gehorsam...

RPM-Pakete speichern neben den eigentlichen Dateien auch den Installationspfad, unter dem diese ins System zu spielen sind. Nicht zuletzt diese Festlegung erschwert zuweilen die Portierung eines für Distribution *A* erzeugten Rpm-Pakets auf die Distribution *B*, da diese womöglich eine abweichende Verzeichnisstruktur bevorzugt. Zumindest Pakete mit rein distributionsunabhängigen Inhalten (bspw. Dokumentationen) bedingen aber keinerlei Annahmen ob des eigentlichen Installationsorts, sodass deren Installationspfad (oft) als »änderbar« (engl.: **relocatable**) vermerkt ist.

»Relocatable« Pakete lassen sich sowohl mit **--prefix < Pfad>** unterhalb eines Verzeichnisses installieren, wobei der komplette Installationspfad dann in diesem erscheint, als auch per **--relocate < Alter_Pfad> = < Neuer_Pfad>** komplett überschreiben.

Um beim Beispiel des Linuxfibel-Basispakets zu bleiben, könnte ein alternativer Installationspfad wie folgt versucht werden:

```
root@sonne> rpm -U --relocate / usr/ share/ doc/ linuxfibel= / usr/ local/ httpd/ htdocs linuxfibel_basis-0.6-0.i386.rpm
path /usr/src/linuxfibel is not relocateable for package linuxfibel_basis-0.6-0
```

Der Versuch scheitert leider, da das Paket nicht als »relocatable« gekennzeichnet wurde. Aber auch hierfür existiert eine Lösung in Form der Option **--badreloc**, die den Pfadwechsel erzwingt:

```
root@sonne> rpm -U --badreloc --relocate / usr/ share/ doc/ linuxfibel= / usr/ local/ httpd/ htdocs linuxfibel_basis-0.6-0.i386.rpm
```

Entfernen von RPM-Paketen

Zum Entfernen von Paketen dient die Option **-e**:

```
root@sonne> rpm -e linuxfibel_basis
Fehler: Das Entfernen dieser Pakete würde Paket-Abhängigkeiten missachten:
        linuxfibel_basis wird von linuxfibel_kap10-0.6-0 gebraucht
        linuxfibel_basis wird von linuxfibel_kap1-0.6-0 gebraucht
...
```

Das Beispiel verdeutlicht, dass **rpm** auch hierbei die Abhängigkeiten zu weiteren installierten Paketen berücksichtigt. Die Deinstallation glückt erst, wenn kein anderes Paket mehr das zu entfernende bedingt. Natürlich kann hier ebenso mit **--nodeps** nachgeholfen werden, was allerdings zu unbrauchbaren Paketen führen kann.

Ist ein Paket gleichzeitig in mehreren Versionen installiert, entfernt **-e** in Kombination mit **--allmatches** alle Vorkommen; einen sehr ausführlichen Report zu den einzelnen Abhängigkeiten bringen die Optionen **-e --test --vv** zum Vorschein.

Rpm-Pakete selbstgebaut

Voraussetzungen

Die meisten Rpm-Pakete beinhalten Programme nebst zugehöriger Bibliotheken, Dokumentationen, Konfigurationsdateien usw. Beim Erzeuger eines Pakets handelt es sich i.A. auch um den Entwickler des jeweiligen Programms, deshalb werden Rpm-Pakete zumeist direkt aus den Quellen gebaut, d.h. die zu verpackenden Dateien werden erst zum Zeitpunkt des Paketbaus erzeugt.

Somit ergibt sich als erste Anforderung, die zum Erstellen eines Pakets erfüllt sein muss, dass die Quellen auf dem System korrekt kompilieren müssen (für das nachfolgend diskutierte Beispiel der Linuxfibel erübrigt sich diese Forderung, da sie keine Kompilierung bedingt).

Die Arbeitsweise des Kommandos **rpm** wird durch eine oder mehrerer Konfigurationsdateien gesteuert. Beim Start durchsucht *Rpm* die Dateien »/usr/lib/rpmrc«, »/etc/rpmrc« und »~/.rpmrc« in benannter Reihenfolge und setzt

alle Optionen anhand der zuletzt vorgefundenen Definition. Da die Datei »~/.rpmrc« aus dem Heimatverzeichnis eines Benutzers stets zuletzt gelesen wird, lassen sich selbst nutzerspezifische Anpassungen vornehmen. Aber vermutlich werden Sie nie in Verlegenheit gelangen, die Voreinstellungen korrigieren zu müssen... Existiert keine der obigen Dateien (auch nicht in anderen Verzeichnissen), so könnte es an einer unvollständigen Installation liegen. Zumindest in neueren RedHat-basierten Distributionen ist die Funktionalität zum Bau oft in ein »rpm-devel-Paket« ausgelagert.

Schon eher Ziel eines Eingriffs dürfte die Forderung nach der Existenz einer Reihe von Verzeichnissen sein, die u.a. als Ablage für die Quellen, für die Bauanleitungen und für die fertigen RPM-Pakete dienen. Bspw. finden Sie in einem SuSE-System (ab 7.0) die folgenden Verzeichnisse vor:

/usr/src/packages/BUILD

In diesem Verzeichnis werden die Dateien vor dem Paketbau erzeugt

/usr/src/packages/RPMS

Enthält die fertigen RPM-Pakete

/usr/src/packages/SOURCES

Enthält die Quellen und Patches

/usr/src/packages/SPECS

Enthält die SPEC-Dateien zu jedem Paket (quasi die Anleitung, wie ein Paket zu erzeugen ist).

/usr/src/packages/SRPMS

Enthält die fertigen SRPM-Pakete (Quellen)

Um Pakete in der vordefinierten Umgebung zu erzeugen, sind Rootrechte erforderlich, was natürlich auf etlichen Entwicklersystemen aus Sicherheitsgründen nicht erwünscht ist. Einem Benutzer steht es deshalb frei, die vom Kommando **rpm** verwendete Struktur anzupassen, indem er die zu ändernden Makros in einer Datei »~/.rpmmacros« definiert. Die Syntax orientiert sich an der aus der »globalen« Makro-Datei »/usr/lib/rpm/macros«:

```
# Auszug aus /usr/lib/rpm/macros
...
%_usr          /usr
%_usrsrc       %{_usr}/src
...
%_topdir       %{_usrsrc}/packages
...
%_rpmdir       %{_topdir}/RPMS
%_sourcedir    %{_topdir}/SOURCES
%_specdir      %{_topdir}/SPECS
%_srcrpmdir    %{_topdir}/SRPMS
```

Wie im Auszug der Datei unschwer zu erkennen ist, würde das Überschreiben der Variablen »_topdir« genügen, um den Paketbau in einem alternativen Verzeichnisbaum zu vollziehen (diese Verzeichnisse müssten Sie natürlich noch anlegen):

```
user@sonne> vi ~/.rpmmacros
%_topdir /home/user/myrpms

user@sonne> mkdir -p ~/myrpms/{RPMS,SOURCES,SPECS,SRPMS}
```

Um auf Ihrem System die Makros mit den Verzeichnisdefinitionen in Erfahrung zu bringen, müssen Sie nicht erst die Konfigurationsdateien durchforsten. Bemühen Sie die Option »--showrc« von *Rpm* und suchen in der Ausgabe nach den Variablen »_rpmdir«, »_sourcedir«, »_specdir« und »_srcrpmdir«.

```
user@sonne> rpm --showrc | egrep '_rpmdir| _sourcedir| _specdir| _srcrpmdir'
RPM_SOURCE_DIR="%{u2p:%{_sourcedir}}"
RPM_SOURCE_DIR="%{_sourcedir}"
-14: _rpmdir    %{_topdir}/RPMS
-14: _sourcedir %{_topdir}/SOURCES
-14: _specdir   %{_topdir}/SPECS
-14: _srcrpmdir %{_topdir}/SRPMS
```

Für die nachfolgende Ausführung gehen wir von oben skizzierter Verzeichnisstruktur eines SuSE-Linux aus.

Der Bauplan

Die entscheidende Datei für den Paketbau ist die so genannte SPEC-Datei, die allgemeine Informationen und eine Art Bauanleitung zum Paket beinhaltet. Die Datei selbst besteht aus mehreren Sektionen:

Die Präambel

Die Präambel enthält allgemeine Informationen zum RPM-Paket und zum Bau. Jeder Eintrag besitzt die Form:

```
<Name> : <Beschreibung>
```

wobei folgende Namen (»Tags«) möglich sind:

Buildroot

(Alternatives) Verzeichnis, indem das Paket gebildet wird. Wird auf die Angabe verzichtet, so werden die Dateien an ihren endgültigen Positionen im Dateisystem erzeugt. Zumindest für Software im Teststadium is dieses Verhalten denkbar ungeeignet. Bei gesetztem »Buildroot« wird die benötigte Verzeichnisstruktur unterhalb des angegebenem Verzeichnisses erzeugt (bezogen auf das Paket der Linuxfibel würde während Paketbaus bspw. ein Verzeichnis »/tmp/usr/share/doc/linuxfibel« angelegt werden, falls »Buildroot« auf »/t stünde).

Copyright

Copyright-Informationen zur enthaltenen Software.

Conflicts

Namen von Paketen, die sich mit diesem RPM-Paket nicht vertragen. Die Installation wird bei einem Konflik abgelehnt, kann aber dennoch mittels **--nodeps** erzwungen werden.

Distribution

Linuxdistribution, für die das Paket erzeugt wurde.

Group

Art der enthaltenen Software (mögliche Einträge werden von RedHat empfohlen und liegen in einer ASCII-D »GROUPS«der Dokumentation zu *Rpm* bei)

Name

Name des RPM-Pakets

Packager

Patch

Name einer Patchdatei, die zum Bau des Pakets erforderlich ist; mehrere Patches können mittels mehrerer Patch-Einträge angegeben werden.

Prefix

»Empfohlener« Installationspfad für ein verschiebbares Paket

Provides

Ein virtueller Paketname, der eine Einordnung in eine Anwendungsgruppe ermöglichen soll. Somit kann der Anwender in manchen Installationsprogrammen aus einer Reihe von Programmen auswählen, die nahezu dasselbe tun (bspw. bieten sowohl »sendmail« als auch »qmail« einen »mail-server«).

Release

Versionsnummer des Pakets; "0" wenn es erstmals gepackt wird, "1" beim zweiten Mal.... D.h. Versionsnummern sollten erhöht werden, wenn das Paket neu erzeugt wird ohne das sich die Version der enthaltenen Software geändert hat.

Requires

Paketnamen und Versionen, die auf dem System vorhanden sein müssen (Paketabhängigkeiten). Hier könn ebenso die Namen virtueller Paketgruppen stehen, falls kein konkretes Programm bedingt wird. Bez. Versionsnummern sind auch relative Angaben erlaubt: *requires perl >= 5.0*.

Source

Name des enthaltenen Quell-Archivs; mehrere Archive lassen sich mittels mehrerer Source-Tags angeben

Summary

Kurze Beschreibung zum Inhalt des Pakets

Url

URL mit weiterführenden Informationen zum Paket.

Vendor

Anbieter der Software

Version

Versionsnummer der enthaltenen Software

Der entsprechende Abschnitt aus der SPEC-Datei zum Linuxfibel-RPM-Paket sieht wie folgt aus:

Vendor:	Saxonia Systems AG
Distribution:	Gnu-Linux (i386)
Name:	linuxfibel
Version:	0.7
Release:	3
Packager:	thomas.ermer@saxsys.de
Copyright:	GFDL
Summary:	Installiert das Linuxfibel Paket

Url:	http://www.linuxfibel.de
Group:	Linux/Dokumentation
Provides:	Linux Basiswissen
Source:	linuxfibel-%{version}.%{release}.tgz
BuildRoot:	/tmp/linuxfibel-%{version}.%{release}-root

Die %description-Sektion

Eine aussagekräftige Beschreibung zum Inhalt des Pakets ermöglicht die Sektion »description«. Der Text kann auf beliebig viele Zeilen ausgedehnt werden.

```
%description
Die Linuxfibel ist ein distributionsunabhängiges Lehrbuch und Referenz zu Themen rund um Linux.
```

Die %prep-Sektion

Hier findet die Vorbereitung zum Bau des Pakets statt. Letztlich beinhaltet die Sektion verschiedene Shellbefehle, die die Quellen für den anschließenden Kompiliervorgang - sofern ein solcher erforderlich ist - aufbereiten.

Ein gebräuchlicher erster Schritt ist das Entfernen der »Reste« früherer Paketbau-Vorgänge. Dann werden i.d.R. die Quelldateien ins BUILD-Verzeichnis entpackt. Ggf. kann in einem weiteren Schritt das Einspielen von Patches erfolgen.

```
%define INSTALL_DIR     /usr/share/doc/linuxfibel

%prep
rm -rf %{name}-%{version}.%{release}
mkdir %{name}-%{version}.%{release}
cd %{name}-%{version}.%{release}
mkdir -p ./%{INSTALL_DIR}
tar xzvf ../../SOURCES/%{name}-%{version}.%{release}.tgz -C ./%{INSTALL_DIR}
```

Die %build-Sektion

Im Falle von Quellpaketen muss die zu installierende Software erst erzeugt werden. Genügt hierfür ein einfacher Aufruf von make, kann die build-Sektion entfallen. Im anderen Fall werden die auszuführenden Schritte angegeben.

%changelog

Beinhaltet Informationen über die wesentlichen Änderungen der enthaltenen Software.

%clean

RPM entsorgt die »Überreste« der Installation stets selbst, mit der Ausnahme der Verwendung eines buildroot-Verzeichnisses. Ein solches sollte im clean-Eintrag entfernt werden.

```
%clean
rm -r %{buildroot}
```

Die %files-Sektion

Hier folgen die Dateien, die das RPM-Paket enthalten soll. Jede Datei steht auf einer eigenen Zeile und jeder Dateiname ist mit einem Zeilenumbruch abzuschließen. Alternativ lassen sich die von den Shells bekannten Wildcards verwenden. Innerhalb der files-Sektion können einzelne Dateien mit weiteren Tags versehen werden:

% attr

Für die anzugebende Datei werden bestimmte Attribute gesetzt:

```
%attr (644, root, root) bla.txt
```

Anstelle des Wertes kann ein Minus verwendet werden (-, root, root), um den aktuellen Wert beizubehalten

% config[(missingok| noreplace)]

Die Datei wird als Konfigurationsdatei gekennzeichnet; die optionalen Tags geben an, ob eine Datei [bspw. beim Entfernen des Pakets] fehlen darf (missingok) oder ob das Ersetzen einer existierenden Datei zu verhindern ist (noreplace).

% defattr

Die angegebenen Attribute gelten für alle Dateien aus der files-Sektion (Angabe wie bei »attr«)

% dir

Das angegebene Verzeichnis wird ins Paket integriert, nicht aber die enthaltenen Dateien und Unterverzeichnisse (ohne die Angabe wird bei Verzeichnissen auch deren Inhalt zum Paket addiert)

% doc

Die Datei wird als Dokumentation ausgewiesen, womit die Installation per Rpm-Option »--excludedocs« verhindert werden kann.

% docdir

Dateien aus dem angegebenen Verzeichnis werden ins Paket integriert und als Dokumente gekennzeichnet. Datei aus /usr/[share/]doc, /usr/[share/]info und /usr/[share/]man müssen nicht explizit als Dokumentation ausgewiesen werden.

% ghost

Die Datei selbst ist nicht im Paket enthalten, sie wird aber bei der Installation als leere Datei erzeugt.

% license

Die angegebene Datei enthält Lizenzinformationen

% readme

Die angegebene Datei enthält lesenswerte Informationen

% verify

Die angegebenen Attribute einer Datei werden bei einer Überprüfung berücksichtigt

```
%verify (md5, owner, group, size) bla.txt
```

Die Liste der einzufügenden Dateien kann auch [teilweise] in einer separaten Datei stehen. Diese wird wie folgt eingebunden:

```
%files -f Datei_mit_Liste
```

Die %install-Sektion

%post, %postun

Die Sektionen beinhalten Bashskripte oder die Namen der dem RPM-Paket beiliegenden Skriptdateien, die nach der Installation (%post) bzw. nach dem Löschen (%postun) des Pakets ausgeführt werden sollen.

```
%post
cd %{INSTALL_DIR}
%{INSTALL_DIR}/print_version.sh
```

```
%postun
cd %{INSTALL_DIR}
test -d images && rm -r images
rmdir %{INSTALL_DIR}
```

%pre, %preun

Die Sektionen beinhalten Bashskripte oder die Namen von dem RPM-Paket beiliegenden Skriptdateien, die vor der Installation (%pre) bzw. vor dem Löschen (%preun) des Pakets ausgeführt werden sollen.

```
%preun
cd %{INSTALL_DIR}
test -d printversion && rm -r printversion
```

Anmerkung: Die im Beispiel verwendeten Befehle erzeugen bzw. entfernen die Dateien zur Druckversion der Linuxfibel. Das Skript *print_version.sh* wird ausführlich im Abschnitt Shells, Bash-Programmierung, Komplexe Anwendungen behandelt.

Der eigentliche Bau

Erzeugt wird ein RPM-Paket mit Hilfe des Kommandos **rpm** selbst:

```
rpm -bStadium [Optionen] SPEC-Datei [SPEC-Datei]
```

Die mit *Stadium* bezeichneten Optionen steuern den Umfang des Paketbaus:

- **p** Ausführen der %prep-Sektion
- **c** Ausführen der %prep- und %build-Sektion
- **i** Ausführen der %prep-, %build- und %install-Sektion
- **b** Ausführen der %prep-, %build- und %install-Sektion; Erzeugen des Binary-Pakets
- **a** Wie **b**; zusätzlich wird ein Quell-RPM-Paket gebaut
- **l** Test der %files-Liste (Existenz der Dateien, Überprüfung benötigter und bereit gestellter Bibliotheken)

Im Zusammenhang mit **c** oder **i** können führende Schritte übersprungen werden, indem mit der Option **--short-circuit** die Startphase des Baus spezifiziert wird. Sinnvoll ist diese Option beim Aufspüren von Fehlern, indem diese nicht an den Orginaldateien des Pakets gefixt werden, sondern die Suche in der Kopie des BUILD-Verzeichnisses erfolgt. Nun sollte natürlich die %prep-Sektion zum folgenden Testlauf nicht erneut ausgeführt werden, da sonst die originalen Quellen den Inhalt des BUILD-Verzeichnisses überschreiben würden...

Als weitere Optionen stehen zur Verfügung:

--timecheck *Sekunde*

Gibt eine Warnung aus, wenn der Zeitstempel einer zu packenden Datei älter als die angegebenen Sekunde ist; somit lassen sich leicht versehentliche Einträge der Dateiliste aufspüren (das Zeitintervall sollte größer s als der Paketbau dauert)

--clean

--rmsource

Entfernt die Quellen und die SPEC-Datei nach erfolgtem Bau

--test

Simuliert den Bau; anhand der Ausgaben können vorab Fehler erkannt werden

--sign

Fügt eine PGP-Signatur ins Paket ein

--target *Plattform*

Spezifiziert die Zielplattform, für die das Paket erzeugt werden soll

--buildroot *Verzeichnis*

Überschreibt den gleichnamigen Eintrag der SPEC-Datei

Beispielhaft sollen die Schritte aufgezeigt werden, die *Rpm* beim Paketbau mit der Option *-ba* durchläuft:

- Lesen und Parsen der Spec-Datei
- Bearbeitung der %prep-Sektion zum Entpacken der Quellen ins Build-Verzeichnis und ggf. Einspielen der Patches
- Bearbeitung der %build-Sektion zum Kompilieren der Quellen
- Bearbeitung der %install-Sektion zur Installation der übersetzen Quellen
- Lesen der %files-Sektion, Suche aller aufgeführten Dateien und Verzeichnisse und Bau der RPM- und SRPM-Pakete
- Bearbeitung der %clean-Sektion

Exemplarisch folgt der Aufruf, mit dem wir die Linuxfibel-RPM-Pakete erstellen:

```
root@sonne> cd / usr/ src/ packages/ SPECS; ls
Linuxfibel.spec
root@sonne> rpm -bb --target noarch Linuxfibel.spec
Erzeuge Zielplattformen: noarch
Paket wird erzeugt für Zielplattform noarch.
Ausführung(%prep): /bin/sh -e /var/tmp/rpm-tmp.38531
+ umask 022
+ cd /usr/src/packages/BUILD
+ rm -rf linuxfibel-0.8.3
+ mkdir linuxfibel-0.8.3
+ cd linuxfibel-0.8.3
+ mkdir -p /usr/share/doc/linuxfibel
+ tar xzvf /usr/src/packages/SOURCES/linuxfibel-0.8.3.tar.gz -C /usr/share/doc/linuxfibel
access.htm
allekapitel.htm
...
xsteuerung.htm
+ exit 0
Verarbeitung der Dateien: linuxfibel-0.8-3
Suche Provides: (benutze /usr/lib/rpm/find-provides)...
Suche Requires: (benutze /usr/lib/rpm/find-requires)...
Provides: Linux Basiswissen
PreReq: /bin/sh
Geschrieben: /usr/src/packages/RPMS/noarch/linuxfibel-0.8-3.noarch.rpm
Ausführung(%clean): /bin/sh -e /var/tmp/rpm-tmp.85181
+ umask 022
+ cd /usr/src/packages/BUILD
+ rm -rf /usr/share/doc/linuxfibel
+ exit 0
```

Bei Verzicht auf Angabe der Zielplattform wird diese Information vom C-Compiler (gcc) übernommen (es würde also stets ein Architektur abhängiges Paket erzeugt werden).

Patch-Rpm

Hintergrund

Während sich bei der Aktualisierung von Quellcode-Paketen das Patchformat bereits seit langem etabliert hat, werden Updates von RPM-Paketen noch immer als vollständige Pakete bereit gestellt. Leider haben RPM-Pakete, da sie oft Programme und Bibliotheken umfassen, die Eigenschaft, recht umfangreich zu sein. Und insbesondere der Anwender, dessen Rechner nur mit einer lahmen Modemanbindung am Internet teilnimmt, stöhnt bei notwendigen Aktualisierungen.

Die SuSE AG hat eine Erweiterung des RPM-Formats vorgeschlagen, um das Patchen derartiger Pakete zu ermöglichen. Dass es funktioniert, beweisen die zahlreichen Patches, die SuSE zu ihren RPM-Paketen bereits anbietet. Bleibt zu hoffen, dass die Erweiterungen bald zum allgemeinen Funktionsumfang von RPM zählen.

Wir demonstrieren nachfolgend die Verwendung und Erzeugung solcher Patches. Denken Sie immer daran, dass dies nur mit einem angepassten RPM-Kommando funktioniert. Konsultieren Sie das Manual zum RPM, ob dieses u.a. die Option **basedon** kennt.

```
user@sonne> man rpm
...
--basedon
      Show what packages a patch-rpm is based on. A
      patch-rpm can only be installed if one of the pack
      ages it is based on is installed.
...
```

Einspielen eines RPM-Patches

Wie auch bei Quelltextpatches kann nicht jeder RPM-Patch auf jedes Paketversion angewandt werden. Für den Fall, dass Sie es dennoch versuchen, würde **rpm** die Installation abweisen.

Mit der eingangs erwähnten Option **--basedon** können Sie vorab erkennen, ob der Patch für die installierte Version »passend« ist:

```
user@sonne> rpm -qp --basedon tk-8.4-64.i586.patch.rpm
tk = 8.4-48
tk = 8.4-51
tk = 8.4-59
```

Der Patch im Beispiel kann auf installierte tk-Rpm-Pakete der Versionen 8.4-48, 8.4-51 oder 8.4-5 angewandt werden. Wenden Sie übrigens die Option *--basedon* auf ein »normales« Rpm-Paket an, wird nichts ausgegeben (Was sollte auch ausgegeben werden?).

Das Einspielen des Rpm-Patches geschieht analog zur Installation eines »normalen« Rpm-Pakets mittels der Option *-i*.

```
root@sonne> rpm -i tk-8.4-64.i586.patch.rpm
```

Ob ein Rpm-Paket gepatcht ist, erfahren Sie mit Hilfe der Option **-P** bzw. **--patches**, die alle ersetzten Dateien auflistet:

```
root@sonne> rpm -i tk-8.4-64.i586.patch.rpm
```

```
user@sonne> rpm -ql --patches tk-8.4-48.i586.rpm
```

```
root@sonne> rpm -i tk-8.4-64.i586.patch.rpm
user@sonne> rpm -ql --patches tk-8.4-48.i586.rpm
/usr/lib/libtk8.4.so
```

Erzeugen eines RPM-Patches

```
user@sonne> cat linuxfibel.patch.spec
#
# spec file for linuxfibel.rpm
#

Vendor:        Saxonia Systems AG
Distribution:   Gnu Linux
Name:          linuxfibel
Version:       __NEW_VERSION__
Release:       __NEW_RELEASE__
Packager:       linuxfibel@gmx.de

Copyright:      GPL
Summary:        Installiert das Linuxfibel Paket
Group:         Linux/Dokumentation
Provides:       linuxfibel

Source:        linuxfibel-%{version}-%{release}.tgz
BuildRoot:      /tmp/linuxfibel-%{version}.%{release}-root
Patches:       Linuxfibel = __INSTALLED_VERSION__

% description
Die Linuxfibel ist ein distributionsunabhängiges Lehrbuch und Referenz zu Themen rund um Linux.
Distribution: Gnu-Linux (i386)

################################################################################
% define INSTALL_DIR     /usr/share/doc/linuxfibel
################################################################################
% prep
rm -rf %{name}-%{version}.%{release}
mkdir %{name}-%{version}.%{release}
cd %{name}-%{version}.%{release}
mkdir -p ./%{INSTALL_DIR}
tar xzvf ../../SOURCES/%{name}-%{version}-%{release}.tgz -C ./%{INSTALL_DIR}

################################################################################
% install
cd %{name}-%{version}.%{release}
rm -f %{buildroot}
ln -s `pwd` %{buildroot}

################################################################################
% clean rm -r %{buildroot}

################################################################################
% files

% dir %{INSTALL_DIR}
```

Deb-Pakete

Der feine Unterschied

Gäbe es keine Debian-Distribution, so könnte der RedHat Package Manager zu Recht den Anspruch eines Standard-Linux-Paketformats erheben. Und würde Debian nicht ausgerechnet bei eingefleischten Linuxkennern so hoch im Kurs stehen, so würden die häufig erklingenden Aussagen, »das Debian-Paket-Format sei dem von RedHat überlegen«, als Pauschalmeinung von Unwissenden ungehört in der Senke des Vergessens verhallen. Doch was unterscheidet dieses Format so von Rpm, dass die Debianer vortrefflich ob der Vorzüge diskutieren mögen?

Als augenscheinlichster Unterschied kommt die Paketverwaltung von Debian mit einer Reihe von Programmen daher, während RedHat's RPM die Funktionalität unter einem Hut vereint. Der Zweck der wesentlichen Werkzeuge sei an dieser Stelle nur kurz vermerkt:

dpkg

Es handelt sich quasi um das »Hauptprogramm« der Debian-Paketverwaltung. Dient der (De)Installation de Pakete, zu dessen Aktualisierung, hilft bei Abfragen und findet außerdem als Frontend zu **dpgk-deb** Verwendung. Wir werden uns im folgenden Abschnitt verstärkt an diesem Werkzeug orientieren, auch wenn meisten administrativen Tätigkeiten bequem mit dem Frontend **dselect** vorgenommen werden können.

dpkg-deb

Erzeugt und verwaltet Debian-Pakete. Obwohl das Kommando separat verwendet werden kann, wird es zur durch **dpkg** gesteuert.

dselect

Ein interaktives Frontend zu **dpkg**. **deselect** modifiziert nur die Aktionen aus der Datenbank, anhand derer **dpkg** entsprechende Handlungen vornimmt.

apt-get

Ein Bestandteil des (in Entwicklung befindlichen) »Advanced Package Tools (APT)«, das als zentrales Paketverwaltungswerkzeug dienen soll. Einige weitere Kommandos aus APT werden im weiteren Text kurz benannt.

Informationen zu installierten Paketen speichert RPM in einer einzigen Datenbank (»/var/lib/rpm«). Debian hingegen legt seine Verwaltungsdateien in Dateien unterhalb von »/var/lib/dpgk/« ab. In »/var/lib/dpgk/available« stehen alle auf dem System verfügbaren (nicht zwingend installierte) Pakete, »/var/lib/dpkg/status« enthält die ggf. auszuführende Aktion und den Status zu jedem verfügbaren Paket.

Der Begriff der Aktion tauchte bereits in der Kurzbeschreibung zu »dselect« auf. Die Paketverwaltung entscheidet anhand dieser Information und dem aktuellen Zustand, wie mit einem Paket zu verfahren ist:

u - Unknown

Keine Aktion

i - Install

Das Paket ist zu installieren

r - Remove

Das Paket ist zu deinstallieren; Konfigurationsdateien bleiben erhalten

p - Purge

Das Paket ist einschließlich seiner Konfigurationsdateien zu löschen

Die Unterscheidung von »Remove« und »Purge« soll verhindern, dass mühsam angepasste Konfigurationsdateien gelöscht werden, falls das Paket während einer Aktualisierung »zeitweilig« entfernt wird.

Der Status zu einem Paket gibt seinen aktuellen Zustand im System an:

config-files

Einzig die Konfigurationsdateien zu einem Paket sind installiert (bspw. per »remove« gelöscht).

half-configured, half-installed

installed/ not-installed

Das Paket ist installiert/nicht installiert.

unpacked

Das Paket ist installiert (entpackt), aber nicht konfiguriert.

Enthält ein Paket Installationsskripte, so werden diese zum Zeitpunkt der Installation in einem Verzeichnis »/var/log/dpkg/*Paketname*« abgelegt.

Das Paketformat selbst ist bei RedHat's RPM eine Eigenkreation, das - vereinfacht formuliert - aus dem eigentlichen Datenpaket im *.tgz-Format und den Konfigurationsinformationen besteht. Debian richtet sein Format an dem der statischen Bibliotheken aus, indem die einzelnen Bestandteile per »ar« ins Archiv gelangen. Letztlich landet aber auch das Paket mit den eigentlichen Daten als *.tgz im Archiv. Hinzu kommen die Dateien »debian-binary« mit der Versionsnummer und »control.tar.gz«, die Installationsskripte, eine Beschreibung und die Dateiliste umfasst.

Ist die Wertung der bisherigen Differenzen rein eine Frage des persönlichen Geschmacks, so ist Möglichkeit der Verwendung von Wildcards in Anfragen des Debian-Paketsystems ein nicht zu unterschätzender Vorteil.

Allgemeine Abfragen

Ebenso wie bei RPM konsultiert Debian's Paketverwaltung die Datenbankeinträge, um Informationen zu einem Paket zu gewinnen. Da Debian auch den Status nicht-installierter - aber verfügbarer - Pakete in den Datenbanken vorhält, erfolgt auch eine diesbezügliche Recherche nur anhand der Einträge.

Die Recherche übernimmt das Kommando **dpgk** in Verbindung mit der Option **-l** bzw. in der Langform **--list**. Eventuell folgende Parameter werden als Paketnamen bzw. Namen mit Wildcards interpretiert. Ein Verzicht auf jegliche Angabe kommt einer Abfrage aller verfügbaren Pakete gleich:

```
user@sonne> dpkg -l
  Desired=Unknown/Install/Remove/Purge/Hold
  |
Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
  |/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:uppercase=bad)
  ||/ Name          Version     Description

+++-==============-============-
=================================================
  ii  adduser       3.11.1      Add users and groups to the system.
  ii  ae            962-26      Anthony's Editor -- a tiny full-screen editor
  ii  anacron       2.1-5.1     a cron-like program that doesn't go by time
  ii  apt           0.3.19      Advanced front-end for dpkg
  ii  asclassic     1.1b-12     A light window manager with the NEXTSTEP look and feel
  ii  at            3.1.8-10    Delayed job execution and batch processing
  ii  autoclass     3.3.2-2     automatic classification or clustering
  ii  balsa         0.6.0-1.1   GNOME email client
  ii  base-config   0.32        Debian base configuration
  ii  base-files    2.2.0       Debian base system miscellaneous files
  ii  base-passwd   3.1.7       Debian Base System Password/Group Files
  ii  bash          2.03-6      The GNU Bourne Again SHell
  ii  bc            1.05a-11    The GNU bc arbitrary precision calculator language
...
```

Die einführende Statuszeile beschreibt kurz die einzelnen Felder der Tabelle. In der ersten Spalte steht die zuletzt erwünschte Aktion; Spalte 2 enthält den aktuellen Zustand. In der Beispielabfrage betraf bei allen angeführten Paketen die letzte Aktion deren Installation (**i**). Dass diese bereits abgeschlossen wurde, markiert im Beispiel das Symbol **i** der zweiten Spalte.

Eine belegte 3. Spalte deutet auf ein Installationsproblem hin. Eventuell benennt das Symbol den konkreten Fehler.

Die weiteren Spalten enthalten den Paketnamen, die Versionsnummer sowie eine Kurzbeschreibung zum Paket.

Die gezielte Betrachtung nur eines Pakets geschieht durch Angabe seines Namens. Sind nur Bestandteile des Namens bekannt, spielt Debian's Recherche-Vorgehen seine Stärke aus, indem die von den Shells her bekannten Wildcards die Angabe von Namensmustern gestatten. Das folgende Beispiel demonstriert eine Anfrage an alle Pakete mit zweistelligem Namen:

```
user@sonne> dpkg -l '??'
  Desired=Unknown/Install/Remove/Purge/Hold
  |
Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
  |/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:uppercase=bad)
  ||/ Name          Version      Description

+++-==============-============-
=================================================
  ii  ae           962-26       Anthony's Editor -- a tiny full-screen editor
  pn  af             <none>       (no description available)
  pn  an             <none>       (no description available)
  ii  at           3.1.8-10     Delayed job execution and batch processing
  pn  bb             <none>       (no description available)
  ii  bc           1.05a-11     The GNU bc arbitrary precision calculator language
...
```

Beachten Sie das in Hochkommata eingeschlossene Suchmuster. So wird eine unerwünschte Interpretation durch die Shell verhindert!

Um weiter gehende Informationen zu einem Paket zu erhalten, bedarf es einer Statusabfrage per **dpgk -s *Paketname***. Hierunter finden Sie auch die Abhängigkeiten und Konflikte, die Sie bei RPM erst durch gezielte Einzelabfragen in Erfahrung bringen:

```
user@sonne> dpkg -s xterm
Package: xterm
Status: install ok installed
Priority: optional
Section: x11
Installed-Size: 580
Maintainer: Branden Robinson <branden@debian.org>
Source: xfree86-1
Version: 3.3.6-10
Replaces: xbase (<< 3.3.2.3a-2)
Provides: x-terminal-emulator
Depends: libc6 (>= 2.1.2), libncurses5, xlib6g (>= 3.3.6-4)
Conflicts: xbase (<< 3.3.2.3a-2)
Conffiles: /etc/X11/Xresources/xterm
Description: X terminal emulator xterm is a terminal emulator for the X Window System. It provides DEC VT102 and
Tektronix 4014 compatible terminals for programs that cannot use the window system directly. This version implements
ISO/ANSI colors and most of the control sequences used by DEC VT220 terminals.
```

Für den Fall, dass das zu konsultierende Paket nicht installiert ist, benötigen Sie die Optionen **-I** bzw. **--info**, um an weiter führende Informationen zu gelangen:

```
user@sonne> dpkg --info xterm_4.2.1-3_i386.deb
new debian package, version 2.0.
size 488764 bytes: control archive= 4039 bytes.
91 bytes, 3 lines conffiles
734 bytes, 17 lines control
1052 bytes, 16 lines md5sums
7914 bytes, 229 lines * postinst #!/bin/sh
147 bytes, 5 lines * postrm #!/bin/sh
7276 bytes, 218 lines * prerm #!/bin/sh
Package: xterm
Version: 4.2.1-3
Section: x11
```

```
Priority: optional
Architecture: i386
Depends: debconf (>> 0.5), libc6 (>= 2.2.4-4), libfreetype6, libncurses5 (>= 5.2.20020112a-1), libxaw7 (>> 4.1.0), xlibs
(>> 4.1.0)
Conflicts: xbase (<< 3.3.2.3a-2), suidmanager (<< 0.50)
Replaces: xbase (<< 3.3.2.3a-2)
Provides: x-terminal-emulator
Installed-Size: 900
Maintainer: Branden Robinson
Source: xfree86
Description: X terminal emulator
xterm is a terminal emulator for the X Window System. It provides DEC VT102 and Tektronix 4014 compatible terminals for
programs that cannot use the window system directly. This version implements ISO/ANSI colors and most of the control
sequences used by DEC VT220 terminals.
```

Die Liste der von einem Paket bereit gestellten Dateien erhalten Sie mittels **dpgk -L *Paketname***:

```
user@sonne> dpkg -L xterm
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/xterm
/usr/share/doc/xterm/copyright
/usr/share/doc/xterm/README.Debian
/usr/share/doc/xterm/changelog.Debian.gz
/usr/share/doc/xterm/xterm.faq.html
/usr/share/doc/xterm/termcap.gz
/usr/share/doc/xterm/ctlseqs.ms.gz
```

Den umgekehrten Weg, zu einer gegebenen Datei das Paket zu finden, welches diese enthält, ist mit **dpgk -S *Dateimuster*** möglich. Im Unterschied zur Abfrage der Dateiliste ist hier die Angabe von Wildcards gestattet:

```
user@sonne> dpkg -S 'bin/*term'
eterm: /usr/bin/Eterm
gnome-terminal: /usr/bin/gnome-terminal
rxvt: /usr/X11R6/bin/rxvt-xterm
hanterm: /usr/X11R6/bin/hanterm
kterm: /usr/X11R6/bin/kterm
util-linux: /usr/bin/setterm
xterm: /usr/X11R6/bin/xterm
```

Abfrage von Abhängigkeiten

Die im vorherigen Abschnitt erwähnte Option **-s** zeigte sowohl die Paketnamen auf, die ein konkretes Paket benötigt als auch jene, deren parallele Installation zu Konflikten führen würde. Auch die Information, was das Paket zur Verfügung stellt, entlockt **dpgk -s** einem Paket.

Wie aber können Sie in Erfahrung bringen, welche Pakete das installierte benötigen? Hier ist *dpgk* mit seinem Latein am Ende. Das aus den *Advanced Package Tools (APT)* stammende Kommando **apt-cache showpkg *Paketname*** kann helfen:

```
user@sonne> apt-cache showpkg xterm
Package: xterm
Versions:
4.2.1-3(/var/lib/dpkg/status)

Reverse Depends:
xfree86-common,xterm
x-window-system,xterm
pgi,xterm
seyon,xterm
gkdebconf,xterm
```

```
tn5250,xterm
debroster,xterm
user-mode-linux,xterm
seaview,xterm
ddd,xterm
lincvs,xterm
texdoctk,xterm
tk-brief,xterm
Dependencies:
4.2.1-3 - debconf (4 0.5) libc6 (2 2.2.4-4) libfreetype6 (0 (null)) libncurses5 (2 5.2.20020112a-1) libxaw7 (4 4.1.0) xlibs (4 4.1.0) xbase (3 3.3.2.3a-2) suidmanager (3 0.50) xbase (3 3.3.2.3a-2)
Provides:
4.2.1-3 - x-terminal-emulator
Reverse Provides:
```

Welche Pakete die Installation von **xterm** bedingen, können Sie dem Eintrag »Reverse Depends:« in obigem Beispiel entnehmen.

Anmerkung: Apt kennt Recherchen analog zu Rpm's Optionen *--requires*, *--provides* und *--whatrequires*, nicht jedoch zu *--whatprovides*.

Alles in Ordnung?

Die Überprüfung der Integrität installierter Dateien geschieht wie auch bei Rpm mittels md5-Prüfsummen. Im Gegensatz zu *Rpm* erledigt *dpkg* jedoch nicht selbst die Aufgabe. Die Existenz der Prüfsummen, anhand derer Dateien kontrolliert werden können, liegt allein in der Verantwortung des Paketerstellers. Falls dieser jene überhaupt vorgesehen hat, finden Sie diese im Verzeichnis »/var/lib/dpkg/info« unter Namen »*Paketname*.md5sum«. Die Überprüfung müssen Sie von Hand vornehmen:

```
user@sonne> cd /
user@sonne> md5sum -c / var/ lib/ dpkg/ info/ xterm.md5sums
md5sum: MD5 check failed for 'usr/X11R6/bin/xterm'
```

Das Beispiel offenbart ein Problem. Ein erneutes Einspielen des Pakets scheint empfehlenswert.

Anmerkung: Die Dateien »/var/lib/dpkg/info/*Paketname*.md5sums« enthalten alle für *md5sum* erforderlichen Informationen (zum Paket gehörigen Dateinamen mit deren Prüfsummen).

Weitere Überprüfungsmechanismen (Rechte, Eigentümer, Modifikationszeit ..) kennt die Debian-Paketverwaltung nicht.

Installation von Deb-Paketen

...noch paar einführende Worte finden...

Installation und Konfiguration mit »dpgk --install«

Die traditionelle Methode zur Installation verwendet das Kommando *dpkg*. Vorhandene Abhängigkeiten werden zwar erkannt aber nicht aufgelöst, d.h. ein Paket lässt sich stets installieren, selbst wenn es wegen fehlender Vorbedingungen nicht nutzbar sein sollte. *Dpkg* arbeitet quasi wie Rpm in Verbindung mit den Optionen *--nodeps --force*.

```
dpkg -i Paketname
dpkg -i [-R] Verzeichnis
```

Sowohl ein einzelnes Paket als auch alle in einem Verzeichnis (einschließlich Unterverzeichnisse bei Angabe der Option »-R«) liegenden Pakete lassen sich mit einem einzigen Befehl installieren. Intern arbeitet *dpkg* folgende Schritte ab:

1. Die Kontrolldateien aus dem neuen Paket werden entpackt
2. Ist eine ältere Version des Pakets installiert, so wird, falls vorhanden, dessen prerm-Skript ausgeführt
3. Das preinst-Skript des neuen Pakets wird ausgeführt (insofern existent)
4. Die neuen Dateien werden entpackt; die alten gleichzeitig gesichert, um bei gescheiterter Installation diese restaurieren zu können
5. Ist eine ältere Version des Pakets installiert, so wird, falls vorhanden, dessen postrm-Skript ausgeführt. Die zuvor gesicherten Dateien werden nun endgültig entfernt.
6. Das neue Paket wird konfiguriert

Auch hierzu ein Beispiel:

```
user@sonne> find / cdrom/ -name 'ddd* deb'
/cdrom/debian/pool/main/d/ddd/ddd_3.3.1-14_i386.deb
user@sonne> dpkg -i / cdrom/ debian/ pool/ main/ d/ ddd/ ddd_3.3.1-14_i386.deb
Selecting previously deselected package ddd.
(Reading database ... 29164 files and directories currently installed.)
Unpacking ddd (from .../d/ddd/ddd_3.3.1-14_i386.deb) ...
Setting up ddd (3.3.1-14) ...
```

Installation und Konfiguration mit »apt-get»

dpgk lässt eine Eigenschaft schmerzlich vermissen: die automatische Auflösung von Abhängigkeiten. Aus diesem Grund ist **apt-get** oft erste Wahl bei den Werkzeugen zur Paketinstallation. Intern deligiert *apt-get* die Arbeit jedoch auch nur an geeignete *dpkg*-Aufrufe.

```
apt-get install Paketname
```

Damit *apt-get* seine Arbeit verrichten kann, benötigt es Kenntnis über alle zur Installation zur Verfügung stehenden Pakete. Hierzu sind sämtliche Quellen (CDROM, ein FTP-Verzeichnis, Verzeichnisse mit Paketen auf lokalen Platten...) in die Datei »/etc/apt/sources.list/« einzutragen. Hierzu können Sie sich eines Editors bedienen oder die Oberfläche von *apt-setup* verwenden.

Apt-get scannt sämtliche Quellen und kann anhand der resultierenden Paketliste entscheiden, ob die Abhängigkeiten bei der Installation eines Pakets erfüllt werden können. Ist dies der Fall, so wird das Kommando sowohl das Paket als auch alle von diesem benötigten Pakete selbsttätig installieren. Bei Nichterfüllung bricht die Installation ab.

```
user@sonne> pwd
/var/lib/dpkg/info
root@sonne> apt-get install ddd
Reading Package Lists...
Building Dependency Tree...
The following NEW packages will be installed:
ddd
0 packages upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 0B/1573kB of archives. After unpacking 4455kB will be used.
Media Change: Please insert the disc labeled 'Debian GNU/Linux 3.0-testing (woody) , disk 3 _2002-08-14/21:54_' in the drive
'/cdrom/' and press enter


Selecting previously deselected package ddd.
(Reading database ... 29165 files and directories currently installed.)
Unpacking ddd (from .../d/ddd/ddd_3.3.1-14_i386.deb) ...
Setting up ddd (3.3.1-14) ...
```

Anstatt des konkreten Paketnamens können Sie bei *apt-get* sogar Namensmuster mit den Shell-üblichen Wildcards angeben (Schützen Sie diese vor der Auswertung durch die Shell!). Das Kommando wird alle Pakete, auf deren Name das Muster zutrifft, installieren.

Eine Liste alle verfügbaren Pakete erhalten Sie mittels »apt-cache dump-avail«. Etwas mehr Komfort bieten **dselect** und das noch in einem frühen Entwicklungsstadium steckende Programm **aptitude**, die wir hier nicht

behandeln werden.

Installation ohne Konfiguration mit »dpgk --unpack«

```
dpkg --unpack Paketname
dpkg --unpack [-R] Verzeichnis
```

Die Option *--unpack* arbeitet wie *-i* mit dem einzigen Unterschied, dass der abschließende Schritt der Konfiguration des neuen Pakets ausgelassen wird.

```
user@sonne> dpkg --unpack /cdrom/debian/pool/main/d/ddd/ddd_3.3.1-14_i386.deb
Selecting previously deselected package ddd.
(Reading database ... 29164 files and directories currently installed.)
Unpacking ddd (from .../d/ddd/ddd_3.3.1-14_i386.deb) ...

Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-====================-====================-
============================================================
iU ddd 3.3.1-14 The Data Display Debugger, a graphical debugger frontend.
```

Eine nachträgliche Konfiguration ist möglich und wird im nachfolgenden Abschnitt beschrieben.

Konfiguration und Reparatur installierter Pakete

Erfolgte eine Installation ohne anschließender Konfiguration eines Pakets, können Sie letzteren Schritt nachholen, indem Sie »dpkg --configure *Paketname*« ausführen:

```
user@sonne> dpkg --configure ddd
Setting up ddd (3.3.1-14) ...

user@sonne> dpkg --list ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-====================-====================-
============================================================
ii ddd 3.3.1-14 The Data Display Debugger, a graphical debugger frontend.
```

Fehlerhafte Installationen identifizieren Sie anhand der Statusspalten beim Listing eines Pakets:

```
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-====================-====================-
============================================================
iHR ddd 3.3.1-14
```

»H« weist auch eine unvollständige Installation hin. Mit »R« wird eine erneute Installation empfohlen, um das Problem zu beheben:

```
user@sonne> apt-get install ddd
Reading Package Lists...
Building Dependency Tree...
1 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
1 packages not fully installed or removed.
```

```
Need to get 0B/1573kB of archives. After unpacking 4455kB will be used.

(Reading database ...
dpkg: serious warning: files list file for package `ddd' missing, assuming package has no files currently installed.
29164 files and directories currently installed.)
Preparing to replace ddd 1:3.3.1-14 (using .../d/ddd/ddd_3.3.1-14_i386.deb) ...
Unpacking replacement ddd ...
Setting up ddd (3.3.1-14) ...
```

Die Statusspalten sollten sich bei Erfolg folgendermaßen präsentieren:

```
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-====================-====================-
================================================================
ii ddd 3.3.1-14 The Data Display Debugger, a graphical debugger frontend.
```

Entfernen von Deb-Paketen

Auch zum Entfernen von Paketen werden wir die Verwendung der beiden Alternativen *dpkg* und *apt-get* demonstrieren. Beide Kommandos ermöglichen auf Wunsch das Beibehalten bestehende Konfigurationsdateien, was mitunter bei späterem Wiedereinspielen von Paketen deren Administrationaufwand vermindern kann.

Mit Erhalt der Konfigurationsdateien mit »dpkg -r«

```
root@sonne> dpkg -r ddd
(Reading database ... 29235 files and directories currently installed.)
Removing ddd ...
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-====================-====================-
================================================================
rc ddd 3.3.1-14 The Data Display Debugger, a graphical debugger frontend.
```

Mit Erhalt der Konfigurationsdateien mit »apt-get remove«

```
root@sonne> apt-get remove ddd
Reading Package Lists...
Building Dependency Tree...
The following packages will be REMOVED:
ddd
0 packages upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
Need to get 0B of archives. After unpacking 4455kB will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 29235 files and directories currently installed.)
Removing ddd ...
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-====================-====================-
================================================================
rc ddd 3.3.1-14 The Data Display Debugger, a graphical debugger frontend.
```

Inklusive der Konfigurationsdateien mit »dpkg -P«

```
root@sonne> dpkg -P ddd
(Reading database ... 29235 files and directories currently installed.)
Removing ddd ...
Purging configuration files for ddd ...
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-==================-====================-
================================================================
pn ddd (no description available)
```

Inklusive der Konfigurationsdateien mit »apt-get --purge remove«

```
root@sonne> apt-get --purge remove ddd
Reading Package Lists...
Building Dependency Tree...
The following packages will be REMOVED:
ddd*
0 packages upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
Need to get 0B of archives. After unpacking 4455kB will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 29235 files and directories currently installed.)
Removing ddd ...
Purging configuration files for ddd ...
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-==================-====================-
================================================================
pn ddd (no description available)
```

Nachträgliches Entfernen der Konfigurationsdateien

Haben Sie die Pakete bereits gelöscht und deren Konfigurationsdateien seinerzeit beibehalten, so müssen Sie auf »dpkg« zurückgreifen, um auch die Konfigurationsdateien aus dem System zu verbannen.

Wir demonstrieren die Arbeitsverweigerung von »apt-get --purge«:

```
user@sonne> dpkg -l ddd
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name Version Description
+++-==================-====================-
================================================================
rc ddd 3.3.1-14 The Data Display Debugger, a graphical debugger frontend.

root@sonne> apt-get --purge remove ddd
Reading Package Lists...
Building Dependency Tree...
Package ddd is not installed, so not removed
0 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

root@sonne> dpkg -P ddd
(Reading database ... 29165 files and directories currently installed.)
Removing ddd ...
Purging configuration files for ddd ...
```

Deb-Pakete selbstgebaut

Wir werden das Thema nicht erschöpfend abhandeln, insbesondere verzichten wir auf die registrierenden Schritte zum Erhalt des Status eines offiziellen Debian-Pakets. Als Ergebnis der folgenden Abschnitte wird ein Debian-konformes Paket der Linuxfibel stehen.

Aufbau von Deb-Paketen

Namenskonvention

Die Namenswahl von (offiziellen) Debian-Paketen ist fest vorgegeben:

```
<Paketname>_<Versionsnummer des Programmes>-<Releasenummer>_<Architektur>.deb
```

Die *Releasenummer* ist gleichbedeutend mit der Versionsnummer der Paketerstellung.

Der Aufbau eines Debianpakets

Eines der gebräuchlichsten Kommandos ist sicherlich tar - der Tape-Archiver. Sie haben es sicherlich schon verwendet, um Dateien und Verzeichnisse in eine einzelne Datei zu packen oder selbige aus einem tar-Archiv zu extrahieren. Im Grunde genommen kennen Sie somit bereits den Aufbau von Debian-Paketen: Sie sind nichts anderes als Archive, die eine Sammlung von Dateien und Verzeichnisse beinhalten. Anstelle des *tar-Archivs* tritt nun das **ar-Archiv**, anstelle des Kommandos *tar* tritt **ar**, das fast dieselben Optionen wie das bereits bekannte Kommando kennt.

Betrachten Sie den Inhalt eines beliebigen Debian-Pakets:

```
user@sonne> ar tvf xterm_4.2.1-3_i386.deb
rw-r--r-- 0/0      4 Oct 19 23:38 2002 debian-binary
rw-r--r-- 0/0   4039 Oct 19 23:38 2002 control.tar.gz
rw-r--r-- 0/0 484532 Oct 19 23:38 2002 data.tar.gz
```

Das Entpacken der Dateien erolgt erfolgt analog zur Option des tar-Kommandos mit **x**:

```
user@sonne> ar xf xterm_4.2.1-3_i386.deb
```

Die Dateien eines Debian-Pakets

debian-binary

```
user@sonne> cat debian-binary
2.0
```

Die Datei beschreibt die Version des verwendeten Formats der Debian-Datei. Im Beispiel verwenden wir die derzeit aktuelle Version 2.0. Wichtig ist, dass der Eintrag der Versionsnummer mit einem Zeilenumbruch abgeschlossen wird.

control.tar.gz

```
user@sonne> tar tzvf control.tar.gz drwxr-xr-x root/root        0 2002-10-19 23:37:13 ./
-rw-r--r-- root/root      734 2002-10-19 23:37:09 ./control
-rwxr-xr-x root/root     7276 2002-10-19 23:36:18 ./prerm
-rw-r--r-- root/root       91 2002-10-19 22:22:51 ./conffiles
-rw-r--r-- root/root     1052 2002-10-19 23:37:13 ./md5sums
-rwxr-xr-x root/root     7914 2002-10-19 23:36:18 ./postinst
-rwxr-xr-x root/root      147 2002-10-19 23:36:18 ./postrm
```

Das Paket der Kontrolldateien umfasst mindestens die Datei **control** mit der Beschreibung des Pakets. Die Datei dient u.a. zur Versionsprüfung und Abhängigkeitsverifizierung. Sie können ebenso *dpkg -p* konsultieren, um

folgende Ausgabe zu erhalten:

```
user@sonne> cat control Package: xterm
Version: 4.2.1-3
Section: x11
Priority: optional
Architecture: i386
Depends: debconf (>> 0.5), libc6 (>= 2.2.4-4), libfreetype6, \
        libncurses5 (>= 5.2.20020112a-1), libxaw7 (>> 4.1.0), xlibs (>> 4.1.0)
Conflicts: xbase (<< 3.3.2.3a-2), suidmanager (<< 0.50)
Replaces: xbase (<< 3.3.2.3a-2)
Provides: x-terminal-emulator
Installed-Size: 900
Maintainer: Branden Robinson <branden@debian.org>
Source: xfree86
Description: X terminal emulator
 xterm is a terminal emulator for the X Window System. It provides DEC VT102
 and Tektronix 4014 compatible terminals for programs that cannot use the
 window system directly. This version implements ISO/ANSI colors and most of
 the control sequences used by DEC VT220 terminals.
```

Optionale Dateien im Kontrollpaket sind die Skripte, die unmittelbar vor bzw. nach einer (De)Installation auszuführen sind. Die Namen der Skripte sind mit *preinst*, *prerm*, *postinst* und *postrm* fest vorgegeben. Jedes der Skripte kann mit konkreten Parametern aufgerufen werden, sodass eine Steuerung des Skriptablaufs nach gewähltem (De)Installationskontext möglich wird. Die wichtigsten Parameter sind:

preinst	install, upgrade
postinst	configure
prerm	remove, upgrade
postrm	remove, purge

Im *Debian Policy Manual* (Chapter 6 - Package maintainer scripts and installation procedure) können Sie sich über die weiteren Parameter informieren.

Ein Skript sollte bei Fehler freiem Ablauf mit dem Wert 0 zurück kehren und mit einem Wert ungleich 0 sonst.

Gehören Konfigurationsdateien zu einem Paket, so werden deren Namen inklusive Pfadangaben in der Datei **conffiles** hinterlegt:

```
user@sonne> cat conffiles
/etc/X11/app-defaults/UXTerm
/etc/X11/app-defaults/XTerm
/etc/X11/app-defaults/XTerm-color
```

Die Unversehrheit der Dateien in einem Paket kann anhand des Inhalts einer Datei **md5sums** verifiziert werden, insofern die Paketbauer diese sinnvolle Eigenschaft vorgesehen hatten:

```
user@sonne> cat md5sums
d672bbf649879f0ce1f340b12f9f8283  usr/lib/menu/xterm
1f68f7ccdb684fb957d7dd983e24e853  usr/X11R6/bin/xterm
ea8870cc936d434239667b3ae5a969bb  usr/X11R6/bin/uxterm
44bb39c5d838a6f062b3735562d459d3  usr/X11R6/man/man1/xterm.1x.gz
096965faecdbdf21e26a667d86a5b415  usr/X11R6/man/man1/uxterm.1x.gz
89fc7c793984ab7c1d8288ceb0dc9f14  usr/share/doc/xterm/xterm.faq.html
526f04b3821954928cde267aeb3306f2  usr/share/doc/xterm/ctlseqs.txt.gz
5ae55d1ac3656abfa4def63d08154852  usr/share/doc/xterm/xterm.faq.text.gz
792df42bff304318c856fd70517c425c  usr/share/doc/xterm/xterm.termcap.gz
3f9346aec8a5fcc6f61d35e0560717fe  usr/share/doc/xterm/xterm.log.html
b6817b9e8bf67218513e02cfec51385a  usr/share/doc/xterm/README.Debian
05469648defd7feef538795654c26b19  usr/share/doc/xterm/xterm.terminfo.gz
2f42d779fe32dc12631e1e5d9a3960c2  usr/share/doc/xterm/copyright
9e3d633a5fb22f4f42b373dd9fc5f45a  usr/share/doc/xterm/ctlseqs.ps.gz
```

```
d18d22063d3c60f1cbf27b8d06015102  usr/share/doc/xterm/changelog.Debian.gz
b41453c1f2d3fec3baeb1209fe33f102  usr/share/doc-base/xterm-faq1
```

data.tar.gz

Dieses Archiv enthält die eigentlichen Paketdateien (Programme, Dokumentationen, Konfigurationsdateien...).

Die Linuxfibel als Debian-Paket

Nach den allgemein gehaltenen Vorbetrachtungen vollführen wir den Schritt zum Debian-Paket anhand des konkreten Beispiels der Linuxfibel.

Vorbereitung der Umgebung

Zunächst benötigen wir ein Verzeichnis, indem der Paketbau vollzogen werden kann. Prinzipiell kann das Verzeichnis irgendwo im Dateisystem liegen und beliebig benannt sein. In Analogie zu Rpm sammeln wir sämtliche Pakete unterhalb eines Verzeichnisses »/usr/src/deb«:

```
root@sonne> mkdir -p / usr/ src/ deb/ linuxfibel
# Die Arbeit als Root ist u.U. gefährlich!
root@sonne> chown user:users / usr/ src/ deb/ linuxfibel
user@sonne> cd / usr/ src/ deb/ linuxfibel
user@sonne> mkdir DEBIAN
user@sonne> chmod 755 DEBIAN
```

In das Unterverzeichnis DEBIAN gelangen später die Beschreibungsdateien und (De)Installations-Skripte.

Kopieren der Dateien

In das neue Arbeitsverzeichnis sind alle Dateien zu kopieren, die zum Paket gehören sollen. Im Fall der Linuxfibel sind das die HTML-Dateien und die Bilder. Wir gehen nachfolgend davon aus, dass im Verzeichnis »/usr/share/doc/linuxfibel« unseres Systems all jene Dateien liegen, die das fertige Paket umfassen soll:

```
user@sonne> (cd / ; tar cf - / usr/ doc/ share/ linuxfibel) | tar xvf -
```

Anlegen der Kontrolldateien

Sämtliche Kontrolldateien und Skripte gehören in das Verzeichnis DEBIAN unterhalb des Build-Verzeichnisses.

debian-binary

Hier hinein gehört die Versionsnummer des verwendeten Paketformats

```
user@sonne> cat DEBIAN/ debian-binary
2.0
```

control

Der im Beispiel verwendete Inhalt der Datei »control« umfasst die minimal erforderlichen Einträge.

```
user@sonne> cat DEBIAN/ control
Package: linuxfibel
Version: 0.8-2
Section: base
Priority: optional
Architecture: all
Maintainer: Thomas Ermer <thomas.ermer@saxsys.de>, Michael Meyer <michael.meyer@saxsys.de>
```

```
Description: Die Linuxfibel
  Die Linuxfibel ist ein distributionsunabhängiges Lehrbuch und Referenz zu Themen rund um Linux.
```

Wichtige Einträge darüber hinaus sind:

Depends

Erfordert Ihr Paket die Installation weiterer Software, so können Sie diese hier inklusive der notwendigen Version angeben. Dem Namen des Softwarepakets folgt die in runde Klammern anzugebende Version. Weit Softwarepakete schließen sich, durch Kommata voneinander getrennt, an ("gcc (>=3.2), glibc (>> 2.2)").

Conflicts

Steht Ihr Paket in Konflikt mit anderen (sodass beide nicht gleichzeitig installiert sein sollten), so können Si auch dieses in der Kontrolldatei vermerken. Die Syntax eines Eintrags ist analog zu *Depends*.

Replaces

Ist das Paket ein vollständiger Ersatz für ein anderes, kann der Name des zu ersetzenden Pakets hier angegeben werden.

Provides

Hier steht ein virtueller Paketname, der eine Einordnung in eine Anwendungsgruppe ermöglichen soll. Somi kann der Anwender in Installationsprogrammen (bspw. dselesct) aus einer Reihe von Programmen auswähl die nahezu dasselbe tun.

Installed-Size

Angabe der Paketgröße.

md5-Prüfsummen

Wir statten unser Paket mit MD5-Prüfsummen über jede der Dateien aus:

```
user@sonne> find / usr/ share/ linuxfibel -type f -exec md5sum {} \; > DEBIAN/ md5sum
```

Der eigentliche Bau

Hierfür wechseln Sie aus dem Build- in das übergeordnete Verzeichnis. Rufen Sie **dpkg-deb** mit der Option **--build** und dem Namen des Buildverzeichnisses als Argument:

```
user@sonne> pwd
/ usr/ src/ deb/ linuxfibel
user@sonne> cd ..
root@sonne> dpkg-deb --build linuxfibel
dpkg-deb: building package `linuxfibel' in `linuxfibel.deb'.
```

Das fertige Paket liegt als »linuxfibel.deb« vor. Wir prüfen dessen Kontrolldatei:

```
user@sonne> dpkg --info linuxfibel.deb
new debian package, version 2.0.
size 7258184 bytes: control archive= 345 bytes.
   303 bytes,     8 lines      control
Package: linuxfibel
Version: 0.8-2
Section: base
Priority: optional
Architecture: all
Maintainer: Thomas Ermer <thomas.ermer@saxsys.de>, Michael Meyer <michael.meyer@saxsys.de>
```

```
Description: Die Linuxfibel
  Die Linuxfibel ist ein distributionsunabhängiges Lehrbuch und Referenz zu Themen rund um Linux.
```

Um der Namenskonvention für Debian-Pakete zu entsprechen, ist das Paket noch umzubenennen:

```
root@sonne> mv linuxfibel.deb linuxfibel_0.8-2_1_i386.deb
```

Alien

Falls ein Programm nicht in einem von »meiner« Distribution unterstützten Format vorliegt, so kann eine Formatwandlung eine Installation doch noch ermöglichen.

Als Voraussetzung für den Einsatz des Perl-Skripts **alien** müssen die Programme zur Verwaltung der »Fremdformate« installiert sein, also:

- Zum Lesen/Erzeugen eines Rpm-Pakets: rpm
- Zum Lesen/Erzeugen eines Tgz-Pakets: tar
- Zum Lesen/Erzeugen eines Deb-Pakets: debmake, dpkg-dev, dpks, make und gcc

Eine Paketverwaltung für ein System

Auch wenn die Programme zur direkten Installation von »Fremdpaketen« vorhanden sind, sollten Sie diese niemals zum Installieren verwenden! Paketverwaltungen basieren auf einer Datenbank und jede Verwaltung verwendet ihre eigene. Leider wissen die Programme nichts von der Existenz der »Konkurrenz«, sodass Vorteile wie Berücksichtigung von Abhängigkeiten, Paketaktualisierung, sauberes Entfernen usw. nicht mehr gegeben sind.

Des Weiteren ist eine Wandlung oft mit einem Verlust an Information behaftet. Deb- und Rpm-Pakete unterscheiden sich im internen Aufbau, sodass manche Aspekte bei der Konvertierung nicht berücksichtigt werden können. Das Tgz-Format beinhaltet gar nur die zu installierenden Dateien ohne jegliche Hinweise auf Version, Abhängigkeiten... Eine Wandlung von Tgz in ein anderes Format ermöglicht einzig die (De)Installation der Daten, aber mehr auch nicht!

Und so gehts...

ie Handhabung von **alien** gestaltet sich spielend einfach. Das Format des Eingabepakets ermittelt das Programm anhand dessen Struktur. Ohne Angabe weiterer Argumente wird das Debian-Format erzeugt. Alternative Formate sind:

- **--to-tgz** Erzeugen des Tgz-Formats
- **--to-rpm** Erzeugen des Rpm-Formats
- **--to-deb** Erzeugen des Deb-Formats (Voreinstellung)
- **--to-slp** Erzeugen des Slp-Formats

Das nachfolgende Beispiel wandelt das Linuxfibel-Basis-RPM-Paket in das Tgz-Format:

```
user@sonne> alien --to-tgz linuxfibel_basis-0.6-0.i386.rpm
Warning: alien is not running as root!
Ownerships of files in the generated packages will probably be messed up.
linuxfibel_basis-0.6.tgz generated
```

Die Warnung ist nicht weiter tragisch. Sie weist nur darauf hin, dass die erzeugte Datei nicht Root gehört und somit ein Sicherheitsrisiko darstellen könnte (da nicht nur Root das Paket bearbeiten darf).

Das Erzeugen von deb-, rpm- oder slp-Formaten bleibt dann aber einzig Root vorbehalten. Um ein Format von

alien lesen bzw. schreiben zu können, muss das entsprechende Werkzeug (bspw. rpm oder debmake) installiert sein. Natürlich kann eine solche Konvertierung nicht sämtliche Informationen sauber in ein Paket integrieren, die diese »normalerweise« enthalten würde. Gerade die Erzeugung der ausgereiften rpm- oder deb-Formate aus einem Tape Archiv kann nur eine saubere (De)Installation der Dateien garantieren. Abhängigkeiten o.Ä. bleiben vollends unberücksichtigt:

```
root@sonne> alien --to-rpm linuxfibel_basis-0.6.tgz
linuxfibel_basis-0.6.2.noarch.rpm generated
```

Zumindest die fehlende Paketbeschreibung kann bei Konvertierung aus dem tgz-Format per Kommandozeile ergänzt werden. Geben Sie hierzu die Option **--description=** ***Text*** an.

Auch können Sie das Paket unverzüglich installieren, indem Sie die Option **-i** verwenden.

Bibliotheken

Mit dem Einspielen von Bibliotheken ins System ist es oft noch nicht getan. Was nützt es, wenn ein Programm abbricht, weil es »seine« Bibliotheken nicht finden kann?

Zunächst bedarf es eines kleinen Ausflugs, wie in Unix der Umgang mit Bibliotheken erfolgt. Nachfolgend diskutieren wir die Möglichkeiten, um Fehlerausgaben, wie die Folgende, zu vermeiden:

```
user@sonne> ./ MyProgram
ld.so: MyProg: fatal: libMylib.so: can't open file
```

Statisch, dynamisch, shared

Ein erster naiver Erklärungsversuch könnte »statisch« mit »unveränderbar« gleich setzen. Hieraus ließe sich schlussfolgern, dass dynamisch gelinkte Programme sich ändern (können), doch zum Glück ist dem dann doch nicht so.

Gebräuchlicher ist die Umschreibung eines statisch gelinkten Programms als ein Programm, das zur Laufzeit eine konstante Größe an Hauptspeicher verkonsumiert. Aus diesem Blickwinkel betrachtet, wäre der Speicherbedarf dynamischer gelinkter Programme änderbar. Und tatsächlich entspricht diese Betrachtungsweise dem wahren Wesen statisch bzw. dynamisch gelinkter Programme. Dem statisch gelinkten Programm wurde die komplette Funktionalität bereits zur Kompilierzeit einverleibt. Es ist unabhängig von jeglichen Bibliotheken und somit prinzipiell auf jedem System lauffähig, das das Programmformat unterstützt. Das dynamisch gelinkte Programm beinhaltet anstatt »verbreiteter« Funktionen nur die Schnittstellen zu diesen; die Funktionen selbst liegen in Form von Bibliotheksroutinen vor. Es ist klar, dass das dynamisch gelinkte Programm i.d.R. weniger Platz auf der Festplatte beansprucht, als es ein statisch gelinktes Pedant tut, das dieselbe Funktionalität erfüllt.

Der Nutzen der dynamischen Realisierung wird erst recht deutlich, wenn man die Gemeinsamkeiten zahlreicher Programme betrachtet. So greift nahezu jedes Programm auf Routinen der Ein- und Ausgabe zu, die meisten Programme verwenden eine dynamische Speicherverwaltung usw. Kandidaten für Funktionen, die besser in Bibliotheken realisiert werden sollten, finden sich reichlich und tatsächlich existieren in einem Linuxsystem nur noch wenige statisch gelinkte Programme (der dynamische Linker ist eines davon).

Die dynamisch gelinkten Programme bedingen jedoch die Anwesenheit der benötigten Bibliotheken. Je nach Zeitpunkt, wann eine Bibliothek geladen wird, werden »dynamisch ladbare« und »geteilte« (engl.: shared) Bibliotheken unterschieden. Diese begriffliche Trennung ist etwas unglücklich, da sich beide Bibliothekenstypen nicht unterscheiden. Ob sie nun als dynamisch ladbare oder als shared Bibliothek fungieren, hängt einzig vom Funktionsaufruf des Programms ab. Lädt dieses eine Bibliotheksfunktion als »shared«, so wird die Bibliothek zum Zeitpunkt des Ladens des Programms in den Hauptspeicher geladen (falls ein anderes Programm diese Bibliothek bereits geladen hatte, entfällt dieser Schritt). Im Falle eines »dynamischen« Aufrufs erfolgt das Laden erst, wenn tatsächlich auf eine Funktion aus dieser Bibliothek zugegriffen wird.

Um noch einmal auf den Unterschied zwischen statisch und dynamisch gelinkten Programmen einzugehen, so lässt sich ein weiterer Vorteil der Bibliotheken nennen. Sie sind durch andere Versionen austauschbar, solange die

Schnittstellen der Funktionen und die Datenstrukturen beibehalten werden. Somit ist es bspw. möglich, einer grafischen Anwendung unter X ein anderes Aussehen zu verleihen, indem einfach die zuständige Grafikbibliothek gewechselt wird.

Ein Übel verbirgt sich dennoch bei dynamisch gelinkten Programmen. Ohne die benötigten Bibliotheken sind sie nicht lauffähig. Der nächste Schritt, die Bibliotheken ins System einzuspielen, genügt meist nicht. Das Programm muss sie auch noch finden können...

Wie sucht ein Programm dynamische Bibliotheken?

Das Aufspüren der Bibliotheken obliegt einem dynamisch gelinkten Programm selbst. Hierzu beinhaltet es den Dateinamen des dynamischen Linkers **ld-linux.so.2** (Glibc2) zu Beginn seines Programmkodes:

```
user@sonne> strings /lib/cat | head -1
/lib/ld-linux.so.2
```

ld-linux.so.2 selbst zeigt als symbolischer Link auf die aktuelle Version des Linkers.

Startet das Programm, lädt der Kernel diesen dynamischen Linker. Jede dynamische Bibliothek, die ein Programm verwendet, hinterlässt in diesem seinen so genannten »so-Namen«. Dabei handelt es sich um den Präfix **lib**, gefolgt vom **Namen** der Bibliothek, wiederum gefolgt von **.so.** und der **Hauptversionsnummer** (bspw.: libXt.so.6). Im Dateisystem selbst ist eine dynamische Bibliothek unter ihrem so-Namen, gefolgt von einer *Nebenversionsnummer* und einer optionalen *Patchnummer* gespeichert (bspw.: libXt.so.6.1.1).

Der dynamische Linker akzeptiert die Anforderung einer Bibliothek, wenn die so-Namen übereinstimmen. Denn eine abweichende Hauptversionsnummer bedeutet eine geänderte Schnittstelle, womit ein Programm mit einer solchen Bibliothek nicht zusammenarbeiten kann.

Der Linker lädt nun die geforderte Bibliothek - falls sie nicht schon dort ist - in den Hauptspeicher. Er kennt nun die relative Lage (Adresse) der Funktionen und Variablen, die das Programm aus der Bibliothek benötigt und manipuliert die Verweise in der »Global Offset Table« (globale Variablen) bzw. in der »Procedure Linkage Table« (Funktionen) des Programms.

Zur Suche einer Bibliothek bedient sich der Linker mehrfacher Mechanismen. Zunächst darf ein dynamisch gelinktes Programm auch die kompletten Pfade zu den Bibliotheken eingebunden haben, womit der Linker genau diese verwenden wird. Ein solches Vorgehen erschwert u.a. die Installation von Programmen einer Distribution auf einem »Fremdsystem«, sodass i.d.R. darauf verzichtet wird.

Als nächstes betrachtet der Linker den Inhalt verschiedener globaler Umgebungsvariablen. Selten von den Distributionen angewandt - aber möglich - ist die Belegung der Variablen **LD_PRELOAD** mit den vollständigen Pfadangaben zu Bibliotheken, die **vor** allen anderen zu laden sind. Somit ist es denkbar, konkrete Funktionen aus anderen Bibliotheken zu überschreiben, da der Linker eine einmal geladene Funktion nicht wieder überschreibt. Aus Sicherheitsgründen wird das Verfahren bei setuid/setgid-Programmen nur zugelassen, wenn Benutzer-ID (UID) und effektive Benutzer-ID (EUID) - bez. Gruppen-ID (GID) - und effektiver Gruppen-ID (EGID) übereinstimmen. In der Praxis setzen vor allem Speicherleck-Prüfprogramme auf den PRELOAD-Mechanismus auf, um (De) Allokationsroutinen durch eigene Versionen zu überladen.

Vor den Standardpfaden werden auf der Suche nach dynamischen Bibliotheken anschließend die in der Umgebungsvariablen **LD_LIBRARY_PATH** aufgeführten Pfade betrachtet.

Schließlich kommen die Bibliotheken aus den Standardpfaden zum Zuge. Aus Gründen der Effizienz erfolgt die Suche nicht in den Pfaden aus /etc/ld.so.conf sondern einzig in der Datei **/etc/ld.so.cache**, die eine Liste aller Bibliotheken aus diesen Pfaden enthält. Das Erzeugen dieser Cache-Datei obliegt dem Kommando **ldconfig**, das uns später noch interessieren soll.

Test mit ldd

Natürlich wird der Aufruf eines Programms, das eine Bibliothek vermisst, den Namen dieser ausgeben, womit

fehlende Abhängigkeiten automatisch ans Licht gelangen. Darüber hinaus gestattet das Kommando **ldd** aber auch einen Einblick in die bereits erfüllten Voraussetzungen, indem es zu den auf der Kommandozeile angegebenen Programmen oder Bibliotheken die benötigten Bibliotheken ausgibt:

```
user@sonne> ldd / bin/ ls
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40024000)
        librt.so.1 => /lib/librt.so.1 (0x40028000)
        libc.so.6 => /lib/libc.so.6 (0x4003a000)
        libpthread.so.0 => /lib/libpthread.so.0 (0x40167000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
user@sonne> ldd -v / lib/ libc.so.6
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)

        Version information:
        /lib/libc.so.6:
                ld-linux.so.2 (GLIBC_2.1.1) => /lib/ld-linux.so.2
                ld-linux.so.2 (GLIBC_2.1) => /lib/ld-linux.so.2
                ld-linux.so.2 (GLIBC_2.2) => /lib/ld-linux.so.2
                ld-linux.so.2 (GLIBC_2.0) => /lib/ld-linux.so.2
```

Die im Beispiel angewandte Option **-v** gibt einige erweitere Informationen preis.

Konfiguration des dynamischen Linkers

Wie bereits erwähnt, wird die Arbeitsweise des Linkers sowohl durch Umgebungsvariablen als auch durch die Datei /etc/ld.so.cache beeinflusst. Letztere Datei wiederum wird durch das Programm **ldconfig** erzeugt, welches in /etc/ld.so.config konfiguriert wird.

Somit bieten sich bei der Installation neuer Bibliotheken zwei Wege an, damit der Linker diese in Zukunft finden kann:

1. Erzeugen einer neuen Datei /etc/ld.so.cache (ldconfig)
2. Setzen der Variablen LD_LIBRARY_PATH

Die Belegung von LD_LIBRARY_PATH ist die »schnelle Methode«, wenn ein Programm mal eben getestet werden soll und das Laden der Bibliothek durch den Linker scheitert, weil dieser sie nicht finden kann. Ansonsten hängt diesem Vorgehen derselbe Nachteil an, der zum Caching-Mechanismus führte: eine Suche im Dateisystem kostet Zeit. Ein Vorteil wäre dennoch zu nennen: mittels LD_LIBRARY_PATH darf jeder Benutzer Einfluss auf den Linker nehmen, während eine Manipulation von /etc/ld.so.cache einzig dem Administrator vorbehalten ist. Die Syntax zur Belegung der Variablen erfolgt analog zur Variablen PATH.

Um die Cache-Datei /etc/ld.so.cache des Linkers zu aktualisieren, genügt nach der Installation neuer Bibliotheken zumeist ein Aufruf von **ldconfig**:

```
root@sonne> ldconfig
```

ldconfig durchsucht zuerst die Verzeichnisse /usr/lib und /lib und nachfolgend alle in der Datei **/ etc/ ld.so.conf** aufgeführten Pfade. Zu jeder gefundenen neuen Bibliothek erzeugt **ldconfig** selbsttätig den symbolischen Link des so-Namens auf den Bibliotheksnamen (also bspw. libX.so.6 als Link auf libX.so.6.1) und aktualisiert die Datei **/ etc/ ld.so.cache**.

Das Erzeugen des Links kann mittels der Option **-l** verhindert werden. Mit **-n *Pfad[e]*** werden einzig die angegebenen Verzeichnisse betrachtet (also auch nicht /usr/lib und /lib).

Eine Konfiguration ist - wenn überhaupt - nur durch Hinzufügen weiterer Suchpfade in die Datei /etc/ld.so.conf erforderlich. Jeder Pfad erscheint in dieser auf einer eigenen Zeile:

```
root@sonne> vi / etc/ ld.so.conf
/usr/X11R6/lib
/usr/local/lib
```

Vergessen Sie nicht, nach Manipulation dieser Datei die Änderung durch Aufruf von **ldconfig** auch zu aktivieren. Auch sollten Sie beachten, dass die Pfade in der aufgeführten Reihenfolge durchsucht und bei gleichnamigen Bibliotheken stets nur die »erste« gefunden wird.

Wohl eher für Entwickler interessant ist die Datei /etc/ld.so.preload, die eine Liste durch Leerzeichen getrennter Bibliotheken enthält, welche vor allen anderen Bibliotheken geladen werden. Analog zur Variablen LD_PRELOAD ist somit das Überschreiben einzelnen Funktionen möglich.

Die Login-Verwaltung

Übersicht

Unter Login-Verwaltung versuchen wir verschiedenen Varianten aufzuzeichnen, wie einem Benutzer der Zugang zum System ermöglicht werden kann.

Jede Standardinstallation kommt entweder mit einem Konsolen-Login oder einer grafischen Anmeldung daher. Im Fall des tristen Konsolenmodus gelangt bei nahezu allen Distributionen ein Programm namens mingetty zum Einsatz. Aber es existieren weitere Varianten eines initialen »Terminals«, welche das Anmelden am System steuern, Gettys, die ihre Stärken vor allem bei Dial-in- und Modem-Anmeldevorgängen beweisen.

An der grafischen Anmeldung schwingt ein X Display Manager das Zepter. In Zeiten von KDE delegieren die verbreiteten Distributionen die Aufgaben dem kdm, seltener wird man noch dem xdm begegnen und der gdm kontrolliert die Anmeldung an ein RedHat-Linux. In diesem Abschnitt sollen nur wenige einleitende Worte die Mächtigkeit des Konzepts verdeutlichen, die Display-Manager werden im Kapitel Login-Manager tiefgründig diskutiert.

Neben den den Anmeldevorgang steuernden Programmen gebührt der Datei /etc/login.defs eine wichtige Rolle. Wie der Name schon verdeutlicht, steuert sie in gewissem Maße das Verhalten der Startup-Programme.

Letztlich tragen auch /etc/securetty und Passwort zum Gelingen oder Scheitern einer Anmeldung bei.

Die Datei / etc/ issue

Alle im folgenden Abschnitt beschriebenen »Getty« sind in der Lage, vor der Login-Aufforderung eine Mitteilung auf den Bildschirm auszugeben. Den Text dazu beziehen sie, wenn nicht explizit etwas anderes angegeben wurde, aus der Datei **/ etc/ issue**. In dieser Datei sind verschiedenste Ersatzdarstellungen zulässig, z.B. um den Rechnernamen oder die aktuelle Zeit einzublenden. Allerdings unterscheiden sich die »Gettys« hinsichtlich der unterstützten Platzhalter. Im Falle von mingetty könnte der Inhalt der Datei so aussehen:

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

Willkommen auf Rechner \n, System \s Version \r!
```

Der Willkommensgruß kann diverse Platzhalter umfassen, die den Text in begrenztem Umfang dynamisch anpassen:

\ b

Fügt die Baudrate der aktuellen Verbindung ein

\ d

\ l

Name des Terminals, an dem mingetty aktiv ist

\ m

Architektur des Rechner

\ n

Voller Rechnername (inklusive Domainname)

\ o

Domainname

\ r

Betriebssystemversion (Kernelrelease)

\ s

Betriebssystemname

\ t

Aktuelle Zeit

\ u

Anzahl am System angemeldeter Benutzer

\ v

Betriebssystemversion (Übersetzungszeit des Kernels)

Der Platzhalter \b wird nur von Gettys verstanden, die auch serielle Schnittstellen überwachen können.

Bei Verwendung von **fbgetty** muss anstatt des Backslashs (\) das Prozentzeichen verwendet werden (%d, %u,...). Da **fbgetty** sich noch in einem frühen Entwicklungsstadium befindet, wird es in der weiteren Diskussion keine Rolle spielen.

Gib mir ein Terminal

In historischen Zeiten der Mainframes ermöglichten physikalische Terminals den Zugang zu den sündhaft teuren Großrechnern. Ein Terminal war nichts weiter als eine Einheit aus Bildschirm und Tastatur, das über einen seriellen Anschluss am Rechner steckte. Als Software lief im Mainframe ein kleines Programm »Get's a TTY«, - oder kurz als **getty** bezeichnet, das die Anmeldung steuerte. Der Name »Getty« für Login-Programme ist geblieben, auch wenn physikalische Terminals heute nur noch in der Computerliteratur verbreitet sind.

Gettys tragen dafür Sorge, dass sich überhaupt jemand am System anmelden kann. Dazu überwachen sie serielle Geräte wie Virtuelle Terminals, Textterminals oder auch Modems. Ein **Getty** bringt die Login-Aufforderung auf den Bildschirm, es fragt nach dem Passwort und startet anschließend das Kommando **login**, das Benutzerkennzeichen und Passwort verifiziert und den Zugang zum Rechner gestattet bzw. ablehnt.

Gettys gibt es reihenweise und eines ist für einen Einsatzbereich besser geeignet, als es ein anderes vermag. Einige Gettys überwachen einzig die virtuellen Terminals, andere sind auf den Faxempfang spezialisiert. Die verbreiteten Vertreter der Gilde werden wir im folgenden Abschnitt kennen lernen. Zuvor seien die Aufgaben

genannt, für dessen Erledigung Gettys herangezogen werden:

- Ermöglichen einer Login-Sitzung (Passwortabfrage usw.)
- Einstellen der Parameter eines Terminals (Spalten- und Zeilenanzahl, Übertragungsgeschwindigkeit, Echo...)
- Ermöglichen von Einwahlverbindungen
- Faxempfang
- Voice-Boxen (Anrufbeantworter)

Automatischer Getty-Start

I.A. erwartet man von einem Getty, dass es mit den Starten des Systems das ihm zugedachte Device eröffnet und mit einer Login-Aufforderung aufwartet. Des Weiteren sollte es erneut starten, sobald eine Sitzung beendet wurde. Der begangene Weg, dies unter Linux zu erreichen, ist ein entsprechender Eintrag in der Datei /etc/inittab:

```
user@sonne> cat / etc/ inittab |  grep respawn
~~:S:respawn:/sbin/sulogin
1:123:respawn:/sbin/mingetty --noclear tty1
2:123:respawn:/sbin/mingetty tty2
3:123:respawn:/sbin/mingetty tty3
4:123:respawn:/sbin/mingetty tty4
5:123:respawn:/sbin/mingetty tty5
6:123:respawn:/sbin/mingetty tty6
#7:2:respawn:+/sbin/init.d/rx tty7
#  mo:23:respawn:/usr/sbin/mgetty -s 38400 modem
#  mo:23:respawn:/usr/lib/fax/faxgetty /dev/modem
#  I6:23:respawn:/usr/sbin/vboxgetty -d /dev/ttyI6
#  I7:23:respawn:/usr/sbin/vboxgetty -d /dev/ttyI7
```

Das obige System ist offensichtlich nur für ein Konsolenlogin konfiguriert (nicht zu Verwechseln mit einer Anmeldung übers Netz mit Programmen wie telnet oder ssh); als Getty gelangt **mingetty** zum Einsatz. Entscheidend ist die Angabe von **respawn**, die den Init-Prozess anweist, im Falle der Beendigung einer Sitzung das jeweilige Getty erneut zu starten (ein Getty-Prozess startet per exec ein Loginprogramm, welches wiederum bei erfolgreicher Anmeldung mittels exec bspw. eine Shell lädt).

Selbstverständlich steht es Root zu, ein Getty per Hand zu starten, jedoch kann er einen solchen Befehl nur absetzen, wenn auf dem entsprechenden Device (bspw. /dev/tty2, /dev/ttyS0...) noch kein Getty-Prozess am werkeln ist. Gerade bei von init kontrollierten Geräten genügt der gezielte Abschuss eines existierenden Prozesses herzlich wenig, da **init** selbst diesen ersetzen wird. Nur eine Änderung der Datei /etc/inittab mit anschließender Benachrichtigung von **init** (kill -1 1) wird hier weiterhelfen.

agetty

Das **Alternative Linux-Getty** ist das einfachste (bedingt) für Einwahlverbindungen geeignete Getty (da es keine Konfigurationsdatei verwendet). Allerdings besitzt es einen entscheidenden Nachteil: Wenn es selbst ein Modemdevice auf eingehende Verbindungen überwacht, ist das Gerät für ausgehende Verbindungen blockiert, selbst wenn noch kein Anruf entgegen genommen wurde. Aus diesem Grund wird **agetty** vornehmlich auf virtuellen Konsolen oder - wegen der Hardware gesteuerten Flusskontrolle - für den Anschluss eines physikalischen Terminals (Nullmodem...) angewandt.

```
agetty [Optionen] Baudrate[n] Port [Terminalemulation]
```

Die minimalen Argumente für »agetty« sind Port und eine Liste von zulässigen Baudraten. Port steht dabei für den Gerätenamen relativ zum Verzeichnis »/dev« oder für »-«. In letzterem Fall nimmt »agetty« an, dass seine Standardeingabe bereits mit einem Port verbunden ist. Bei Baudrate handelt es sich um eine komma-separierte Liste der unterstützten Geschwindigkeiten. Im Fall einer Modemverbindung startet »agetty« mit dem ersten angegeben Wert. Wird dieser von der Gegenstelle zurückgewiesen (brEAK), schaltet »agetty« auf die nächste Baudrate um.

Die Terminalemulation überschreibt den Wert der Shellvariablen TERM (wird oft initial von **init** gesetzt).

Des Weiteren werden folgende Optionen unterstützt:

-h

Aktiviert die Hardware gesteuerte Flusskontrolle

-i

Unterdrückt die Ausgabe des Inhalts von /etc/issue. Dies kann bei Modemverbindungen erforderlich sein, fa die Gegenseite nach konkreten Zeichenfolgen (Login, Password) sucht.

-f *Datei*

Verwendet die angegebene Datei anstatt /etc/issue

-I *Initstring*

Zeichenkette (zur Initialisierung eines Modems), die vor dem eigentlichen Senden an den Port geschickt wir

-l *Programm*

Alternatives Login-Programm (anstatt /sbin/login).

-m

»agetty« versucht die Baudrate anhand der CONNECT-Zeichenkette zu erkennen.

-n

»agetty« fragt nicht nach einem Login-Namen. Sinnvoll ist die Option bspw, wenn als Login-Programm sulo genutzt wird.

-t *Sekunden*

Wird innerhalb dieser Zeitspanne kein Nutzername eingegeben, bricht »agetty« ab.

-L

Das zu überwachende Terminal ist eine physikalische/virtuelle Konsole. »agetty« wartet nicht auf das Verbindungssignal (das bei Modemverbindungen eine neue Sitzung einleiten würde...).

-w

»agetty« wartet auf den Empfang eines Zeilenumbruchs oder Wagenrücklaufs, bevor es den Inhalt von /etc/issue an den Port sendet.

faxgetty

»Faxgetty« ist Bestandteil des Hylafax-Pakets zum Betrieb eines Faxservers unter Linux. Eine Abhandlung der Thematik findet der interessierte Leser im Kapitel »Netzwerk-Grundlagen« im Abschnitt Allgemeine Dienste, Faxserver.

getty und uugetty

Beide Programme stammen aus dem Paket »getty_ps«. »getty« dient der Anmeldung an der Virtuellen Konsole oder über ein physikalisches Terminal, während das Programm »uugetty« mit Einwahlleitungen umzugehen vermag.

Da das nachfolgend beschriebene »mgetty« die Funktionen beider Programme nicht nur in sich vereint, sondern mit weiteren Fähgkeiten aufwartet, werden »getty« und »uugetty« heute kaum noch verwendet. Auch wir werden

diesen Getty-Vertretern nicht weiter auf den Grund gehen.

mgetty und vgetty

»Mgetty« ist eine Art Multitalent in Verbindung mit Modems. Es ermöglicht sowohl einen Einwahlserver (Terminalserver oder PPP-Server) als auch den Versand und Empfang von Faxen. Zum Betrieb eines Anrufbeantworters über ein analoges Modem ist »vgetty« aus demselben Paket erforderlich. Beide Gettys werden wir im entsprechenden Abschnitt in Allgemeine Dienste im Kapitel »Netzwerk-Grundlagen« abhandeln.

mingetty

mingetty verwenden heute alle gängigen Distributionen, um den Anmeldevorgang an den virtuellen Konsolen zu steuern. Der Name deutet den minimalen funktionellen Umfang bereits an, sodass **mingetty** sich nicht eignet, um bspw. eine serielle Modemleitung anzusprechen.

mingetty erwartet als Argument das Terminal-Device. Optional ermöglicht **--long-hostname** die Anzeige des vollständigen Rechnernamens vor dem Login-Prompt (Voreinstellung ist der Name vor dem ersten Punkt). **--noclear** verhindert das Löschen des Bildschirm vor Anzeige der Login-Aufforderung.

In der Voreinstellung verwendet **mingetty** »/sbin/login« als Login-Programm. Mit **--login < Programm«** kann ein alternatives Programm und **--logopts < Optionen>** Parameter an dieses angegeben werden.

mingetty gibt den Inhalt der Datei /etc/issue (falls sie existiert) vor dem Login aus.

Damit sind schon alle Optionen von **mingetty** genannt; der Präfix ist Programm...

vboxgetty

»Vboxgetty« ist wichtigster Bestandteil eines auf ISDN basierten Anrufbeantworters. Da dieses Getty nur in Zusammenhang mit einem Anrufbeantworter von Interesse ist, erfolgt die Beschreibung erst im Kapitel »Netzwerk-Grundlagen« unter Allgemeine Dienste, ISDN-Anrufbeantworter.

Die Datei / etc/ motd

Der Inhalt dieser Datei (»Message of the day«) wird vom Kommando **login** im Anschluss an ein erfolgreichen Anmeldevorgang auf den Bildschirm ausgegeben. So hilft dem Administrator bswp., um allen Benutzern, die Zugang auf diesen Rechner haben, etwas mitzuteilen.

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

Am kommenden Montag wird der Rechner wegen Wartungsarbeiten vorübergehend abgeschalten. Ihre lokal gespeicherten
Daten stehen ihnen deshalb nicht zur Verfügung!
```

Die Datei / etc/ login.defs

Über diese Datei kann das Login-Verhalten der Shadow-Suite konfiguriert werden. Beachten Sie, dass die Parameter MOtd_FILE, DIALUPS_CHECK_ENAB, LASTLOG_ENAB, MAIL_CHECK_ENAB, OBSCURE_CHECKS_ENAB, PORTTIME_CHECKS_ENAB, CONSOLE, SU_WHEEL_ONLY, CRACKLIB_DICTPATH, PASS_CHANGE_trIES, PASS_ALWAYS_WARN, MD5_CRYPT_ENAB, CONSOLE_GROUPS, ENVIRON_FILE, NOLOGINS_FILE, ISSUE_FILE und PASS_MIN_LEN bei Verwendung von Pluggable Authentication Modules von der dortigen Konfiguration überschrieben werden.

Eine Zeile der Datei beginnt mit dem Konfigurationsparameter. Ihm folgt, durch Leerzeichen oder Tabulator(en) getrennt, der Wert. Ein Doppelkreuz leitet einen Kommentar ein, der mit einem Zeilenumbruch endet.

Die nachfolgende Beschreibung enthält nur eine Auswahl aller Parameter. Im Wesentlichen verzichten wir auf die Erläuterung zu Parametern, die in der derzeitigen Implementierung zwar vorhanden, aber deren Anwendung nicht empfohlen wird.

CHFN_AUTH

Ist der Wert »yes«, so fordern die Kommandos chsh und chfn die Eingabe des Passworts.

CHFN_RESTRICT

Dieser Parameter bestimmt, welche Einträge des GCOS-Feldes der /etc/passwd ein normaler Benutzer mit H von chfn ändern darf. Die vier erlaubten Buchstaben sind: **f** für den vollständigen Namen, **r** für die Raumnummer, **w** für die Bürotelefonnummer und **h** für die private Telefonnummer. Fehlt dieser Parameter darf einzig Root Änderungen vornehmen.

CLOSE_SESSIONS

Bei Verwendung von Pluggable Authentication Modules ermöglicht dieser Parameter dem Kommando login, das Ende der Sitzung zu warten und anschließend die PAM-Ressourcen freizugeben (durch Aufruf von pam_close_session(...)). Die meisten PAM-Module räumen selbst auf, aber eben nicht alle (u.a. Kerberos).

CONSOLE

Root darf sich nur an den hier erwähnten Terminals anmelden. Neben der direkten Angabe zulässiger Term kann auch der Pfad zu einer Datei angegeben werden, die die Terminals enthält. Üblich ist /etc/securetty.

CRACKLIB_DICTPATH

Pfad zu den Cracklib-Wörterbüchern.

DEFAULT_HOME

Die Werte »true« bzw »false« legen fest, ob ein Benutzer sich anmelden darf, falls sein Home-Verzeichnis r verfügbar ist (bspw. wenn es per NFS gemountet wird, die Verbindung zum NFS-Server aber nicht hergeste werden kann).

DIALUPS_CHECK_ENAB

Eine Datei /etc/dialups kann Terminals enthalten, an denen auf eingehende Anrufe gewartet wird. Zu jeden Terminal kann ein Passwort festgesetzt werden. Steht nun der Parameter auf »yes« wird die Abfrage dieses Passworts vorgenommen.

ENVIRON_FILE

Die hier angegebene Datei kann einen Satz vordefinierter Umgebungsvariablen enthalten.

ENV_PATH

Hier wird die initiale Belegung der Variablen PATH vorgenommen. Dieser Eintrag ist zwingend erforderlich, noch keine Shell aktiv ist und somit noch kein Suchpfad für Programme existiert (login muss bspw. die Shel selbst finden).

ENV_ROOTPATH

Initiale Belegung von PATH für Root.

ERASECHAR

Das angegebenem Zeichen ermöglicht das Löschen in einem Terminal. Fehlt die Angabe ist [Backspace] da Löschzeichen.

FAILLOG_ENAB

Bei »yes« werden fehlgeschlagende Anmeldeversuche in der Datei /var/log/faillog protokolliert.

FAIL_DELAY

Nach einem fehlgeschlagenen Login-Versuch wird die anegegebene Zeitspanne (in Sekunden) gewartet, be ein erneutes Login-Prompt erscheint.

GID_MIN, GID_MAX

Minimaler und maximaler Wert für eine Gruppennummer, die das Kommando groupadd automatisch kalkuli darf.

ISSUE_FILE

Pfadname zu einer Datei, deren Inhalt vor dem Loginprompt angezeigt wird.

LASTLOG_ENAB

Steht hier »yes«, werden nach erfolgreichem Anmelden Informationen zum Zeitpunkt der letzten Anmeldun und ggf. zu zwischenzeitlichen fehlgeschlagenen Anmeldeversuchen ausgegeben.

LOGIN_RETRIES

Bei fehlgeschlagenem Anmeldeversuch lässt das Kommando login die angegebene Anzahl erneuter Versuch zu, bis es sich selbst beendet. Bei lokalen Login-Konsolen startet i.d.R. ein **getty** anschließend erneut den Anmeldevorgang; bei einer Anmeldung übers Netz wird jedoch meist die Verbindung getrennt.

LOGIN_TIMEOUT

Anzahl Sekunden, die login auf die Eingabe eines Passworts wartet. Nach Ablauf der Zeitspanne gilt der Ver als gescheitert.

MAIL_CHECK_ENAB

Steht der Wert auf »yes«, wird ein Benutzer nach dem Login über den Status seiner Mailbox informiert.

MAIL_DIR

Enthält das Verzeichnis mit den Mailboxen der Benutzer. Das Benutzerkennzeichen wird automatisch ergänz Weicht die Namensgebung von diesem üblichen Schema ab, muss die Mailbox-Datei mit MAIL_FILE angege werden.

MAIL_FILE

Enthält die Datei mit der Mailbox eines Benutzers. Sie muss im Heimatverzeichnis des Benutzers liegen; der Pfad zum Heimatverzeichnis wird automatisch ergänzt. Dieser Parameter sollte nicht gleichzeitig mit MAIL_I verwendet werden.

MD5_CRYPT_ENAB

Bei »yes« wird das Passwort nicht per herkömmlichen DES-Algorithmus verschlüsselt, sondern mittels eines MD5-Verfahrens. Somit sind Passwortlängen bis zu 256 Zeichen möglich.

MOTD_FILE

Enthält den vollständigen Pfad zu einer Datei mit der »Nachricht des Tages«. Mehrere Dateien können - per Doppelpunkt getrennt - angegeben werden. Existieren sie, wird ihr Inhalt nach dem erfolgreichen Anmelden angezeigt.

NOLOGINS_FILE

OBSCURE_CHECKS_ENAB

Steht der Wert auf »yes«, wird ein neues Passwort erst akzeptiert, nachdem es einfachen Test unterzogen wurde (minimale Passwortlänge). Führt Root das Kommando passwd aus, wird die Prüfung ausgesetzt.

PASS_ALWAYS_WARN

Ändert Root ein Passwort, das einer Überprüfung mittels den Mechanismen von OBSCURE_CHECKS_ENAB r standhalten würde, wird er gewarnt, falls der Wert des Parameters auf »yes« steht.

PASS_CHANGE_trIES

Anzahl Versuche, das Passwort zu ändern, bevor passwd abbricht.

PASS_MIN_DAYS, PASS_MAX_DAYS

Minimale/maximale Zeitspanne, die zwischen zwei Passwortänderungen vergehen muss/darf.

PASS_MIN_LEN, PASS_MAX_LEN

Mindestlänge bzw. maximale Länge eines Passworts.

PASS_WARN_AGE

Ab so vielen Tagen vor Erreichen der PASS_MAX_DAYS wird ein Benutzer gewarnt, dass sein Passwort demnächst abläuft.

QMAIL_DIR

Gibt den Pfad zum Mailverzeichnis bei Verwendung von qmail an.

QUOTAS_ENAB

Steht hier »yes«, so werden die Werte des GCOS-Feldes der Datei /etc/passwd verwendet, um Limits den jeweiligen Benutzer zu setzen. Selbstverständlich wirkt der Eintrag nur, wenn die /etc/passwd die entsprechenden Angaben auch enthält.

SULOG_FILE

In diese Datei werden Aktivitäten von **su** aufgezeichnet. Wird die Datei nicht angegeben, findet keine Protokollierung statt.

SYSLOG_SG_ENAB

Steht hier »yes«, werden alle Aufrufe von **sg** über den syslogd protkolliert.

SYSLOG_SU_ENAB

Steht hier »yes«, werden alle Aufrufe von **su** über den syslogd protkolliert.

TTYGROUP

Das Login-Terminal kann mit der angegebenen Gruppe als besitzende Gruppe gestartet werden. In Verbind mit TTYPERM lassen sich somit die Rechte detaillierter steuern.

TTYPERM

Die Rechte, mit denen ein Login-Terminal versehen wird. Fehlt der Eintrag, so werden die Rechte intern au

gesetzt. Somit sind andere Benutzer bemächtigt, auf das Terminal zu schreiben (bspw. mit dem Kommando write, talk...).

TTYTYPE_FILE

Hier wird der Pfad zu einer Datei mit den Terminal-Spezifikationen angegeben. Diese Datei enthält Zeilen, die einem Terminal einen konkreten Typ (»Terminalemulation«) zuordnen. So sind lokale Login-Konsolen (tty1..tty6) meist vom Typ »linux«; Pseudoterminals (ttyp*X*) zur Anmeldung übers Netz verwenden häufig die »vt100«-Emulation.

UID_MIN, UID_MAX

Minimale bzw. maximale Nummer, die vom Kommando **useradd** bei der automatischen Vergabe der UID gewählt werden kann.

ULIMIT

Dateien dürfen maximal diese Größe annehmen.

UMASK

Die Voreinstellung der Zugriffsrechte für neu erstellte Dateien und Verzeichnisse; siehe umask im Abschnitt Zugriffsrechte.

Die Datei / etc/ securetty

Haben Sie schon einmal versucht, unter dem Nutzerkennzeichen **root** eine Verbindung zu einem Rechner über Telnet zu initiieren? Dann habe Sie schon Bekanntschaft mit der Ausgabe »Keine Berechtigung« gemacht?

Mit der Sicherheit der Administrator-Zugangs steht und fällt die Sicherheit Ihres gesamten Systems. So wird es nicht verwundern, dass man besondere Maßnahmen getroffen hat, um die Möglichkeiten zum Root-Zugang zu erschweren. Die Datei /etc/securetty enthält nun die »vertrauenswürdigen« Terminal, an denen ein Root-Zugang gewährt wird:

```
user@sonne> cat / etc/ securetty
tty1
tty2
tty3
tty4
tty5
tty6
```

Obige Konstellation gestattet ein Root-Login nur von einer der virtuellen Konsolen aus. Ein Versuch über ein Pseudo-Terminal (z.B. über das Netz) wird somit immer abgewiesen.

/etc/securetty wird vom Kommando **login** ausgewertet.

Die Datei / etc/ shells

Wenn ein Benutzer sich erfolgreich am System anmeldet, dann startet die in der Datei /etc/passwd fest gelegte Shell. Ein Benutzer hat nun die Möglichkeit, eine andere so genannte Default-Shell einzustellen.

Diese Loginshell ist nun ein zentraler Angelpunkt, denn alle Prozesse, die ein Benutzer im Laufe der Sitzung kreiert, sind Abkömmlinge des die Shell ausführenden Prozesses. Eine mangelhafte implementierte Shell könnte somit unter Umständen die Stabilität des gesamten Systems gefährden.

Aus diesem Grund werden alle Shells, die ein Benutzer als seine default-Shell verwenden darf, in der Datei **/ etc/ shells** zusammengefasst. Eine dort nicht erwähnte Shell wird somit niemals vom Kommando **chsh** akzeptiert werden. Beachten Sie, dass die Shells mit vollständigem Pfad anzugeben sind:

```
user@sonne> cat / etc/ shells
/bin/bash
/bin/csh
/bin/false
/bin/sh
/bin/tcsh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/passwd
/usr/bin/tcsh
/usr/bin/zsh
```

X-Login

Passwort

Die Benutzerverwaltung

Übersicht

In diesem Abschnitt erfahren Sie die notwendigen Maßnahmen zur Administration von Benutzerzugängen auf dem System. Nach dem Kennenlernen der wichtigen Dateien widmen wir uns den Möglichkeiten zum Anlegen und Löschen von Zugängen. Da neben den allgemeinen Werkzeugen eine Reihe distributionspezifischer Administrationshilfen existieren, finden auch diese Erwähnung. Weiterhin besprechen wir Kommandos, die die einzelnen Einträge der Passwortdatei manipulieren. Abschließend erfahren Sie, wie Sie einzelnen Benutzern Sonderrechte (z.B. Root-Rechte) für bestimmte Zwecke einräumen können.

Die Datei / etc/ passwd

Die Daten zur Benutzerverwaltung findet man in /etc/passwd. 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):

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

Die Einträge bedeuten:

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 (GCOS)

Das Feld **kann** mehrere Einträge enthalten. Dazu zählen:

- Vollständiger Name des Benutzers
- Telefonnummer des Benutzers

- Zimmernummer des Benutzers

Diese Angaben werden von zahlreichen Programmen wie finger, mail abgefragt. Zusätzlich können, per Kor getrennt, 3 weitere Angaben enthalten sein:

- **umask= ...** - Die Maske für neu erstellte Dateien
- **pri= ...** - Ein Nice-Faktor für Prozesse
- **ulimit= ...** - Verschiedene Beschränkungen der Systemressourcen

Home

Startverzeichnis nach Login

Shell

Default-Shell

Die Datei / etc/ shadow

Dass das verschlüsselte Passwort in der /etc/passwd steht, wird man nur noch auf älteren Systemen vorfinden.

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

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

Die Einträge bedeuten:

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 das Passwort trotz Ablauf der MaxD?

Dis

Bis zu diesem Tag (gezählt ab 1.1.1970) ist dieser Account gesperrt

Res

Reserve, Feld wird derzeit nicht ausgewertet

Konvertieren der Einträge

Eine manuelle Konvertierung der Einträge aus der Datei /etc/passwd in die Datei /etc/shadow und umgekehrt kann aus drei Gründen notwendig sein:

1. Sie verwenden noch immer das herkömmliche Passwortsystem, d.h. mit Speicherung der verschlüsselten Passwörter in der /etc/passwd, und möchten nun auf das Shadow-Passwort-System umstellen.
2. Sie sind ein Verfechter der Konsole und verwalten ihre Benutzer in der /etc/passwd von Hand.
3. Aus irgendeinem Grund möchten Sie vom Shadow-Passwort-System auf das herkömmliche zurückstellen.

Es ist sicher schwierig und in Systemen mit mehreren hundert Benutzern vielleicht sogar unmöglich, die Konsistenz von /etc/passwd und /etc/shadow manuell zu gewährleisten. Zum Glück existieren zwei Kommandos, die die Konvertierung der Formate automatisch vornehmen:

```
root@sonne> pwconv
```

pwconv vergleicht die Einträge aus der Datei /etc/passwd mit denen in der Datei /etc/shadow. Findet das Programm Unstimmigkeiten, wird die /etc/shadow so manipuliert, dass sie dem Stand der /etc/passwd entspricht. D.h. zu jedem Eintrag der Passwortdatei existiert anschließend ein zugehöriger Eintrag in der Shadowdatei, wobei eventuell vorhandene Passwörter in der /etc/passwd durch ein »x« ersetzt und diese in der /etc/shadow gespeichert werden.

```
root@sonne> pwunconv
```

pwunconv überträgt die verschlüsselten Passwörter aus der Datei /etc/shadow in die Datei /etc/passwd und löscht anschließend die Shadowdatei.

Eventuell ist eine Überprüfung der Konsistenz der beiden Dateien notwendig. Das Kommando **pwck** gibt eventuelle Unstimmigkeiten aus und fordert ggf. zu Korrekturen auf:

```
root@sonne> pwck
user man: directory /var/catman does not exist
user fixadm: program /bin/ksh does not exist
user fib: program /bin/ksh does not exist
pwck: no changes
```

Das Verzeichnis / etc/ skel

Selbst der Unix-Kenner ist mit der Konfiguration aller wichtigen Programme oft überfordert. Deswegen liegen solchen Programmen meist Konfigurationsdateien bei, die eine arbeitsfähige Grundkonfiguration bieten. Der Systemverwalter kann solche Dateien, die er für die Anwender als sinnvoll erachtet, in das Verzeichnis /etc/skel kopieren. Beim Anlegen eines neuen Zugangs kopiert man dann dessen gesamten Inhalt ins Heimatverzeichnis des neuen Benutzers.

Die Namen der Konfigurationsdateien beginnen üblicherweise mit einem Punkt, um sie beim normalen Listing zu verbergen:

```
root@sonne> ls / etc/ skel
root@sonne> ls -a / etc/ skel
.           .dayplan.priv  .kermrc     .tex          .xinitrc
..          .dvipsrc       .lyxrc      .uitrc.console .xserverrc.secure
.Xdefaults  .emacs         .muttrc     .uitrc.vt100  .xsession
.Xmodmap    .exrc          .nc_keys    .uitrc.vt102  .xtalkrc
.Xresources .gimprc        .profile    .uitrc.xterm  .zsh
```

```
.bash_history  .grok          .seyon      .urlview
.bashrc        .hotjava       .stonxrc    .xcoralrc
.dayplan       .jazz          .susephone  .xfm
```

Anlegen neuer Benutzer

Das Anlegen neuer Benutzer kann prinzipiell auf drei Wegen erfolgen:

- mittels reiner Handarbeit
- mit Hilfe der allgemeinen Werkzeuge
- durch Verwendung distributionseigener Verwaltungstools

Die Handarbeit

1. Zunächst bearbeitet man mit einem Editor die Datei /etc/passwd. Als Editor eignet sich **vipw**, da das Kommando gleichzeitig die notwendige Dateisperre setzt und somit eine Mehrfachbearbeitung ausschließt (der Editor arbeitet tatsächlich auf einer Kopie und schreibt erst beim Speichern das Original). Der sich hinter dem Kommando verbergende Editor kann mittels der Shellvariable VISUAL bzw. EDITOR eingestellt werden. Erst wenn beide Variablen nicht gesetzt sind, wird der vi verwendet. Bei manuellem Eintrag ist insbesondere auf die Eindeutigkeit von Benutzerkennung und die UID zu achten.

```
root@sonne> vipw
...
user:x:501:100:Testuser:/home/user:/bin/bash
newuser:x:502:100:Neuer Benutzer:/home/newuser:/bin/bash
-- INSERT --                          47,55         Bot
```

2. Bei Verwendung des Shadow-Passwort-Systems ist nun die Datei /etc/shadow anzupassen. Dies geschieht am einfachsten mittels eines Aufrufs von pwconv. Um von den Default-Einstellungen abweichende Werte zu verwenden, kann auch eine manuelle Bearbeitung notwendig sein. Hier kann das Kommando **vipw -s** genutzt werden.

```
root@sonne> pwconv
```

3. Das Passwort des Benutzers ist zu setzen.

```
root@sonne> passwd newuser
New password:
New password (again):
Password changed
```

4. Das Heimatverzeichnis des neuen Benutzers ist anzulegen. Um dem Anwender eine voreingestellte Konfiguration zur Verfügung zu stellen, sollte der Inhalt des Verzeichnisses /etc/skel in dessen Home kopiert werden.

```
root@sonne> mkdir /home/newuser
root@sonne> cp -R /etc/skel ~newuser
```

Allgemeine Werkzeuge

Hierbei ist zwischen Shadow- und herkömmlichen Passwort-Systemen zu unterscheiden. Die traditionelle Passwortverwaltung verwendet das Kommando **adduser**, dessen Bedienung dem neueren **useradd** stark ähnelt. Der wesentliche Unterschied ist die automatische Anpassung der Datei /etc/shadow durch das Kommando »useradd«. Wir beschränken uns hier auf die Beschreibung des letzteren Befehls.

Aufruf: useradd [-c comment] [-d home_dir] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group[,...]] [-m [-k skeleton_dir]] [-p passwd] [-s shell] [-u uid [-o]] login

Wichtige Optionen von **useradd** sind:

-c "Infos"

Das »Info«-Feld des Eintrags in der /etc/passwd wird gesetzt.

-d "Verzeichnis"

Das Heimatverzeichnis des Benutzers wird gesetzt. Damit kann die Voreinstellung »/home/<Benutzerkennung>« überschrieben werden.

-g "Gruppe"

Gruppennummer oder -name der Default-Gruppe des neuen Benutzers.

-G "Gruppenliste"

Durch Komma getrennte Liste der Gruppen, denen der Benutzer angehört.

-m

Das Heimatverzeichnis wird angelegt (falls es nicht so existiert) und der Inhalt von /etc/skel hinein kopiert.

-p "Passwort"

Das Passwort des Benutzers in **verschlüsselter** Form!

-s "Shell"

Die Default-Shell des Benutzers.

-u "UID"

Die NutzerID des Anwenders. Wird sie nicht angegeben, wird die nächsthöhere noch freie UID gewählt.

Um denselben Benutzereintrag, wie unter »Handarbeit« beschrieben, zu erzeugen, ist folgende Kommandofolge notwendig:

```
root@sonne> useradd -m -u 502 -c "Neuer Benutzer" newuser
root@sonne> passwd newuser
New password:
New password (again):
Password changed
```

Alle nicht explizit angegebenen Werte werden mit Default-Werten belegt, die in der Datei /etc/default/useradd zu finden sind:

```
user@sonne> cat /etc/default/useradd
GROUP=100
HOME=/home
INACTIVE=0
EXPIRE=10000
SHELL=/bin/bash
SKEL=/etc/skel
```

Der Systemverwalter kann mit Hilfe des Aufrufes »useradd -D« die Voreinstellungen u.a. für die Default-Gruppe »-g«, das Basisverzeichnis für die Homes »-b« und die Shell »-s« ändern.

Hinweis zum Passwort: Die Verwendung der Option »-p Passwort« erwartet das verschlüsselte Passwort. In einigen Unix-Versionen existiert hierzu das Kommando **crypt**, das aus dem Klartextpasswort den verschlüsselten Text erzeugt. Den meisten Linux-Distributionen liegt allerdings nur die gleichnamige Bibliotheksfunktion bei, so

dass man sich erst ein eigenes crypt-Kommando schreiben müsste. Angenommen, das Programm hieße »mycrypt«, dann ließe sich das Passwort automatisch generieren:

```
root@sonne> useradd newuser -p 'mycrypt Klartext-Passwort Salz'
```

Eine Beispielimplementierung des Programmes »mycrypt« finden Sie im Anhang Skriptsammlung.

Distributionseigene Werkzeuge

Manchen Distributionen liegen spezielle Administrationswerkzeuge bei, die das Vorgehen auf der Kommandozeile hinter grafischen Eingabemasken verbergen.

Allen **RedHat**-basierenden Linuxen liegt das Tools **userconf** bei, dessen Konsolen-Frontend hier dargestellt sei:

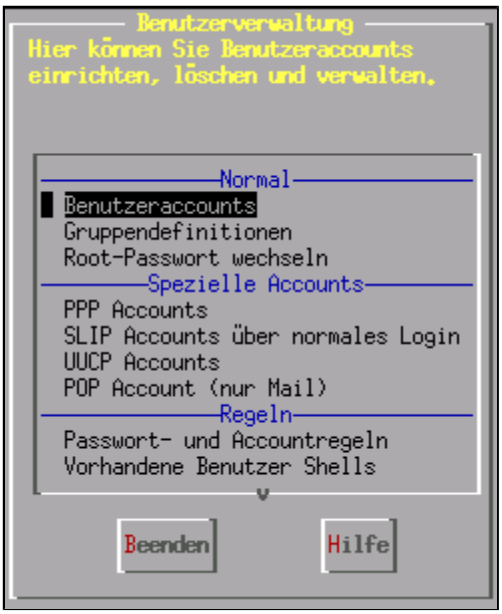


Abbildung 1: Nutzerverwaltung mit userconf

SuSE integriert die Benutzerverwaltung in ihr Adminstrationswerkzeug **Yast**. Die Eingabemaske erreicht man über Administration des Systems → Benutzerverwaltung

Abbildung 2: Nutzerverwaltung mit yast1

Löschen eines Benutzers

Die Handarbeit

1. Der Eintrag des Benutzers wird aus der Datei /etc/passwd entfernt. Dies geschieht am besten wiederum mit dem Kommando »vipw«.
2. Der Eintrag des Benutzers wird aus der Datei /etc/shadow entfernt. Entweder löscht man ihn durch Editieren der Datei (»vipw -s«) oder durch Aufruf von »pwconv«, so dass die Datensätze der Shadowdatei mit denen der /etc/passwd abgeglichen werden.
3. Das Heimatverzeichnis des Benutzers ist zu löschen. Denken Sie an eine vorherige Archivierung!

Allgemeine Werkzeuge

Bei herkömmlichen Passwortsystemen heißt das zu »adduser« korrespondierende Kommando **deluser** und bei Shadow-Passwort-Systemen **userdel.** »userdel« arbeitet analog zu »deluser«, entfernt aber den Benutzereintrag gleichzeitig aus der Shadowdatei.

Die einzige Option von »userdel« ist »-r«, womit gleichzeitig das Heimatverzeichnis des Benutzers entfernt wird. Vorsicht: Die Daten sind damit für immer verloren!

```
root@sonne> userdel -r newuser
```

Distributionseigene Werkzeuge

Die zum Anlegen neuer Benutzer gedachten Werkzeuge lassen sich ebenso zu Entfernen dieser benutzen. Verwenden Sie also »userconf« bei RedHat-Systemen und »yast« bei SuSE.

Modifzieren eines Benutzereintrags

Es sollte einleuchtend sein, dass sich die Einträge der /etc/passwd mittels eines Editors modifizieren lassen. Auch sollte das Vorgehen mit den distributionseigenen Werkzeugen leicht nachzuvollziehen sein.

Zum Ändern der Felder »Info« und »Shell« der /etc/passwd stehen spezielle Kommandos zur Verfügung, wobei einige Anpassungen sogar dem Benutzer selbst möglich sind.

Ändern der voreingestellten Shell mit chsh: Jeder Anwender darf die von ihm favorisierte Shell als Default-Shell in die Datei /etc/passwd eintragen. Voraussetzung ist nur, dass diese Shell in der Datei /etc/shells aufgeführt ist:

```
user@sonne> cat / etc/ shells
/bin/ash
/bin/bash
/bin/bash1
/bin/csh
/bin/false
/bin/sh
/bin/tcsh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/passwd
/usr/bin/tcsh
/usr/bin/zsh
```

Die Liste kann bei den einzelnen Installationen differieren, dennoch sollte man als normaler Benutzer zwei der Einträge nicht verwenden.

- **/ bin/ false** Diese Shell verhindert das Einloggen eines Benutzers in das System; er landet immer wieder bei der Anmeldeaufforderung (nicht bei grafischem Login).
- **/ usr/ bin/ passwd** Der Benutzer wird beim nächsten Anmeldevorgang zum Ändern seines Passwortes aufgefordert. Anschließend wird die Sitzung beendet (nicht bei grafischem Login).

Zum Ändern der Shell ist unbedingt deren vollständiger Zugriffspfad anzugeben, da zum Zeitpunkt des Logins noch

keine PATH-Variable existiert:

```
user@sonne> chsh -s /bin/tcsh
Password:
```

Ändern des Info-Feldes mit chfn: Eine Modifikation des vollen Namens des Benutzers »-f name« und der Limits »-o options« (man vergleiche die Beschreibung des Info-Feldes) sind dem Administrator vorbehalten. Die Raumnummer »-r nummer«, Firmentelefonnummer »-w nummer« und Privattelefonnummer »-h nummer« dürfen auch vom betreffenden Benutzer gesetzt werden.

```
newuser@sonne> grep ^newuser /etc/passwd
newuser:x:502:100:Neuer Benutzer:/home/newuser/newuser:/bin/bash

newuser@sonne> chfn -r 0815 -w 0815/110
Password
newuser@sonne> grep ^newuser /etc/passwd
newuser:x:502:100:Neuer Benutzer,0815,0815/110,:/home/newuser/newuser:/bin/bash
```

In der Datei /etc/shadow lassen sich durch den Systemverwalter die Werte zur Passwortalterung manipulieren. Das hierzu verwendete Kommando ist **chage**. Mit der Option **-l *Benutzerkennung*** liest **chage** die aktuellen Einstellungen aus:

```
root@sonne> chage -l root
Minimum:        0
Maximum:        10000
Warning:        -1
Inactive:       -1
Last Change:            Jan 27, 2000
Password Expires:       Never
Password Inactive:      Never
Account Expires:        Never
```

chage ist immer die Benutzerkennung mitzugeben, für das die Aktion vollzogen werden soll. Die einzelnen Werte zur Passwortalterung lassen sich mit folgenden Optionen setzen:

-m *Mindesthaltbarkeit*

Ein neues Passwort behält mindestens für die angegebene Anzahl Tage seine Gültigkeit. Steht hier 0, kann Passwort stets geändert werden.

-M *Maximale Lebensdauer*

Erreicht ein Passwort dieses Alter (Angabe in Tagen), wird der Zugang gesperrt.

-d *Letzter Tag*

An diesem Tag wurde das Passwort letztmalig geändert.

-E *Verfallstag*

An diesem Tag wird ein Zugang definitiv gesperrt. Die Angabe kann als Tage seit dem 1.1.1970 oder in For von JJJJ-MM-TT (Jahr-Monat-Tag) erfolgen.

-I *Inaktiv*

Ist ein Passwort abgelaufen, so wird der Zugang erst nach Ablauf dieser Anzahl Tage gesperrt. Ein Nutzer h somit die Möglichkeit, sein Passwort noch zu ändern.

-W *Tag der Warnung*

Ab so vielen Tagen vor Ablauf des Verfalls eines Passwortes wird der Benutzer bei einem Anmeldevorgang

darüber informatiert.

Sonderrechte für einen Benutzer

Die mit Gruppen verbundene Möglichkeit zur Vergabe von speziellen Rechten an eine Auswahl von Benutzern ist Gegenstand des Abschnittes Gruppenverwaltung. An dieser Stelle möchten wir demonstrieren, wie man einem Benutzer/einer Benutzergruppe die Rechte eines beliebigen anderen Benutzers bei der Ausführung eines einzelnen Kommandos einräumen kann.

Das Kommando zum Ausführen eines Kommandos mittels Rechten eines bestimmten Benutzers - auch des Administrators - heißt **sudo**.

Aufruf: sudo [OPTIONEN] KOMMANDO

Wichtige Optionen sind:

-l

Gibt eine Liste der Kommandos aus, die der Benutzer im Auftrag von Root verwenden darf:

```
user@sonne> sudo -l
Sorry, user user is not allowed to execute "list" as root on sonne.
```

-b

Das »sudo« übergebene Kommando wird im Hintergrund ausgeführt

-u "user"

Das Kommando wird als Benutzer »user« anstatt als aufrufender Benutzer gestartet. Ein solcher »user« wir weiteren Text als **Zielbenutzer** bezeichnet.

Doch zunächst ist es am Systemverwalter, eine Datenbank anzulegen, die eine Zuordnung von Benutzern und den Kommandos, die diese im Auftrag von Root ausführen dürfen, beinhaltet. Die Datei nennt sich **/ etc/ sudoers** und sollte mit dem Kommando **visudo** (einige Syntaxprüfungen werden automatisch von dem Werkzeug vorgenommen) bearbeitet werden.

Die Datenbasis /etc/sudoers

/ etc/ sudoers ist in fünf Sektionen gegliedert, von denen die ersten vier nur Aliasfunktionen ausüben, um die fünfte - und damit auch die Konfiguration an sich - übersichtlich zu halten.

```
# Benutzer Alias Spezifikation
User_Alias <USER> = ...

# Rechner Alias Spezifikation
Host_Alias <HOSTS> = ...

# Zielbenutzer Alias Spezifikation
Runas_Alias <RUN_AS> = ...

# Kommando Alias Spezifikation
Cmnd_Alias <COMMAND> = ...

# Benutzer Spezifikation
...
```

In den **Alias-Sektionen** werden symbolische Namen für Rechner-, Benutzer-, Kommando- und Zielbenutzer-Gruppen erzeugt. Eine derartige Zusammenfassung ist sinnvoll, wenn die Gruppen mehrfach Verwendung finden oder aber auf komplexe Definitionen wie u.a. IP-Adressbereiche zurückgegriffen wird.

Hintergrund der recht ausführlichen Möglichkeiten der Rechner- bzw. Netzwerk-Selektion und Varianten von Benutzergruppen ist die zentrale Konfiguration von UNIX-Systemen in größeren Netzwerken mit einer einzigen sudoers-Datei, die hierbei nur einmal generiert und auf die einzelnen Maschinen verteilt werden muss.

Um die einzelnen Bereiche genauer zu erklären ist es am günstigsten, das Feld von hinten aufzurollen, da in der **letzten Sektion** - der Benutzer-Spezifikation - alle zuvor definierten Einträge erst ihre Wirksamkeit erlangen. Hier wird die Zuordnung realisiert, die die Funktion von **sudo** ausmacht, dass »ein bestimmer Benutzer« als »ein bestimmter anderer Benutzer« »bestimmte Kommandos« auf »einem bestimmten Rechner« ausführen darf.

Der Aufbau einer *Benutzer Spezifikation* gestaltet sich wie folgt:

```
# Benutzer Spezifikation
<USER> <HOSTS>=[(<RUN-AS>)][NOPASSWD:]<COMMAND>[,<COMMAND>]
```

Die einzelnen Einträge stehen dabei sowohl für einen einzelnen Benutzer, einen Rechnernamen, ein Kommando oder aber für einen Alias der gleichnamigen Alias-Spezifikation. Die optionale Angabe von **NOPASSWD:** berechtigt die angegeben Benutzer, das (oder die) Kommando(s) ohne Eingabe eines Passwortes (Passwort des »Zielbenutzers«) zu starten. Fehlt *RUN-AS*, so ist der Zielbenutzer immer Root.

Abgesehen vom einleitenden Schlüsselwort **User_Alias** bzw. **Runas_Alias** besitzt die Spezifikation der *Benutzer-* und *Zielbenutzer-*Aliasse denselben Aufbau. Dem Schlüsselwort folgt eine freie wählbarer Aliasname (Großbuchstaben!), dem eine kommaseparierte Liste aus Benutzerkennung, Gruppennamen (durch ein vorangestelltes »%« gekennzeichnet) oder Netzgruppen (Benutzer der genannten Netzgruppe aus der Datei »/etc/netgroups«; ein Eintrag wird durch ein vorangestelltes »+« als Netzgruppe gekennzeichnet) folgt.

```
# Benutzer Alias Spezifikation
User_Alias LOCALUSER= tux,user,newbie
User_Alias NETGROUP= +intranet,tux

# Zielbenutzer Alias Spezifikation
Runas_Alias WORKAS= operator,%printer
```

Ein Eintrag der *Rechner* Alias Spezifikation wird mit einem **Host-Alias** eingeleitet. Dem Aliasnamen (Großbuchstaben) folgt eine Liste der Mitglieder, jeweils durch Komma voneinander getrennt. Ein Mitglied kann ein Rechnername, eine Netzgruppe, eine IP-Adresse oder ein Netzwerk sein. Ein Netzwerk wird entweder über seinen Namen (/etc/networks) oder über eine Netzwerkadresse (194.168.17.0), die mittels einer optionalen Netzmaske (194.168.17.0/255.255.255.127) weiter eingeschränkt werden kann. Letztes Beispiel betrifft alle Rechner mit IP-Adressen 194.168.17.1 - 194.168.17.127. Ohne Angabe der Netzmaske wird die des lokalen Rechners angenommen.

```
# Rechner Alias Spezifikation
Host_Alias SOMEHOSTS= erde,sonne,melmac.outside.all
Host_Alias NETWORKS= localnet,194.168.17.0/255.255.255.127
Host_Alias MIXED= erde.galaxis.de,128.233.1.12,linuxnet
```

Die *Kommando* Alias Spezifikation besitzt die komplexeste Syntax, da hier auch die shell-üblichen Metazeichen (für Argumente) Anwendung finden. Ein Kommando muss immer mit vollständigem Zugriffspfad angegeben werden. Ihm können optionale Argumente folgen, um die Verwendung des Kommandos auf eben diese Optionen einzuschränken. Mehrere Kommandos werden durch Komma voneinander getrennt.

Die Metazeichen *, ?, [...] und [!...] besitzen dieselbe Bedeutung, wie im Abschnitt Bash Kommandoeingabe beschrieben. Sonderzeichen verlieren durch einen vorangestellten Backslash »\« ihre Wirkung. Sollen alle Optionen eines Kommandos verboten werden, so ist dem Kommando eine leere Zeichenkette "" nachzustellen.

```
# Kommando Alias Spezifikation
Cmnd_Alias BOOT= /sbin/shutdown -r now,/sbin/reboot
Cmnd_Alias USERADMIN= /usr/sbin/vipw "",/usr/sbin/vigr ""
Cmnd_Alias SHELLS= /bin/sh,/bin/csh,/bin/tcsh,/bin/ksh
```

Kommen wir noch einmal zur Benutzer Spezifikation zurück, die letztlich die Verbindung aller Alias-Spezifikationen herstellt. Stellvertretend für die Aliasse von Rechnern, Zielbenutzern und Kommandos kann das Schlüsselwort **ALL**

stehen, das als Platzhalter für »Alles« steht. Anstelle eines einzelnen Kommandos oder Kommandoaliasses darf hier eine Liste von Kommandos oder Kommandoaliassen stehen. Steht einem Kommando(alias) ein Ausrufezeichen »!« zuvor, so wird genau dieses Kommando gesperrt.

Abschließend soll diese Thematik eine vollständige Datei »/etc/sudoers« demonstrieren:

```
# Benutzer Alias Spezifikation
User_Alias LOCALUSER= tux,user,newbie
User_Alias NETGROUP= +intranet,tux

# Zielbenutzer Alias Spezifikation
Runas_Alias WORKAS= operator,%printer

# Rechner Alias Spezifikation
Host_Alias SOMEHOSTS= erde,sonne,melmac.outside.all
Host_Alias NETWORKS= localnet,194.168.17.0/255.255.255.127
Host_Alias MIXED= erde.galaxis.de,128.233.1.12,linuxnet

# Kommando Alias Spezifikation
Cmnd_Alias REBOOT= /sbin/shutdown -r now,/sbin/reboot
Cmnd_Alias HALT= /sbin/shutdown -h now,/sbin/halt
Cmnd_Alias USERADMIN= /usr/sbin/vipw "",/usr/sbin/vigr ""
Cmnd_Alias SHELLS= /bin/sh,/bin/csh,/bin/tcsh,/bin/ksh

# Benutzer Spezifikationen
# tux darf alles
tux      ALL = (ALL) ALL

# notux erhält Root-Rechte, darf aber den Rechner nicht booten. Und damit er diese Beschränkung durch Öffnen einer neuen Shell nicht umgehen kann, werden diese ausgeklammert (siehe Anmerkungen im Anschluss)
notux       ALL = (ALL) ALL,!REBOOT,!HALT,!SHELLS

# Lokale Benutzer dürfen auf Rechner "sonne" das Modem ohne Passwortangabe benutzen (dies stellt insofern eine Alternative zu den Suid-Bits dar, da die Berechtigung detaillierter konfiguriert werden können und die Benutzung protokolliert wird)
LOCALUSER sonne = NOPASSWD: /usr/bin/wvdial

# "newbie" darf auf allen Rechnern in der Gruppe NETGROUP ein "su" mit allen Argumenten außer "root" ausführen. Oder anders formuliert, er kann mit su jeder Nutzer werden ausgenommen root.
newbie    NETGROUP = /bin/su ?*,!/bin/su *root*

# Jeder darf auf jedem Rechner ohne Eingabe eines Passwortes den Automounter zum Unmounten der nicht benutzten Dateisysteme auffordern
ALL       ALL = NOPASSWD:/usr/bin/killall -USR1 automount
```

Anmerkungen:

Auch wenn die Konfiguration vielgestaltig und fehlerträchtig ist, so besitzt **sudo** gegenüber dem Kommando **su** und den Suid-Bits verschiedene Vorteile. Wesentlich ist die Möglichkeit der Protokollierung sämtlicher Aktivitäten, womit etwaige ungewöhnliche Vorgänge im System, deren Ursache man im Missbrauch von Sonderrechten vermutet, im Nachhinein resümiert werden können. Und während man mittels Suid-Bits die Rechte nur auf Gruppenbasis (»allen Mitgliedern dieser Gruppe«) oder allen Benutzern einräumen kann, ermöglicht **sudo** eine abgestufte Administration.

Dennoch erfordert die Konfiguration in kritischen Umgebungen enorme Sorgfalt, da oben benannte Eigenschaften nur für die direkt mit **sudo** eingegebenen Kommandos gelten und **nicht** für mittels **sudo** eröffnete Shells und den daraus gestarteten Kommandos. Deshalb **Vorsicht** mit der Vergabe von solchen allgemeinen Rechten wie mit »ALL«; hier sollten zumindest die Ausführung von Programmen untersagt werden, die irgendwelche Manipulationen im Dateisystem ermöglichen (cp, chmod, chown, ln, ...) ebenso wie alle Programme, die außerhalb der Standard-Pfade liegen!

Gehen Sie besser sehr restriktiv an die Freigabe von Berechtigungen. Ihre wichtigste Devise lautet: **»Weniger ist mehr!«**.

Zugang sperren

Zugang allen Benutzern verwehren

Manchmal ist es für den Systemadministrator erforderlich, den Benutzern 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 Benutzer beim System anzumelden, wird der Inhalt dieser Datei ausgegeben.

Zugang einem bestimmten Benutzer verwehren

Der häufigere Fall wird wohl sein, dass man einen einzelnen Benutzer den Zugang verwehren möchte. So ist es durchaus sinnvoll, falls ein Kollege für längere Zeit auswärts seinen Job ausübt, dessen ungenutzten Zugang zu sperren, um so potentiellen Hackern einen in Frage kommenden Angriffspunkt zu entziehen.

Es gibt mehrere Wege, dies zu realisieren:

- Mit Hilfe des Kommandos passwd kann Root das Passwort des Benutzers als ungültig kennzeichnen:

```
root@sonne> passwd -l user
Password changed.
root@sonne> grep user: /etc/shadow
user:!HbFHWOedOYXwI:11103:0:99999:7:::
```

  Dem verschlüsselten Passwort wurde ein Ausrufezeichen vorangestellt, somit ist das alte Passwort nutzlos. Der Aufruf:

```
root@sonne> passwd -u user
Password changed.
```

  entfernt das Ausrufezeichen wieder.
- Mit einem Editor kann Root natürlich manuell das Passwort in der Shadowdatei »versalzen«.
- Die Shell des Benutzers kann auf /bin/false geändert werden:

```
root@sonne> chsh -s /bin/false user
```

  Das funktioniert allerdings nur, wenn das System für Konsolenlogin (Runlevel!) konfiguriert ist. Unter einem grafischen Login hätte ein solcher Benutzer weiterhin Zugang. Allerdings kann er dann keine Shell (keine Terminalanwendungen) mehr starten.

Passwort vergessen - was nun?

Es ist eine Gratwanderung zwischen dem Sicherheitsanspruch an ein Passwort und dessen Einfachheit. Ein simples Passwort lässt sich leicht merken, ist aber ebenso einfach von einem potentiellen Hacker zu erraten. Ein kompliziertes Passwort verschwindet womöglich auf Nimmer Wiedersehen in den grauen Zellen des Vergessens... Und nun?

Einfach ist es, hat man sein Passwort als normaler Benutzer »verlegt«. Dann bittet man einfach den Systemverwalter, den alten Passworteintrag zu entfernen.

Und wenn einem das Root-Passwort entfallen ist?

Der erste Versuch sollte die Erlangung einer Root-Shell zum Ziel haben. Booten Sie hierzu Linux neu und übergeben dem Boot-Image am Bootmanager-Prompt folgenden »init«-Parameter:

```
# Das Bootimage im Beispiel heißt »vmlinuz«
Lilo: vmlinuz init=/bin/sh
```

Anstatt die Runlevel-Skripte abzuarbeiten, startet Linux einzig die Shell /bin/sh. Das Root-Passwort kann

nachfolgend mit **passwd** neu vergeben werden.

Falls jedoch obige Methode versagt - weil bspw. die Übergabe von Bootparametern unterbunden wurde - kann das Passwort nur aus einem anderen Linux-System heraus geändert werden. Booten Sie hierzu ein anderes System. Vielen Distributionen liegt hierzu ein so genanntes **Rettungssystem** bei. Melden Sie sich auf diesem System als Root an und mounten die Rootpartition des Systems mit dem verschollenen Passwort auf ein beliebiges Verzeichnis. Mit dem Kommando **chroot** weisen Sie das System an, als Wurzel das angegebene Verzeichnis zu verwenden. Ändern Sie anschließend das Passwort und booten das System neu.

Beispielhaft seien die Schritte des Mountens, Wechseln des Rootverzeichnisses und Ändern des Passwortes skizziert, wobei als Wurzelverzeichnis des zu korrigierenden Systems /dev/hda2 angenommen wird:

```
# Wir sind Administrator des Rettungssystems
root@sonne> mount / dev/ hda2 / mnt

root@sonne> chroot / mnt passwd
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully

root@sonne> reboot
```

Bemerkung: Nach dem Aufruf von »chroot« befinden wir uns in einer anderen Wurzel, d.h. alle nachfolgend eingegebenen Kommandos verwenden diese aus diesem System! Ein Aufruf von »passwd« bewirkt - aus Sicht des Rettungssystems - einen Aufruf von »/mnt/usr/bin/passwd«. Im Beispiel differiert daher die Ausgabe des passwd-Kommandos (das hier die MD5-Verschlüsselung verwendet) von denen der weiter oben angeführten Beispiele.

Verwendet man »chroot« ohne Angabe des Kommandos, erhält man eine Shell im System der neuen Wurzel. Diese kann man nur über **exit** verlassen.

Protokollierung

Übersicht

Die Protokollierung wird gerne vernachlässigt, dabei ist eine saubere Buchführung im System in vielen Situationen hilfreich.

Was ist, wenn z.B. ein Dienst versagt, dessen Fehlerausschrift aber eher nichts besagt? Ein lapidarer Ausspruch des Clients wie: »Server antwortet nicht«, lässt die Suche nach dem Fehler überall und nirgends ansetzen. Aber... mit großer Sicherheit finden Sie in einer der Protokolldateien detaillierte Hinweise auf eine mögliche Ursache.

Oder vermuten Sie einen Hacker in Ihrem System? Dann haben Sie hoffentlich alle Anmelde-Vorgänge automatisch notieren lassen? Hacker sind vielleicht keine symphatischen, aber ganz sicher keine dummen Zeitgenossen und haben sie einmal Zugang zum System gefunden, so werden sie alles ihnen Mögliche tun, um ihre Spuren zu verwischen... Und auch dem können Sie durch eine geschickte Protokollierung vorbeugen.

Der Alptraum eines jeden Administrators... Ihre Maschine reagiert nicht mehr! Was den Zustand verursacht hat, steht vielleicht irgendwo geschrieben und Sie sind nach dem Neustart in der Lage, dem Übel nachzugehen...

Dieser Abschnitt erklärt Ihnen die Möglichkeiten, welche Ereignisse Sie in welcher Art und Weise protokollieren können.

Der Syslog-Dämon

Der **Syslog-Dämon** »syslogd« bietet eine Methode, um Fehler- und/oder Protokollausgaben von Programmen abzufangen und diese nach konkreten Regeln zu verwalten. Daneben existiert der **Klog-Dämon** »klogd«, der die Syslog-Funktionalität auf die speziellen Belange des Linux-Kernels erweitert. Letzterer ist Dreh- und Angelpunkt eines späteren Abschnitts.

Um die Meldungen beim Systemstart nicht zu versäumen, wird der **klogd** als erster Dämonen-Prozess überhaupt gestartet, noch bevor der Kernel seine Überprüfung und Initialisierung der Hardware beginnt. Der **syslogd** wird erst hochgefahren, wenn der Kernel seine »Hausaufgaben« erledigt hat.

Was der Syslogd kann

Letztlich liegt es im Ermessen der Programmierer eines Programmes, inwiefern dieses Mitteilungen über außergewöhnliche Ereignisse erzeugt. Bei Anwendungen, die normalerweise mit der Standardausgabe verbunden arbeiten, ist es durchaus Brauch, auch Fehler auf das Terminal des Benutzers zu senden. Was aber ist mit Prozessen, die keinerlei Ausgaben erzeugen? Dass z.B. ein Dämon seine Probleme jedem Anwender offenbart, kann nicht die Lösung sein. Oder nützt Hans aus Hinterfindmichnicht die Meldung über einen Papierstau des Druckers seines Internet Services Providers etwas, wo er sich doch nur zum Abholen seiner Mails angemeldet hat?

Viele Programme unter Unix greifen deshalb auf den Systemruf **syslog()** zurück. Dieser Systemruf schreibt nun die Protokollnachricht in das Device »/dev/log« und der **syslogd** liest aus diesem. »Programme« umfasst hier sowohl Prozesse, als auch Hardwaretreiber oder den Benutzer, der mit Hilfe des Kommandos logger selbst Meldungen einbringen kann. Und wird der **syslogd** mit der Option **-r** gestartet, nimmt er auch Meldungen anderer **syslogd** (über UDP-Port 514) aus dem Netzwerk entgegen.

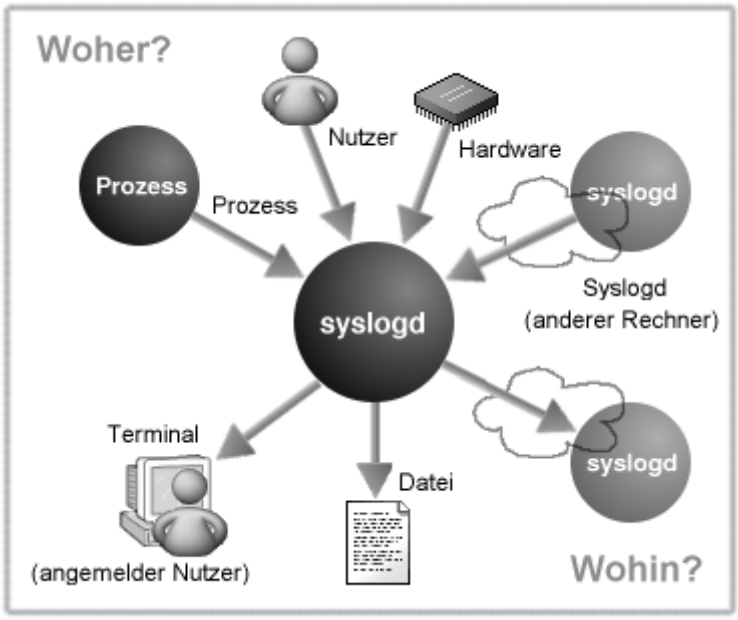


Abbildung 1: Quellen und Ziele von Protokolldaten

Wohin mit den Meldungen? Das liegt an der jeweiligen Konfiguration. Prinzipiell sind folgende Ziele möglich:

- Die Meldung wird in eine bestimmte Datei geschrieben
- Die Meldung wird auf das Terminal ausgewählter (oder aller) Benutzer geschrieben, insofern sie am System angemeldet sind
- Die Meldung wird zur Behandlung an den **syslogd** eines anderen Rechners weiter geleitet

Vor allem wenn der lokale **syslogd** auch Meldungen aus dem Netz entgegen nimmt und diese in lokale Dateien verstreut, nehmen die Daten rasch überhand. Auch heute ist es ein beliebter Angriffspunkt von Hackern, durch Provokation von Fehlermeldungen, die Platten eines Rechners »vollzumüllen« und somit das System in die Knie zu zwingen. Aus diesem Grund enthält jede Protokollnachricht neben dem eigentlichen Text noch eine **Herkunft** (Typ der Nachricht) und eine **Priorität** (Dringlichkeit der Nachricht).

Eine Protokollnachricht kann von folgenden Quellen stammen:

auth bzw. security

Authentifizierung (z.B. login)

authpriv

Vertrauliche Nachrichten der »inneren« Sicherheitsdienste

cron

Meldungen des »cron«-Dämons

daemon

Meldungen von anderen Dämonen

kern

Kernelmitteilungen

lpr

Meldungen des Druckerdämonen (lpd)

mail

Meldungen vom Mailsystem

news

Meldungen des Nachrichtensystems

syslog

Meldungen vom »syslogd« selbst

user

Meldungen von Anwenderprogrammen

local0-local7

Zur freien Verfügung

Als Dringlichkeit sind möglich:

debug

Debugmeldungen, innere Programmzustände

info

Informationen über das alltägliche Geschehen...

notice

Etwas Auffälliges im Normalbetrieb...

warn bzw. warning

Alle möglichen Warnungen...

err bzw. error

Fehlermeldungen aller Art

crit

Kritische Fehlermeldungen, gerade noch gutgegangen...

alert

Dringend, sofortiger Eingriff notwendig

emerg bzw. panic

Unmittelbar vor einem Systemcrash

Die Namensgebung für Herkunft und Priorität in den Tabellen entspricht dabei den möglichen Angaben in der nachfolgend beschriebenen Konfigurationsdatei »/etc/syslog.conf«.

Sicherlich ist mit einer »debug«-Meldung von einem »user« anders zu verfahren als mit einem »alert« vom »kernel«. Letztere ist sichere Kandidatin, um unverzüglich auf der Konsole ausgegeben zu werden. Vielleicht kann der Systemverwalter den drohenden Absturz noch abwenden? Außerdem ist es einen Versuch wert, eine solche kritische Situation auf einem anderen Rechner protokollieren zu lassen, um im Falle des lokalen Crashs noch einen

Hinweis auf die Fehlerursache vorzufinden. Mit welcher Meldung letztlich wie verfahren wird, wird in der Datei **/ etc/ syslog.conf** festgelegt.

Das Format der Protokolldaten

Der **syslogd** ergänzt, bevor er die Meldung protokolliert, zu jeder Nachricht weitere Informationen: Zum einen ist dies die Zeit des Eintreffens der Meldung, der Rechnername, von dem die Meldung kam und zum anderen deren Verursacher, also »kernel« bzw. der Name des protokollierenden Programmes. Typische Einträge in den Protokolldateien sehen also wie folgt aus:

```
Jun 14 15:51:19 erde ACCT: ISDN: 14.06.2000,13:34:56,13:51:23,984,2435661,74327,5036,2076,
,O,489953,21032038329708330,7/0,0,0,saxsys ag
Jun 22 14:48:07 sonne ktalkd[1425]: connect from 192.168.99.127
Jun 22 15:10:13 sonne kernel: hda: no DRQ after issuing WRITE
Jun 22 15:58:52 sonne -- MARK --
Jun 22 15:58:54 sonne logger: Meldung von user
```

Die Zeile »-- MARK --« ist eine interne Meldung vom **syslogd** selbst, eine Art Zeitstempel, der die Aktivität des Protokollanten bezeugen soll (in der Voreinstellung wird aller 20 Minuten ein Eintrag vorgenommen).

Die Optionen des Syslogd

Der **syslogd** kommt mit verschiedenen Kommandozeilenoptionen klar. In den meisten Fällen begibt sich das Programm sofort nach Aufruf in den Hintergrund, einzig die Option **-d** verhindert dies. In Folge dessen wird der Protokollant allerlei Debugmeldungen auf dem Terminal produzieren.

Sollen Meldungen aus dem Netzwerk entgegen genommen werden, ist das Programm mit der Option **-r** zu starten. Der **syslogd** verarbeitet jetzt alle Pakete, die am UDP-Port 514 eintreffen.

Die lokalen Meldungen erhält das Programm aus dem Socket »/dev/log«, mit **-p socket** kann ein alternativer Socket und mit **-a socket** bis zu 19 zusätzlich zu überwachende Sockets angegeben werden.

Als letzte wichtige Option soll **-m Zeit** genannt werden, die die Zeit zwischen den Zeitstempeln auf das angegebene Intervall (in Minuten) festlegt. Ein Wert "0" schaltet die Generierung des Zeitstempels ab.

Die Datei / etc/ syslog.conf

Dem **syslogd** muss man genau sagen, was er protokollieren soll. Alles, was nicht explizit in seiner Konfigurationsdatei - der /etc/syslog.conf - benannt wurde, schickt er unbeachtet ins Nirwana.

Eine Zeile der Datei »/etc/syslog.conf«, die nicht mit einem Doppelkreuz beginnt (Kommentar), besteht aus zwei, durch Leerzeichen oder Tabulatoren voneinander getrennten Teilen.
Der zweite, rechts stehende Teil beschreibt das **Ziel der Nachricht**. Hier kann eine Datei stehen (Protokolldatei, Device-Datei, Pipe) oder der Name eines Rechners, falls die Protokollierung von einem entfernten **syslogd** erledigt werden soll.
Der erste Teil besteht aus Paaren von **Herkunft.Priorität** (durch den Punkt getrennt), mehrere solcher Paare können, jeweils durch ein Semikolon getrennt, angegeben werden (es dürfen keine Leerzeichen enthalten sein!).

Jede Priorität ist einem bestimmten **Level** zugeordnet, wobei die Ordnung gemäß der Anordnung in obiger Tabelle gilt, d.h. die Priorität »debug« besitzt das niedrigste Level und »panic« das höchste. **Die Angabe einer Priorität schließt alle Prioritäten höherer Level ein**.

Beispiel: Der folgende Eintrag protokolliert alle Meldungen des Mailsystems »mail« mit der Priorität »warn« und höher (also »err«, »crit«, »alert», »emerg«) und des Newssystems »news« mit den Prioritäten »err«, »crit«, »alert« und »emerg« in einer Datei »/var/log/info.log«:

```
mail.warn;news.err          /var/log/info.log
```

Die Protokollierung in Dateien im Verzeichnis »/var« anzusiedeln, ist der gebräuchliche Weg (Vergleiche Bemerkungen zum Filesystem Hierarchie Standard).

Nun ist es doch nicht immer ideal, mit einer Priorität gleich den ganzen »Schwanz« der übergeordneten Level nach sich zu ziehen. Deswegen existieren **vier Modifzierer**, mit denen die Angaben ausgeweitet, eingeschränkt, ausgeschlossen oder gar negiert werden können:

- Der **Stern** »*« darf dabei sowohl für die Herkunft als auch für die Dringlichkeit eingesetzt werden und bewirkt die »Ausweitung« auf alle Quellen bzw. alle Prioritäten
- Um Meldungen einer bestimmten, einzelnen Dringlichkeit zu behandeln, ist der Prioritätsangabe ein **Gleichheitszeichen** "**=**" voran zu stellen
- Um Meldungen einer konkreten Herkunft oder Priorität von der Protokollierung auszuschließen, kann das Schlüsselwort **none** eingesetzt werden
- Und um die Auswahl bestimmter Prioritäten zu negieren oder eine einzelne Herkunft auszuschließen, kann dem Eintrag ein **Ausrufezeichen** "**!**" voran gestellt werden

Beispiel: Alle kritischen Meldungen werden in einer einzelnen Datei gesammelt, wobei Nachrichten vom Kernel ausgeschlossen werden sollen (erste Zeile). Des Weiteren wünschen wir alle Prioritäten »alert« und höher dem Systemadministrator und Benutzer »tux« auf dessen Konsolen mitzuteilen (zweite Zeile). In der dritten Zeile sammeln wir alle Nachrichten geringerer Priorität (»warn« und kleiner) in einer extra Datei.

```
*.=crit;kern.none       /var/log/crit.log
*.alert                 root,tux
*.!err;                 /var/log/stuff.log
```

Beispiel: Um alle Meldungen des Mailsystems zu unterdrücken sind zahlreiche Syntaxvarianten möglich:

```
mail.!*                 /var/log/mail
mail.none               /var/log/mail
mail.!debug             /var/log/mail
mail.*                  /dev/null
```

Beispiel: Es sollen nur die Meldungen des Newssystems protokolliert werden, die mindestens die Priorität »info«, aber höchstens die Priorität »warn« besitzen:

```
news.info,news.!err     /var/log/news
```

Etwas Schreibarbeit erspart man sich, indem man mehrere Quellen, die mit identischen Prioritäten zu protokollieren sind, per Komma getrennt an die Stelle der Herkunft einsetzt. Im folgenden Beispiel werden die Meldungen der Sicherheitsdienste gleichwertig behandelt und an den **syslogd** des Rechners »erde« geleitet:

```
auth,authpriv.*         @erde
```

Zwei weitere Varianten zur Angabe des Ziels der Protokollnachricht vervollständigen die Möglichkeiten. Zum einen kann ein einzelner Stern »*« dort stehen (es steht für das Kommando **wall**), dann erhalten alle aktuell angemeldeten Benutzer die Meldung auf dem Terminal angezeigt. Und zum zweiten darf eine Fifo-Datei als Ziel angegeben werden (ihr ist das Pipe-Zeichen »|« voran zu stellen). Aus einer solchen können beliebige Programme dann lesen. Eine typische Anwendung ist das Programm »xconsole«, das aus der Fifo-Datei »/dev/xconsole« seine Eingaben bezieht. Alle Zielangaben sollen nochmals zusammengefasst dargestellt werden:

/ datei

In die angegebene Datei, der Name ist als vollständiger Zugriffspfad anzugeben

user

Auf das Terminal des angegebenen Benutzers

@rechner

*****

Auf die Terminals aller angemeldeten Nutzer

| / fifo_datei

In die angegebene Fifo-Datei

Bekanntlich werden modifzierte Dateien im Normalfall nicht sofort auf die Festplatte zurückgeschrieben. Im Sinne der Protokollierung kritischer Situationen entpuppt sich das Vorgehen für unzureichend, da im Falle eines Systemausfalls wichtige Informationen womöglich verloren gehen. Aus diesem Grund kann dem Ziel, insofern es sich um eine Dateiangabe handelt, ein **Minus** vorangestellt werden, was die sofortige **Synchronisation** der Datei veranlasst.
Das Beispiel einer vollständigen Datei /etc/syslogd soll den Abschnitt beenden:

```
# /etc/syslog.conf - Configuration file for syslogd(8)
#
# For info about the format of this file, see "man syslog.conf".
#


#
#
# print most on tty10 and on the xconsole pipe
#
kern.warn;*.err;authpriv.none    /dev/tty10
kern.warn;*.err;authpriv.none   |/dev/xconsole
*.emerg                          *

# enable this, if you want that root is informed
# immediately, e.g. of logins
#*.alert                         root

#
# all email-messages in one file
#
mail.*                          -/var/log/mail

#
# all news-messages
#
# these files are rotated and examined by "news.daily"
news.crit                        -/var/log/news/news.crit
news.err                         -/var/log/news/news.err
news.notice                      -/var/log/news/news.notice
# enable this, if you want to keep all news messages
# in one file
#news.*                         -/var/log/news.all

#
# Warnings in one file
#
*.=warn;*.=err                   -/var/log/warn
*.crit                          /var/log/warn

#
# save the rest in one file
#
*.*;mail.none;news.none         -/var/log/messages

# enable this, if you want to keep all messages
# in one file
*.*                             -/var/log/allmessages
*.*                             /dev/tty11
```

Tipp: Vergessen Sie nicht, die Protokolldateien hin und wieder zu sichern und das Original zu leeren, sonst werden sich binnen kürzester Zeit die Megabytes in ihrem /var-Verzeichnis ansammeln. Nach dem Bearbeiten der /etc/syslog.conf ist der **syslogd** zum Einlesen der neuen Konfiguration zu bewegen:

```
root@sonne> killall -HUP syslogd
```

Der klog-Dämon

In den üblichen Konfigurationen von Linux wird als eine der ersten Aktionen der **klogd** gestartet (meist unmittelbar im Anschluss an das Mounten des /proc-Dateisystems). Dieser Dämon ist es nun, der die zahlreichen Meldungen, die während des Bootvorganges über den Bildschirm rauschen, veranlasst. Der **klogd** tut auf den ersten Blick nichts anderes, als wir es vom **syslogd** her kennen, abgesehen davon, dass er nur die vom Kernel ausgehenden Nachrichten verarbeitet.

In Abhängigkeit ihrer Verfügbarkeit bezieht der **klogd** seine Informationen aus einer von zwei möglichen Quellen. Sobald der Dämon startet, sucht er das gemountete /proc-Dateisystem. Wird er fündig, ist die dortige Datei /proc/kmsg Quelle der Nachrichten. Existiert das /proc-Dateisystem nicht, erhält der Kernel-Log-Dämon alle Nachrichten über einen Systemruf »sys_syslog()«.

Der wesentliche Unterschied zum **syslogd** ist die Fähigkeit der weiteren Verarbeitung der Kernelmeldungen. Der Dämon extrahiert zunächst eine enthaltene Priorität (es werden dieselben Level wie beim »syslogd« unterschieden), anhand derer über die weitere Verfahrensweise entschieden wird. Genau das tut der **syslogd** ebenso... doch der **klogd** vermag nun aus der Meldung ebenso den Namen der verursachenden Funktion heraus zu lesen. Die Ursache kann somit wesentlich exakter eingegrenzt werden, als es die vielsagenden Ausschriften - wie »Modul xyz verursachte eine allgemeine Schutzverletzung« - vermögen.

Um diese Zuordnung zwischen Adresse und Funktionsname vornehmen zu können, schaut der **klogd** in einer internen Tabelle nach, die er zu Beginn anhand der Informationen der Datei /boot/System.map (oder, falls sie nicht existiert aus /System.map bzw. /usr/src/linux/System.map) generiert.

In der Voreinstellung des Kernel-Log-Dämons werden alle Meldungen der Priorität »info« und höher auf der Konsole ausgegeben. Auf die Dauer ist es sicherlich sehr ermüdend, Nachrichten vom Mounten eines Dateisystems und vom Laden eines Modules ständig vom Terminal löschen zu müssen. Aus diesem Grund wird der **klogd** (eigentlich immer) mit der Option **-c Levelnummer** angewiesen, nur wirklich problematische Situationen den Konsolen zuzuführen. In vielen Distributionen lautet der Aufruf **klogd -c 1**, so dass der Nutzer nur noch den »letzten« Aufschrei vor dem Systemabsturz miterleben darf.

Neben der Ausgabe auf der Konsole werden alle Nachrichten nach Aufbereitung durch den **klogd** an den **syslogd** weiter gereicht, der dann gemäß seiner Regeln diese verarbeitet (in den Protokolldateien erkennt man die Kernelmeldungen anhand der Herkunft "kernel").

Folgende wichtige Optionen steuern den **klogd**:

-c Level

Unterdrückt Konsolenausgaben von Nachrichten mit niedrigerem Level

-d

Der **klogd** erzeugt auch Debugmeldungen (sonst nur Level »info« und höher)

-f Datei

Die Ausgaben werden in die angegebene Datei geleitet

-n

Der Dämon startet nicht im Hintergrund

-o

Der **klogd** liest alle ausstehenden Meldungen, verarbeitet diese und beendet sich

-s

Der **klogd** muss den Systemruf »sys_syslog()« verwenden, anstatt die Datei /proc/kmsg

-k Symboltabelle

Die Symboltabelle wird aus der angegebenen Datei gelesen

Zum Abschluss noch ein praktisches Beispiel der Verwendung des **klogd** in einem SuSE-System. Wer ein solches vor sich hat, findet eine Datei »/var/log/boot.msg« vor, die genau die während des Bootvorgangs erzeugten Kernelmeldungen enthält. Zu einem späteren Zeitpunkt allerdings erkennt man anhand der Ausgaben des Kommandos **ps**, dass der **klogd** mit der Option **-c 1** aktiv ist. Wie wurde das realisiert?

Indem als eine der ersten Maßnahmen der **klogd** im so genannten One-Shot-Mode gestartet wurde, er die bis dahin aufgehäuften Nachrichten verarbeitet, in die Datei schreibt und sich anschließend beendet (der Aufruf dazu lautet "/usr/sbin/klogd -s -o -n -f /var/log/boot.msg"). Später, nachdem die Initialisierung abgeschlossen ist, wird der Dämon erneut als Hintergrundprozess gestartet "klog -c 1".

Logger

Mit dem Kommando **logger** steht eine Nutzerschnittstelle zum **syslogd** bereit. Seine Anwendung macht in erster Linie innerhalb von Shellskripten Sinn, wenn diese Fehlermeldungen oder außergewöhnliche Situationen für notierenswert erachten. Es steht jedoch auch dem »normalen« Benutzer zu, mit dem Kommando eine Protokollierung seiner Nachrichten dem **syslogd** aufzuerlegen.

Aufruf: logger [OPTIONEN] [Nachricht]

Die Kommandozeilenoptionen setzen in erster Linie die Parameter der Nachricht:

-i

Die Prozessnummer des logger-Prozesses wird zusätzlich protokolliert

-s

Die Nachricht wird zusätzlich (!) auf die Standardfehlerausgabe umgeleitet

-f Datei

Die Nachricht wird zusätzlich (!) in die angegebene Datei umgeleitet

-p Herkunft.Priorität

Die Nachricht erhält die angegebene Herkunft und Priorität. Voreinstellung ist »user.notice«. Als Angaben s die in den Tabellen zum **syslogd** aufgeführten Werte zulässig.

-t tag

Jede Zeile wird mit dem »tag« eingeleitet

-u Socket

Die Nachricht wird in den angegebenen Socket geschrieben (Vergleiche auch Option »-a« des »syslogd«)

--

Kennzeichnet das Ende der Optionen, damit kann eine nachfolgende Nachricht auch mit einem Minus begin

Wird auf der Kommandozeile keine Nachricht eingegeben, erwartet **logger** diese von der Standardeingabe (Eingabeende mit [Ctrl]-[D]), sonst wird alles bis zum Zeilenende als Inhalt der Meldung betrachtet.

Einige Beispiele sollen das Thema beenden. Bitte beachten Sie beim Nachvollziehen der Beispiele, dass die tatsächliche Protokollierung von der konkreten Konfiguration des **syslogd** abhängt.

```
user@sonne> logger -s -t "###" Vorsicht, Anwender ist übermüdet
###: Vorsicht, Anwender ist übermüdet
user@sonne> logger -p local7.alert -- -Anwender hat Durst-

root@sonne> tail -2 /var/log/messages
Jun 23 11:00:01 sonne ###: Vorsicht, Anwender ist übermüdet
Jun 23 11:02:58 sonne logger: -Anwender hat Durst-
```

Logrotate

Die detaillierte Aufzeichnung der Vorgänge im System wirft allerdings auch eine neues Problem auf: »Wohin mit all den Daten?«.

Wenn Sie den unmittelbar nach einer frischen Installation aufgezeichneten Speicherbedarf des Verzeichnisses /var/log mit dem Stand einige Wochen später vergleichen, so werden Sie ein Wachstum von einigen Kilobyte bis zu mehreren hundert Megabyte verzeichnen. Selbst wenn letztere Werte wohl eher auf einem frequentierten Server gemessen werden, so führen auch die kleineren Datenmengen auf ihren privaten Rechner irgendwann zu einer Sättigung der Speicherkapazität der Partition.

Ein Volllaufen einer Partition kann drastische Folgen für die Stabilität Ihres Systems haben. Mitunter ist sogar ein Einfrieren dessen denkbar, was einen Neustart und manuellen Eingriff des Administrators erfordert. Im privaten Kämmerlein hakt man das kleine Ärgernis mit einem Schulterzucken ab, aber einen Ausfall des Webservers wird der Chef wohl wenig erbaulich registrieren...

Besser agieren als reagieren und einer solchen Situation zuvor kommen!

Ein Ausweg aus dem Übel ist die regelmäße Archivierung der Protokolldateien und damit eine Beschränkung ihres Wachstums. Neben der Realisierung der Aufgaben durch per Cronjob gestartete Shellskripte hat das Programm **logrotate** eine weite Verbreitung erlangt.

Was kann Logrotate?

Wie der Name schon andeutet, »rotiert« das Programm die Protokolldateien. Typisch ist der Aufruf von **logrotate** über einen Cronjob; die Frequenz hängt vom Datenaufkommen ab und sollte zwischen einmal im Monat und täglich gewählt werden. Darüber hinaus vermag **logrotate** eine Protokolldatei erst zu bearbeiten, wenn sie eine bestimmte Dateigröße überschritten hat.

logrotate kann die Logdateien **komprimieren**. Es kann archivierte Dateien **löschen** oder sie per **Mail** an einen Administrator versenden. Wird **logrotate** mehrfach am Tag gestartet, so modifiziert sie eine Logdatei nur, wenn die Option **-f** angegeben wurde oder die betreffende Datei die maximal zulässige Größe erreicht hat.

Treten bei der Arbeit von **logrotate** Fehler auf, ist das Kommando in der Lage, eine Nachricht darüber zu versenden.

Wird das Kommando mit der Option **-d** gerufen, so werden alle Aktionen »simuliert«, die Protokolldateien und die Statusdatei (/var/lib/logrotate.status) aber nicht verändert. Durch **-s *Datei*** lässt sich **logrotate** zur Verwendung einer alternativen Statusdatei überreden.

Die **Arbeitsweise** von Logrotate lässt sich am einfachsten anhand des Beispiels der Datei /var/log/messages erläutern. Angenommen, die Konfiguration sieht vier Rotationsstufen vor. Auch soll vorerst auf eine Komprimierung verzichtet werden. Nach Ablauf einer bestimmten Zeit - bzw. bei Erreichen einer konkreten Dateigröße - verschiebt **logrotate** die Datei /var/log/messages nach /var/log/messages.1 und erzeugt eine leere /var/log/messages. Sind die Voraussetzungen für eine weitere Rotation erfüllt, wird aus /var/log/messages.1 /var/log/messages.2, aus /var/log/messages wird /var/log/messages.1 und eine neue leere Datei entsteht. Das Ganze setzt sich fort, bis /var/log/messages.4 verschoben werden soll. Da die Rotation allerdings auf 4 Durchläufe beschränkt wurde, wird diese Datei durch /var/log/messages.3 überschrieben, ohne ihren Inhalt weiterhin zu archivieren.

Zumindest bei der auf Dateigröße basierenden Konfiguration ist somit ein maximaler Speicherverbrauch durch die von **logrotate** erfassten Protokolldateien garantiert.

Konfiguration von Logrotate

Das Verhalten von **logrotate** wird üblicherweise durch mehrere Dateien beschrieben. Dabei ist **/ etc/ logrotate.conf** der ursprüngliche Anlaufpunkt für Konfigurationen. Diese Datei selbst bindet jedoch die Dateien des Verzeichnisses **/ etc/ logrotate.d/** ein, die zumeist auf einzelne RPM-Pakete zugeschnittene Konfigurationen beinhalten (alternativ werden alle dem Kommando als Argumente übergebenen Dateien als Konfigurationsdateien eingelesen).

Jede Option, die außerhalb der Definition zu einer Protokolldatei steht, ist eine globale Option und gilt für alle Dateien, insofern sie dort nicht wieder überschrieben wird. Eine spätere Bezugnahme auf eine Option überschreibt frühere Zuweisungen. D.h. die Reihenfolge des Einlesens mehrerer Konfigurationsdateien ist durchaus entscheidend für das Ergebnis.

Betrachten wir zunächst eine typische Datei /etc/logrotate.conf:

```
user@sonne> cat / etc/ logrotate.conf
# Siehe "man logrotate" für Details
# Protokolldateien werden einmal pro Woche rotiert:
weekly

# Die Protokolldateien werden 4 Wochen ("weekly") aufgehoben
rotate 4

# Fehlermeldungen gehen an root
errors root

# Nach dem Verschieben einer Protokolldatei wird eine neue (leere) erzeugt
create

# Die archivierten Dateien werden komprimiert
compress

# Einbinden der Dateien aus /etc/logrotate.d
include /etc/logrotate.d

# Diese eine Datei wird direkt hier beschrieben
/var/log/wtmp {
   monthly
   create 0664 root utmp
   rotate 1
}
```

Für die Datei /var/log/wtmp (»lastlog«) wurden die globalen Werte überschrieben, sodass sie einmal im Monat rotiert wird. Sie wird nur einmal rotiert (es existiert also jeweils nur die Sicherungsdatei des letzten Monats). Und eine leere Datei wird mit den Rechten 644 (ohne Angabe gibt umask diese vor), dem Eigentümer root und der besitzenden Gruppe utmp angelegt.

In den Dateien unter /etc/logrotate.d werden oft die globalen Vorgaben überschrieben:

```
user@sonne> cat / etc/ logrotate.d/ ftpd
/var/log/xferlog {     # ftpd hat Probleme mit dem Signal SIGHUP
   nocompress
}

user@sonne> cat / etc/ logrotate.d/ syslog
/var/log/messages {
   postrotate
     /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
   endscript
}

/var/log/secure {
   postrotate
     /bin/kill -HUP `cat /var/run/syslogd.pid 2> /dev/null` 2> /dev/null || true
   endscript
```

```
}
...
```

Das Senden des Signals SIGHUP an den Syslog-Dämon bewirkt, dass dieser all seine offenen Dateideskriptoren schließt, seine Konfigurationsdatei neu einliest und anschließend mit seiner Arbeit fort fährt. Ohne diese Maßnahme wäre der Deskriptor ungültig (da logrotate dem Syslog seine Datei entrissen hat) und das Ergebnis vermutlich fatal. Zwischen den Schlüsselwörtern **prerotate...endscript** und **postrotate...endscript** können beliebige Kommandos gestartet werden, die im ersten Fall unmittelbar vor dem Rotieren und in letzterem Fall direkt anschließend ausgeführt werden. Denkbar wären hier Skripte, die die Logdateien scannen und verdächtige Zeilen an den Administrator senden.

Die Angabe von Dateinamen darf **Wildcards** enthalten! Nachfolgende Konfiguration ist durchaus legitim und meint »alle Dateien des Verzeichnisses«:

```
user@sonne> cat / etc/ logrotate.d/ news
/var/log/news/* {
  rotate 2
  nomail
  postrotate
    kill -HUP `cat /var/run/inn.pid`
  endscript
}
```

Die Optionen für **logrotate** sind (mit vorangestelltem »no« wird die Bedeutung negiert):

[no]compress

Komprimierung der archivierten Logdateien.

[no]copytruncate

Das voreingestellte Verfahren ist das Verschieben der alten Logdatei mit anschließendem Erzeugen einer ne
Dies funktioniert aber nur, wenn der entsprechende Dienst, der die Logdatei verwendet, in der Lage ist, na
Empfang eines an ihn gerichteten Signals SIGHUP diese Dateien ordnungsgemäß zu schließen und erneut z
öffnen.

Mittels **copytruncate** wird die Protokolldatei zunächst kopiert und das Original abgeschnitten. Der Effekt is
dass der schon offene Dateideskriptor seine Gültigkeit behält und der Dienst problemlos weiter arbeiten kar

[no]create [Rechte] [Eigentümer] [Gruppe]

Erzeugt eine neue leere Logdatei mit den entsprechenden Parameter. Im Falle von nocreate müssen die Parameter entfallen.

daily | weekly | monthly

Die Datei wird täglich | wöchentlich | einmal im Monat rotiert.

[no]delaycompress

Eine erstmals rotierte Datei wird nicht komprimiert, erst im nächsten Durchlauf.

error Mailadresse

Fehler gehen an den angegebenen Empfänger.

extension Dateiendung

Die archiverten Logdateien erhalten die angegebene Endung

[no]ifempty

include Datei | Verzeichnis

Die Konfigurationsdatei bzw. die Dateien des angegebenen Verzeichnisses werden eingelesen.

[no]mail Mailadresse

Die älteste Logdatei wird an die Adresse gemailt (sonst ist sie durch das Löschen für immer verloren).

mailfirst, maillast

Die Optionen arbeiten nur in Verbindung mit **mail** zusammen und bewirken das Versenden des in einer Ru neu erzeugten und des »neuen ältesten« Archivs.

[no]missingok

Fehlt eine Protokolldatei, wird sie übersprungen, ohne eine Fehlermeldung zu generieren.

[no]olddir Verzeichnis

Anstelle im Verzeichnis der originalen Logdatei wird das Archiv in das angegebene Verzeichnis verschoben.

postrotate/ endscript

Alle zwischen den beiden Schlüsselworten eingeschlossenen Kommandos werden unmittelbar nach dem Rotieren ausgeführt.

prerotate/ endscript

Alle zwischen den beiden Schlüsselworten eingeschlossenen Kommandos werden unmittelbar vor dem Roti ausgeführt.

rotate Anzahl

So oft wird eine Protokolldatei rotiert, bevor sie aendgültig gelöscht oder per Mail versandt wird.

size Dateigröße

Übersteigt die Dateigröße der Logdatei den angegebenen Wert, so wird sie rotiert.

[no]sharedscripts

Bezieht sich ein Eintrag einer Konfiguration auf mehrere Dateien (wenn die Angabe Metazeichen enthält), s werden die Skripte zwischen »[post|pre]rotate/endscript« nur einmalig aufgerufen (anstatt für jede Datei einzeln).

X - Konfiguration

Allgemeines

Erfolgte die Konfiguration der grafischen Oberfläche nicht bereits automatisch während der Installation des Systems, so ist es wohl eine der ersten administrativen Maßnahmen, der man sich zuwendet. Wer mag schon freiwillig auf der tristen Konsole verharren?

Stellt sich die Frage: Welche Administrationshilfen stehen unter Linux zur Verfügung? Die Antwort ist gar nicht so leicht in knappe Worte zu fassen, kocht doch jeder Distributor sein eigenes Süppchen und verzichtet auch gern einmal auf Beilegung der »allgemeinen« Werkzeuge. Gar noch verflixter gestaltet sich die Auswahl, da mit dem Versionswechsel von Xfree86 Version 3.3.6 auf Version 4.0 (2000) die alten Werkzeuge nicht mehr zur Konfiguration der aktuellen Server herangezogen werden können, da sie weder die abweichende Syntax der Konfigurationsdatei XF86Config beherrschen noch die erweiterten Möglichkeiten unterstützen.

Warum wir uns nicht auf die neue Version beschränken? Weil noch genügend »alte« (Grafik-) Hardware kursiert und die neue Server-Architektur bewusst auf Unterstützung so manchen Urgesteins verzichtet. Wer so eine in die Jahre gekommene Platine sein Eigen nennt, der wird zwangsläufig einen Server der 3.3.6er Serie aufsetzen müssen.

Vorbereitende Schritte

Bevor Sie sich an die Konfiguration wagen, sollten Sie die Kenndaten Ihrer Hardware parat halten. Wichtig sind vor allem die horizontalen und vertikalen Frequenzbereiche Ihres Monitors, die Sie aus dessen Handbuch erfahren sollten. Manche Hersteller vermerken die Daten auch auf der Rückseite des Geräts. Einige der nachfolgend vorgestellten Konfigurationshilfen enthalten gut funktionierende Mechanismen, um die Werte automatisch zu ermitteln. Da dies allerdings nur bei neueren Modellen zuverlässig klappt und Sie die Fähigkeiten des Bildschirms auch auszunutzen gedenken, ist die manuelle Angabe der Werte der sicherste Weg.

Weniger Aufwand ist bei der Maus zu treiben. Endet ihr Schwanz in einem relativ klobigen Stecker, dann handelt es sich wohl um eine serielle Maus. Die PS/2-Modelle ziert ein runder Stecker und ähnelt in seiner Form einem Telefonstecker, dann wird es eine USB-Maus sein. Ob es eine Rad-Maus ist, lässt sich ebenso leicht feststellen, wie die Anzahl der Tasten, womit wir alle relevanten Informationen beisammen hätten.

Welcher Länderspezifik eine Tastatur folgt, zeigt schon ein Blick auf die Tastenbeschriftung. Nun können Sie einmal die Tasten zählen oder sich gleich 104 vormerken, womit Sie jede gängige Tastatur unter X zähmen können (auch wenn 104 nicht der wirklichen Anzahl entspricht). Für einige spezielle Modelle finden Sie in manchem Werkzeug direkte Entsprechungen.

Die Grafikkarte bereitet zumeist die größten Kopfschmerzen, liegen ihr doch selten irgendwelche Informationen bei, sieht man einmal von der Karte mit den Glückwünschen der Hersteller ab, dass man sich gerade für dieses Modell entschieden hat. Aber auch ohne jegliche Dokumentation muss die Grafikkarte einige Kenndaten preisgeben. Schauen Sie sich hierzu die Ausgabe des Kommandos **lspci** an:

```
user@sonne> /sbin/lspci
...
00:07.3 Host bridge: VIA Technologies, Inc. VT82C596 Power Management (rev 20)
00:09.0 Ethernet controller: Intel Corporation 82557 [Ethernet Pro 100] (rev 08)
01:00.0 VGA compatible controller: S3 Inc. 86c368 [Trio 3D/2X] (rev 02)
```

Die mit *VGA compatible controller* beginnende Zeile verrät, dass es sich im Beispiel um einen S3 Trio 3D/2X Chip handelt. Noch etwas gesprächiger lässt das Kommando **SuperProbe** die Grafikkarte erscheinen:

```
root@sonne> SuperProbe

SuperProbe Version 2.23 (2000 November 28)
      (c) Copyright 1993,1994 by David Wexelblat lt;dwex@xfree86.org>
```

```
      (c) Copyright 1994-1998 by The XFree86 Project, Inc

### gekürzt ###

WARNING - THIS SOFTWARE COULD HANG YOUR MACHINE.
        READ THE SuperProbe.1 MANUAL PAGE BEFORE
        RUNNING THIS PROGRAM.

        INTERRUPT WITHIN FIVE SECONDS TO ABORT!
First video: Super-VGA
      Chipset: S3 Trio3D/2X (PCI Probed)
      Memory:  8192 Kbytes
      RAMDAC:  Generic 8-bit pseudo-color DAC
            (with 6-bit wide lookup tables (or in 6-bit mode))
```

Tatsächlich sind nun der Ausbau an VideoRAM und die RAMDAC bekannt.

Vermutlich werden Sie obige Umwege gar nicht beschreiten müssen, da bei einigermaßen gängiger Hardware die modernen Konfigurationswerkzeuge doch recht zuverlässig die Werte ermitteln. Aber wenn einmal gar nichts geht, dann starten Sie das jeweilige Werkzeug in einem Standardmodus (im Zweifelsfall den 16-Farben-Server) und stellen von Hand die korrekten Werte ein.

Die Werkzeuge

xf86config - Für den Konsolen-Liebhaber

Übersicht

Im Unterschied zu den nachfolgend beschriebenen X-Konfigurations-Werkzeugen sollte das Programm »xf86config« jedem X-Server-Paket beiliegen. Für die neueren »X-Server-Versionen 4.x« wurde ein angepasstes Programm »xf86config4« geschrieben. Sind beide Server-Versionen auf einem System installiert, ist »xf86config« meist ein Link entweder auf »xf86config3x« oder eben auf »xf86config4«. Abgesehen von geringen kosmetischen Änderungen besteht in der Bedienung beider Programme kein Unterschied. »xf86config4« schreibt die Konfigurationsdatei nur im neuen Format, wobei das Programm nicht die Konfiguration der wesentlichen Neuerungen von XFree86-4 (Multihead, DRI, DGA...) beherrscht. Wir beschreiben nachfolgend die »originale« Version von »xf86config«.

Der Einsatz von »xf86config« kommt vor allem in Frage, wenn der Versuch XF86Setup bzw. Xconfigurator oder Sax zu starten, scheitert. Dies ist bei relativ alter Grafikhardware durchaus denkbar, was zum Glück noch lange nicht heißen muss, dass man die grafische Oberfläche nicht doch noch zum Laufen überredet...

Auswahl und Konfiguration des X-Servers

Die Verwendung von »xf86config« erfordert vorab die Kenntnis der Parameter von Grafikkarte und Monitor. Die technischen Daten lassen sich meist aus der diesen Geräten beigelegten Dokumentation entnehmen oder wie im einführenden Abschnitt beschrieben ermitteln. »xf86config« selbst enthält keinerlei Routinen zur automatischen Hardware-Erkennung.

Anhand der Informationen zum Chipsatz, sollte der entsprechendene X-Server installiert werden (dieser Schritt gilt nur für Xfree Version 3.3.6). Die meisten Kartentreiber sind im SVGA-Server enthalten. Für andere Karten existieren eventuell spezielle X-Server, deren Namen zumeist auf den unterstützten Chipsatz bzw. Kartenhersteller hindeuten (z.B. XF86Com_Matrox für Matrox-Karten).

Ist der entsprechende Server installiert, kann mit der Konfiguration mittels »xf86config« begonnen werden. (wird der Server zu einem späteren Zeitpunkt installiert, ist »xf86config« nicht in der Lage, einige Links korrekt anzulegen).

```
root@sonne> xf86config
This program will create a basic XF86Config file, based on menu selections you make.
# ...
```

Mauseinstellung

Zunächst fordert das Programm Sie zur Spezifikation der Maus auf (im Beispiel entsprechen die Eingaben einer nach dem Microsoft-Protokoll arbeitenden 2-Tasten-Maus):

```
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
```

```
10.  Acecad tablet

# ...

Enter a protocol number: 1
```

Den entsprechenden Typ entnehmen Sie am einfachsten aus der Beschreibung. Beachten Sie, 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 eine dritte Taste simuliert werden soll (indem beide Tasten gleichzeitig gedrückt werden). Diese Option ist bei 2-Tasten-Mäusen zu bejahen.

```
Do you want to enable ChordMiddle? y
```

Verfügen Sie über eine Maus mit drei Tasten, sollte die folgende Option gewählt werden:

```
Do you want to enable Emulate3Buttons? n
```

Das Device kann in den meisten Fällen in der Voreinstellung übernommen werden.

```
Mouse device: /dev/mouse
```

Tastatur

Möchten Sie voreingestellte Tastaturtabellen verwenden, ist die folgende Frage zu bejahen:

```
Do you want to use XKB? y
```

Als Nächstes ist die gewünschte Tastaturtabelle anzugeben:

```
List of preconfigured keymaps:

1  Standard 101-key, US encoding
2  Microsoft Natural, US encoding
3  KeyTronic FlexPro, US encoding
4  Standard 101-key, US encoding with ISO9995-3 extensions
5  Standard 101-key, German encoding
6  Standard 101-key, French encoding
7  Standard 101-key, Thai encoding
8  Standard 101-key, Swiss/German encoding
9  Standard 101-key, Swiss/French encoding
  10  Standard 101-key, US international
  11  Brazilian ABNT2
  12  None of the above
Enter a number to choose the keymap.

5
```

Später lässt sich die Tastaturbelegung z.B. mit dem Kommando »xmodmap« anpassen (so könnten Sie auch die ansonsten unnützen Tasten einer Windows-Tastatur mit eigenen Funktionen belegen).

Monitor

Die Frequenzen für den Monitor erfahren Sie aus dem Handbuch; für etliche Modelle finden Sie 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
 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

Enter your choice (1-11): 11
#...
Horizontal sync range:  30-96
```

Analog verfähren Sie mit den vertikalen Frequenzen.

```
 1  50-70
 2  50-90
 3  50-100
 4  40-150
 5  Enter your own vertical sync range

Enter your choice: 5

Vertical sync range: 50-150
```

Die drei folgenden Angaben bez. des Monitornamens u.a. können Sie ignorieren (nur im Falle des Ansteuerns mehrerer Monitore ist ein eindeutiger Bezeichner erforderlich; allerdings benötigen Sie für derartige Konfigurationen ohnehin ein anderes Konfigurationswerkzeug).

Grafikkarte / Server

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

Jetzt sollten Sie hoffen, dass Ihre Karte in der Liste der unterstützten Grafikkarten auch enthalten ist. Die gewählte Option sollte der Karte zu einhundert Prozent entsprechen, sonst ist für das Überleben dieser nicht zu garantieren!

```
#...
  23  AOpen PA50E                 SiS6326
  24  AOpen PA50V                 SiS6326
  25  AOpen PA80/DVD                SiS6326
  26  AOpen PG128                 S3 Trio3D
  27  AOpen PG975                 3dimage975
#...

26
Your selected card definition:

Identifier: AOpen PG128
Chipset:    S3 Trio3D
Server:  XF86_SVGA
Do NOT probe clocks or use any Clocks line.
```

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, sollten Sie hier entweder den Monochrome- oder den 16-Farben-Server auswählen, ansonsten nehmem Sie den am besten geeigneten (den unter »Server:« spezifizierten) Server.

```
1  The XF86_Mono server. This a monochrome server that should work on any
VGA-compatible card, in 640x480 (more on some SVGA chipsets).
2  The XF86_VGA16 server. This is a 16-color VGA server that should work on
any VGA-compatible card.
3  The XF86_SVGA server. This is a 256 color SVGA server that supports
a number of SVGA chipsets. On some chipsets it is accelerated or
supports higher color depths.
4  The accelerated servers. These include XF86_S3, XF86_Mach32, XF86_Mach8,
XF86_8514, XF86_P9000, XF86_AGX, XF86_W32, XF86_Mach64, XF86_I128 and
XF86_S3V.

These four server types correspond to the four different "Screen" sections in
XF86Config (vga2, vga16, svga, accel).

 5  Choose the server from the card definition, XF86_SVGA.

Which one of these screen types do you intend to run by default (1-5)? 5
```

Haben Sie sich für einen Server entschieden, so lassen Sie gleich vom Programm den Link von »/usr/x11R6/bin/X« auf den Server setzen:

```
Do you want me to set the symbolic link? y
```

Ein 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

```
How much video memory do you have on your video card:

 1  256K
 2  512K
 3  1024K
 4  2048K
 5  4096K
 6  Other

Enter your choice: 6

Amount of video memory in Kbytes: 8192
```

- Name, Hersteller, Typ - kann ignoriert werden

- RAMDAC und Clock-Chip bei beschleunigten Servern (die Werte sollten nur angegeben werden, wenn diese in der Grafikkartenbeschreibung auch exakt spezifiziert sind):

```
 1  Chrontel 8391                                        ch8391
 2  ICD2061A and compatibles (ICS9161A, DCS2824)         icd2061a
 3  ICS2595                                              ics2595
 4  ICS5342 (similar to SDAC, but not completely compatible)    ics5342
 5  ICS5341                                              ics5341
 6  S3 GenDAC (86C708) and ICS5300 (autodetected)        s3gendac
 7  S3 SDAC (86C716)                                     s3_sdac
 8  STG 1703 (autodetected)                              stg1703
 9  Sierra SC11412                                       sc11412
10  TI 3025 (autodetected)                               ti3025
11  TI 3026 (autodetected)                               ti3026
12  IBM RGB 51x/52x (autodetected)                       ibm_rgb5xx

Just press enter if you don't want a Clockchip setting.
What Clockchip setting do you want (1-12)?
```

- Die Clock-Lines sind nur für alte Grafikkarten noch von Bedeutung. Sie werden zur Berechnung der Bildwiederholfrequenzen herangezogen. Haben Sie weiter oben bereits eine konkrete Grafikkarte angegeben, wird meist nachgefragt, ob ein Autoprobing notwendig ist:

```
The card definition says to NOT probe clocks.
Do you want me to run 'X -probeonly' now?  n
```

Zuletzt lassen sich die verschiedenen Grafikauflösungen für jede Farbtiefe angeben.

```
"640x480" "800x600" "1024x768" "1280x1024" for 8bpp
"640x480" "800x600" "1024x768" "1280x1024" for 16bpp
"640x480" "800x600" "1024x768" "1280x1024" for 24bpp
"640x480" "800x600" "1024x768" for 32bpp

Note that 16, 24 and 32bpp are only supported on a few configurations.
Modes that cannot be supported due to monitor or clock constraints will
be automatically skipped by the server.

 1  Change the modes for 8pp (256 colors)
 2  Change the modes for 16bpp (32K/64K colors)
 3  Change the modes for 24bpp (24-bit color, packed pixel)
 4  Change the modes for 32bpp (24-bit color)
 5  The modes are OK, continue.

Enter your choice:
```

In den ersten Zeilen werden die verfügbaren Auflösungen für die jeweiligen Farbtiefen angegeben. Werden die Daten nicht geändert, so wird beim Start von X Window die niedrigste Farbtiefe bei geringster Auflösung gewählt.

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.

X-Konfiguration mit XF86Setup

Übersicht

»XF86Setup« ist ein vom XFree86-Projekt entwickeltes Konfigurationstool für die X-Server der Versionen 3.3.x und sollte damit jeder »umfangreichen« Distribution beiliegen. In älteren Debian-Distributionen (vor »Potato«) ist dieses Werkzeug die einzige komfortable Hilfe zum Einrichten des X-Servers.

Wie so ziemlich alles, was etwas mit Konfiguration zu tun hat, darf auch dieses Programm nur von Root ausgeführt werden:

```
root@sonne> XF86Setup
```

Wenn es sich bei der Grafikkarte nicht gerade um ein exotisches Modell handelt, sollte zumindest der 16-Farben-Server starten und der Startbildschirm erscheinen:

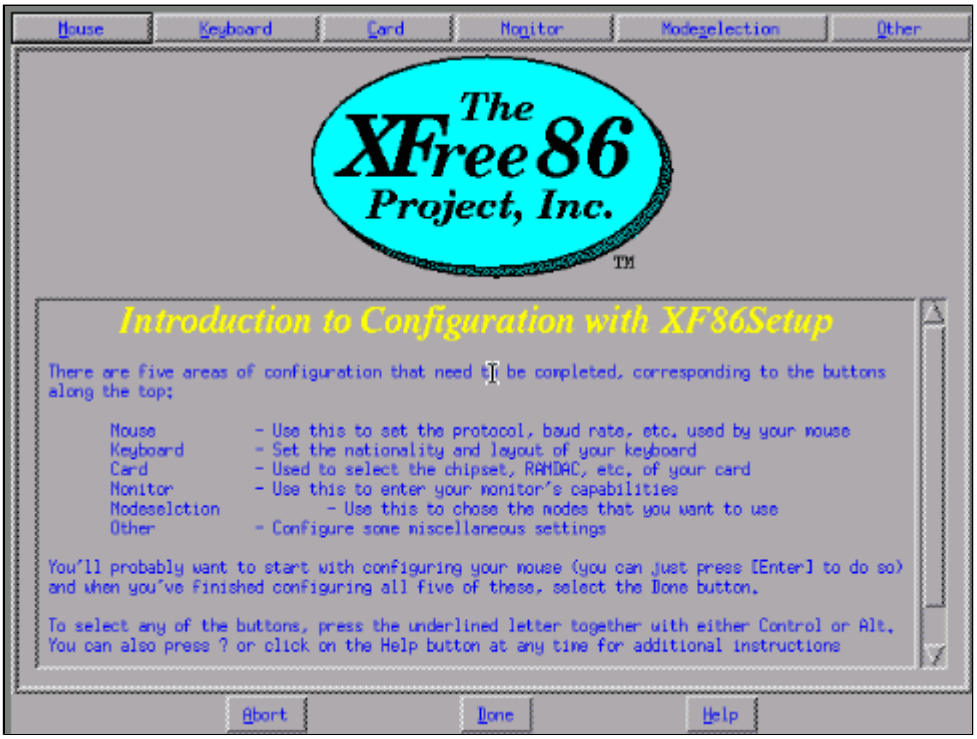


Abbildung 1: Startbildschirm von XF86Setup

Die Reihenfolge des Einrichtens der einzelnen Komponenten ist nicht vorgeschrieben. Wir betrachten an dieser Stelle die einzelnen Eingabemasken von links nach rechts.

Die Maus

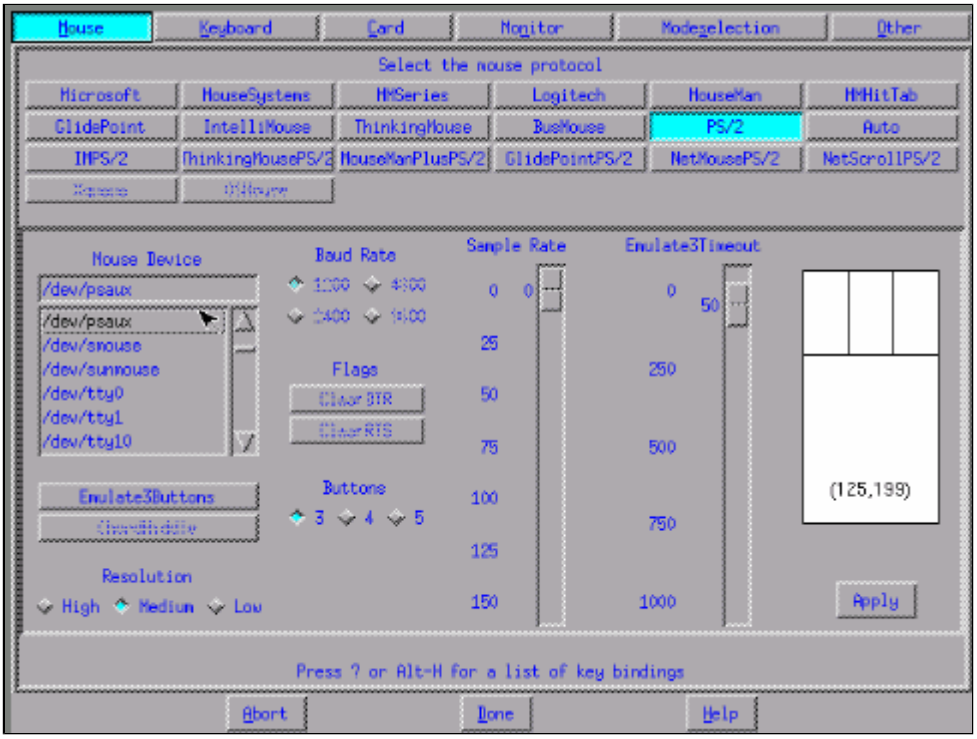


Abbildung 2: Mauskonfiguration

Mit etwas Glück wurde die Maus schon beim Starten erkannt, dann sollten die Voreinstellungen den Gegebenheiten entsprechen. Ist dies nicht der Fall, so wechseln Sie über die Taste [p] zum entsprechenden Protokoll (die oberen Felder). Verwechseln Sie nicht Hersteller und Protokoll; auch viele Logitech-Mäuse arbeiten nach dem Microsoft-Protokoll! Mittels [Tab] erreichen Sie die Liste der Devices. Aktivieren Sie hier den Eintrag, an dem Ihre Maus installiert ist. So hängt eine serielle Maus i.d.R. an der ersten seriellen Schnittstelle (/dev/tty0); eine PS/2-Maus ist an /dev/psaux angeschlossen. Wechseln Sie weiter mittels [Tab] zum Feld *Apply*. Nach Aktivierung dessen sollte Ihre Maus nutzbar sein. Wenn nicht, so versuchen Sie es mit einem anderen Protokoll oder einer anderen Schnittstelle.

Verfügt Ihre Maus nur über 2 Tasten, sollte *Emulate3Buttons* selektiert werden. Die übrigen Werte dienen zum Optimieren der Klickzeiten (wie schnell muss eine Taste zweifach betätigt werden, um dies als Doppelklick zu interpretieren usw.).

Das Rad von Wheel-Mäusen lässt sich mit XF86Setup nicht konfigurieren!

Die Tastatur

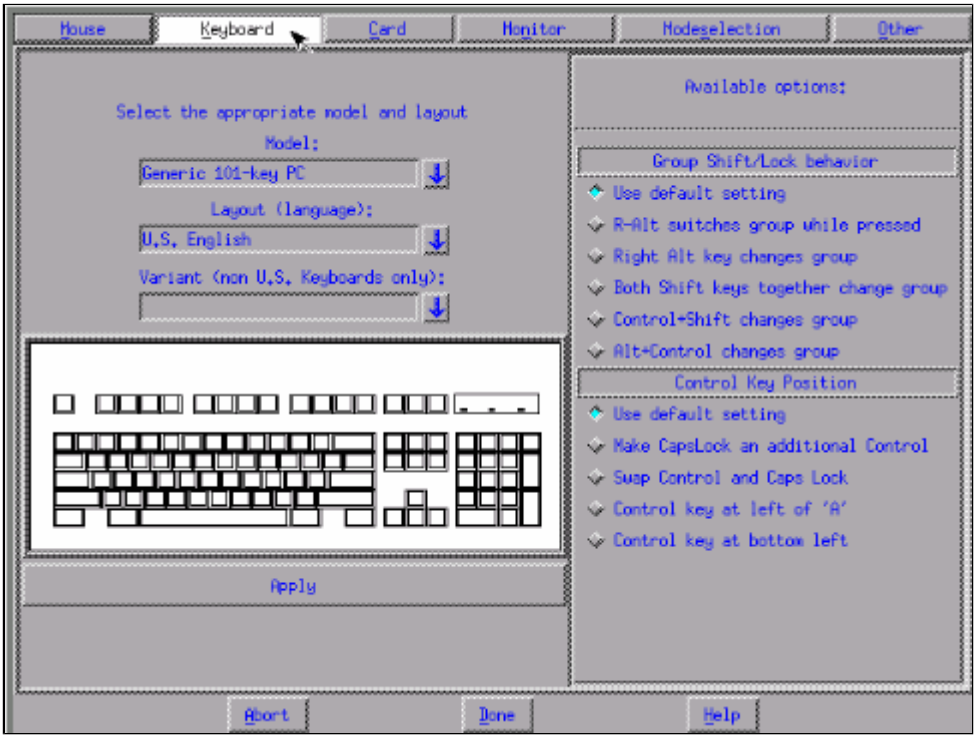


Abbildung 3: Die Tastatur einrichten

Bei den Standard-Tastaturen mit Windows-Tasten handelt es sich um das als *Generic 104-key PC* bezeichnete *Modell*. Als *Layout* sollten Sie *German* und *Eliminate dead keys* als *Variante* wählen.

Die Einträge der rechten Seite sind nützlich, um die Taste [Ctrl] zu verlegen oder um die Taste [AltGr] zu (de) aktivieren. In den meisten Fällen sollten Sie es bei der default-Einstellung belassen.

Die Grafikkarte

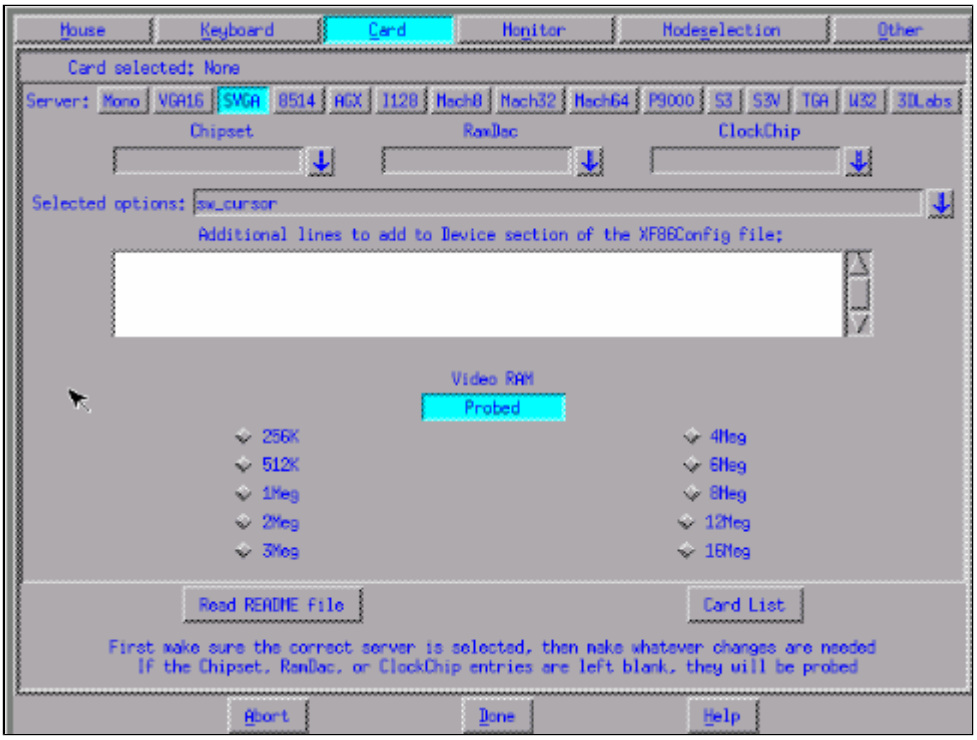


Abbildung 4: Grafikkarte, detailliertes Setup

Die Auswahl der Grafikkarte ist der heikelste Schritt der Konfiguration. Mitunter werden Sie feststellen, dass neuere

Grafikkarten-Modelle in der Liste nicht aufgeführt sind. In einem solchen Fall können Sie versuchen unter *Detailed Setup* die Werte für Chipsatz, Ramdac etc. von Hand einzugeben (die Abbildung zeigt die Maske der erweiterten Eingabe). Diese erfahren Sie eventuell durch das bereits erwähnte Superprobe. Seien Sie vorsichtig mit Angaben, die nicht exakt Ihrer Karte entsprechen!

Eine Fülle von Einstellungen ist über *Selected options* möglich. Falls bspw. der Mauscursor verschoben dargestellt wird, kann die Option "sw_cursor" den Fehler richten. In den Manuals zu X finden Sie nähere Erläuterungen zu einzelnen Optionen.

Der Monitor

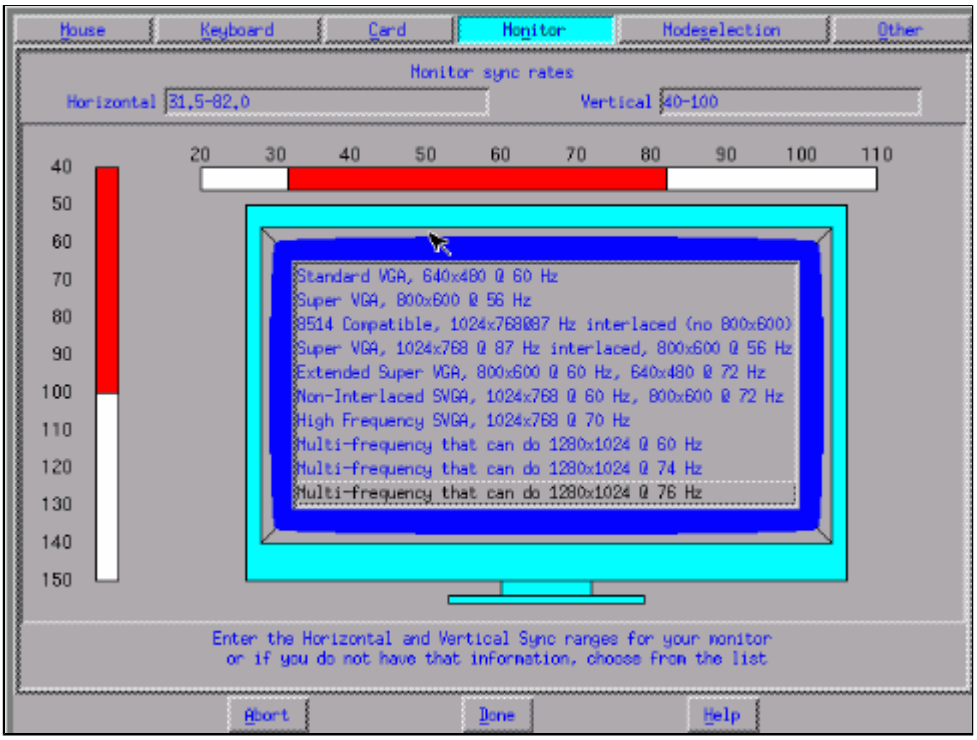


Abbildung 5: Monitor-Frenquenzangabe

Es genügt die Angabe der horizontalen und vertikalen Frequenzen gemäß den Angaben aus dem Monitorhandbuch (bei vielen Modellen finden Sie die Werte auch auf der Gehäuserückseite).

Treten später Darstellungsfehler auf, können Sie eventuell durch eine geringere Frequenzangabe korrigiert werden.

Die Modi

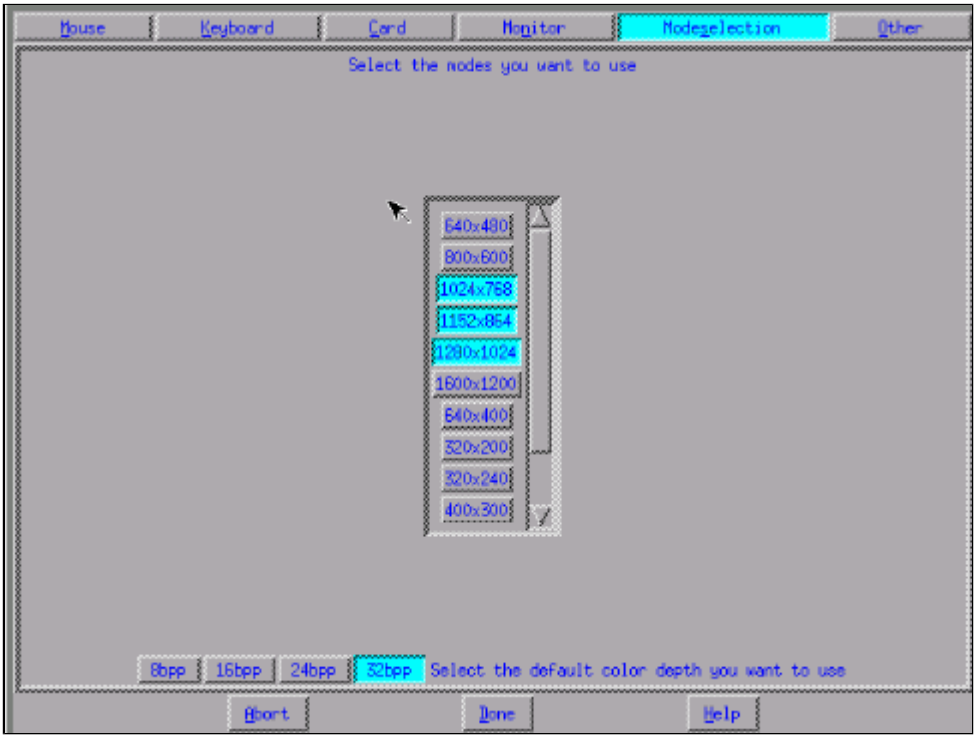


Abbildung 6: Auflösung und Farbtiefen

Die gewählten Auflösungen werden für jede Farbtiefe (8,16,24,32 Bit) konfiguriert. Mit der voreingestellten Farbtiefe spezifizieren Sie, welchen Wert der X-Server verwenden soll, wenn keine Farbtiefe auf der Kommandozeile angegeben wurde.

Beachten Sie, dass die größte gewählte Auflösung gleichzeitig die Ausmaße des virtuellen Desktops bestimmt, d.h. wenn Sie unter X eine kleinere Auflösung wählen, wird nur ein Ausschnitt des virtuellen Bildschirms entsprechender Größe dargestellt.

Sonstiges

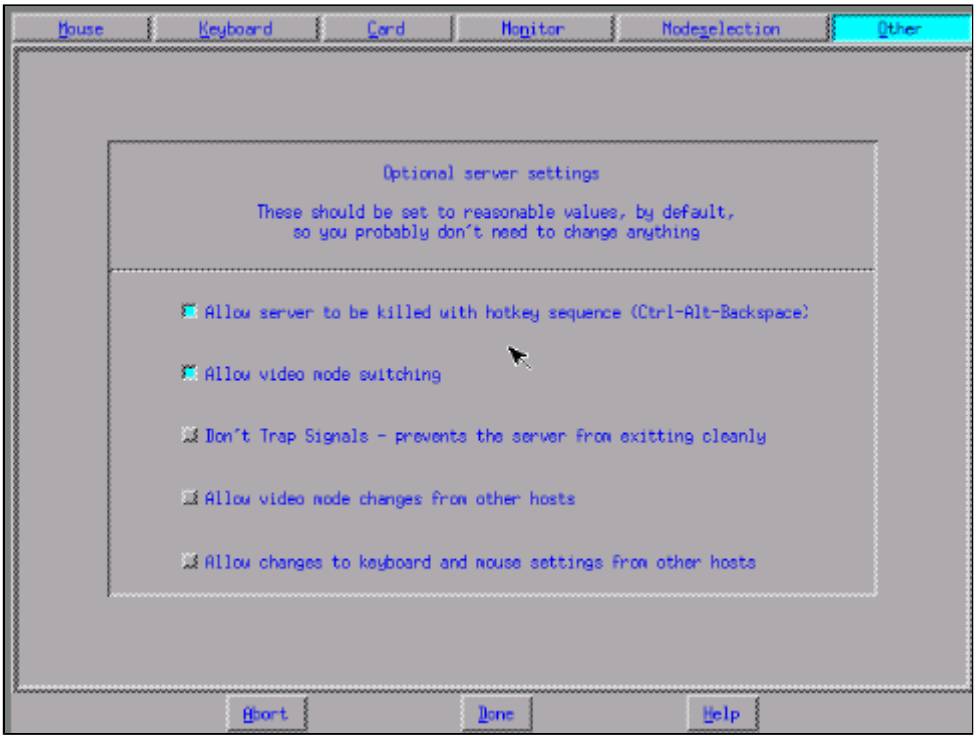


Abbildung 7: Sicherheitseinstellungen

Die letzten Einstellungen betreffen einige Sicherheitsaspekte. So erlaubt Punkt 1 das Beenden des Servers durch [Ctrl][Alt][Backspace].

Über den Button »Done« des Hauptfensters gelangen Sie zum Abschluss der Konfiguration... naja nicht ganz. Zunächst wird die Möglichkeit geboten, mittels »xvidtune« noch eine Feinjustierung vorzunehmen...

Die Feinabstimmung

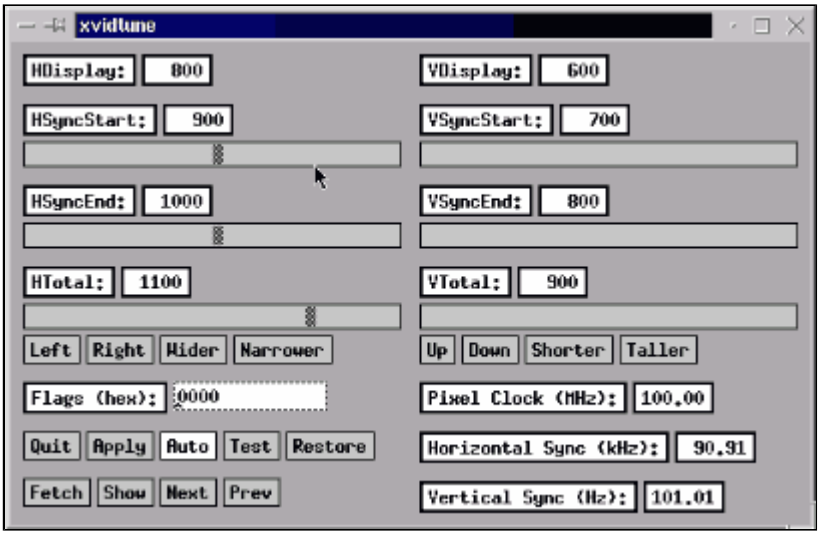


Abbildung 8: Feinabstimmung mit »xvidtune«

Das Programm (Abbildung 8) kann auch zu einem späteren Zeitpunkt über den Namen »xvidtune« gestartet werden.

Hilfreich ist die Aktivierung des Schalters »Auto«, um die Änderungen direkt am Bildschirm verfolgen zu können. In der linken Hälfte der Anwendung konfigurieren Sie die horizontale Ausrichtung der Bildschirmanzeige. Mit »Left« wird die Darstellung schrittweise gegen den linken Bildschirmrand verschoben, mit »Right« nach rechts. »Wider« verbreitert und »Narrower« schmälert die Ausgabe.

Analog verfahren Sie mit der vertikalen Ausrichtung im rechten Teil des Anwendungsfensters. Mit »Up« rückt das Bild nach oben, mit »Down« geht's nach unten. »Shorter« lässt die Höhe schrumpfen und »Taller« wachsen.

Jederzeit können Sie die alten Werte mittels »Restore« restaurieren, vor einem Abschluss mit »Quit« sollten die neuen Parameter mittels »Apply« übernommen werden.

xf86cfg - Konfigurieren von XFree86 Version 4

Übersicht

Da sich die Syntax der Konfigurationsdatei XF86Config für die X-Server-Versionen 4.*x* von denen ihrer Vorgänger unterscheidet, wurden auch neue Konfigurationswerkzeuge erforderlich, die auf die neuen Möglichkeiten eingehen können. **xf86cfg** ist eines der Werkzeuge, das zwar in der derzeitigen Implementierung noch nicht alle Feinheiten unterstützt, aber zumindest die Ansätze einer kompletten Konfigurationshilfe aufweist.

xf86cfg bedingt einen vorkonfigurierten X-Server. Damit ist es erforderlich, die Ersteinrichtung als **Root** vorzunehmen. Das Programm entscheidet anhand der Shellvariablen »DISPLAY«, ob eine existierende Datei XF86Config zum Start von X herangezogen werden soll (DISPLAY ist gesetzt) oder ob zuvor eine neue Konfigurationsdatei erzeugt werden muss (DISPLAY nicht gesetzt). **xf86cfg** startet in letzterem Fall implizit »XFree86 -configure«, was allerdings der Rootberechtigung bedarf.

Die durch »XFree86 -configure« ermittelten Daten landen in einer Datei »~root/XF86Config«, die **xf86cfg** nachfolgend zum Serverstart verwendet. Ein Problem könnte nun sein, dass die resultierende Auflösung zur raschen Ermüdung der Augen führt, da der X-Server mit der maximal möglichen Auflösung startet und das Konfigurationsprogramm winzig klein in der Ecke des Desktops sein Dasein fristet. Um eine vernünftige Darstellung zu erzwingen, könnten Sie die Datei »~root/XF86Config« editieren und in der *Section Screen* die hohen Auflösungen durch geringere ersetzen (bspw. »1600x1200« durch »800x600«).

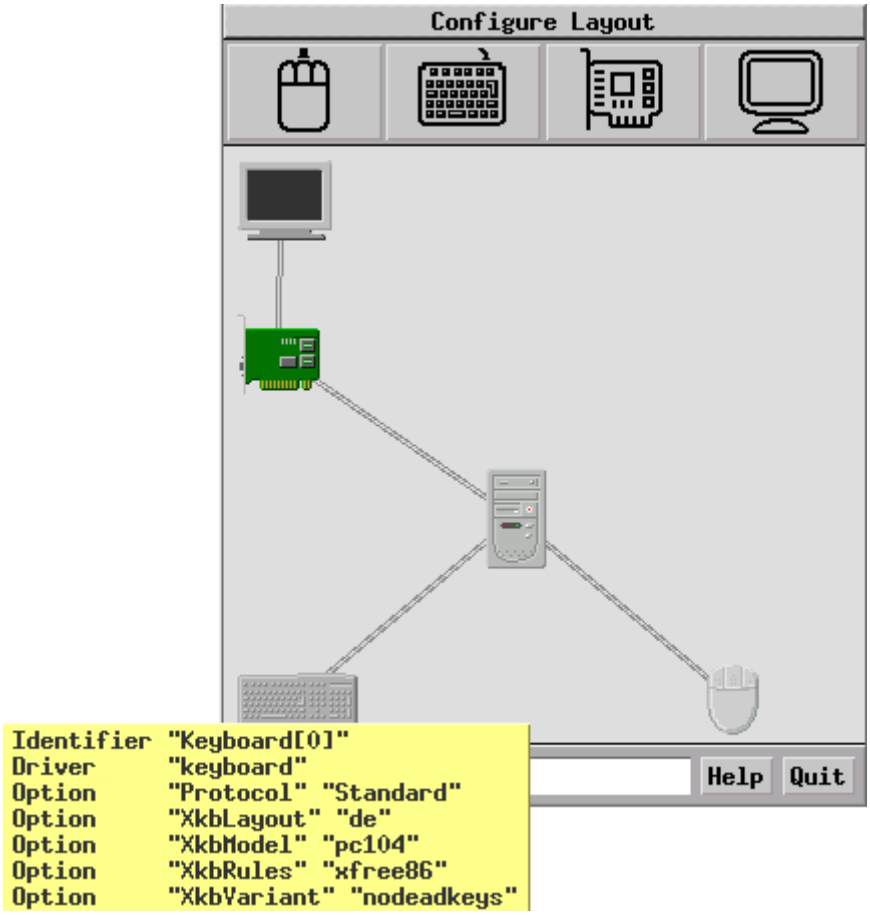


Abbildung 1: xf86cfg - Hauptbildschirm

Wenn Sie mit **xf86cfg** erstmals die X-Server-Konfiguration vornehmen, sollten Sie sich nach dem Programmstart in einer Ansicht analog zur obigen Abbildung vorfinden, d.h. die Geräte Tastatur, Maus und Grafikkarte sind symbolisch mit dem X-Server (Rechner) und der Monitor mit der Grafikkarte verbunden. Eine solche Konfiguration von Geräten wird **Layout** genannt.

Indem Sie mit der rechten Maustaste auf den mit *Configure Layout* bezeichneten Schalter drücken, gelangen Sie in ein Menü, das den Wechsel in weitere Ansichten erlaubt. Wir werden in den Abschnitten Modelines, Auflösungen und Farbtiefen (screen) und Server-Zugriff (access) darauf zurück kommen.

Die Zeile der *Gerätesymbole* ermöglicht die Hinzunahme weiterer Hardware zu einem Layout. Dies ist bspw. sinnvoll, wenn Sie mehrere Monitore an mehreren Grafikkarten (oder einer Dualhead-Grafikkarte) betreiben und diese unter X verwenden möchten (»Xinerama«). Wir kommen auch darauf zurück. Vorläufig nehmen wir an, dass jedes Gerät nur einmal im System vorhanden ist.

Wenn Sie mit der Maus über einem der Symbole verweilen, so wird ein Textfeld mit der aktuellen Konfiguration eingeblendet. Betätigen Sie hingegen über dem Symbol die **rechte Maustaste**, so öffnet sich ein Menü mit folgenden Einträgen:

configure

Führt zum Konfigurationsdialog des Gerätes; zumeist bietet der Dialog nur die wichtigen Optionen und Standardeinstellungen an

options

Ermöglicht die manuelle Konfiguration; hier kann jede Option mit jedem Wert eingegeben werden

enable

Aktiviert das Gerät; symbolisch wird eine Verbindung vom Gerät zum Server hergestellt

disable

Deaktiviert das Gerät

remove

Entfernt das Gerät aus der Konfiguration

Nicht alle Menüeinträge sind für alle Geräte zugänglich. Andere Einträge werden kontextabhängig freigeschaltet, so ist ein (De)Aktivieren erst nach erfolgter Konfiguration sinnvoll.

Konfiguration der Maus

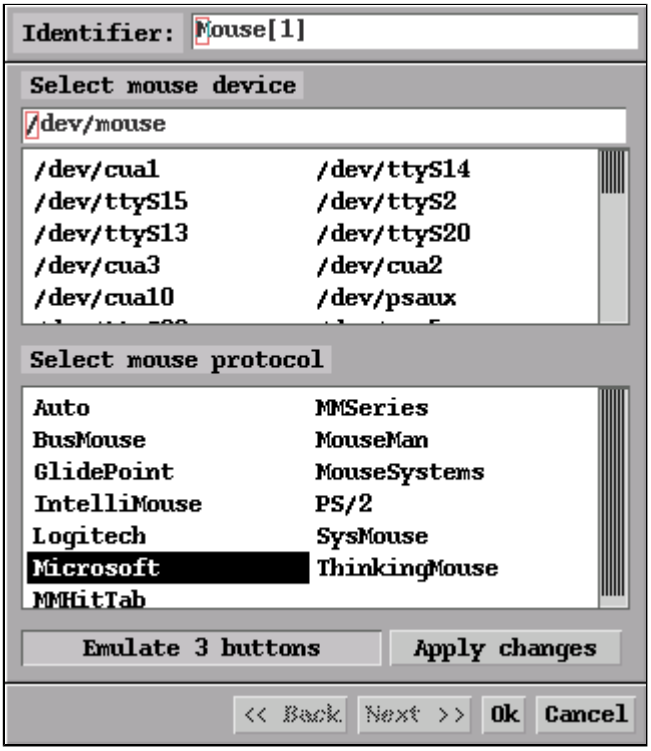


Abbildung 2: xf86cfg - Mauskonfiguration

Falls Sie überhaupt bis in den Konfigurationsdialog vorgedrungen sind, heißt das, dass die Grundfunktionalität Ihrer Maus bereits gegeben ist (das Programm lässt sich nicht per Tastatur bedienen). Für 2-Tasten-Modelle sollten Sie sich noch vergewissern, dass *Emulate 3 buttons* aktiviert ist, womit sich die sinnvollen Änderungen in diesem Dialog auch schon erschöpfen.

Das Rad von **Wheelmäusen** bringen Sie hier nicht zum Rollen. Hierzu sollten Sie in den unter **options** erreichbaren Dialog wechseln und zunächst das Protokoll anpassen. Tragen Sie hierzu in das linke Eingabefeld »Protocol« ein und in das rechte den Protokolltyp. Für viele Mäuse sollte hier »IMPS/2« oder »IntelliMouse« stehen, für eine Logitech MouseMan+ lautet der Eintrag »MouseManPlusPS/2« und für Microsofts IntelliMouse Explorer »ExplorerPS/2«.

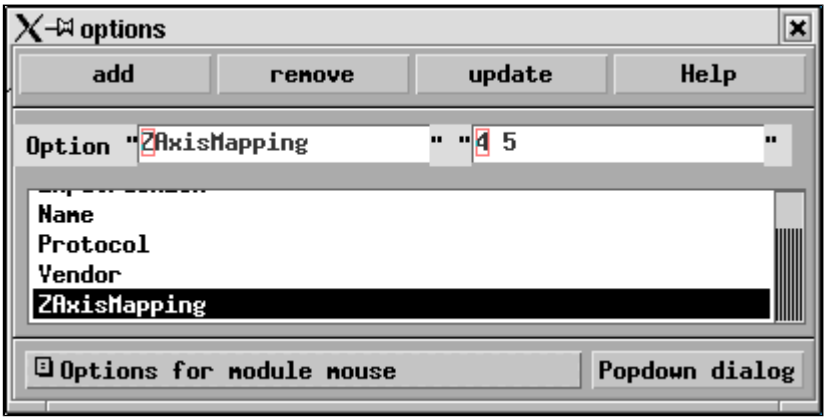


Abbildung 3: xf86cfg - Mausrad-Konfiguration

Als weitere Option tragen Sie in das linke Feld (Abbildung 3) »ZAxisMapping« und rechts zwei Werte »*x y*« ein, wobei *x* eine um 1 höhere Zahl ist als die Maus Tasten besitzt. *y* ist immer der Wert von *x+1* (bei einer 3-Tastenmaus also »4 5«).

Hinweis: Für Anwendungen, die die Unterstützung von Wheel-Mäusen nicht integriert haben, benötigen Sie zusätzlich das Programm **imwheel**.

Konfiguration der Tastatur

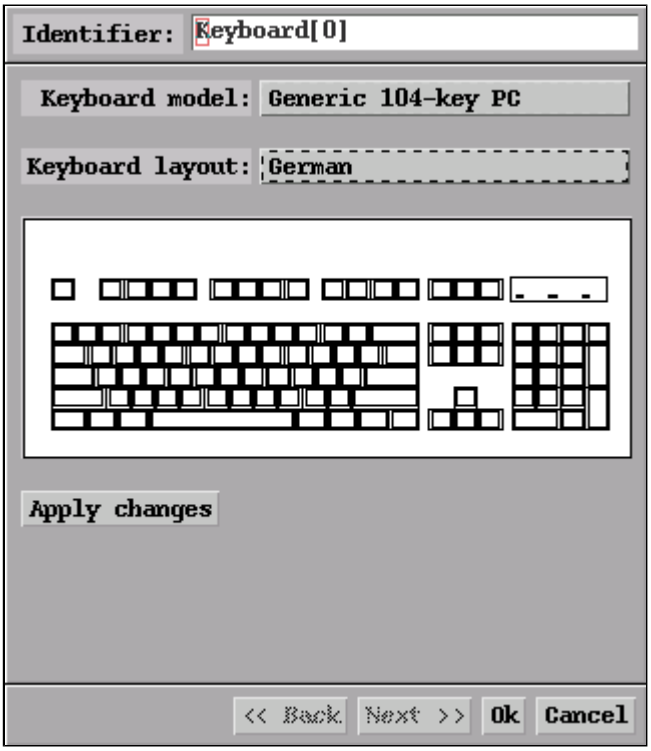


Abbildung 4: xf86cfg - Tastaturkonfiguration

Jede Änderung, die Sie hier vornehmen, wird bei Betätigung des Schalters *Apply changes* sofort wirksam. Unter einem laufenden X-Window-System könnten Sie somit in einem Terminalfenster das Verhalten der Tastatur kontrollieren. Falls die so bezeichneten »toten Tasten« (^ ~ `) nicht direkt verfügbar sind (sondern erst nach einem folgenden Drücken der Leertaste), so empfiehlt es sich, anschließend in den **options**-Dialog zu wechseln und den Wert **XkbVariant** auf **nodeadkeys** zu setzen.

Konfiguration der Grafikkarte

Die Einrichtung der Grafikkarte ist der komplexeste Teil der gesamten Konfiguration. Mit etwas Glück finden Sie Ihr Modell in der Liste aufgeführt, womit sich die Administration auf die Selektion des Listeneintrags beschränkt. Ist die Karte nicht enthalten, kann vielleicht der Treiber des Vorgängermodells den Ausweg bedeuten.

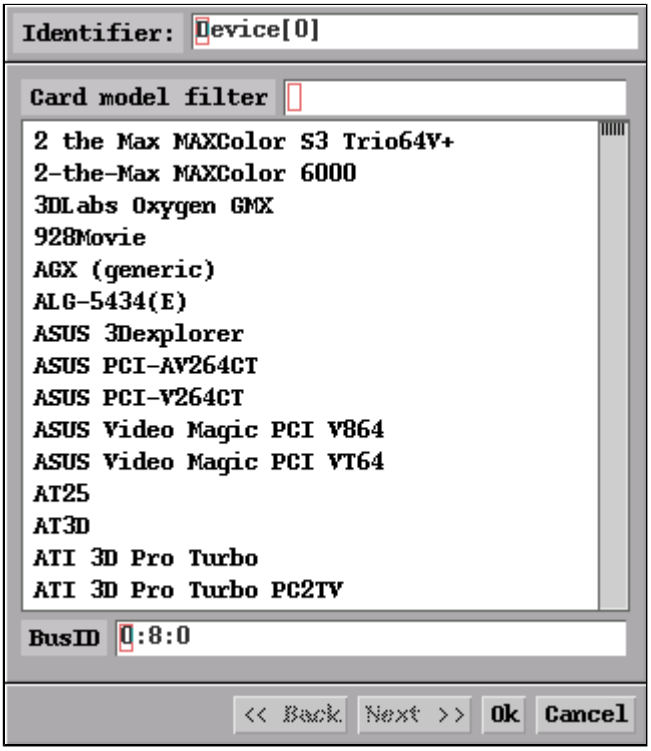


Abbildung 5: xf86cfg - Konfiguration der Grafikkarte

Für den Fall, dass Ihre Karte in keinster Weise erwähnt wurde und die aktuelle Konfiguration (mit der xf86cfg startete) keinesfalls zufrieden stellend ist, sollten Sie alle Werte per Hand eintragen, indem Sie in den Dialog **options** wechseln. Informationen über Ihre Grafikkarte erhalten Sie zumeist aus deren Handbuch. Eventuell helfen auch die Ausgaben der Kommandos **SuperProbe**, **X -scanpci** oder **lspci -v** weiter.

In der Beschreibung zur Datei XF86Config erfahren Sie, welche Optionen in der zuständigen Sektion »Device« möglich sind.

Konfiguration des Monitors

Auch das Menü zur Monitoreinrichtung erreichen Sie per Mausklick mit der rechten Taste auf das Monitorsymbol.

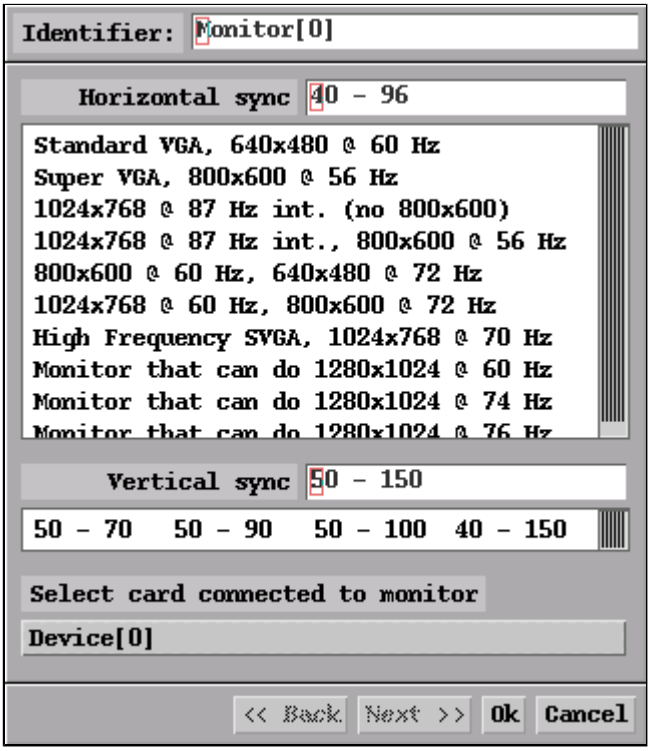


Abbildung 6: xf86cfg - Konfiguration des Monitors

Tragen Sie am einfachsten die Werte für die horizontale und vertikale Freuqenz ein. Diese sollten entweder auf der Monitorrückseite oder im Handbuch zu finden sein. Wählen Sie keinesfalls höhere als die zulässigen Werte!

Um den Monitor einer konkreten Grafikkarte zuzuorden, müssen Sie diese unterhalb des Feldes *Select card connected to monitor* auswählen (rechte Maustaste)!

Weitere Optionen (...im **options**-Dialog) werden nur selten von Nöten sein; eventuell kann bei TFT-Bildschirmen eine Option **gamma** erforderlich werden, falls die Darstellung dunkler (ein Wert zwischen 0.1 und 1.0) bzw. heller (1.0 bis 10.0) erfolgen soll.

Serveroptionen

Im Menü des Servers ist einzig der Eintrag **options** zugänglich. Hinweise zu möglichen Einträgen finden Sie im Zusammenhang mit der Beschreibung der XF86Config-Datei.

Abbildung 7: xf86cfg - Serveroptionen

Modelines

Die Ansicht der Modelines erreichen Sie, indem Sie auf dem obersten Schalter per rechtem Mausklick den gleichnamigen Menüpunkt markieren.

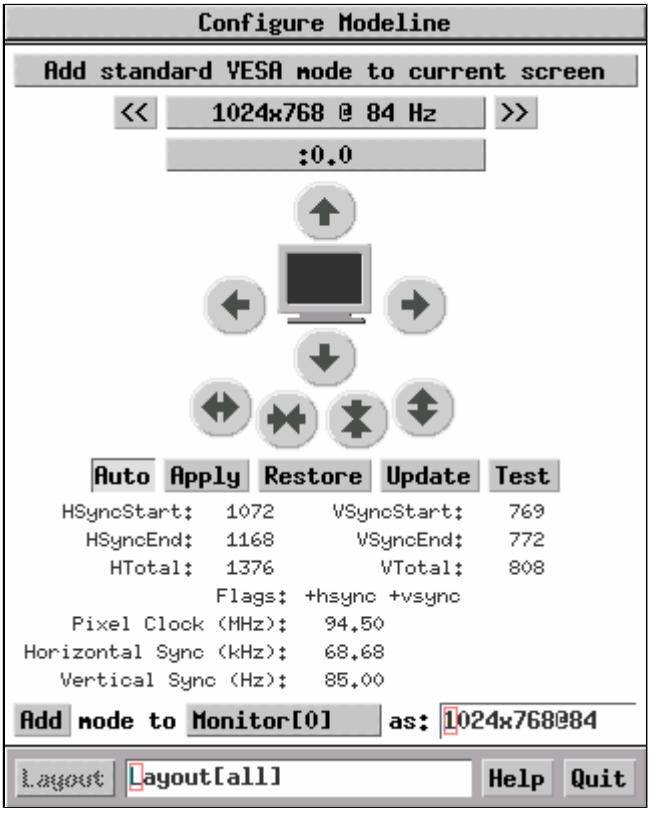


Abbildung 8: xf86cfg - Modelines

Die so genannten Modelines steuern die Lage und das Format des Bildschirms und beeinflussen damit maßgeblich die Bildwiederholfrequenz des Monitors. Spätestens seit Werkzeugen wie xvidtune oder eben **xf86cfg** haben die Modelines viel von Ihrem Schrecken verloren. Wer dennoch hinter den Sinn der vielen Werte steigen möchte, der sei auf den Abschnitt Monitor aus der Datei **XF86Config** verwiesen.

Für jede Kombination aus Auflösung und Monitor kann die Ausrichtung des Bildes separat vorgenommen werden. Welche Auflösungen überhaupt verfügbar sind, wird in der Konfiguration der »Screens« fest gelegt; der Wechsel zwischen diesen erfolgt mittels der Schaltflächen [<<] bzw. [>>]. Bei mehreren konfigurierten Monitoren sind diese über den in der Abbildung mit *Monitor[0]* beschrifteten Schalter einstellbar.

Es empfiehlt sich die Aktivierung von *Auto*, um die Wirkung einer Änderung unverzüglich an der Reaktion des Servers nachvollziehen zu können. Benutzen Sie die einfachen Pfeile, um das Bild in die entsprechende Richtung zu verschieben. Mit Hilfe der doppelten Pfeile kann das Bild vergrößert bzw. verkleinert werden. Versuchen Sie, das Bild mittig anzuordnen, sodass die größtmögliche Darstellung erzielt wird, ohne dass die Ränder verzerrt erscheinen. Füllt das Ergebnis den Bildschirm nicht komplett aus, so regulieren Sie »den Rest« per Einstellung am Monitor nach.

Eine jede Einstellung wird über *Add* gespeichert.

Auflösungen und Farbtiefen

Die Screen-Ansicht ist über den Eintrag *Screen* des Menüs erreichbar, das bei Klick der rechten Maustaste auf den obersten Schalter aufklappt. Dargestellt werden alle Monitore, die bislang konfiguriert wurden. Klicken Sie zum Bearbeiten der unterstützten Farbtiefen und Auflösungen mit der rechten Maustaste auf einen Monitor. Die Einstellungen betreffen somit genau diesen Bildschirm.

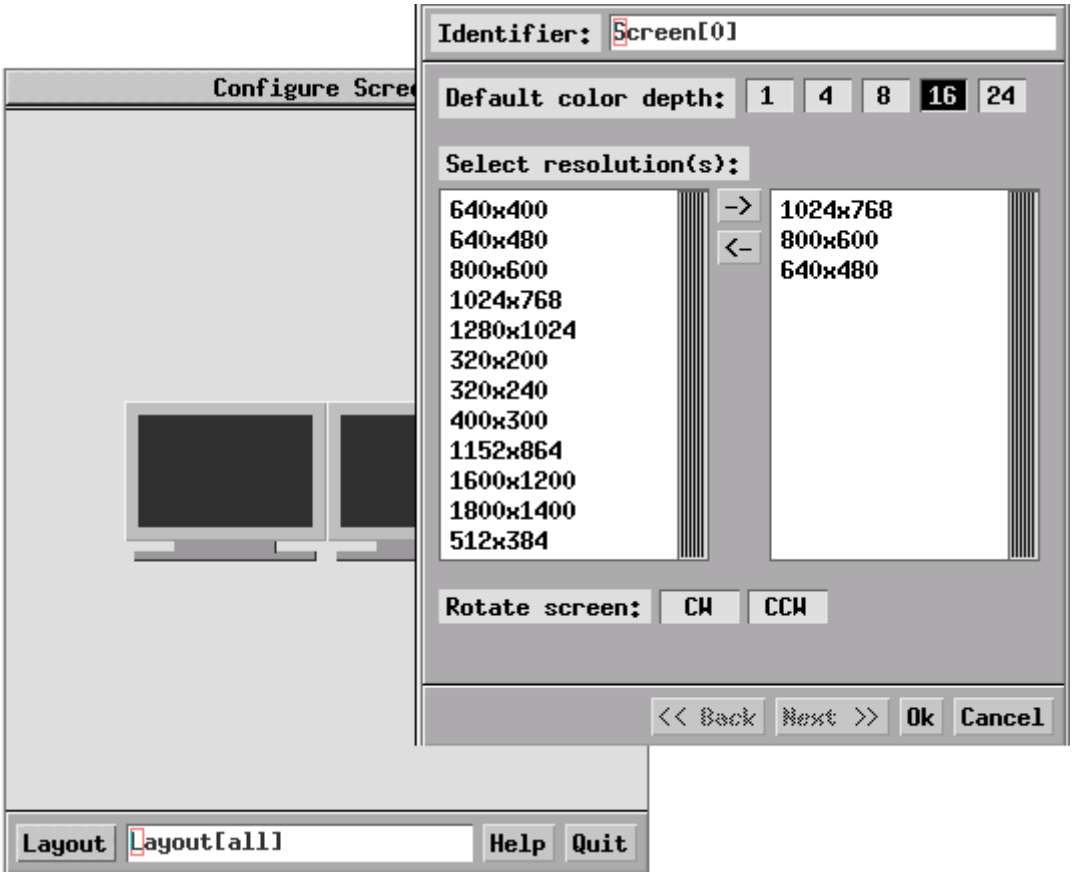


Abbildung 9: xf86cfg - Auflösungen und Farbtiefen

Alle Auflösungen der rechten Liste stehen Ihnen später zur Verfügung. Beachten Sie, dass die höchste hier stehende Auflösung später die Dimension des so genannten virtuellen Desktops bestimmt. D.h. wann immer Sie eine geringere Auflösung verwenden, wird nur ein Ausschnitt des virtuellen Desktops entsprechender Größe dargestellt, den Sie verschieben, indem die Maus über den Rand hinaus geführt wird. Dieses Verhalten empfinden viele Benutzer als lästig, sodass die höchste Auflösung auch ihrer »Arbeitsauflösung« entsprechen sollte.

Die erste Auflösung in der Liste bestimmt später die Auflösung, mit der der Server startet; eine absteigende Anordnung ist daher sinnvoll.

Default color depth ist die Farbtiefe, die der Server in der Voreinstellung verwendet. 1 und 4 Bit sind wohl eher für Monochrom-Bildschirme gedacht. Die 8 Bit-Einstellung ist zwar keine Augenweide, aber bei älteren Grafikkarten die einzige Chance, mit einem mageren Speicherausbau (<1MB) eine höhere Auflösung (>800x600) zu erzielen. Der Unterschied zwischen 16 und 24 Bit ist nur bei großflächigen Farbverläufen wahrzunehmen. Bei schnellen Grafikkarten ist durchaus die 24 Bit-Option zu empfehlen, aber prinzipiell leidet die Geschwindigkeit der Darstellung mit jeder Erhöhung der Farbanzahl (...und manche Spiele laufen nur mit einer konkreten Farbtiefe).

Mit *Rotate screen* lässt sich die Darstellung um 90° im Urhzeigersinn [CW] bzw. entgegen [CCW] des Uhrzeigersinns drehen.

Server-Zugriff

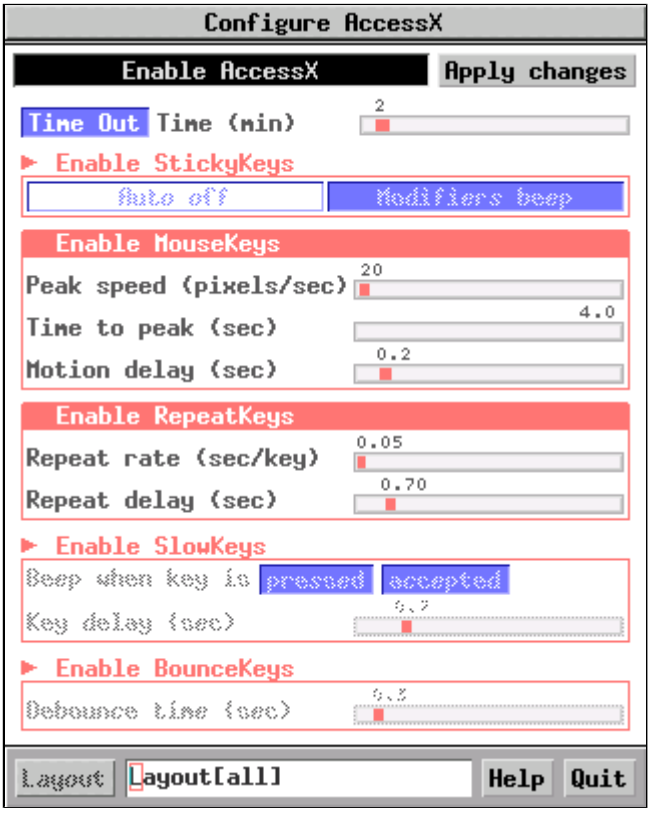


Abbildung 10: xf86cfg - Server-Zugriff

Die Einstellungen dieser Maske berühren in keinster Weise die Zugriffsrechte des Servers, auch wenn der Name dieses vermuten lässt. Hierüber lässt sich das Verhalten der Tastatur anpassen. Die getroffenen Einstellungen landen nach Abschluss in der Datei **/ etc/ X11/ xkb/ X0-config.keyboard**.

Um AccessX zu aktivieren, muss **Enable AccessX** gesetzt sein. **Time Out** bestimmt die Zeit, nach der, ohne dass eine Eingabe am Rechner erfolgte, die AccessX-Eigenschaften automatisch abgeschaltet werden. Dies kann bei Maschinen, an denen sich mehrere Benutzer abwechseln, sinnvoll sein, da nicht jeder dieselben Einstellungen vorteilhaft empfindet.

Unter **Enable sticky keys** erfolgt die Konfiguration, ob Mehr-Tasten-Kombinationen (bspw. [Ctrl][S]) in sequentieller Form möglich sind, d.h. anstatt solche Tasten gleichzeitig betätigen zu müssen, wird derselbe Effekt durch nacheinander folgendes Drücken derselben erreicht. Ob diese Eigenschaft automatisch aktiviert werden kann, ist unter **Auto [on| off]** konfigurierbar. Wenn ja, lässt sich das Einschalten noch akkustisch durch einen Piepton untermalen (**Modifiers beep**). Mit jedem wiederholten Drücken einer der Modifier-Tasten (bspw. [Ctrl], [Alt]...) wird die Eigenschaft im Wechsel aktiviert bzw. deaktiviert.

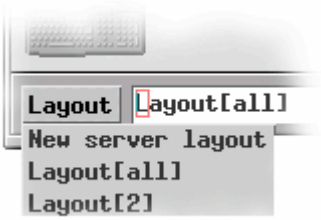
Die Option **MouseKeys** konfiguriert den Nummernblock der Tastatur so, dass die Mausbewegungen mittels der Cursortasten simuliert werden können, [5] wirkt dann wie ein Einfach-Klick, [0] ahmt das Halten einer Taste nach und [/], [*], [-] symbolisieren die drei Maustasten. Um demzufolge die rechte Maustaste zu drücken, müssen gleichzeitig [5] und [/] betätigt werden. Die Parameter **Peek speed**, **Time to peek** und **Motion delay** konfigurieren die Geschwindigkeit, mit der der Mauszeiger über den Bildschirm eilt.

RepeatKeys bestimmt die Wiederholrate, die eine dauerhaft betätigte Taste auslöst. **Repeat rate** bestimmt, wie viele Zeichen in einer Sekunde maximal erzeugt werden und **Repeat delay** ist die Zeitspanne, die mindestens vergehen muss, bis eine dauerhaft betätigte Taste ein erneutes Signal erzeugt.

Ist **SlowKeys** aktiviert, erzeugt ein Tastenanschlag nur ein Zeichen, wenn die Taste auch innerhalb der angegebenen Zeitspanne wieder losgelassen wird. Ob ein Zeichen akzeptiert wurde oder nicht, lässt sich durch die zusätzlichen Optionen akkustisch verdeutlichen.

BounceKeys konfiguriert letztlich eine Zeitspanne, die mindestens zwischen zwei Anschlägen auf einunddieselbe Taste verstreichen muss, bis dieses als zwei Anschläge gewertet wird.

Xinerama



Zum Abschluss möchte ich skizzieren, wie der X-Server konfiguriert werden kann, um mehrere Grafikkarten und Monitore zu verwenden, wobei wir im Beispiel von einer zweiten Grafikkarte und einem zweiten Monitor ausgehen.

Wählen Sie im unter *Layout* befindlichen Menü den Eintrag für *New server layout*. Sie können diesem Layout einen beliebigen (aber eindeutigen) Namen geben.

Fügen Sie jeweils eine weitere Grafikkarte und einen weiteren Monitor hinzu, indem Sie das jeweilige Symbol aus der oberen Leiste anwählen. Konfigurieren Sie die Geräte wie oben beschrieben. Vergessen Sie nicht das Anlegen einer eigenen Screen-Section und die Konfiguration der Modelines für den neuen Monitor!

Klicken Sie nun der Reihe nach auf die einzelnen Geräte (außer Monitore) und wählen Sie **enable** aus dem per rechter Maustaste erreichbaren Menü. Die Monitore verbinden Sie im Monitor-Konfigurations-Dialog mit einer Grafikkarte.

Die voreingestellte Anordnung der beiden Monitors ist nebeneinander. Möchten Sie den zweiten Monitor anders positionieren, so müssen Sie in die **Screen-Ansicht** wechseln. Verschieben Sie die Symbole der Monitore entsprechend der gewünschten Anordnung.

Beenden Sie nun **xf86cfg** und Bestätigen das Speichern der Dateien. Abschließend sollten Sie den Device-Abschnitt der Datei XF86Config überprüfen. In den mir vorliegenden Programmversionen schlich sich ein Fehler in der Screen-Zeile ein:

```
Section "Device"
  BoardName   "MGAG400"
  BusID       "1:0:0"
  Driver      "mga"
  Identifier  "Device[1]"
  VendorName  "Matrox"
  Videoram    32768
  Screen      "1"
EndSection
```

Die angegebene Screen-Nummer darf **nicht** in Anführungsstriche geschrieben werden, da die Server (in den von mir getesteten Versionen 4.0-4.0.2) mit einem Syntaxfehler abbrechen. Ändern Sie derartige Zeilen:

```
Section "Device"
...
  Screen    1
EndSection
```

Haben Sie weitere Grafikkarten konfiguriert, sollten Sie deren Abschnitte ebenfalls anpassen.

Beim Start von X wählt der Server automatisch das Layout des ersten Abschnitts aus. Um ein alternatives Layout zu nutzen, muss dieses in der Kommandozeile angegeben werden:

```
user@sonne> startx -- -layout "Layout[2]"
```

War die Konfiguration erfolgreich, sollte jetzt der zweite Monitor einen grauen Hintergrund zeigen. Der Desktop selbst ist noch immer vollständig auf dem ersten Monitor vorhanden, Sie können allerdings schon mit der Maus zwischen den Bildschirmen wechseln.

Um den Desktop über mehrere Bildschirme zu verteilen, muss erst noch die **Xinerama**-Erweiterung des Servers aktiviert werden. Per Kommandozeile genügt ein:

```
user@sonne> startx +xinerama
```

Möchten Sie Xinerama permanent mit einem Layout verwenden, so ergänzen Sie die Zeile

```
Section "ServerLayout"
    Identifier   "Layout[2]"
...
    Option   "Xinerama"  "on"
EndSection
```

in der entsprechende »Section ServerLayout« der Datei XF86Config.

anXious - Das Werkzeug von Debian

Übersicht

Mit **anXious** hat Debian ein eigenes Konfigurationswerkzeug herausgebracht, um die Möglichkeiten der Hardwareerkennung in den Konfigurationsprozess einfließen zu lassen. Eine weitere Verbindung besteht zur Paketverwaltung von Debian, so dass fehlende Pakete (XServer, Windowmanager) für die Pakettools vorgemerkt werden. Das Programm ist letztlich ein grafisch verpacktes xf86config mit diversen Erweiterungen.

Meist wurde man während der Installation bereits mit anXious konfrontiert, in dem Fall entfällt die anfängliche Abfrage, ob es sich um eine »Umkonfiguration« handelt:

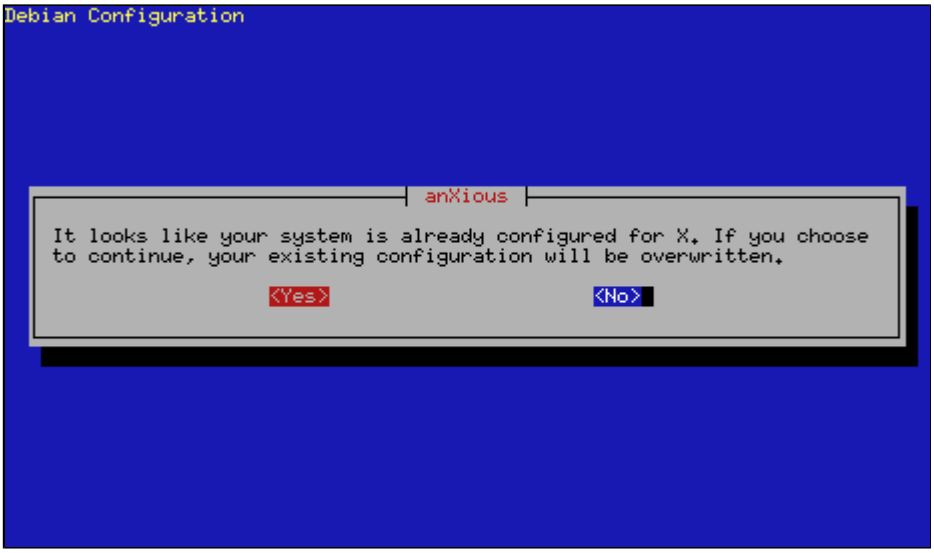


Abbildung 1: Warnung vor Überschreiben einer bestehenden Konfiguration

Automatische Erkennung der Grafikkarte

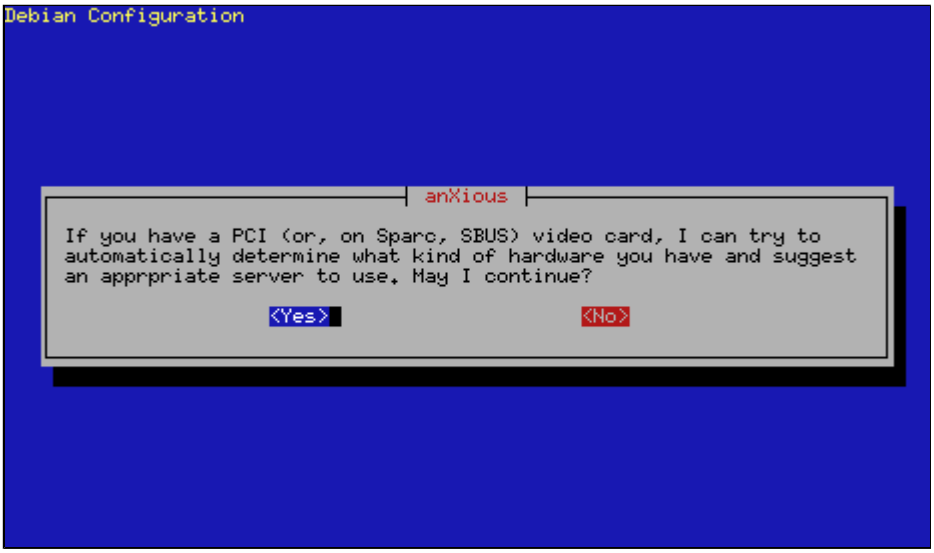


Abbildung 2: Automatische Erkennung der Grafikkarte

Handelt es sich um eine PCI-Grafikkarte, sollte man die automatische Erkennung starten. Scheitert diese, oder verneint man die Frage, beendet sich anXious und man muss auf die allgemeinen Werkzeuge wie xf86config oder XF86Setup zurückgreifen.

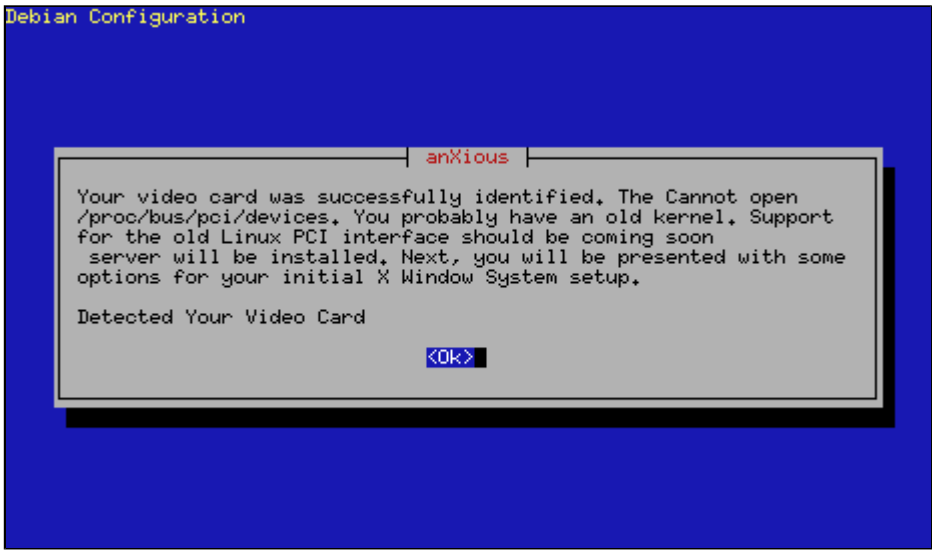


Abbildung 3: Geglückte Erkennung

Software

X Window besteht nicht allein aus dem Server. Zur Arbeit sind bestimmte Zeichensätze (Schriftart/-größe), Terminal-Emulationen (jede Terminalemulation reagiert anders auf bestimmte Tastatureingaben), Windowmanager (Wie sollen die Anwendungen generell dargestellt werden?), ... notwendig. Wählen Sie aus den folgenden Menüs jeweils mindestens einen Eintrag aus:

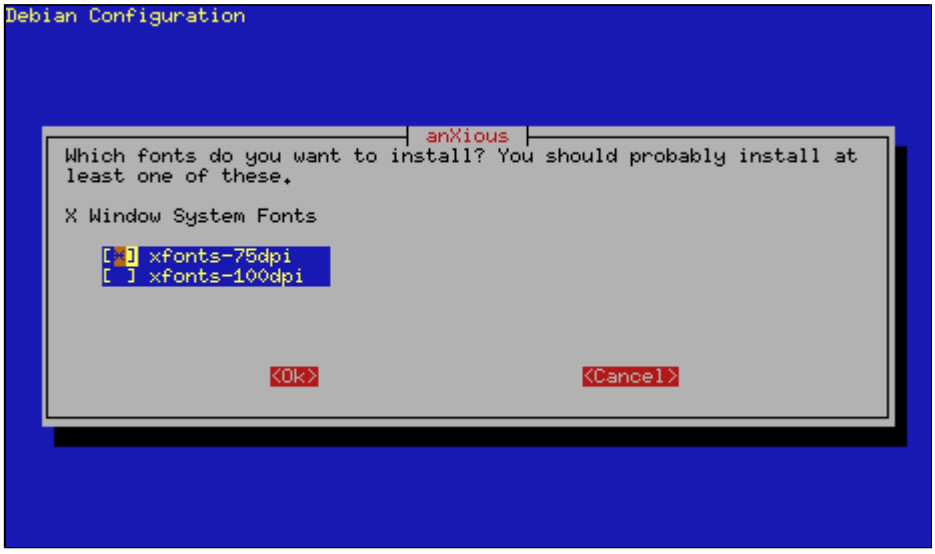


Abbildung 4: Bildschirmfonts auswählen

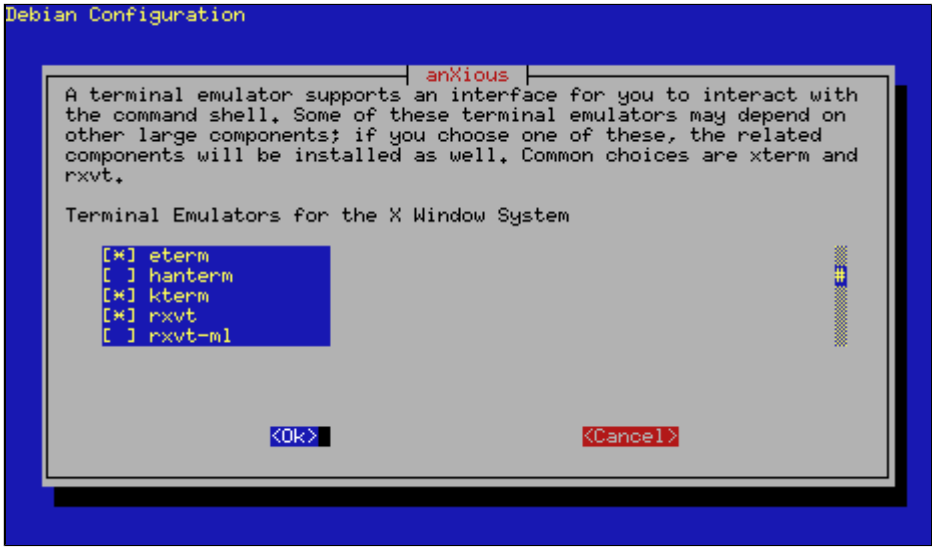


Abbildung 5: Terminal-Emulationen auswählen

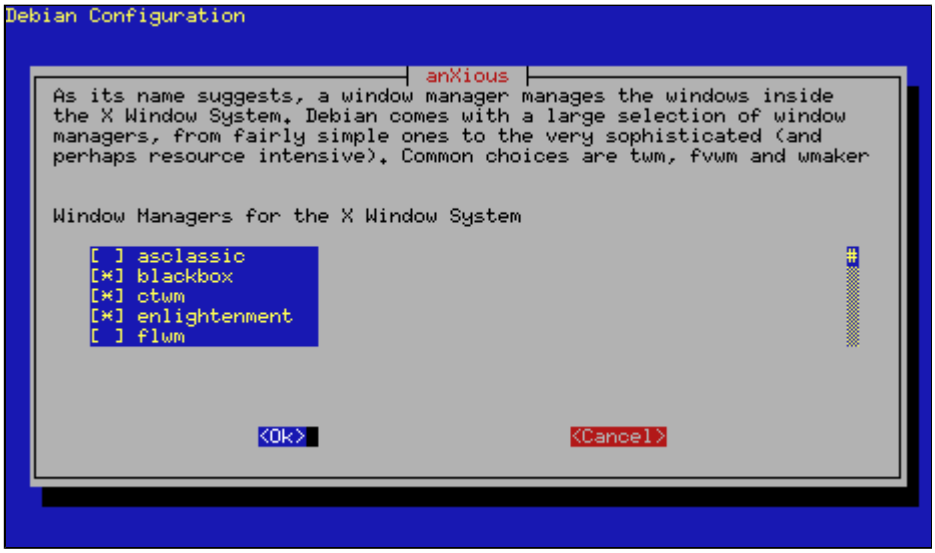


Abbildung 6: Windowmanager auswählen

Der Xdm ermöglicht ein grafisches Login. Wenn Sie diese Frage bejahen, wird der »xdm« in jedem Multiuser-Runlevel gestartet.

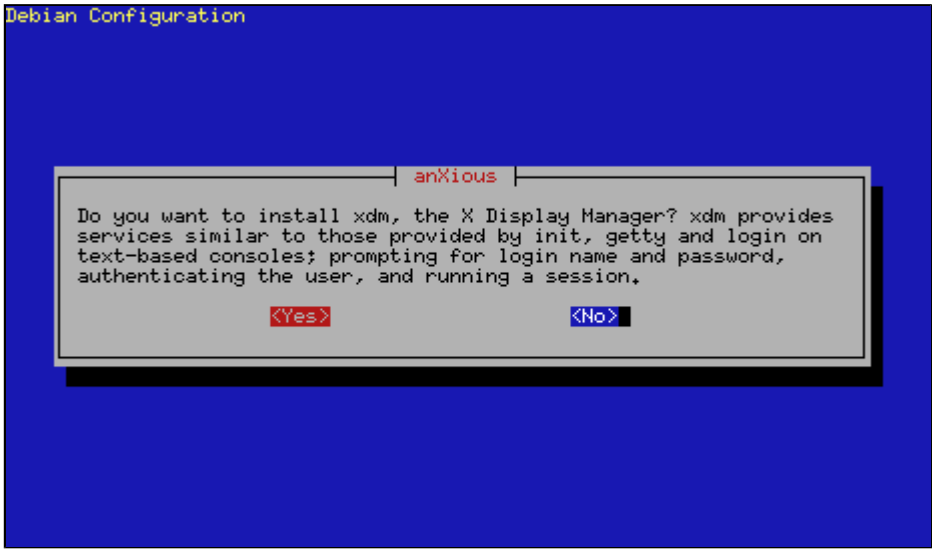


Abbildung 7: Grafisches Login?

Maus und Tastatur

Die nächste Abfragemaske betrifft das Mausprotokoll (nicht den Hersteller!). Als Faustregel können Sie davon ausgehen, dass es sich bei einer am seriellen Port installierten Maus um das »Microsoft«-Protokoll handelt und in den meisten anderen Fällen um PS2.

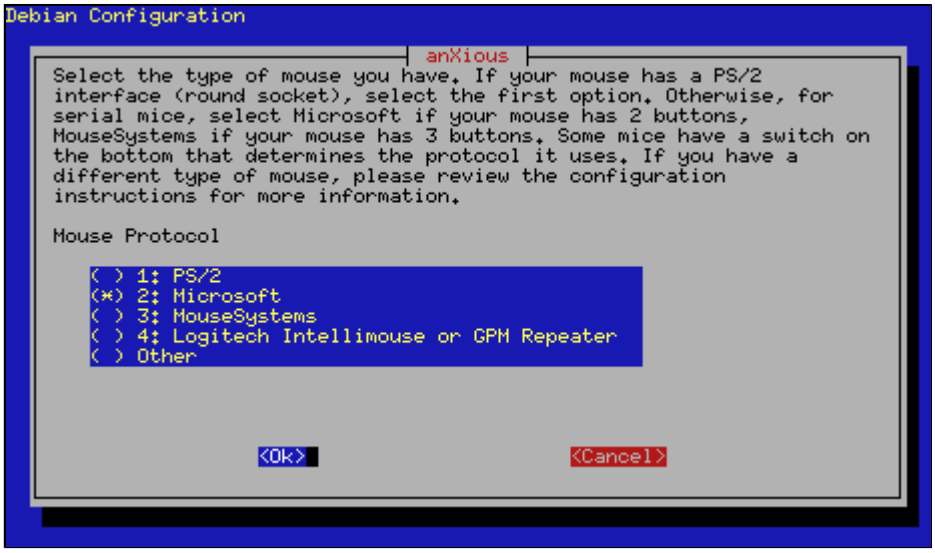


Abbildung 8: Mausprotokoll auswählen

Verfügt Ihre Maus über drei Tasten, so verneinen Sie die folgende Frage nach [Emulate3Buttons]. Besitzt sie nur zwei Tasten, sollte [Yes] Ihre Wahl sein, um durch gleichzeitiges Drücken der Tasten die dritte zu emulieren.

Das zugehörige Device ist bei einer Maus am seriellen Port /dev/ttyS0 (in DOS »COM1«), /dev/ttyS1 (»COM2«)... und /dev/psaux bei einer PS2-Maus. Das angebotene /dev/mouse ist ein Link auf das richtige Device. Sie können dieses verwenden, wenn der Link korrekt gesetzt ist.

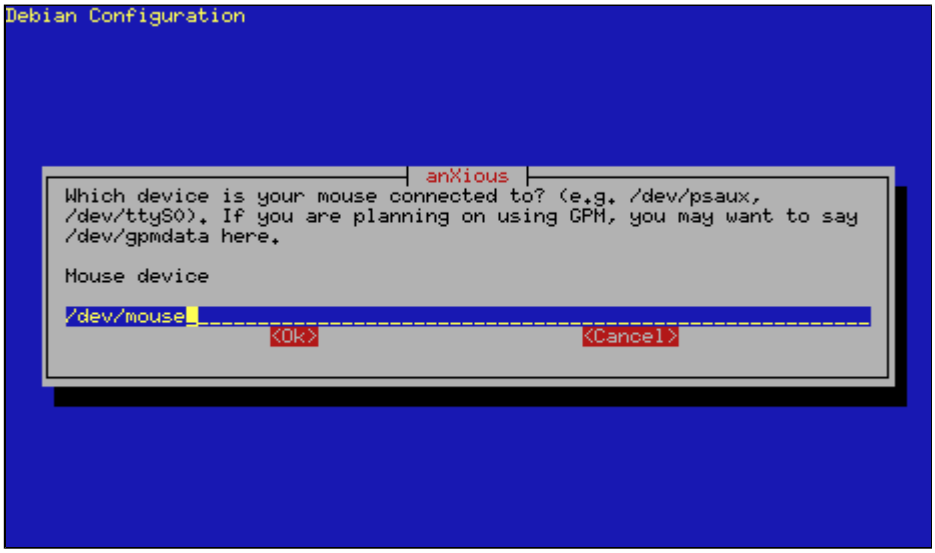


Abbildung 9: An welchem Device hängt die Maus?

In der Liste der verfügbaren Tastaturmodelle (Abbildung 10) steht das von Ihnen verwendete Modell womöglich nicht zur Auswahl. Wählen Sie dann aauf jeden Fall [German], um die korrekte Tastaturbelegung für deutsche Sonderzeichen zu erhalten.

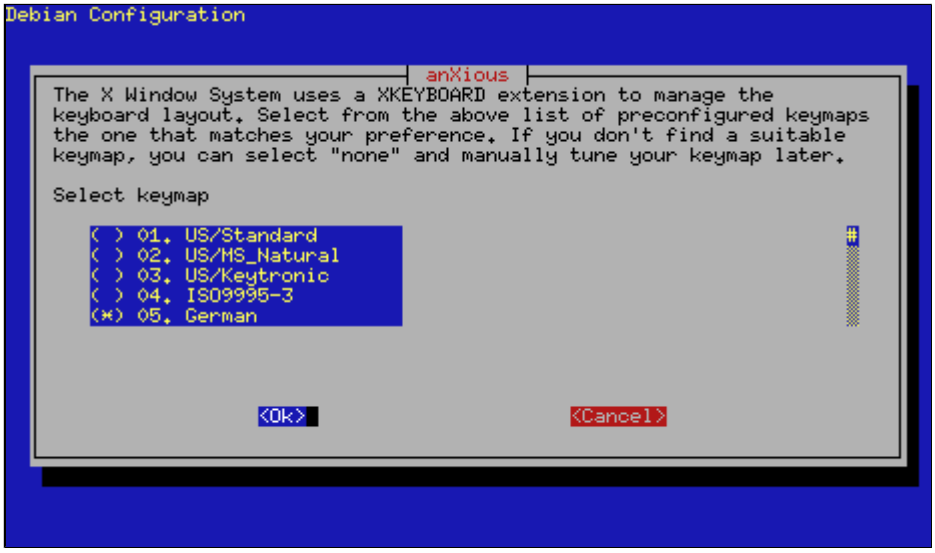


Abbildung 10: Tastaturkonfiguration

Monitor und Grafikkarte

Die folgende Maske verlangt die Angabe der Vertikalfrequenzen ihres Monitors. Diese Werte, wie auch die Horiztontalfrequenz, erfahren Sie aus dem Handbuch Ihres Bildschirms. Weichen die Werte Ihres Monitors von den Vorgaben ab, so wählen Sie [other] und geben in der nachfolgenden Maske die Werte an (einen Bereich legen Sie bspw. mit "30-107" fest).

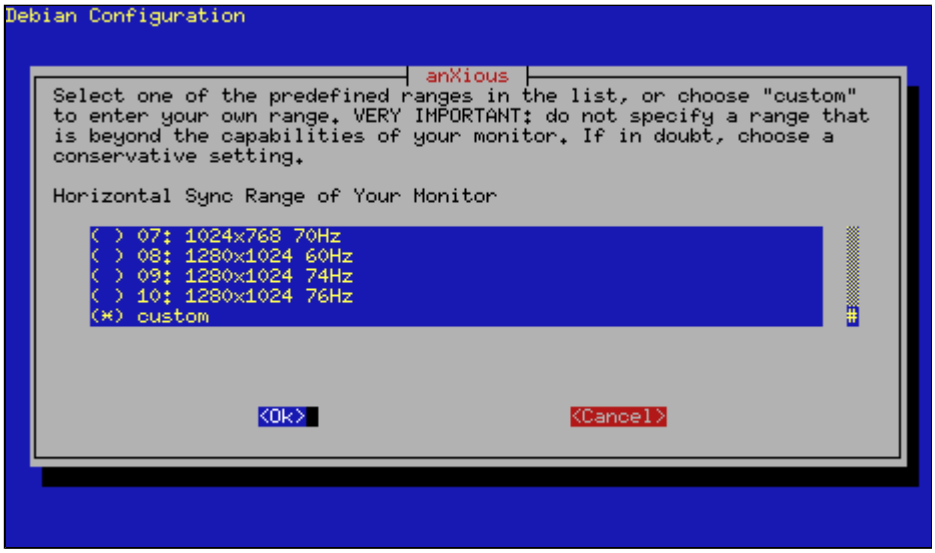


Abbildung 11: Monitor Frequenzen

Dasselbe Vorgehen wiederholt sich nachfolgend mit den Horizontalfrequenzen. Abschließend werden Sie aufgefordert, einen Monitornamen festzulegen. Nur wenn Sie mehrere Monitore angeschlossen haben, ist hier ein eindeutiger Bezeichner erforderlich.

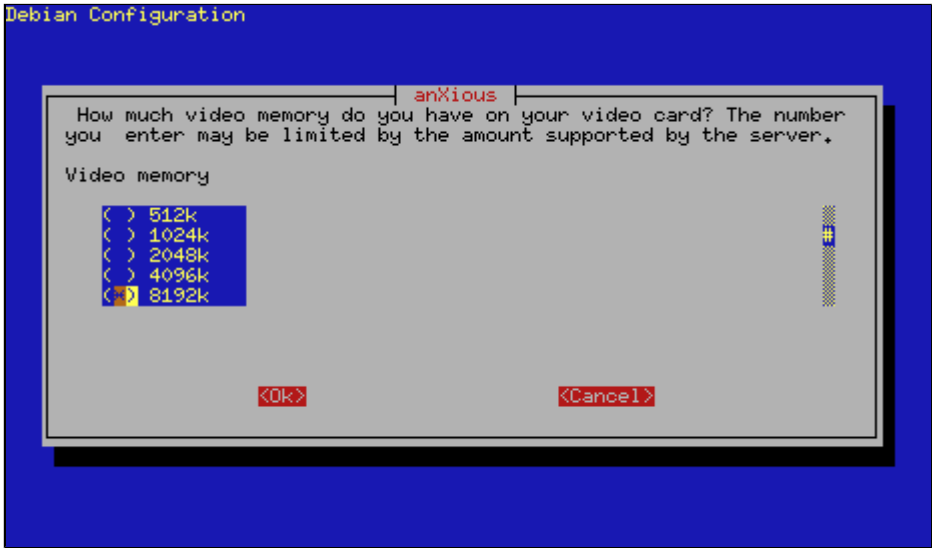


Abbildung 12: Grafikspeicher

Ihre Grafikkarte betreffen die folgenden Masken. Zunächst müssen sie den VideoRAM angeben, dann einen Bezeichner für die Grafikkarte.

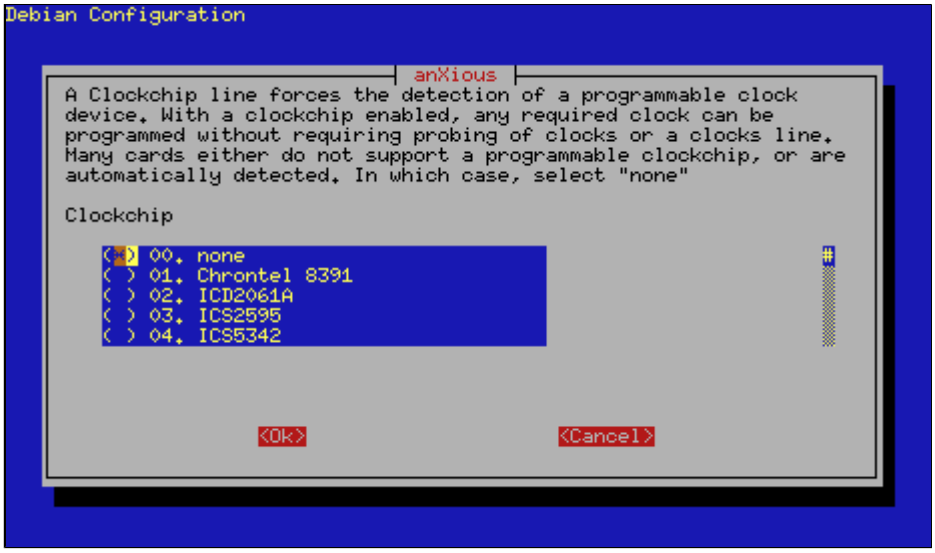


Abbildung 13: Clockchip

Einen Clockchip dürfen Sie nur angeben, wenn Sie die exakte Bezeichnung kennen und diese in der Liste erscheint. In allen anderen Fällen ist [none] ihre Wahl.

Die Automatische Erkennung der Clocklines, zu der die nächste Eingabemaske ermuntert, ist bei aktuellen Grafikkarten nicht sinnvoll, da sie zumeist einen programmierbaren Clockchip verwenden, der keine festen Vorgaben erfordert.

Farbtiefe und Auflösung

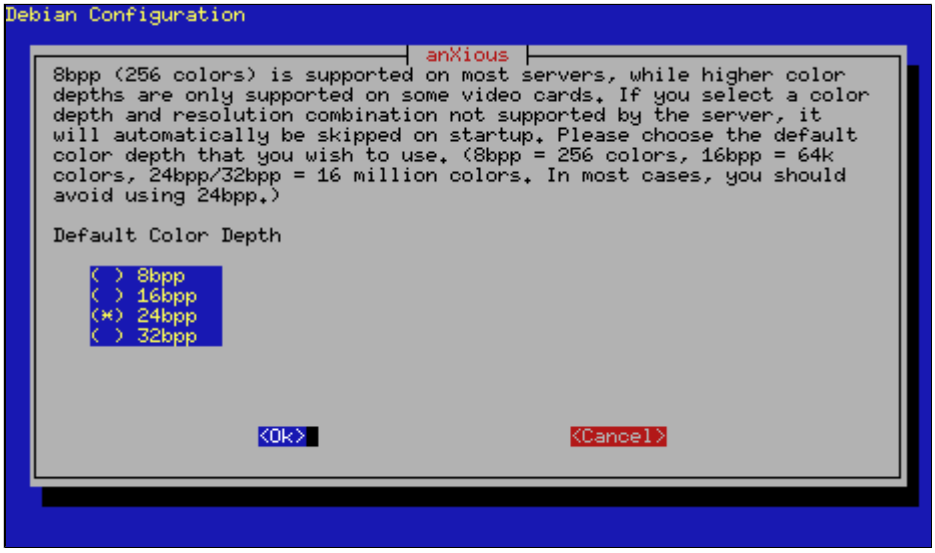


Abbildung 14: Default-Farbtiefe

Sie können im späteren Verlauf den X-Server mit jeder Farbtiefe starten. Ohne dieses Startargument startet der Server jedoch mit der voreingestellten Farbtiefe. Je höher Sie Farbtiefe und Auflösung wählen, desto mehr VideoRAM benötigt Ihre Grafikkarte. So reichen 8 MByte Videospeicher bei 1280x1024 Bildpunkten und 32 Bit Farbtiefe nicht mehr aus!

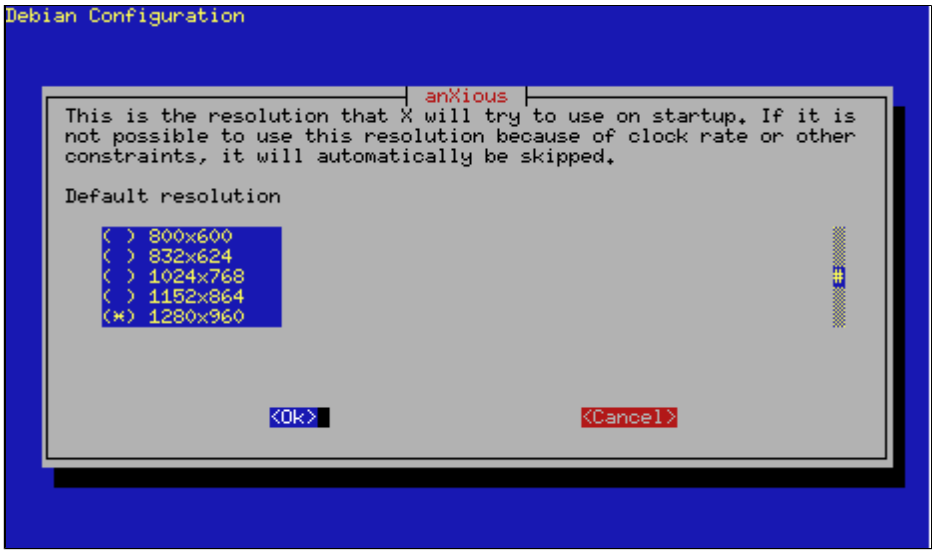


Abbildung 15: Default-Auflösung

Wie es eine voreingestellte Farbtiefe gibt, startet Ihr X Server mit einer voreingestellten Auflösung. Sinnvoll ist hier die Angabe Ihrer »Arbeitsauflösung«, also nicht das Maximum sondern die Werte, bei denen Sie nicht mit der Nasenspitze die Mattscheibe berühren, um die klitzekleine Schrift entziffern zu können.

Anschließend können Sie alle Auflösungen markieren, die Ihr X Server später unterstützen soll. Allerdings sollten Sie keine höhere wählen als die aus der vorherigen Maske, es sei denn, sie mögen virtuelle Bildschirme (es wird nur ein Ausschnitt des Desktops projeziert und Sie erreichen die anderen Bereiche, indem Sie die Maus »über den Rand« hinaus bewegen).

Abschluss

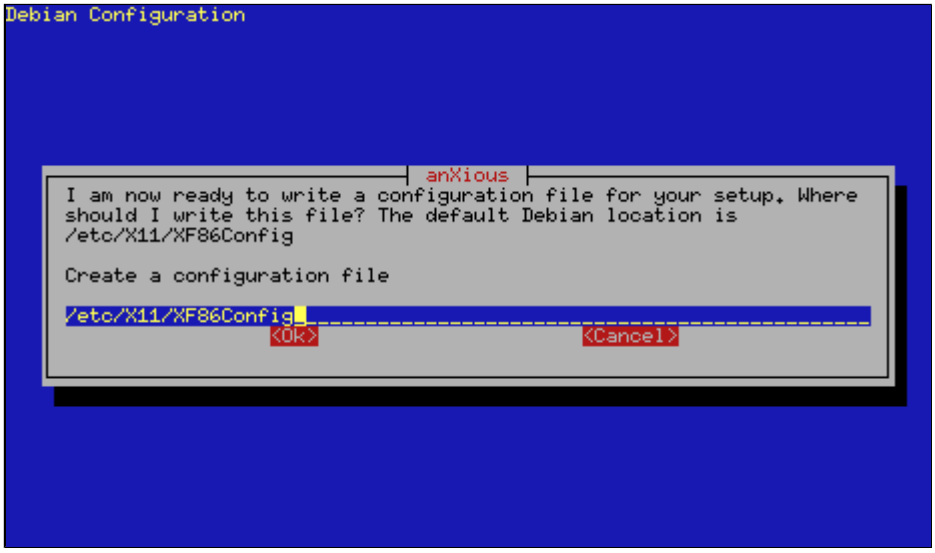


Abbildung 16: Zum Abschluss

Die letzte Frage wird bestätigt, wenn Sie mit ihrer Konfiguration einverstanden sind. Dann wird ggf. die alte XF86Config überschrieben.

Xconfigurator

Optionen beim Start

Beim »Xconfigurator» handelt es sich um ein von RedHat entwickeltes Werkzeug zur Konfiguration des X-Servers, das sämtlichen RedHat-basierenden Distributionen beiliegen sollte. Das Programm sollte, sofern die notwendigen Bibliotheken und Pfade vorhanden sind, auch auf anderen Linux-Systemen verwendet werden können.

Wird das Kommando ohne Argumente aufgerufen, arbeitet es im **interaktiven Modus**. Nahezu alle Parameter sind einstellbar. Ausnahme ist die Maus. Hierfür werden die Werte der Datei »/etc/sysconfig/mouse« verwendet. Existiert die Datei nicht, wird eine Microsoft-Standard-Maus mit zwei Tasten konfiguriert.

Mit der Option **--expert** gestartet, lässt Xconfigurator zu, dass der Benutzer sämtliche Werte überschreiben kann. Vorsicht: Falsche Werte können Ihre Hardware zerstören!

Wird das Programm mit der Option **--kickstart** gestartet, versucht es selbstständig, alle notwendigen Parameter der Hardware zu ermitteln. Alle nicht erkannten Werte werden durch Standardwerte ersetzt (z.B. eine Auflösung von 640x480 Bildpunkten bei 16 Farben). Am Ende schreibt Xconfigurator die Datei /etc/XF86Config, ohne dass der Benutzer Einfluss auf die Werte nehmen kann.

Mit weiteren Optionen lassen sich die konkrete Grafikkarte »--card«, der zu verwendende X-Server »--server«, der Monitor »--monitor« u.a.m. einstellen.

Abbildung 1: Startbildschirm im interaktiven Modus

Im weiteren Verlauf betrachten wir das Vorgehen einer normalen interaktiven Konfiguration.

Automatische Erkennung der Grafikkarte

Abbildung 2: Erfolgreiche Erkennung der Grafikkarte

Mit etwas Glück hat das Programm die installierte Grafikkarte von allein erkannt. Falls nicht, oder falls eine falsche Karte »identifiziert« wurde, sollte Xconfigurator abgebrochen und mit der Option »--card <Kartenname>« neu gestartet werden.

Bildschirm

Abbildung 3: Auswahl des Monitors

Erscheint der Bildschirm in der Liste der Typen, sollte dieser selektiert werden. Finden Sie Ihn nicht, dann wählen Sie den Eintrag **Benutzerdefiniert** und wählen die Werte, die den Frequenzen zu Ihrem Monitor am ehesten entsprechen. Diese Werte finden Sie entweder im Handbuch zum Bildschirm oder manchmal auch auf dessen Rückseite.

Grafikmodus

Das Programm versucht nun den X-Server in verschiedenen Modi zu starten. Zwischen den Umschaltungen wird der Bildschirm flackern, abschließend sollte der Vorschlag einer »optimalen« Einstellung erscheinen.

Abbildung 4: Konfiguration des Grafikmodus

Wählen Sie nun **Manuell auswählen**, so müssen nachfolgend die Auflösungen der einzelnen Farbtiefen eingegeben werden. Werden mehrere Farbtiefen selektiert, so wird der Server später mit der niedrigsten starten, es sei denn, ihm wird über Optionen eine andere Farbtiefe mitgeteilt. Werden zu einer Farbtiefe mehrere Auflösungen eingestellt, kann unter X zwischen diesen gewechselt werden (»[Ctrl]-[Alt]-[+]« bzw. »[Ctrl]-[Alt]-[-]«).

Abbildung 5: Modiauswahl

Fertig! ...oder doch noch nicht?

Xconfigurator bietet nun an, den X-Server automatisch zu starten. Gelingt dies, kann die Konfiguration abgeschlossen und zusätzlich das Programm angewiesen werden, Linux für grafisches Login einzurichten.

Haben Sie zufällig in der letzten oder vorletzten Maske **Zurück** selektiert, so gelangen Sie in Masken, die eine detaillierte Konfiguration der Grafikkarte erlauben.

Der erste Dialog fordert zur Angabe des Grafikspeichers auf, anschließend folgt der Clockchip. Geben Sie diesen nur ein, falls der auf ihrer Karte befindliche Chip in der Liste erscheint.

Die nachfolgende Frage nach »X --probeonly« versucht die **Modelines** automatisch zu erkennen. Modelines bestimmen das Timing des Elektronenstrahls bei der Darstellung des Bildes. Durch unterschiedliche Werte lassen sich das Bild ausrichten und dessen Größe ändern. Weitere Information stehen im Abschnitt zur Beschreibung der /etc/XF86Config.

Jetzt befinden Sie sich wieder in der Auswahlmaske der Farbtiefen und Auflösungen und verfahren wie weiter oben beschrieben.

X-Konfiguration mit Sax

Übersicht

SuSE engagiert sich stark bei der Entwicklung von XFree86. Ihren eigenen Distributionen liegen komfortable Werkzeuge bei, die das Einrichten des X-Servers stark vereinfachen. Zumindest solange die Hardware nicht allzu exotische Einstellungen erwartet. Die hier vorgestellten Konfigurationshilfen ermöglichen nicht die Konfiguration sämtlicher Parameter (die allerdings nur in seltenen Fällen benötigt werden).

Der »SuSE Advanced XF86-Configurator« in der Version 1 (Sax1) konfiguriert die Server der XFree86 Version 3.3.6. Ggf. müssen Sie auf diese alte Version zurückgreifen, falls Sie eine relativ alte Grafikkarte in Ihrem System betreiben, auf deren Unterstützung in den neuen Servern bewusst verzichtet wurde (solche Karten sind auch nicht in der Lage, die Vorteile der neuen Architektur zu nutzen).

Um die Möglichkeiten des aktuellen XFree86 Version 4.x auszuschöpfen, wurde Sax2 geschaffen, das wiederum nicht zur Administration der älteren Version eingesetzt werden kann!

Allgemeines zur Verwendung der Werkzeuge

Beide Werkzeuge können Sie genauso gut einsetzen, um eine bestehende X-Konfiguration zu ändern. Auch wenn dies aus einer laufenden grafischen Oberfläche heraus denkbar ist, empfiehlt sich das vorherige Beenden von X. Wechseln Sie hierzu auf eine der Textkonsolen ([Ctrl]-[Alt]-[F*x*]), melden sich als Root an und überführen das System in eines der »Konsolen-Runlevel« (in dem kein Login-Manager gestartet wird):

```
# SuSE <= 7.0
root@sonne> init 2
# SuSE > 7.0
root@sonne> init 3
```

Während der nachfolgenden Konfiguration mittels **sax** bzw. **sax2** können Sie den X-Server zu jedem Zeitpunkt mittels [Strg][Alt][Backspace] beenden. Notwendig kann dies werden, wenn der Server sich »festgefahren« hat oder der Monitor unerwartet auf schwarz schaltet. Nach einem solchen manuellen Abbruch kann eventuell der Statusbericht aus der Datei **/ var/ log/ ServerLog** bei der Ursachenforschung nach dem Fehler behilflich sein.

Sax1

Ab der Version 6.0 liegt den Distributionen von SuSE das Konfigurationstool SaX bei, das in den meisten Fällen schnell zu einer befriedigenden X-Installation führen sollte. Aufrufen kann es der Systemverwalter unter so ziemlich allen "sax"-Buchstabenkombinationen sax, Sax,..., saX. Das ausführbare Programm heißt SaX, alles andere sind Links darauf.

```
root@sonne> sax
```

SaX versucht zunächst, den Typ der Grafikkarte zu bestimmen, um ggf. mit dem richtigen X-Server hochzufahren. Gelingt dies nicht, startet SaX den 16 Farben Server (falls installiert). Schlägt auch dieses fehl, so bleibt Root noch der Versuch, dem Tool den entsprechenden Server als Argument mitzuteilen oder er muss auf die in den vorangegangenen Abschnitten vorgestellten Hilfsmittel zurückgreifen.

```
# Start von SaX mit dem XFCom_Matrox Server
root@sonne> sax -s XFCom_Matrox
```

Glückte der Start von SaX, so lädt das Tool nun eine Reihe von Daten zu bekannter Hardware. Bei erstmaliger Konfiguration des X-Servers erzwingt **Sax**, die einzelnen Geräte in der durch die Menüleiste vorgegebenen Reihenfolge einzurichten. Existierte bereits eine X-Konfiguration, so erscheint eine Nachfrage, ob die bisherigen Daten eingelesen werden sollen.

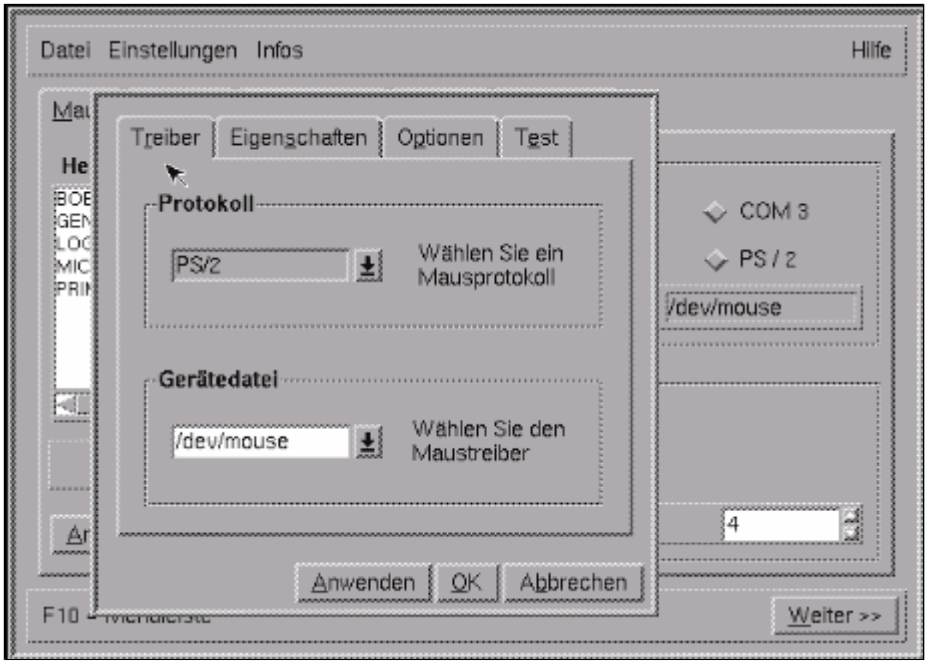


Abbildung 1: Sax(1): Mauskonfiguration

Funktioniert die Maus bereits zufriedenstellend, kann der erste Punkt sofort über [Weiter] verlassen werden. Ist dem nicht so, hilft die Tabulatortaste bei der Selektion der einzelnen Einträge. Die Bezeichnung »Hersteller« ist etwas unglücklich gewählt, da hier das Protokoll und nicht der konkrete Produzent der Maus enscheidend ist (Logitech-Mäuse arbeiten oft nach dem Microsoft-Protokoll!). Mit [Anwenden] wird die aktuelle Konfiguration sofort wirksam.

Was sich hier nicht konfigurieren lässt, ist die korrekte Funktion des Rades von Wheel-Mäusen (die aktuellen Modelle fehlen leider auch in der Datenbank). Für die gängigen Modelle sollten Sie als Protokoll **IMPS/2** einstellen, wobei nach [Anwenden] mit Ausnahme des Rades die Maus dennoch funktionieren sollte. Um das Rad zum Scrollen zu bringen, ist nach Abschluss der Konfiguration Handarbeit angesagt. Bearbeiten Sie hierzu die »Section "Pointer"« der Datei /etc/XF86Config:

```
root@sonne> vi /etc/XF86Config
...
Section "Pointer"
   Protocol        "PS/2"
   Device          "/dev/psaux"
...
   ZAxisMapping    "4 5"
EndSection
...
```

Ergänzen Sie die Zeile **ZAxisMapping**, wobei Sie anstatt "4 5" die Ziffern wählen, die um 1 bzw 2 höher sind als die Anzahl der Maustasten (oft ist das Rad selbst als Taste realisiert).

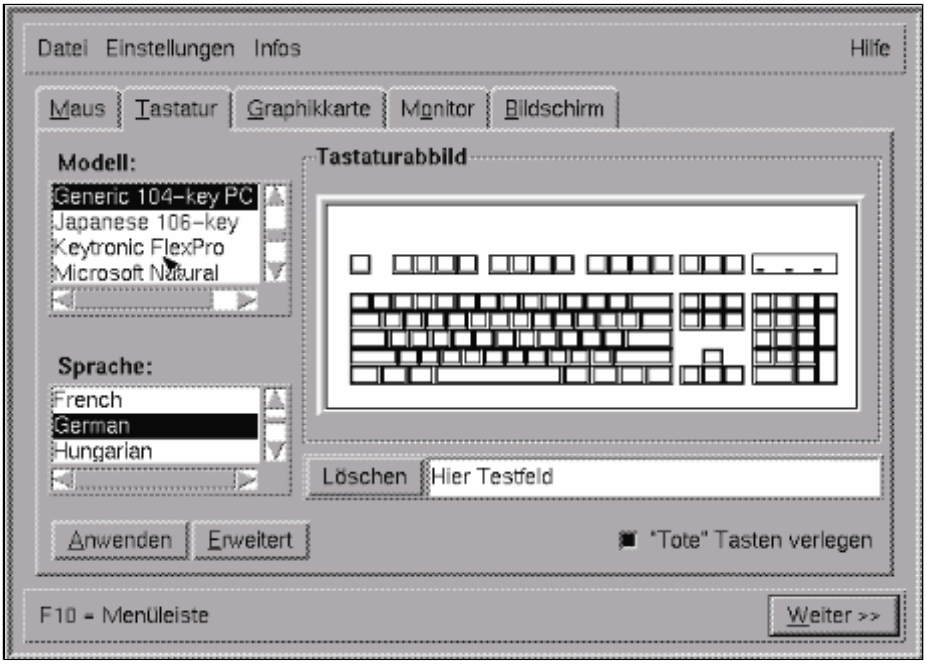


Abbildung 2: Sax(1): Tastaturkonfiguration

Der Typ der Tastatur und die entsprechende Belegung sind in der nächsten Maske zu spezifizieren. Für deutsche Tastaturen empfiehlt sich die Option "Tote Tasten verbergen".

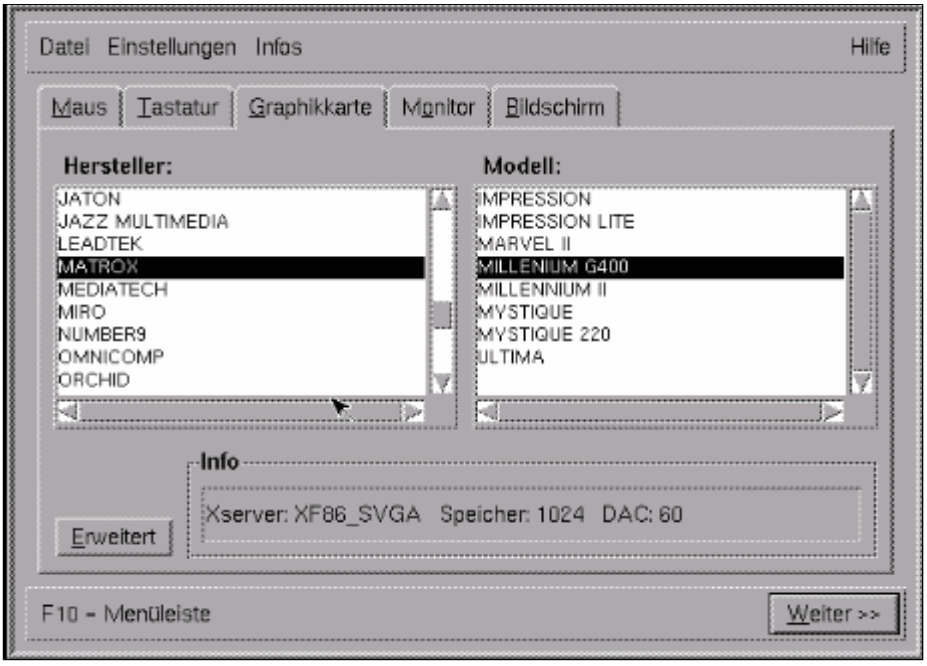


Abbildung 3: Sax(1): Konfiguration der Grafikkarte

In der Regel erkennt SaX die Grafikkarte selbsttätig. Dann werden Servername, RAMDAC und VideoRAM in der Statuszeile angezeigt. Ist der richtige Server nicht installiert, weist Sax darauf hin. Leider klappt es mit der Erkennung hin und wieder doch nicht so richtig, ein sehr geringer RAMDAC-Wert (aktuelle Karten arbeiten mit jenseits der 100) könnte u.a. darauf hindeuten... Sind Hersteller und exakte Bezeichnung der Grafikkarte bekannt, sollten in einem solchen Fall diese Werte (sofern vorhanden) in den beiden Boxen eingestellt werden. Der Schalter [Erweitert] ermöglicht die manuelle Einstellung von RAMDAC und RAM...

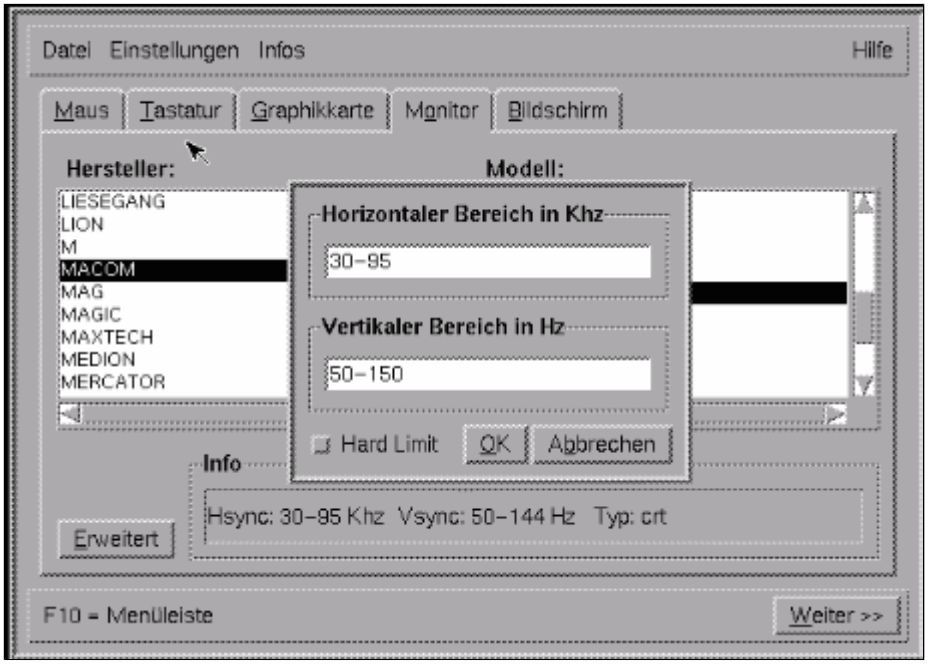


Abbildung 4: Sax(1): Angaben zum Monitor

Erscheint das Monitormodell in der Auswahlliste, so werden die möglichen Frequenzen aus der Datenbank entnommen. Alternativ lassen sich auch die im Handbuch zum Monitor aufgeführten Frequenzen eingeben. Stellen Sie beim Servertest im weiteren Verlauf eine stark verzerrte Darstellung fest, dann versuchen Sie, die oberen Grenzwerte der Frequenzen zu verringern.

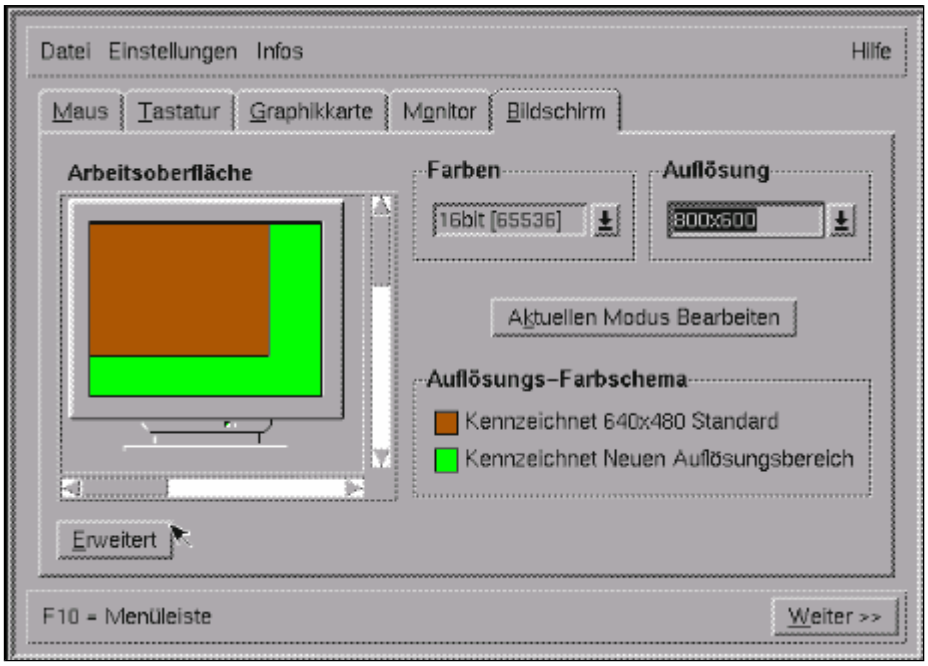


Abbildung 5: Sax(1): Konfiguration der Grafikmodi

Der letzte Punkt bestimmt Auflösung und Farbtiefe der Darstellung. Dabei werden nur solche Konstellationen zugelassen, die Grafikkarte und Monitor auch verarbeiten können. Für jede Farbtiefe lassen sich auch mehrere Auflösungen konfigurieren; zwischen diesen kann im laufenden Serverbetrieb mit den Tastenkombinationen [Ctrl][Alt][+] bzw. [Ctrl][Alt][-] gewechselt werden (die jeweils höchste Auflösung bestimmt die Größe des virtuellen Desktops!).
Abschließend sollten die aktuellen Einstellungen getestet werden. Es erscheint ein blauer Bildschirm, dessen Größe über die Buttons verändert werden kann. Wichtig ist, dass alle vier Ecken vollständig und ohne Verzerrungen sichtbar sind!

Sax2 - Startbildschirm

Sax2 führt durch die Konfiguration des neuen X Servers XFree86 Version 4. Das Kommando zum Start lautet **sax2**:

```
root@sonne> sax2
SaX: detecting please wait...
SaX: startup
```

Sax2 ermittelt die Grafikhardware (und Maus, Tastatur) Ihres Rechners und startet - falls vorhanden - bereits den richten X-Server. Rufen Sie **sax2 --help** auf, um die Kommandozeilenoptionen zu erfahren, mit denen Sie die automatische Erkennung ganz oder teilweise deaktivieren können.

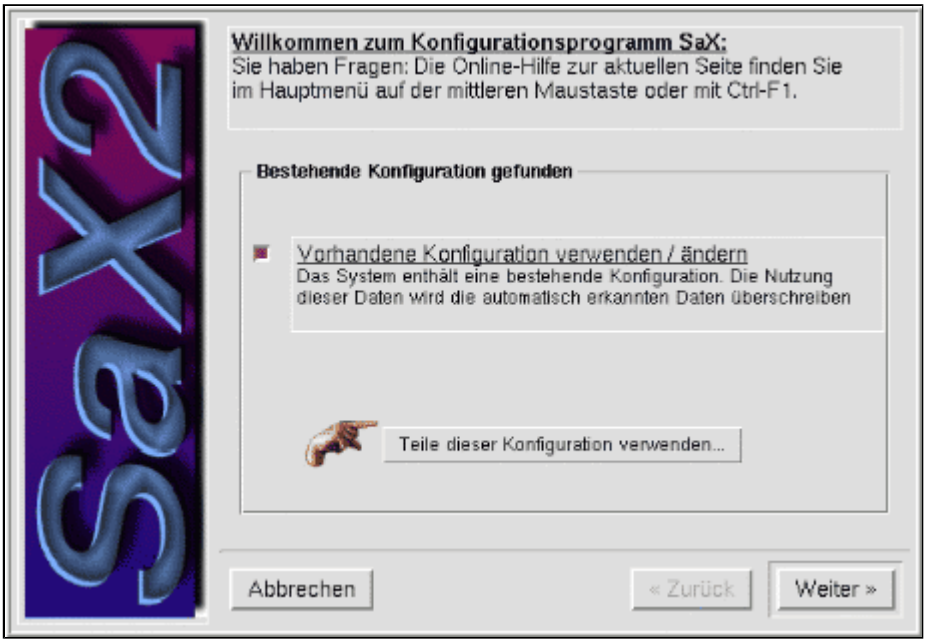


Abbildung 6: SaX2 Startmenü

Der dargestellte Startbildschirm wird nur sichtbar, falls die Konfigurationsdatei **/ etc/ X11/ XF86Config** bereits existiert. In dem Fall bietet das Werkzeug Ihnen an, die dortigen Einstellungen zu importieren. Bei der kompletten Übernahme werden alle durch die automatische Hardwareerkennungen ermittelten Werte durch die Angaben aus der Datei ersetzt. Mittels »Teile dieser Konfiguration verwenden...« erlaubt ein Dialog die Auswahl der zu importierenden Sektionen:

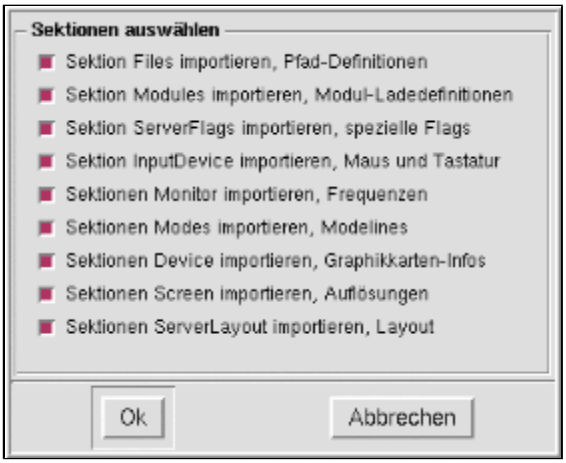


Abbildung 7: SaX2 - Import einzelner Sektionen

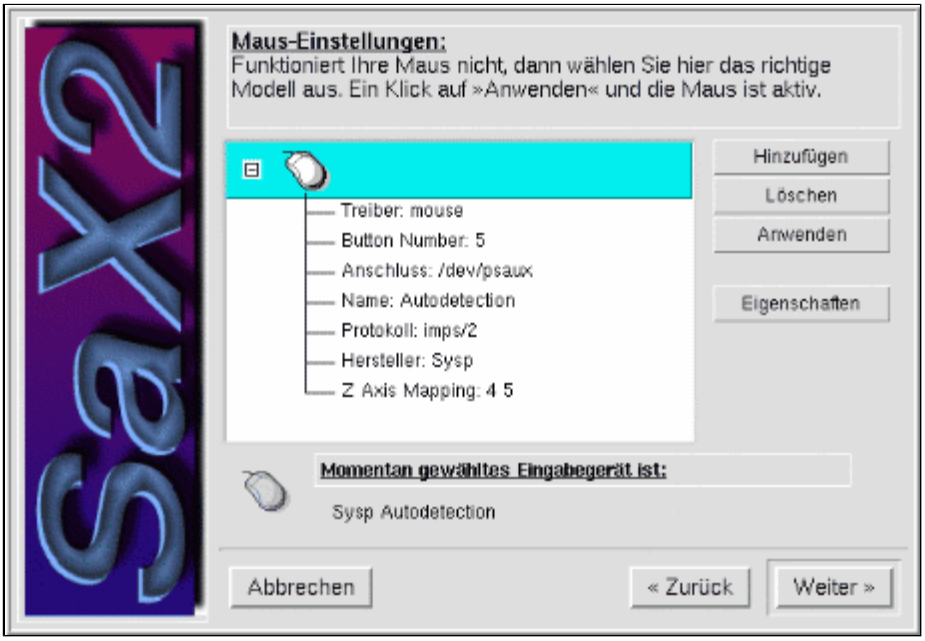


Abbildung 8: SaX2 - Die Maus einstellen

XFree86 ab Version 4 unterstützt die Verwendung mehrerer Eingabegeräte. Sind die von Sax2 vorgegebenen Werte fehlerhaft oder favorisieren Sie eigene Feinabstimmungen, so besteht über [Eigenschaften] die Möglichkeit des manuellen Eingriffs. Sowohl die Auswahl gängiger Mausmodelle als auch die Beeinflussung einzelner Parameter wie Doppelklick-Zeitspanne, Rad-Aktivierung einer Wheelmaus oder die Zuordnung einer Maus zu einem konkreten »Screen« ist gegeben.

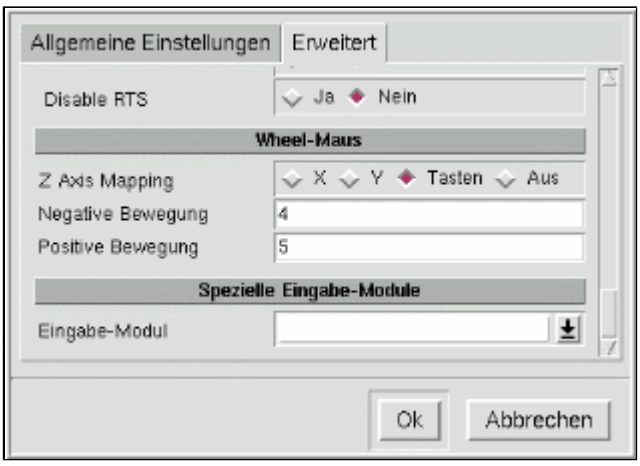


Abbildung 9: SaX2 - Die Maus einstellen, Expertenmodus

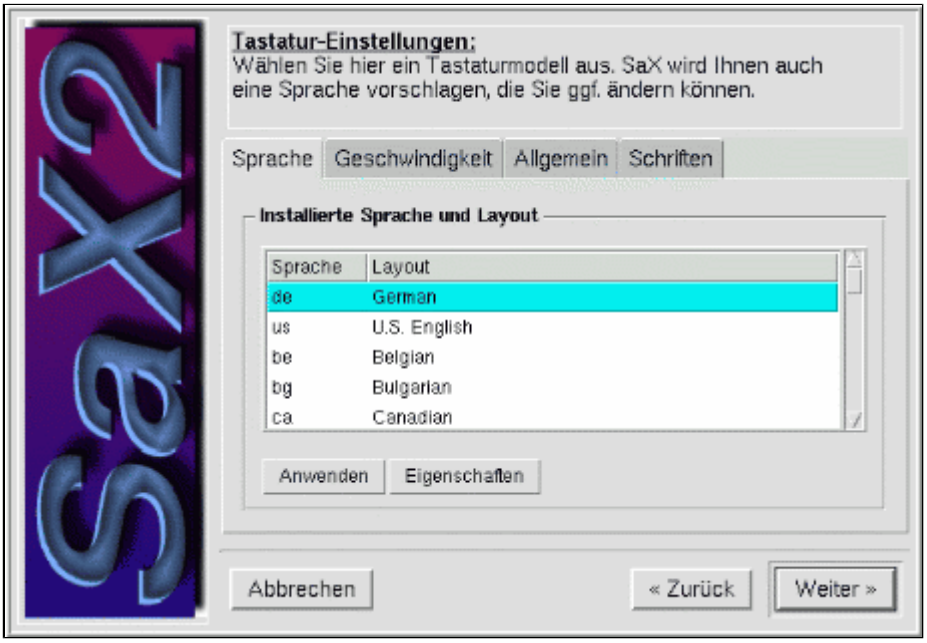


Abbildung 10: SaX2 - die Tastaturtabelle einstellen

Unter [Allgemein] finden Sie die Auswahl des Tastaturmodells. Bei den heute üblichen Standardtastaturen handelt es sich um den mit »Standard Keyboard [pc104]« bezeichneten Eintrag. Bei [Sprache] erfolgt die Konfiguration der zu verwendenden Tastaturtabelle, wobei »de German« automatisch die »nodeadkeys«-Variante impliziert.

Ein wenig Feineinstellung ist unter [Geschwindigkeit] möglich, was wohl eher dem geübten Schnelltipper zum Vorteil gereicht.

Wichtiger sind die [Schriften], denn auf Schriftarten, deren Installationspfade Sie hier vergessen, wird keine Anwendung unter X zugreifen können. Fügen Sie daher sämtliche Pfade hinzu, unter denen Schriften in Ihrem System installiert sind (i.d.R. finden Sie diese unter /usr/X11R6/lib/X11/fonts).

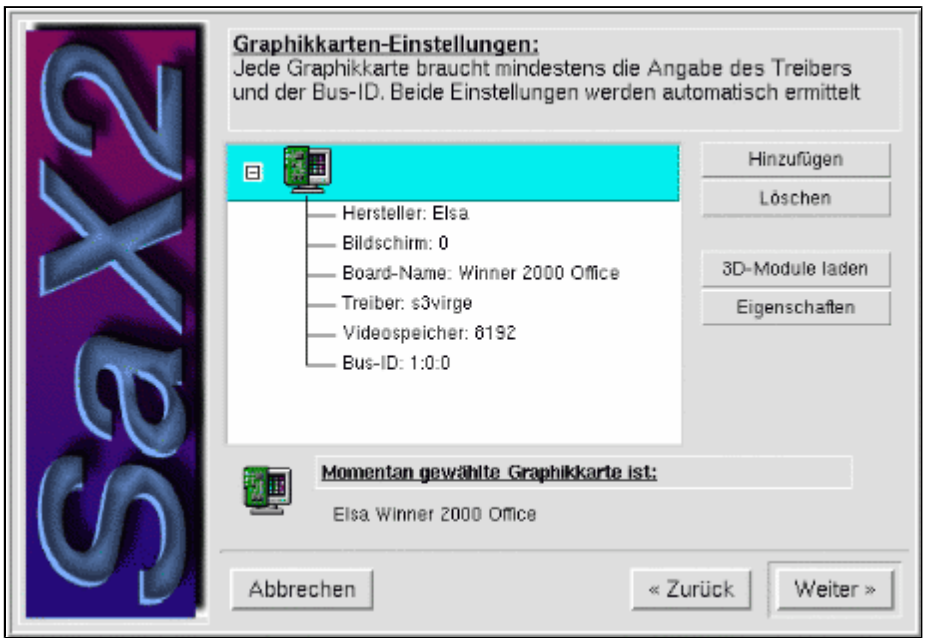


Abbildung 11: SaX2 - Auswahl der Grafikkarte

Neben mehreren Grafikkarten lassen sich auch mehrere Server pro Grafikkarte konfigurieren. Deswegen befinden sich selbst mehrere Grafikkarten in der Liste, obwohl das System nur über eine verfügt. Üblich ist, für eine Grafikkarte neben dem »optimalen« Server noch den VGA-Server zu konfigurieren. Dieser arbeitet mit jeder VESA-kompatiblen Karte zusammen und dient deshalb als »Notbehelf«. Dieser X-Server wird starten, falls der Aufruf des »eigentlichen« Servers (warum auch immer) scheitert.

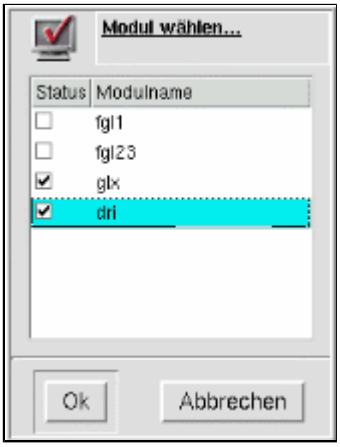


Abbildung 12: SaX2 - 3D-Eigenschaften aktivieren

Aktuelle Grafikkarten verfügen zumeist über eine hardwarebasierte 3D-Unterstützung. Um diese verwenden zu können, muss der X-Server verschiedene Module laden. **GLX** (Graphic Library for X) ist hierbei für die Bereitstellung der 3D-Fähigkeiten über das Netzwerk zuständig; da X von Haus aus netzwerktransparent arbeitet, ist diese Erweiterung erforderlich. **DRI** (Direct Rendering Infrastructure) ermöglicht es den Anwendungen, unter Umgehung des X-Servers auf die Grafikkarte zuzugreifen. Nur dieses Vorgehen ermöglicht effizientes 3D; da es aber der eigentlichen Philosophie von X zugegen läuft; kümmert DRI sich um die Synchronisation der Zugriffe auf die Hardware zwischen X-Server und der jeweiligen Hardware. Die *Fire GL*-Module sind einzig für Intel Pentium III-Systeme nützlich, womit die Verwendung der ISSE-Erweiterung (Internet Streaming Single-Instruction, Multiple-Data Extensions) zu einer Beschleunigung von 3D-Darstellungen führen kann.

Abbildung 13: SaX2 - Allgemeine Eigenschaften

Unter [Eigenschaften] erreichen Sie die Details zu Ihrer Grafikkarte. Wenn die automatische Erkennung von Sax2 versagte, finden Sie das Modell eventuell doch in der Liste. In der Registerkarte [Erweitert] lassen sich gar alle relevanten Daten direkt manipulieren, was Sie nur tun sollten, wenn Sie sich dieser Werte absolut sicher sind. Falls Sie mehrere Grafikkarten - oder eine Karte mit zwei Ausgängen - nutzen und somit mehrere Bildschirme ansteuern möchten, müssen Sie die Nummer (Zählung beginnt bei 0) des Desktops ("Bildschirm") eintragen, der mit einer Grafikkarte verbunden sein soll. In der folgenden Maske zur Desktop-Konfiguration wird automatisch die Mindestanzahl der Desktops angepasst.

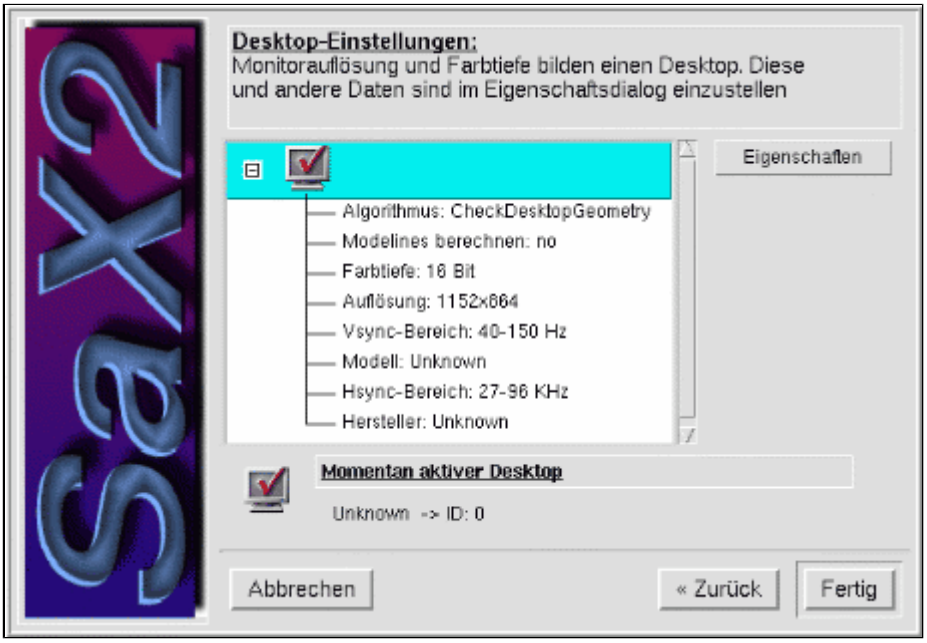


Abbildung 14: SaX2 - Einrichten des Desktops

Die Liste der Desktops kann mehrere Einträge umfassen, falls Sie mehrere Grafikkarten konfiguriert und diesen verschiedene Desktops zugewiesen haben. Überprüfen Sie zunächst, ob die angegeben Werte für »VSync-Bereich« und »HSync-Bereich« den Angaben im Handbuch Ihres Monitors entsprechen. Mit zu geringen Werten werden Sie nicht die Fähigkeiten des Monitors ausnutzen und unter Umständen ein flimmerndes Bild erhalten. Liegen die Werte jedoch über der Spezifikation, ist eine Zerstörung des Bildschirms möglich (die automatische Erkennung von Sax2 arbeitet sehr konservativ und setzt im Zweifelsfall eher zu geringe Werte). Änderungen zum Monitor lassen sich in der Registerkarte [Monitor] unter [Eigenschaften] tätigen, wobei - die Kenntnis vorausgesetzt - die direkte Angabe der Frequenzen (Registerkarte [Frequenzen]) der Auswahl des Monitormodells vorzuziehen ist (falls diese voneinander abweichen).

Die weiteren erforderlichen Abstimmungen betreffen Farbtiefe und Auflösung. Letztere ist sicherlich von der Qualität und Größe Ihres Monitor abhängig und ebenso Geschmacksache. Die heutigen Grafikkarten ermöglichen auch bei höchsten Auflösungen noch hohe Farbtiefen. Wer auf Performance (Spiele) Wert legt, ist mit 16Bit Farbtiefe bestens bedient, stellt diese doch weniger Anforderungen an die Grafikkarte als 24Bit. Ein optischer Unterschied ist ohnehin nicht auszumachen (vor allem einige ältere X-Anwendungen arbeiten nicht mit jeder Farbtiefe zusammen!). Die Werte für Farbtiefe und Auflösung variieren Sie ebenso unter [Eigenschaften].

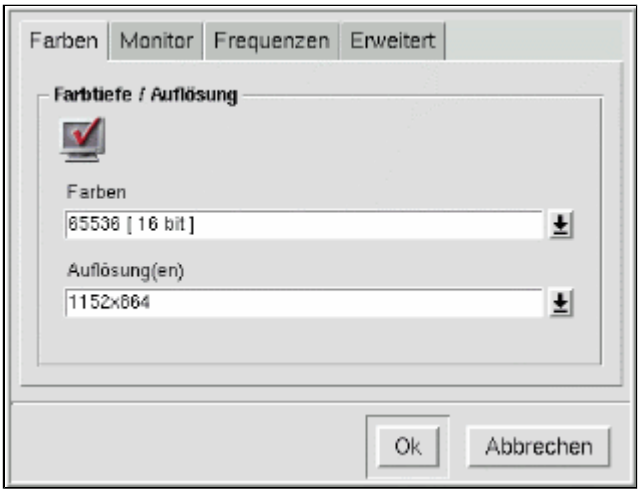


Abbildung 15: SaX2 - Desktop Feineinstellungen

Abbildung 16: SaX2 - Layout-Auswahl

Die Maske »Server-Layout« erscheint nur, wenn Sie mehrere Grafikkarten und mehrere Desktops konfiguriert haben. Für jeden der Desktops ist symbolisch ein Monitor vor einem Gitternetz dargestellt, wobei die angezeigte Nummer der Nummer des jeweiligen Desktops entspricht. Mit der Maus lassen sich diese Desktops in ihrer relativen Lage zueinander verschieben, sodass Sie das Layout beliebig anpassen können.

Relativ unwichtig ist die Anordnung, wenn Sie den [Clone Modus] anschalten. Dann wird auf jedem Desktop ein identisches Bild erzeugt.

Mit [Traditionell] erscheint der X-Server auf alle Fenster verteilt; jedoch läuft der Windowmanager einzig auf Desktop 0. Die anderen Desktops zeigen den typischen grauen X-Hintergrund. Fenster gestarteter Anwendungen sind auf den Desktop ihrer Erzeugung beschränkt.

Erst für [Ein Fenster (Xinerama)] kommt die Anordnung zum tragen, hierbei wird die sichtbare Arbeitsfläche über alle Desktops verteilt.

Zum Abschluss geht es ans Testen der getroffenen Einstellungen, wobei ein blauer Bildschirm sichbar werden sollte. Justieren Sie dessen Lage und Größe so, dass die Ecken vollständig und ohne Verzerrungen sichtbar sind.

Die Datei XF86Config

Übersicht

Vermutlich gibt es heute nur noch wenige Hardware-Konstellationen, bei denen alle verfügbaren Konfigurationwerkzeuge versagen. Für den seltenen Fall, dass dem doch so ist, kann eine manuelle Bearbeitung der XF86Config-Datei die Rettung bedeuten. Aber Vorsicht: **Falsche Einstellungen zerstören unter Umständen die Hardware**!

Ich selbst hatte vor Jahren das TFT-Displays eines Notebooks zur Mitarbeit animiert, nachdem ich etwas mit den Modelines herum experimentierte. Zugegeben, es war eine riskante Angelegenheit, aber auf das Notebook gab's ja noch Garantie;-)

Die Lage der Datei XF86Config

»XF86Config« ist die einzige für den X-Server relevante Konfigurationsdatei. In den meisten Systemen werden Sie sie im Verzeichnis »/etc/X11/« finden; ältere SuSE-Distributionen (< 7) legten sie direkt im Verzeichnis »/etc/« ab. Seit XFree86 Version 4 ist die Namensgebung leider nicht bei allen Distributionen einheitlich; mitunter nennt sich die Datei auch »XF86Config-4«. Gar noch größere Vielfalt ist zu beobachten, wenn die X-Server in beiden Versionen 3.3.x und 4.x parallel installiert sind. Oft konfiguriert »/etc/XF86Config« dann den älteren Server, während »/etc/X11/XF86Config« die Version 4 betrifft.

Der Aufbau der Datei XF86Config

Die Datei »XF86Config« ist in so genannte »Sektionen« gegliedert, die in diesem Abschnitt vorgestellt seien. Einige Sektionen sind erst in den Versionen ab XFree86 4.0 relevant, andere kennen nur die »älteren« X-Implementierungen, wobei mit Ausnahme von XInput alle Sektionen weiterhin unterstützt werden. Bei den versionsabhängigen Sektionen geben die Überschriften der einzelnen Abschnitte die zugehörige Version an; gilt eine Sektion in jeder Version, entfällt diese Angabe.

Abweichende Syntax

Mit dem Wechsel von XFree86 Version 3.3.6 zu Version 4 wurde die Syntax mancher Angaben in der Datei XF86Config geringfügig geändert. Genau genommen betrifft es die Einträge der Sektionen, die als »Optionen« bezeichnet werden. Ab Version 4 werden diese Optionen nicht nur bei ihrem Namen benannt, sondern zusätzlich durch das Schlüsselwort »Option« eingeleitet. Eine »Section ServerFlags» in Syntax der Version 4 sähe dann wie folgt aus:

```
Section "ServerFlags"
  Option    "AllowMouseOpenFail"
  Option    "DontZap"
EndSection
```

In der Konfigurationsdatei für X-Server der Versionen 3.3.x lautet ein gleichwertiger Eintrag hingegen:

```
Section "ServerFlags"
  AllowMouseOpenFail
  DontZap
EndSection
```

In den nachfolgenden Beispielen verwenden wir mit Ausnahme der 3.3.x-er spezifischen Sektionen ausschließlich die neuere Syntax; der Leser sollte die Änderungen leicht nachvollziehen können.

Die Sektion Files

Bestimmte Variablen enthalten Pfadangaben, in denen der Server nach Dateien sucht:

FontPath	Verzeichnisse, die Schriften enthalten
RGBPath	Verzeichnis mit der Datenbasis der Rgb-Farbtabelle
ModulePath	Verzeichnis mit Modulen

Sollten Sie bspw. nachträglich neue Schriften installieren, so fügen Sie den neuen Pfad zu [FontPath] hinzu, und alle X-Programme können nachfolgend diese verwenden.

```
Section "Files"
  RgbPath     "/usr/X11R6/lib/X11/rgb"
  FontPath    "/usr/X11R6/lib/X11/fonts/Type1"
  FontPath    "/usr/X11R6/lib/X11/fonts/URW"
  FontPath    "/usr/X11R6/lib/X11/fonts/misc"
  FontPath    "/usr/X11R6/lib/X11/fonts/75dpi"
  FontPath    "/usr/X11R6/lib/X11/fonts/100dpi"
  FontPath    "/usr/X11R6/lib/X11/fonts/freefont"
  FontPath    "/usr/X11R6/lib/X11/fonts/sharefont"
EndSection
```

Auch der Port, an dem ein möglicher Fontserver seine Dienste anbietet, wird dem X-Server über einen FontPath-Eintrag bekannt gegeben. Für den Standardport 7100 sähe ein Eintrag wie folgt aus:

```
Section "Files"
...
  FontPath    "unix/:7100"
...
EndSection
```

Die Sektion Module

Module sind Erweiterungen, die der X-Server dynamisch nachladen kann, um bestimmte Geräte zu unterstützen. In Versionen vor 4.0 existieren einzig Module, die den Zugriff auf verschiedene Eingabegeräte wie Joysticks, Touchscreens,... ermöglichen; diese Geräte selbst müssen in der Sektion XInput angegeben werden.

Ab Version 4 des X-Servers sind weitere Teile dessen als Module realisiert.

Module werden entweder mit ihrem vollständigen Zugriffspfad oder nur mittels ihres Namens angegeben. In letzterem Fall muss das Modul in einem der durch »ModulePath« genannten Verzeichnisse oder in einem Standardpfad (/usr/X11R6/lib/modules/fonts oder /usr/X11R6/lib/modules/extensions) enthalten sein.

```
Section "Modules"
  Load  "dri"
  Load  "glx"
  Load  "extmod"
  Load  "freetype"
```

```
EndSection
```

Verwenden Sie **XFree86 Version 4** und verfügen über eine Grafikkarte mit 3D-Unterstützung, so sollten Sie die Module für das Direct Rendering Interface (load "dri") und die X-Window-Erweiterungen von OpenGL (load "glx") einbinden, um die Fähigkeiten ihrer Grafikkarte auch nutzen zu können.

Die Sektion ServerFlags

Verschiedene Optionen beeinflussen die Arbeit des X-Servers. Die interessantesten sind:

DontZap

Verhindert das Beenden des X-Servers über die Tastenkombination [Ctrl]-[Alt]-[Backspace]. Ein solches Verhalten kann in Verbindung mit grafischem Login nützlich sein.

DontZoom

Verhindert das Ändern der Auflösung während einer X-Sitzung. Das Umschalten zwischen den konfigurierte Auflösungen ist sonst mit den Tastenkombinationen [Ctrl]-[Alt]-[-] bzw. [Ctrl]-[Alt]-[+] möglich.

AllowMouseOpenFail

Der Server startet auch, wenn er das Maus-Device nicht öffnen konnte. In der Voreinstellung beendet sich Server in einem solchen Fall.

Eine ServerFlag-Sektion könnte wie folgt aufgebaut sein:

```
Section "ServerFlags"
  Option    "AllowMouseOpenFail"
  Option    "DontZap"
EndSection
```

Die Sektion Keyboard [3]

Bemerkung: Ab XFree86 Version 4.0 ist diese Sektion in der Sktion »InputDevice« enthalten.

Die Sektion ist hauptsächlich für die Belegung der Tastatur verantwortlich. Zwar können die einzelnen Tasten auch nachträglich neu belegt werden, doch wird man das kaum für alle 104 Tasten vollziehen wollen. Das manuelle Mapping dient nur, um einzelnen Tasten des Standardlayouts mit neuen Funktionen zu versehen.

Die nachfolgend beschriebenen Optionen sind nur eine Auswahl.

Protocol

Hier wird wohl immer **Standard** stehen. Mit **Xqueue** wird das Verhalten von System V Unix eingestellt, wo Eingaben zunächst in einer "Event Queue" landen.

LeftAlt, AltGr, ScrollLock, RightCtl

Die Standardbelegung der benannten Funktionstasten kann verändert werden. Die Voreinstellung ist *Meta* f *LeftAlt* und *AltGr*, *Compose* für *ScrollLock* und *Control* für *RightCtl*. Als neue Werte sind gestattet: *Meta*, *Compose*, *ModeShift*, *ModeLock*, *ScrollLock* und *Control*.

XkbRules

Solange der X-Server von XFree86-Projekt stammt, sollte hier immer "xfree86" stehen. Dies ist auch die Voreinstellung, falls die Option nicht angegeben ist.

XkbModel

Die Voreinstellung ist "pc102", sollte aber auf "pc104" gesetzt werden, um die zusätzlichen Tasten einer 10
Tasten-Tastatur verwenden zu können.

XkbLayout

Hier steht der Ländercode, also "de" für Unterstützung der deutschen Tastaturbelegung. Die Voreinstellung
"us".

XkbVariant

Einige Tasten (~, ^, ´) sind "tote" Tasten, d.h. sie werden beim einfachen Anschlag nicht sofort in einen
Zeichencode umgesetzt, sondern nur, wenn nachfolgend die Leertaste gedrückt wurde. Dieses Verhalten di
z.B. zum Setzen von Akzenten in vielen Sprachen. Für deutsche Tastaturbelegungen empfiehlt sich die Opti
"nodeadkeys", die die "toten" Tasten aktiviert.

```
Section "Keyboard"
  Protocol     "Standard"
  XkbRules     "xfree86"
  XkbModel     "pc104"
  XkbLayout    "de"
  XkbVariant   "nodeadkeys"
EndSection
```

Die Sektion Pointer [3]

Bemerkung: Ab XFree86 Version 4.0 ist diese Sektion in InputDevice enthalten.

Hier erfolgen die Angaben über die Maus. Hierunter zählen Mausprotokoll, Anzahl unterstützter Tasten, die Zeitspanne zwischen zwei Klicks... Ich beschränke mich auf die gebräuchlichen Optionen:

Protocol

Bei neueren Modellen und vor allem aktuelleren XFree86-Versionen (> 3.3.5) kann im Falle serieller Mäuse **auto** angegeben werden. X sollte in der Lage sein, den konkreten Typ selbsttätig zu bestimmen. Weitere verbreitete Mausprotokolle sind "Logitech", "Microsoft" und "PS/2". Verwechseln Sie nicht den Hersteller mi dem Protokoll, auch viele Logitech-Mäuse arbeiten nach dem Microsoft-Protokoll!

Device

Die Schnittstelle, an der die Maus angeschlossen ist. Steht hier "/dev/mouse", so sollte dies ein Link auf das entsprechende Device sein. Für serielle Mäuse lauten die richtigen Namen "/dev/tty0" (1. serielle Schnittste "/dev/tty1" (2. serielle Schnittstelle) usw. Eine PS/2-Maus ist am Device "/dev/psaux", eine USB-Maus an "/dev/usbmouse" (veraltet: /dev/input/mice) angeschlossen.

BaudRate

Die Geschwindigkeit, mit der Daten von der Maus empfangen oder zur Maus übertragen werden. 1200 ist d normale Wert.

Buttons

Anzahl der Maustasten, wenn sich diese von 3 (Voreinstellung) unterscheidet.

Emulate3Buttons

Bei 2-Tasten-Mäusen wird die dritte Taste durch gleichzeitiges Drücken der beiden Tasten emuliert. Um sin mit X arbeiten zu können, sind 3 Tasten notwendig.

Emulate3Timeout

Diese Zeitspanne in Millisekunden kann das Betätigen beider Tasten maximal auseinander liegen, so dass d Server dies als 3. Taste akzeptiert. Voreinstellung ist 50 ms.

Resolution

Mausgeschwindigkeit; sinnvolle Werte liegen - je nach Geschmack - zwischen 200..600 (Hz).

SampleRate

Anzahl der Mausereignisse (Tastendruck, Bewegung), die die Maus maximal in einer Sekunde senden kann.

ZAxisMapping

Dient der Konfiguration eines Mausrads. Die beiden Werte sind so zu wählen, dass sie um 1 bzw. 2 höher s als die Anzahl der Maustasten.

Ein entsprechender Abschnitt der Konfigurationsdatei ist wie folgt aufgebaut:

```
Section "Pointer"
  Protocol      "Microsoft"
  Device        "/dev/ttyS0"
  SampleRate    "60"
  BaudRate      "1200"
  ZAxisMapping  "4 5"
EndSection
```

Die Sektion Monitor

Bemerkung Ab XFree86 Version 4.0 sind die ModeLine-Zeilen in eine eigene Sektion Modes ausgelagert.

Zum Verständnis des folgenden Abschnitts ist eine kurze Einführung in die prinzipielle Arbeitsweise von Monitoren unumgänglich:

Eine Kathode erzeugt in Zusammenarbeit mit einer Anode einen Elektronenstrahl, der beim Auftreffen auf die Bildschirmoberfläche (Anode) deren Beschichtung zum Erleuchten bringt. Der Elektronenstrahl beginnt in der linken oberen Bildschirmecke, bewegt sich, gesteuert durch ein Ablenksystem, nach rechts, bis ein **Hsync**-Signal ihn zurück zum Zeilenanfang der folgenden Zeile bringt. Erreicht der Elektronenstrahl die rechte untere Bildschirmecke, ist ein **Vsync**-Signal notwendig, um einen vertikalen Rücklauf zum Bildschirmanfang zu bewirken. Die verschiedenen Kontraste und Farben der Darstellung werden durch die Intensität des Elektronenstrahls variiert, beim Rücklauf bewirkt die Intensität 0 dessen "Unsichtbarkeit". Das Nachleuchten der angeregten Teilchen auf der Bildschirmoberfläche ist nur von kurzer Dauer, so dass der Strahl mehrfach in der Sekunde den gesamten Bildschirm überstreifen muss, um ein flimmerfreies Bild zu erzeugen (Refresh oder vertikale Bildschirmfrequenz). 70 Bilder sind das Minimum, das man seinen Augen in der Sekunde zumuten sollte.

Zurück zur Konfiguration...

Mit **Identifier** ist ein eindeutiger Name einzugeben, über den in der später beschriebenen Sektion "Screen" der Monitor referenziert wird. Die beiden Angaben zum Monitor-Hersteller "VendorName" und Modell "ModelName" besitzen rein informativen Charakter und können entfallen.

Enorm wichtig ist die korrekte Angabe der horizontalen **HorizSync** und vertikalen **VertSync** Frequenzen, die der Monitor unterstützt. Ein zu kleiner Wert ist nicht kritisch, allerdings leidet die Qualität der Ausgabe, da die Möglichkeiten des Monitors nicht ausgereizt werden. Zu hohe Angaben können allerdings auf die Dauer den Bildschirm beschädigen. Welche Frequenzbereiche für einen Monitor zulässig sind, erfahren Sie aus dem Handbuch oder bei manchen Modellen durch einen Herstelleraufdruck auf der Monitor-Rückseite. Bei Festfrequenzmonitoren werden die möglichen Werte durch Komma getrennt angegeben, bei variablen Frequenzen erfolgt die Angabe als Bereich *Min-Max*.

Die "ModeLine"-Zeilen sind das Kernstück der Konfiguration. Sie bestimmen, wie sich der Elektronenstrahl über die Mattscheibe bewegt und bestimmen somit neben der Bilddimension dessen Ausrichtung auf dem Bildschirm (Breite, Höhe, Abstände zum Rand). Die Angabe der Modelines kann nach 2 Varianten erfolgen:

```
# 1. Syntaxvariante
ModeLine "800x600" 97.55 800 816 928 1040 600 600 615 626

# 2. Syntaxvariante
Mode "800x600"
    DotClock    97.55
    HTimings    800 816 928 1040
    VTimings    600 600 615 626
    Flags
EndMode
```

DotClock ist die Videobandbreite in MHz und gibt an, wie viele Pixel binnen einer Sekunde von der Grafikkarte zum Monitor übertragen werden können. Angenommen das Bild soll mit 100 Hz (100 Bilder in der Sekunde) dargestellt werden, so ist bei einer Auflösung von 800x600 Bildpunkten eine Videobandbreite von 48 + *x* MHz notwendig, wobei *x* eine durch den horizontalen und vertikalen Rücklauf des Elektronenstrahls bestimmte Komponente (Faustregel ca. 10 MHz) ist. Der angegebene Wert muss mit einem der Werte aus der Sektion Device übereinstimmen (eine Abweichung in der Nachkommastelle ist zulässig), sonst handelt es sich um einen ungültigen Modus. Bei modernen Grafikkarten lassen sich die Frequenzen frei programmieren, so dass der Modus mit der höchst möglichen "DotClock" bei gleicher Auflösung gewählt wird.

HTimings steuert die horizontale Bewegung des Elektronenstrahls. Im obigen Beispiel soll dieser 800 Pixel anzeigen, nachfolgend 16 Pixel (801-816) "schwarz" darstellen. Für die Zeitspanne der Darstellung von 928-816=112 Pixel, wird ein Hsync-Signal erzeugt, das den Strahl an den Beginn der nächsten Zeile platziert. Auf dieser sind nochmals 1040-928=112 Pixel mit einer Intensität von "0" zu zeichnen, bevor der Vorgang wiederholt wird.

VTiming ist dasselbe Prinzip bezogen auf Zeilen. Im Beispiel sind also 600 Zeilen "normal" anzuzeigen, bevor "keine" Zeile "schwarz" dargestellt wird. Anschließend folgt ein Vsync-Impuls, der solange währt, wie die Darstellung von 15 Zeilen andauern würde, um mit 11 Zeilen "unsichtbarer" Darstellung einen Durchlauf abzuschließen.

Flags definiert spezielle Optionen, z.B. **Interlace**, womit billige Monitore versuchen, die Frequenzen teurer Exemplare zu simulieren und den Benutzer mit Kopfschmerzen quälen. Bei eingermaßen moderner Technik sollten die Flags nicht mehr notwendig sein.

TFT-Bildschirme neigen meist zu einer zu hellen Darstellung, die mit den bildschirmeigenen Einstellmöglichkeiten nur unzureichend auszugleichen ist. Ein Eintrag **Gamma *Wert*** (Wert=0.1..10) kann durchaus zu akzeptablen Ergebnisses führen. Der genaue Wert muss durch Tests ermittelt werden.

Was nützt die Kenntnis der einzelnen Werte der Timings?

Bleiben wir beim Beispiel der obigen HTimings-Werte 800 816 928 1040. Das Verhältnis vom ersten zum letzten Wert bestimmt die Breite des Bildes. Ist die Differenz kleiner, wird die Darstellung breiter, ist sie größer, wird das Bild schmaler. Mit Verringerung des HSync-Signals (Differenz von zweitem zu drittem Wert) verschieben Sie das Bild nach rechts und mit einer Verlängerung nach links. Die besten Ergebnisse erzielen Sie, indem Sie den zweiten Wert erhöhen, während Sie den dritten Wert um den gleichen Betrag vermindern. Dieses "Finetuning" funktioniert allerdings nur in geringen Grenzen, wenn Sie die Werte zu stark ändern, kann der Monitor die Darstellung verzerren und im schlimmsten Fall verweigern.

Sie können sich zurücklehnen... es ist gar nicht notwendig, dass Sie die Werte von Hand errechnen. Verwenden Sie z.B. das Programm **xvidtune** (unter X und eventuell mit der Option "--display 0:0") und spielen Sie mit den Werten bis sich ein stabiles Bild ergibt.

Mit einem Beispiel verabschieden wir uns aus der Sektion Monitor:

```
Section "Monitor"
  Identifier    "Mein Monitor"
  VendorName    "Unwichtig"
```

```
  ModelName     "Interessiert nicht"
  HorizSync     30-96
  VertRefresh   50-150
  Modeline "1600x1000" 199.68 1600 1616 1968 2080 1000 1000 1015 1044
  Modeline "1280x960" 159.74 1280 1296 1552 1664 960 960 975 1003
  Modeline "1024x768" 127.49 1024 1040 1216 1328 768 768 783 802
  Modeline "640x480" 62.48 640 656 720 832 480 480 491 501
  Modeline "1600x1200" 199.68 1600 1616 1968 2080 1200 1200 1215 1253
  Modeline "1280x1024" 159.74 1280 1296 1552 1664 1024 1024 1039 1070
  Modeline "1152x864" 143.62 1152 1200 1416 1536 864 864 879 902
  Modeline "800x600" 97.55 800 816 928 1040 600 600 615 626
EndSection
```

Die Sektion Modes [4]

Der Aufbau der ModeLine-Zeilen ist analog zur Beschreibung aus der Sektion Monitor; eine (verkürzte) Sektion könnte damit so aussehen:

```
Section "Modes"
  Identifier    "Diese Modelines"
  Modeline "1600x1200" 199.68 1600 1616 1968 2080 1200 1200 1215 1253
...
  Modeline "800x600" 97.55 800 816 928 1040 600 600 615 626
EndSection
```

Wichtig ist bei mehreren Modes-Sektionen die Eindeutigkeit des **Identifier**, über den der Zugriff in der Sektion Monitor vorgenommen wird:

```
Section "Monitor"
...
  UseModes "Diese Modelines"
EndSection
```

Die Sektion Device

Für jede im System installierte Grafikkarte kann eine eigene Sektion Device enthalten sein. Wichtig ist der eindeutige **Identifier**, der einen frei wählbaren Bezeichner enthält. Die Angaben zum Hersteller **VendorName** und Modell **BoardName** können entfallen.

Ab **XFree86 Version 4** ist die Angabe von **Driver <Treibername>** zwingend erforderlich, um dem Server die korrekte Auswahl des Treibers für diese Grafikkarte zu ermöglichen. Die gültigen Treibernamen erfährt man aus den gleichnamigen Manualpages der Sektion 4 (meist unter /usr/X11R6/man/man4). Befinden sich mehrere Grafikkarten im System, benötigt der Server Kenntnis über deren Anordnung auf dem Bus. Mit **BusID <Slot>** wird die Zuordnung zwischen Grafikkarte und ihrem PCI- bzw. AGP-Steckplatz getroffen. Die korrekte Angabe für "Slot" kann mittels "X -scanpci" ermittelt werden. Der "BusID"-Eintrag kann bei Verwendung von nur einer Grafikkarte entfallen.

Für viele aktuelle Grafikkarten wurden damit schon alle erforderlichen Angaben erfasst, der Rest sollte automatisch ermittelt werden können.

Leider klappt es mit der Identifizierung der Parameter dann doch nicht immer so genau... vor allem ältere Grafikkarten halten sich mit ihren Informationen sehr bedeckt. Deswegen sollten die interessantesten Optionen kurz vorgestellt werden:

Chipset

Der Name des Chipsatzes der Grafikkarte kann hier angegeben werden. Sie erfahren diesen entweder anha der Beschriftung des Chips auf der Karte oder aus dem Handbuch oder mit Hilfe des Kommandos "SuperPro Kennen Sie den Chipsatz nicht, dann lassen Sie die Option weg.

Ramdac

Der Ramdac bestimmt die Taktfrequenz ihrer Grafikkarte. Auch diesen sollten Sie nur angeben, wenn Sie de Typ exakt kennen. Wieder können die unter Chipset angeführten Quellen die Information enthalten.

Clocks

Besitzt die Grafikkarte keinen programmierbaren Clock-Generator (bestimmt die Videobandbreite), so müss hier alle Frequenzen angegeben werden, die die Grafikkarte unterstützt. Die entsprechenden Werte finden mit etwas Glück in der Datei "/usr/X11R6/lib/X11/Cards". Eventuell entlockt auch ein Aufruf von "X -probeo die korrekten Angaben. Wenn nicht, könnten Sie mit Werten ähnlicher Grafikkartentypen experimentieren. Allerdings sollten Sie das nur als letzte Möglichkeit in Betracht ziehen und auch nur dann, wenn noch Garan auf die Hardware besteht. Eine **Clock**-Zeile schaut in etwa wie diese aus:

```
Clocks  25.20 28.32 40.0 30.0 50.10 77.0 36.10 45.0
```

Möchten Sie sehr viele Werte angeben, lassen sich diese auf mehrere Clock-Zeilen verteilen. Die Angabe ei Wertes macht nur Sinn, wenn in der Sektion Monitor ein entsprechender Modus existiert.

VideoRam

Der wohl unkritischste Wert... Hier tragen Sie einfach die Größe des RAMs ihrer Karte ein (in kByte).

Eine weitere (teils wichtige) Möglichkeit ist die Angabe von speziellen Optionen, die einem Schlüsselwort "Options" folgen können. Teilweise sind diese Optionen in den Dateien des Verzeichnisses "/usr/X11R6/lib/X11/doc/" dokumentiert. Aus eigener Erfahrung kann ich nur auf "sw_cursor" verweisen, da einige Grafikkarten ohne Verwendung dieser Option den Mauszeiger nur als flimmerndes Karree projezierten.

```
Section "Device"
  Identifier   "Primary-Card"
  VendorName   "---AUTO DETECTED---"
  BoardName    "---AUTO DETECTED---"
  VideoRam     8192
  Option       "sw_cursor"
EndSection
```

Für XFree86 in der Version 4 könnte eine Device-Sektion wie folgt aussehen, wobei wir die Existenz von zwei Grafikkarten (eine AGP, eine PCI) annehmen:

```
Section "Device"
  Identifier  "AGP-Karte"
  Driver      "ati"
  VideoRam    8192
  BusID       "PCI:1:0:0"
EndSection
Section "Device"
  Identifier  "PCI-Karte"
  Driver      "mga"
  VideoRam    4096
  BusID       "PCI:0:11:0"
EndSection
```

Die Sektion Screen

In dieser Sektion findet die Zuordnung von Grafikkarte, Monitor und X-Server zu den unterstützen Auflösungen statt. Es ist nicht ungewöhnlich, für mehrere Konstellationen mehrere Screen-Sektionen zu verwenden.

Jede Screen-Sektion beginnt mit 3 Einträgen:

Identifier (nur in Version 4)

Driver (nur in Version 3)

Hier steht der Name des zu verwendenden Treibers, also z.B. "svga" für den SVGA-Server, "accel", "mono", "vga2", "vga16" entsprechend (In der Version 4 findet die Zuordnung zum Treiber in der Device-Section sta

Device

Hier steht der Name der Grafikkarte für die dieser Abschnitt gelten soll. Dieser Name ist identisch mit dem Identifier-Eintrag der Sektion Device.

Monitor

Hier steht der Name des Monitors für den dieser Abschnitt gelten soll. Dieser Name ist identisch mit dem Identifier-Eintrag der Sektion Monitor.

Zwei weitere Optionen verdienen erwähnt zu werden. Dies ist zum einen **DefaultColorDepth**, das die Server anweist, mit der angegebenen Farbtiefe zu starten. Fehlt der Eintrag, ist der erste **SubSection "Display"**-Eintrag Ausschlag gebend. Mit **BlankTime** kann die Zeitspanne (in Minuten) zum Aktivieren des Bildschirmschoners eingestellt werden (wenn dieser nicht explizit gesetzt wird, erscheint ein schwarzer Bildschirm). Voreingestellt sind 10 Minuten.

Entscheidend für die Darstellung sind nun die verschiedenen **SubSection "Display"... EndSubSection**- Einträge. Für jede Farbtiefe (8,16, 24, 32 Bit) existiert meist eine eigene SubSection da es u.a. vom VideoRAM-Ausbau abhängt, welche Auflösung bei welcher Farbtiefe tatsächlich erzielt werden kann.

Mit **Depth** wird die **Farbtiefe** eingestellt, für die der Eintrag relevant ist. **Modes** enthält nun eine **Liste der Auflösungen**, die der X-Server unterstützen soll. Zwischen diesen kann im laufenden Betrieb mit [Ctrl][Alt][+] und [Ctrl][Alt][-] umgeschalten werden. Der X Server startet immer mit der ersten in der Liste angegebenen Auflösung, sofern es sich um eine gültige Angabe handelt. Dazu schaut der Server in der Sektion "Monitor" nach, ob ein passender ModeLine-Eintrag existiert, dessen DotClock wiederum in der "Device"-Sektion aufgeführt sein muss.

Abbildung 1: Virtuelle und physische Auflösung

Der X Server unterscheidet zwischen **virtueller** und **physischer** Auflösung. Letztere ist jene, die tatsächlich vom Bildschirm dargestellt werden kann. Mit **Virtual** kann eine virtuelle Auflösung eingetragen werden. Dabei wird der X Server nur einen Ausschnitt in Größe der gewählten physischen Auflösung darstellen. Und indem man mit der Maus »über« den Rand des Bildschirms hinausscrollt, wird der Ausschnitt automatisch in die vorgegebene Richtung verschoben. Ist die Angabe unter »Virtual« kleiner als der größte Wert in der Liste der Auflösungen, so wird jener Wert aus der Liste als virtuelle Auflösung genommen!

Das Beispiel zeigt die Screen-Sektion der Version 3.x; ab Version 4 tritt anstelle des Eintrags »driver« der »identifier«:

```
Section "Screen"
  Driver      "SVGA"
  Device      "Primary-Card"
  Monitor     "Mein Monitor"
  DefaultColorDepth   32
  SubSection "Display"
    Depth     32
    Modes     "1152x864" "1028x768" "800x600"
    Virtual   1152 864
  EndSubSection
  SubSection "Display"
    Depth     24
    Modes     "1152x864" "1028x768" "800x600"
    Virtual   1600 1200
  EndSubSection
  SubSection "Display"
    Depth     16
    Modes     "1600x1200"
  EndSubSection
  SubSection "Display"
    Depth     8
    Modes     "1600x1200"
  EndSubSection
EndSection

Section "Screen"
  Driver      "Accel"
  Device      "Primary-Card"
  Monitor     "Mein Monitor"
  DefaultColorDepth   32
  SubSection "Display"
    Depth     32
    Modes     "1152x864"
    Virtual   1152 864
  EndSubSection
  SubSection "Display"
    Depth     24
    Modes     "640x480"
  EndSubSection
  SubSection "Display"
    Depth     16
    Modes     "1600x1200"
  EndSubSection
  SubSection "Display"
    Depth     8
    Modes     "640x480"
  EndSubSection
EndSection
```

Die Sektion XInput [3]

Bemerkung: Diese Sektion wird ab XFree86 Version 4.0 nicht mehr unterstützt.

Die Sektion XInput wird einzig zur Unterstützung weiterer Eingabegeräte benötigt. Moderne Eingabegräte bringen ihre eigenen Treiber in Form von Modulen mit, so dass diese in der Sektion Module anzugeben sind. Wichtigstes Gerät wird wohl ein Joystick sein, der unter Linux mit folgenden Einschränkungen unterstützt wird:

- Bewegungen werden relativ ausgewertet
- Zwei Richtungen werden ausgewertet (x und y)
- Zwei Tasten werden unterstützt

Jedes Gerät wird innerhalb einer **SubSection-EndSubSection**-Umgebung beschrieben, für einen Joystick sind die Angaben **Port**, das Device, an dem der Joystick angeschlossen ist und **DeviceName**, der Name des X-Devices, zwingend. Die anderen Angaben überschreiben die voreingestellten Werte und werden einzig zur Feinabstimmung verwendet.

Die Sektion InputDevice [4]

Diese Sektion ist als Ersatz der alten Sektionen "Keyboard", "Pointer" und "XInput" gedacht; wobei die beiden ersteren weiterhin unterstützt werden.

Neu hinzu kommende Schlüsselwörter sind **Identifier <Eindeutiger_Name>** und **Driver <Treibername>** . Die weiteren Einstellungen erfolgen analog zu den Angaben der "alten" Sektionen.

Eine zu den Beispielen in Keyboard und Pointer identische Konfiguration würde somit wie folgt aussehen:

```
Section "InputDevice"
  Identifier      "Meine Tastatur"
  Driver          "keyboard"
  Option "Protocol"          "Standard"
  Option "XkbRules"          "xfree86"
  Option "XkbModel"          "pc104"
  Option "XkbLayout"         "de"
  Option "XkbVariant"        "nodeadkeys"
EndSection

Section "InputDevice"
  Identifier      "Meine Maus"
  Driver          "mouse"
  Option "Protocol"          "Microsoft"
  Option "Device"            "/dev/ttyS0"
  Option "SampleRate"        60
  Option "BaudRate"          1200
EndSection
```

Die Sektion VideoAdaptor [4]

Die derzeitige Aufnahme dieser Sektion scheint in weiser Voraussicht auf die weitere Entwicklung von XFree86 geschehen zu sein. Vorgesehen ist hier die direkte Unterstützung von Videokarten (Fernsehen). Der Status der jetzigen Implementierung lässt keine Rückschlüsse auf eine mögliche Konfiguration zu, selbst auf der Homepage der XFree86-Organisation finden sich keinerlei Hinweise.

Die Sektion ServerLayout [4]

Diese Sektion trägt der Möglichkeit von XFree86 Version 4 Rechnung, den Desktop wahlweise auf einem oder mehreren ("Multi Head Modus") Bildschirm(en) darzustellen. Die ServerLayout-Sektion verbindet eine oder mehrere Screen-Sektionen mit einer oder mehreren Input-Sektionen. Bei Verwendung nur eines Monitors, einer Tastatur und einer Maus kann sie entfallen.

Eine jede ServerLayout-Sektion kann folgende Einträge enthalten:

Identifier

Eindeutiger Name für diese Sektion

Screen

Für jeden Screen, der in dieser Sektion genutzt werden soll, ist ein eigener Eintrag anzulegen. Dem Schlüsselwort »Screen« folgt der **Name** der Screen-Sektion, die hiermit referenziert werden soll (Identifier "Section Screen").

Alle weiteren Angaben sind optional.

Dem Screen-Namen kann eine **Screen-Nummer** (Zählung beginnt mit 0) folgen. Dies ist nur notwendig, w die Nummerierung von der Reihenfolge der Screen-Einträge abweicht.

Bei mehreren Screen-Zeilen kann die **Positionsangabe** eines Screens (Screen-Number = 0) entfallen; alle weiteren Screens erfordern die Angabe der Lage bez. dieses "Referenzscreens". Mit **Absolute x y** können verbindliche Koordinaten für den Screen vereinbart werden, wobei die Lage ausgehend von der linken ober Desktopecke (0,0) beschrieben wird. Das Gegenstück ist **Relative "Screen-Name" x y**, das die Lage ausgehend vom mit Screen-Name benannten Eintrag kalkuliert (anstatt des Namens kann auch die Screen-Nummer angegeben werden).

Gebräuchlich sind aber die Angaben **RightOf "Screen-Name"**, **LeftOf "Screen-Name"**, **Above "Screer Name"** und **Below "Screen-Name"**, die für rechts, links, oberhalb bzw. unterhalb des benannten Screen stehen.

InputDevice

Mit diesen Zeilen werden die Eingabegeräte der jeweiligen Screen-Sektion zugeordnet. Zumindest die Einträ für Tastatur und Maus sollten nicht fehlen.

Dem Schlüsselwort **InputDevice** muss der Name der zugeordneten InputDevice-Sektion folgen (der dortig Identifier). Werden mehrere Eingabegeräte vom selben Typ zugeordnet, sollte eines als "Haupteingabegerä gekennzeichnet werden, indem im Falle einer Tastatur als dritter Eintrag der Zeile **CoreKeyboard** und bei Maus **CorePointer** folgt. Ohne diese Angaben werden implizit die jeweils ersten Einträge des Typs als Haupteingabegerät angenommen.

Soll bspw. eine zweite Maus gleichzeitig benutzt werden, steht **SendCoreEvents** an Stelle des dritten Parameters. Sie löst dieselben Aktionen aus, wie die "default"-Maus.

Options

Hier dürfen alle Optionen stehen, die auch in **ServerFlags** erlaubt sind, wobei die hiesigen Angabe frühere überschreiben.

Das abschließende Beispiel definiert zwei ServerLayout-Sektionen; die erste für einen Monitor und eine zweite für zwei Bildschirme. Bei Start des Servers, wird dieser immer die zuerst angegebene Sektion wählen, es sei denn, er wird mit der Option "-layout *Layout_Identifier*" aufgerufen.

```
Section "ServerLayout"
  Identifier  "Einer alleine"
  Screen      "Dieser Screen"
  InputDevice "Meine Maus" "CorePointer"
  InputDevice "Meine Tastatur" "CoreKeyboard"
EndSection

Section "ServerLayout"
  Identifier  "Zwei gemeinsam"
  Screen      "Dieser Screen"
  Screen      "Jener Screen" RightOf "Dieser Screen"
  InputDevice "Meine Maus" "CorePointer"
  InputDevice "Meine Tastatur" "CoreKeyboard"
EndSection
```

Anmerkung: In der "Section Screen" müssen die Sektionen unter den Namen "Dieser Screen" und "Jener Screen" existieren (entspricht dem dortigen Identifier); das Gleiche gilt für "Meine Maus" und "Meine Tastatur", die in der "Section InputDevice" spezifiziert sein sollten.

Die Sektion DRI [4]

Die Sektion dient der Regulierung des Zugangs zur 3D-Hardware-Unterstützung. Vorraussetzung ist natürlich, dass Ihre Grafikkarte über die entsprechenden Fähigkeiten verfügt und X diese Karte vollends unterstützt. In der Datei "/usr/X11R6/lib/X11/doc/README.DRI" finden Sie eine Auflistung der von Ihrer X-Version verwendbaren Karten.

Die Schlüsselworte der Sektion sind **Mode < Rechte>** und **Group < Gruppe>** . Letztere Angabe ist optional; entfällt sie, gelten die Rechte für alle Benutzer.

Die Angabe der Rechte geschieht analog zu den Datei-Zugriffsrechten. Um also alle Benutzer am 3D-Genuss teilhaben zu lassen, genügt eine einfache Sektion:

```
Section "DRI"
  Mode  0666
EndSection
```

Den Zugang auf Gruppenbasis zu ermöglichen, erfolgt analog; sinnvoll erscheint, die berechtigten Benutzer in eine neue Gruppe aufzunehmen:

```
Section "DRI"
  Group  "group3d"
  Mode  0660
EndSection
```

Die Sektion Vendor [4]

Diese Sektion ist für herstellerspezifische Erweiterungen vorgesehen. Es bleibt abzuwarten, ob sich auch für XFree86 Version 4 kommerzielle Server-Implementierungen einstellen werden.

Zeit und zeitgesteuerte Abläufe

Übersicht

Zeit ist Geld.

Das magische Jahr 2000 entpuppte sich für die Informatikbranche als wahre Goldgrube. Viel zu spät hatten die Betreiber von Datenbanken die wahre Dimension des Problems »Jahrtausendwechsel« begriffen. Die Folge war eine fieberhafte Reaktivierung schon längst in Rente gegangener Programmierer, die die eigentlich schon als historisch einzustufenden COBOL-Quellen (u.a. Programmiersprachen) nach Datumsfeldern durchforsteten und diese von 2 Stellen auf 4 aufblähten. Jeder Entwickler weiß, was es heißt, aus tausenden Zeilen undokumentiertem Quellcode die relevanten Passagen zu extrahieren...

Linux selbst ließ dieses »Zeitproblem« relativ ruhig, da es, wie alle Unix-Systeme, für die Zeitzählung einen 32 Bit langen Zähler verwendet, der, am 01.01.1970 gestartet, erst im Jahre 2038 überlaufen wird. Bis dahin, so hofft man, wird der 32 Bit Rechner nur noch als Museumsstück überleben, und die dann verbreitete 64 Bit Zählung sollte bis »in alle Ewigkeit« reichen. Das J2K-Problem betraf also höchstens die Anwendungen unter Unix und auch hier, so lehrte die Vergangenheit, hat es keine nennenswerten Effekte gegeben.

Dennoch ist, zumindest für einen vernetzten Recher, die korrekte Zeit das A und O eines funktionierenden Systems.

Die Telefonkosten purzeln weiter, so spielt es in den heutigen Tagen nicht mehr die entscheidende Rolle, den Download des großen Datenpaketes zum billigsten Tagestarif anzuschieben. Oder doch? Oft ist das Netz tagsüber hoffnungslos überlastet, so dass sich die Übertragungsrate der Daten bei Null einpendelt. Also doch besser in den frühen Morgenstunden den FTP-Server kontaktieren!

Aber wer steht schon gern mitten in der Nacht auf? Da lässt man das System doch lieber selbsttätig die Arbeit verrichten. Weitere Beispiele für zeitgesteuerte Arbeiten lassen sich beliebig angeben und im folgenden Abschnitt werden solche auch genannt sein. Neben der Handhabung der zeitgesteuerten Abläufe, soll auch das Stellen der Systemzeit Thema dieses Textes sein.

Zuvor noch die Erläuterung zweier Begriffe: Die so genannte **Hardware-Zeit** ist die Zeit, die im BIOS des Rechners gespeichert ist. Sie ist batteriegepuffert und läuft auch weiter, wenn der Computer ausgeschaltet ist. Im Normalfall handelt es sich hierbei um die lokale Zeit des Standorts und leider ist es auch die Norm, dass diese Zeit - dank fernöstlicher Billigquarze - einer permanenten Gangungenauigkeit unterliegt. Die **Systemzeit** ist letztlich die Zeit, die Linux für seine Arbeit verwendet. Beim Systemstart wird sie anhand der Hardware-Uhr gesetzt und ggf. nach verschiedenen Methoden korregiert. Einige Werkzeuge setzen nur die Systemzeit, diese hat aber keinen Einfluss auf die Zeit im BIOS!

Prozess zu bestimmter Zeit starten - at

Eine Beschreibung zur Anwendung von **at** und seiner verwandten Kommandos **atq atrm** und **batch** finden Sie im Abschnitt Weiteres des Kapitels »Nutzerkommando«. Wir beschränken uns hier auf die administrativen Aspekte.

Start des Dämonen

Für die Bearbeitung der Jobs ist der at-Dämon **atd** verantwortlich, der bereits während des Systemstarts aktiviert

werden sollte. Mit Hilfe des Kommandos ps sollte ein Zeichen des Geistes zu erhalten sein:

```
user@sonne> ps ax | grep atd
 232 ?        S      0:00 /usr/sbin/atd
 518 ?        S      0:00 /usr/sbin/rpc.rstatd
4316 ?        S      0:00 /usr/sbin/rpc.kstatd
1967 pts/8    S      0:00 grep atd
```

Fehlt eine den Dämonen betreffende Zeile, kann dieser von Hand gestartet werden. Die interessanten Optionen sind hierbei **-l Auslastungsfaktor** und **-b Sekunden**. Der **atd** ist so ausgelegt, dass er einen Auftrag nur zur vereinbarten Zeit startet, wenn die Systemauslastung einen bestimmten Wert (0.8) unterschritten hat. Ist das System zu diesem Zeitpunkt zu beschäftigt, stellt der **atd** den Auftrag solange zurück, bis der Lastwert unter die Schranke gefallen ist. Mit ersterer Option kann ein anderer als der voreingestellte Wert vereinbart werden, dies ist bei Mehrprozessorsystemen zu empfehlen. Die momentane Auslastung ermittelt der Dämon anhand der Datei »/proc/loadavg«. Mit der zweiten genannten Option kann das Zeitintervall variiert werden, indem der **atd** die Auftragswarteschlangen nach fälligen Jobs durchsucht. Voreinstellung ist 60 Sekunden.

Um den Dämonen schließlich nicht immer von Hand starten zu müssen, sollten Sie einen Link in allen Runleveln auf ein entsprechendes Skript anlegen. Bei den meisten Distributionen sollte das Skript »at« heißen und im Verzeichnis unterhalb der Runlevel liegen.

- Debian und RedHat: Verzeichnis /etc/rc.d/init.d
- SuSE: Verzeichnis /sbin/init.d

Zugangskontrolle

Der Zugang zum Dienst kann vom Systemverwalter eingeschränkt werden, in dem er die Benutzerkennzeichen in einer der Dateien »/etc/at.allow« oder »/etc/at.deny« aufnimmt. Dabei wird die zweite Datei nur ausgewertet, falls erstere fehlt. In »at.allow« trägt Root genau die Benutzer ein, die den Dienst beanspruchen dürfen. Nichterwähnte Benutzer werden somit ausgeschlossen. Sollen jedoch nur einige wenige Personen ausgenommen werden, so ist es einfacher, diese in der Datei »at.deny« einzutragen. Alle dort nicht benannten Benutzer haben dann Zugriff auf **at**. Existiert keine Datei, ist der Dienst für jeden nutzbar.

Wiederkehrende Abläufe mit cron

Schon die Standardinstallation von Linux birgt für manchen Benutzer Überraschungen. Da fährt man nichtsahnend sein System hoch, freut sich auf eine gute Performance und wundert sich, dass die CPU am Limit kreiselt, obwohl man doch nur seine Mails begutachtet. Geht man der Ursache auf den Grund, findet man heraus, dass »nobody« das Kommando »find« ausführt und sämtliche Ressourcen zu verschlingen scheint. Wer ist dieser »nobody«? Und wie kommt er dazu, meinen Dateibaum zu durchstöbern?

Erst einmal sei der Leser beruhigt; in jenem Fall ist es unwahrscheinlich, dass sich hinter »nobody« ein arglister Eindringling verbirgt. Hier ist vermutlich irgendein Systemprogramm am werkeln, das irgendeine Datenbank aktualisiert.

Und wer rief das Programm? Mit ziemlicher Sicherheit identifiziert man den **cron** als den Übeltäter. Und »Übeltäter« ist für diesen Pünklichkeitsfanatiker sicher die unpassendste Bezeichnung, er verrichtet schließlich nur gewissenhaft die ihm angetragenen Aufgaben.

Beim **cron** handelt es sich um einen Dämonen, auch wenn sein Name nicht mit dem sonst üblichen »d« endet. Warum dieser Name? Wer weiß... schieben wir es seinem Entwickler *Paul Vixie* in die Schuhe. Dieser **cron** sollte schon während des Systemstarts aktiviert werden. Fortan wird er minütlich zum Leben erwachen und in seinen Terminkalendern nachschlagen, ob etwas zu erledigen ist. Wenn nicht, versetzt er sich für eine weitere Minute in den Schlaf.

Der Start des Dämonen

In den meisten Fällen sollte nach der Linuxinstallation der Dämon bereits mit dem Start des Systems aktiv werden.

Die verschiedenen Distributionen regeln das über ein Skript, das in allen Runleveln gestartet wird. Debian speichert das Skript unter »/etc/init.d/cron«, bei RedHat liegt es als »/etc/rc.d/init.d/cron« auf der Platte und SuSE hält »/sbin/init.d/cron« für das richtige Konzept. Sollte bei Ihnen der **cron** nicht aktiv sein, so überprüfen Sie, ob in den jeweiligen Runlevel-Verzeichnissen ein Link auf das Skript existiert.

Mit »/usr/sbin/cron« vermag der Systemadministrator den Dämon von Hand ins Leben zu berufen, da dessen Arbeit allerdings von unschätzbarem Nutzen ist, sollten ggf. die fehlenden Links erzeugt werden. Als Linknamen bieten sich »S21cron« für das Startskript und »K19cron« für das Stoppskript an. Wo sich die Runlevel-Verzeichnisse befinden und was es mit der Namensgebung auf sich hat, ist Thema des Abschnitts Bootvorgang.

Mehrere Terminkalender

Ist der Geist erst einmal losgelassen, so schaut er als erstes in der Datei »/etc/crontab« nach, welchen Spuk er im System als nächstes zu treiben hat. Und beim weiteren Vorgehen scheiden sich die Geister... Hier treiben die verschiedenen Distributionen allerlei eigenen »Unsinn« und kompilieren die verschiedensten Suchpfade in den Code des Dämons ein.
Entsprechend den Richtlinien des Filesystem Hierarchie Standards sollten die benutzereigenen Tabellen im Verzeichnis »/var/spool/cron« angesiedelt sein, bei Debian und RedHat findet man sie unter »/var/spool/cron/crontabs« und SuSE liegt mit »/var/cron/tabs« völlig daneben. Die letzten Dateien, die zum üblichen Auftragsvolumen des **cron** gehören, sind »cron.hourly«, »cron.daily«, »cron.monthly« und »cron.weekly«, die distributionsabhängig entweder unterhalb von »/etc/cron.d« oder direkt in »/etc« (SuSE) liegen.

Zu bemerken ist, dass die Dateien nicht existieren müssen, da alle Aufträge für den Dämon auch in einer einzigen Datei enthalten sein könnten. Und außerdem sind die zu überwachenden Dateien und Pfade durch Herumspielen im Quellcode beliebig anpassbar.

Der Eine darf, der Andere nicht...

Dem Systemadministrator steht es frei, bestimmte Benutzer von den Diensten des **cron** auszuschließen. Hierzu kann er die Benutzerkennzeichen in die Dateien »allow« und »deny« aufnehmen. Jeder Benutzer, der in der allow-Datei enthalten ist (ein Benutzer je Zeile), darf eine eigene »crontab« anlegen. Im Falle einer leeren Datei »allow« kann also niemand den **cron** benutzen. Existiert die »allow«-Datei nicht, kommt »deny« ins Spiel. Sie enthält genau jene Benutzer, denen der Dienst versagt wird.

allow und **deny** sind hier nur beispielhafte Bezeichnungen für die Dateien. SuSE und Debian benennen sie tatsächlich so, im ersten Fall liegen sie im Verzeichnis »/var/cron« und bei Debian unter »/var/spool/cron«. RedHat regelt den Zugriff über die beiden Dateien »/etc/cron.allow« und »/etc/cron.deny«.

Die Datei crontab

Wie muss nun der Inhalt einer crontab-Datei aussehen?

Eine Zeile, die nicht mit dem Doppelkreuz (»#« - Kommentar) beginnt, ist entweder eine Variablenzuweisung oder eine Anweisung der Art: »Starte das Kommando zur dieser Zeit an diesem Datum«. Jede Zeile entspricht einem Eintrag, d.h. eine Anweisung oder Variablenzuweisung darf sich nicht über mehrere Zeilen erstrecken. Auch ist die Zeile auf 1024 Zeichen begrenzt. Beginnt eine Anweisungszeile mit einem Minus »-«, so wird kein syslog-Eintrag zur Protokollierung des Kommandostarts vorgenommen.

Eine **Variablenzuweisung** legt Umgebungsvariablen für den **cron** neu fest. So lassen sich dem Dämon mit **PATH** alternative Pfade angeben, wo er die Kommandos zu suchen hat. Mit **SHELL** kann eine von der »/bin/sh« abweichende Shell zur Kommandoausführung benannt werden und mit **MAILTO** ist es möglich, die Ausgabe, die der **cron** per Mail an den Eigentümer der »crontab« ausliefert, einem anderen Benutzer zukommen zu lassen (oder gar zu unterdrücken »MAILTO=«).

Eine **Anweisung** besteht aus 7 Feldern (in den privaten Tabellen nur 6), wobei diese durch Leerzeichen oder Tabulatoren getrennt sind. Die ersten 5 Felder betreffen die Angabe des Zeitpunkts, wann das Kommando zu starten ist:

Feld 1: Minute

0-59

Feld 2: Stunde

0-23

Feld 3: Tag

0-31

Feld 4: Monat

0-12, jan, feb, ..., dec

Feld 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 auf Minuten:

	Syntax	Beispiel/ Bemerkung
Voller Bereich	*	0, 1, 2, ..., 59
Ausgewählter Bereich	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.

Die Datei »/etc/crontab« enthält nun als 6. Feld den Benutzer, in dessen Auftrag das Programm auszuführen ist. Bei allen anderen Tabellen entfällt der Eintrag, da die enthaltenen Kommandos immer mit den Rechten des Eigentümers der Datei gestartet werden.

Das letzte Feld beinhaltet in jeder »crontab« das auszuführende Kommando. Ein Prozentzeichen »%« kann benutzt werden, um einen Zeilenumbruch zu simulieren, alle folgenden Daten werden dem Kommando als Argumente übergeben.

Die Datei »/etc/crontab« könnte wie folgt ausschauen:

```
root@sonne>  cat / etc/ crontab
SHELL=/bin/sh
PATH=/usr/bin:/usr/sbin:/sbin:/bin:/usr/lib/news/bin
MAILTO=root
# Langwierige Jobs sollten besser Nachts ausgeführt werden...
# Um 21.00 Uhr soll die Warteschlange der Faxe bearbeitet und geleert werden
 0 21 * * *     root  test -x /usr/sbin/faxqclean && /usr/sbin/faxqclean
# Reports über das Faxgeschehen sind um 23.25 Uhr zu generieren
25 23 * * *     root  test -e /usr/sbin/faxcron && sh /usr/sbin/faxcron | mail FaxMaster

# check scripts in cron.hourly, cron.daily, cron.weekly and cron.monthly
#
# Das nächste Kommando soll alle 15 Minuten starten und nicht protokolliert werden
-*/15 * * * *    root  test -x /usr/lib/cron/run-crons && /usr/lib/cron/run-crons
# 0.00 Uhr jeden Tag
 0 0  * * *      root  rm -f /var/cron/lastrun/cron.daily
# 0.00 Uhr jeden Sonntag
```

```
0 0  * * 7     root  rm -f /var/cron/lastrun/cron.weekly
# 0.00 Uhr jeden ersten Tag im Monat
0 0  1 * *     root  rm -f /var/cron/lastrun/cron.monthly
```

Benutzereigene crontab

Bei all der Pingelichkeit mit der Sicherheit in einem Unix-System wird sich niemand darüber wundern, dass Otto-Normalverbraucher seine cron-Jobs nicht direkt in das entsprechende Spoolverzeichnis schreiben kann. Aus diesem Grund steht ihm das Kommando **crontab** zur Verfügung.

Dem Systemadministrator steht es zu, die nachfolgenden Optionen mit **-u Benutzer** zu koppeln und somit die Datei eines beliebigen Benutzers zu bearbeiten. Die simplen Optionen sind **-l** zur Anzeige des Inhalts seiner eigenen Tabelle und **-r** zum Löschen jener. Zum Editieren der Datei ist das Kommando mit der Option **-e** zu starten. Kommt jetzt die Ausgabe:

```
user@sonne> crontab -e
You (user) are not allowed to use this program (crontab)
Contact your sysadmin to change /var/cronallow or /var/crondeny
See crontab(1) for more information
```

...so trete man seinem Systemadministrator entgegen und frage ihn, warum man von der Benutzung des Dienstes ausgeschlossen wurde.

Gesetzt den Fall, der Aufruf glückte, so findet man sich in einem Editor wieder. Welcher das ist, steht in den Shellvariablen $VISUAL und $EDITOR (nur wenn erstere nicht gesetzt ist, wird die 2. ausgewertet) und kann nach persönlichen Wünschen angepasst werden. Übrigens scheitert ein Kommandoaufruf auch, wenn die Variable nicht oder auf einen nicht installierten Editor gesetzt ist.

Beispiel: Einmal pro Woche (Mittwoch) sollten wir ein Backup unseres Heimatverzeichnisses in Erwägung ziehen. Wir archivieren also alle Dateien und schicken das Archiv per Mail an eine Adresse. Zusätzlich soll die normale Nachricht über die erfolgte Ausführung des Kommandos unterdrückt werden.

```
user@sonne> crontab -e
MAIL=
0 0 * * 3 tar czvf - /home/user | uuencode backup.tgz | mail user@outside.all -s "Backup"
```

Nach dem Abspeichern der Datei meldet **crontab** entweder den Vollzug »crontab: installing new crontab« oder aber einen aufgetretenen Fehler. Wir provozieren einen solchen, indem wir einen Zeilenumbruch angeben:

```
user@sonne> crontab -e
*/30 14-16 * * 1-5 echo "Feier
abend"
Speichern der Datei
crontab: installing new crontab
"/tmp/crontab.2171":2: bad minute
errors in crontab file, can't install.
Do you want to retry the same edit?
```

crontab weist uns auf die mögliche Fehlerursache hin (Zeile 2:bad minute). Klar, dass »abend« keine gültige Minutenangabe ist. Weiterhin verweigert das Kommando das Abspeichern der Datei und bietet ein erneutes Bearbeiten an. Durch Eingabe von »y« gelangen wir zurück zum Editor.

Systemzeit ändern mit date

Die Verwendung des Kommandos zur Anzeige der Systemzeit wurde bereits im Abschnitt Weiteres zu Nutzerkommandos demonstriert, jetzt stellen wir Ihnen die Möglichkeiten zum Stellen der Uhr vor.

Dem Systemverwalter stehen zwei Varianten zur Eingabe der neuen Systemzeit zur Verfügung. Als erstes kann er die Zeit als einzige Zahl angeben, wobei die Reihenfolge der Ziffern wie folgt festgeschrieben ist:

```
<Monat><Tag><Stunde>[<Jahreszahl (2 Stellen)>][<Jahreszahl (2 Stellen)>][<Sekunden>]
```

Die drei ersten Angaben sind verbindlich, die weiteren werden, sodenn sie fehlen, durch die aktuellen Einstellungen ersetzt. Möchten Sie die Sekunden neu einstellen, so ist die Angabe der Jahreszahl mit vier Stellen erforderlich. Wählen Sie aus den 3 optionalen Parametern nur einen aus, wird dieser als Jahreszahl des aktuellen Jahrhunderts interpretiert.

Das nächste Beispiel setzt die Zeit auf den 01.01.2000, 12.00 Uhr:

```
root@sonne> date 010112002000
Sam Jan    12:00:00 MET 2000
```

Etwas verständlicher ist da die Verwendung eines Formatstrings, so wie es die Option **-s** ermöglicht. Welche Platzhalter enthalten sein können, wurde bereits im Abschnitt Weiteres zu Nutzerkommandos vorgestellt. Der folgende Aufruf setzt die Zeit wie im letzten Beispiel beschrieben:

```
root@sonne> date -s "Jan 1 12:00:00 2000"
Sam Jan  1 12:00:00 MET 2000
```

Warum gleich die ganze Eingabe vornehmen, wenn man die Uhr nur um ein paar Minütchen korregieren will? Hier kann das Schlüsselwort **now**, gefolgt von der Zeitspanne um die die Uhr vorgestellt werden soll, eingesetzt werden. Um bspw. die Systemzeit um 20 Minuten vor zu stellen, nützt diese Eingabe:

```
root@sonne> date; date -s "now 20 min"
Sam Jan  1 12:02:07 MET 2000
Sam Jan  1 12:22:07 MET 2000
```

Um nach obigem Vorgehen die Zeit zurückzustellen, ist dem Formatstring das Schlüsselwort **ago** nachzustellen:

```
root@sonne> date; date -s "now 20 sec ago"
Sam Jan  1 12:24:17 MET 2000
Sam Jan  1 12:23:57 MET 2000
```

Hardwareuhr setzen mit hwclock

Um Missverständnissen vorzubeugen: Unter Linux sind die beiden Kommandos **clock** und **hwclock** identisch, d.h. das Programm selbst ist »hwclock« und »clock« ist als Link auf dieses realisiert.

Mit **hwclock** lässt sich die Hardwareuhr auslesen und setzen. Das Kommando vermag sowohl die Systemzeit der Hardwarezeit anzupassen als auch umgekehrt, außerdem ermöglicht es mit Hilfe der Datei /etc/adjtime eine »automatische« Zeitkorrektur.

Die Option **--show** zeigt die **lokale** Hardwarezeit des Rechners an. Wird zusätzlich die Option **--utc** gewählt, handelt es sich um die Standardweltzeit (wenn die Hardwareuhr nach letzterer läuft, sind die beiden Angaben identisch):

```
root@sonne> hwclock --show
Mon Jun 26 13:10:41 2000 -0.811165 seconds
```

Um die Systemzeit nach der Hardwareuhr zu stellen, ist die Option **--hctosys** zu benutzen. Gleichzeitig wird ggf. die Zeitzone angepasst. Die entgegen gesetzte Richtung ermöglicht die Option **--systohc**, d.h. das Stellen der Hardwareuhr nach der Systemzeit.

Mit der Option **--set** kann eine beliebige gültige Zeit eingestellt werden. Die Option verlangt einen Formatstring, der mittels **--date= "Zeit"** anzugeben ist. Die Syntax der Zeichenkette entspricht dabei der des Kommandos »date -s Zeit«. Um die Hardwareuhr 23 Stunden in die Zukunft zu versetzen, gibt man z.B. Folgendes ein:

```
root@sonne> hwclock --set --date= "now 23 hours"
```

Einige Parameter von **hwclock** sind architekturabhängig und sie werden eher selten benötigt. Abschließend soll die Option **--debug** genannt sein, die bei der Fehlersuche recht hilfreich sein kann:

```
root@sonne> hwclock --debug
hwclock 2.4c/util-linux-2.10f
Using /dev/rtc interface to clock.
Last drift adjustment done at 962023508 seconds after 1969
Last calibration done at 962023508 seconds after 1969
Hardware clock is on UTC time
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
...got clock tick
Time read from Hardware Clock: 12:45:15
Hw clock time : 100/06/26 12:45:15 = 962023515 seconds since 1969
Mon Jun 26 14:45:15 2000 -0,484035 Sekunden
```

Die Datei / etc/ adjtime

Auch wenn die ständige Abweichung der Hardwareuhr sehr unbefriedigend ist, so ist ihr doch etwas Gutes abzugewinnen: Sie ist relativ konstant, d.h. wenn die Uhr an einem Tag um 2 Sekunden vorgeht, dann eilt sie der realen Zeit in einer Woche um 14 Sekunden und in einem Jahr (kein Schaltjahr) um 12 Minuten und 10 Sekunden voraus.

Es liegt also nahe, diese kalkulierbare Größe zur automatischen Zeitkorrektur zu verwenden. **Welche Informationen muss man kennen?** Zum einen die **tägliche Abweichung** und zum anderen den **Zeitpunkt der letzten Korrektur**. Beide Werte u.a.m. stehen in der Datei **/ etc/ adjtime**.

Die Datei /etc/adjtime besteht aus nur **drei Zeilen**:

1. Zeile: Hier stehen drei Werte, zuerst die tägliche Abweichung in Sekunden (als Gleitkommazahl), als nächstes der Zeitpunkt der letzten Korrektur (Angabe seit 01.01.1970) in maschinenlesbarer Form und zuletzt der Wert »0« , der nichts zu sagen hat und nur aus Kompatibilitätsgründen enthalten ist
2. Zeile: Zeitpunkt der letzten Korrektur (Angabe seit 01.01.1970) in maschinenlesbarer Form oder »0« , falls noch keine Korrektur stattfand
3. »LOCAL« oder »UTC« , je nachdem, ob die Hardwareuhr auf lokale Zeitzone oder »Coordinated Universal Time« eingestellt ist

Sie wissen hoffentlich, wie man die Zeit in einem für Maschinen verständlichen Format angibt? Nein? Dann können Sie sich glücklich schätzen, dass eine manuelle Bearbeitung der Datei /etc/adjtime nicht notwendig ist. Vollziehen Sie einfach folgende Schritte:

1. Ein Aufruf von **hwclock** mit der Option **--systohc** stellt die Hardwareuhr auf die aktuelle Systemzeit und aktualisiert (bzw. legt an) gleichzeitig die Datei /etc/adjtime. Stellen Sie zuvor sicher, dass die Systemzeit stimmt.

   ```
   root@sonne> hwclock --systohc
   root@sonne> cat / etc/ adjtime
   0.000000 962005907 0.000000
   962005907
   LOCAL
   ```

2. Lassen Sie ein paar Tage verstreichen (kümmern sich bspw. mal wieder um Ihre Familie) - je länger der Zeitraum, desto aussagekräftiger die ermittelte Abweichung - und berechnen die Abweichung der

Hardwareuhr von der realen Zeit. Setzen Sie die Zeit nun neu, indem Sie »hwclock« mit der Option **-set --date= »richtige Zeit«** aufrufen, um z.B. um 23 Sekunden vorzustellen, geben Sie Folgendes ein:

```
root@sonne> hwclock --set --date= "now 23 sec"
root@sonne> cat / etc/ adjtime
2.300000 962016844 0.000000
962016844
LOCAL
```

3. Um in Zukunft die Zeit zu korregieren, genügt ein Aufruf von:

```
root@sonne> hwclock --adjust
```

Wenn Sie diesen Aufruf in eines der Bootskripte eintragen, wird die Zeit automatisch bei jedem Systemstart angepasst.

Die Zeit aus dem Netz

Thematisch gehört dieser Punkt in den Bereich des Netzwerks, aber wegen der Einfachheit sei er dem Abschnitt über die Zeit angegliedert.

Bei einem einzelnen Rechner spielt es letztlich keine entscheidende Rolle, wie exakt die Hardwareuhr tickt. Der Benutzer wird die auf seinem Desktop eingeblendete Uhrzeit anhand seiner Erfahrungen auf die korrekte Zeit interpolieren.

Bei der Anbindung an ein Netzwerk sieht es schon anders aus... Sendet man eine Mail über den lokalen Maildämon, so erhält die ausgehende Nachricht die aktuelle Systemzeit als Stempel. Beim Empfnger heißt das, dass die Mail in die Empfangsliste gemäß ihres Sendedatums eingeordnet wird (die meisten Benutzer ordnen die Nachrichten nach dem Datum). Angenommen, die lokale Systemzeit hinkt der realen Zeit drastisch hinterher, so kann die Mail in der Anzeige zwischen den »alten« Nachrichten »untergehen«.
Viel drastischer äußert sich das Problem, wenn eine Datei von verschiedenen Leuten an verschiedenen Orten bearbeitet wird. Womöglich wird eine alte Version für die aktuellere gehalten, nur weil deren Zeitstempel dies nahe legt. Und das, weil die Uhr des einen Rechners ein »paar« Stunden voraus eilte...

In einem Netzwerk also ist der Wunsch nach Synchronisation der einzelnen Rechner untereinander wesentlich ausgeprägter als an der heimischen Spielekonsole. Die Lösung liegt im **Einrichten eines Zeitservers**, also eines Rechners, dessen Zeit als aktuell angesehen wird (z.B. weil er eine Funkuhr besitzt) und dem periodischen Abgleich der Zeiten zwischen Server und Clients.

Einen Zeitserver einrichten

Wie schon angedeutet, handelt es sich hierbei um die Konfiguration eines Netzwerkdienstes. Wenn Sie einige Schritte nicht verstehen sollten, so informieren Sie sich in den Netzwerkkapiteln.

Als erster Schritt sind die entsprechenden Dienste zu aktivieren. Hierzu sind in der Datei /etc/inetd.conf folgende 4 Zeilen zu entkommentieren (das Doppelkreuz am Beginn der Zeilen ist zu entfernen):

```
root@sonne> vi / etc/ inetd.conf
...
daytime      stream  tcp    nowait  root    internal
daytime      dgram   udp    wait    root    internal
...
time    stream  tcp    nowait  root    internal
time    dgram   udp    wait    root    internal
...
```

Als zweiter Schritt ist der **inetd** neu zu starten:

```
root@sonne> killall -HUP inetd
```

Der Rechner ist nun als Zeitserver aktiv.

Die Clients

Auf Clientseite sind zur Synchronisation genau zwei Befehle auszuführen:

```
root@erde> / usr/ sbin/ netdate -v sonne.galaxis.de
sonne.galaxis.de -3446.062 Tue Jun 20 08:01:58.000 at 08:59:24.062 delay +0.058
root@erde> / sbin/ hwclock -wu
```

Um den Vorgang z.B. bei jedem Rechnerstart automatisch auszuführen, könnten Sie die Befehle in ein kleines Skript fassen und dieses an entsprechender Stelle des Bootvorgangs automatisch starten lassen.

Hinweis: Im Internet existieren eine Reihe frei zugänglicher Zeitserver (z.B. wrzx03.rz.uni-wuerzburg.de), die Sie zum Abgleich ihres eigenen Zeitservers verwenden können (falls dessen Uhr nicht richtig tickt).

Zugriffsrechte

Was schon bekannt sein sollte...

Wer dem Buch durch die einzelnen Kapitel folgte, hat schon Bekanntschaft mit den wichtigsten Zugriffsmechanismen unter Unix geschlossen. Diese Rechte betreffen (die Darstellung entspricht der ersten Kolonne des Kommandos `ls -l`):

- `-rwxrwxrwx` den Eigentümer,
- `-rwxrwxrwx` die besitzende Gruppe und
- `-rwxrwxrwx` alle anderen.

Ein **r** in der entsprechenden Dreiergruppe, erlaubt das Lesen für den betreffenden Nutzerkreis, ein **w** steht für die Befugnis, die Datei zu ändern und das **x** erlaubt das Ausführen der Datei bzw. - im Falle von Verzeichnissen - die Erlaubnis, in jenes zu wechseln.

Ziel dieses Abschnitts

Zunächst werden wir zwei spezielle Zugriffsrechte kennen lernen, die für gewisse Anwendungsfälle die üblichen Rechte "verbiegen". Des Weiteren werden Sie in der Lage sein, die Zugriffsrechte Ihrer Dateien nach Ihren Wünschen zu gestalten. Ebenso in diesem Abschnitt behandeln wir Dateiattribute, die einen anderweitigen Schutz der Daten gewährleisten.

Nach dem Studium dieses Abschnitts wissen Sie:

Welche 5 Zugriffsrechte existieren
Wie Sie Zugriffsrechte ändern
Warum neue Dateien/Verzeichnisse offensichtlich immer dieselben Rechte erhalten
Was Dateiattribute sind
Wie Sie Ihre Daten vor versehentlichem Modifzieren schützen können

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 in diesem Fall für »save

program text on swap device« und bewirkt, dass in dieses Verzeichnis zu schreibende Daten so lange 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 zeitaufwändige Schreiboperationen vermieden werden. Wird das Flag auf ein normales Verzeichnis angewandt, darf dessen Besitzer Dateien anderer Benutzer aus diesem Verzeichnis nicht löschen (das trifft auch auf /tmp! zu). In diesem Zusammenhang spricht man auch vom »Sticky Bit«.

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 Benutzer 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 Benutzerkennung von user erhält.

Mit Hilfe des s-Flags kann die Benutzerkennung des Prozesses vom rufenden Benutzer/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 Binärdateien gesetzt werden (keine Shellskripte).

Chmod -Symbolischer Modus

Zum Ändern der Zugriffsrechte steht das Kommando chmod zur Verfügung:

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

Für die möglichen Optionen sei auf die Manuals verwiesen.

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
-	Recht(e) entfernen
=	Genau diese Recht(e) setzen

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
```

Chmod -Numerischer Modus

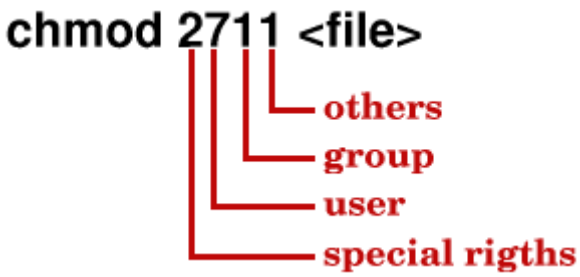


Abbildung 1: Nummerischer Modus von chmod

Rechte werden bitweise dargestellt, für Eigentümer, Gruppe und Andere.

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 übergeben (Abbildung). 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-Flag	0000 0001	1
s-flag der Gruppe	0000 0010	2
s-Flag 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
```

Voreinstellung für neue Dateien

Neue Dateien/Verzeichnisse werden offensichtlich stets mit ein und denselben 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 so genannte 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:

	Verzeichnisse	Dateien
Maximalmaske	777	666
umask	022	022
Ergebnis	755	644

Mit dem Kommando umask lässt sich die Voreinstellung ändern.

Besitzer ändern

Wird eine Datei neu erzeugt, so ist ihr Eigentümer immer derjenige, der die Aktion veranlasste und die Datei gehört zur Default-Gruppe des Benutzers. Zum Ändern der Werte dienen die Kommandos **chown** (change owner) und **chgrp** (change group).

Den **Besitzer** einer Datei **darf einzig Root ändern**. Wäre dem nicht so, könnten hintertriebene Hacker einem beliebigen Benutzer ein modifziertes Programm unterschieben, das dann, mit dessen Rechten gestartet, ggf. allerlei Schindluder im System treiben könnte.
Etwas liberaler geht es bei Gruppen zu, wo der **Besitzer** Dateien beliebig **zwischen Gruppen, denen er angehört**, hin und her jonglieren kann.

Beginnen wir mit dem Kommando **chgrp**:

Aufruf: chgrp [OPTIONEN]... GRUPPE DATEI...

Um die Gruppenzugehörigkeit einer Datei zu wechseln, geben wir Folgendes ein:

```
user@sonne> ls -l datei
-rw-r--r--   1  user    users       120 May 30 11:25   datei
user@sonne> chgrp fibel datei
user@sonne> ls -l datei
-rw-r--r--   1  user    fibel       120 May 30 11:25   datei
```

Das Beispiel funktioniert allerdings nur, wenn "user" Mitglied in den Gruppen "users" und "fibel" und Eigentümer von "datei" ist. "chgrp" kann auch auf Verzeichnisse angewandt werden, in dem Fall ist die Option "-R" hilfreich, um rekursiv alle enthaltenen Dateien/Verzeichnisse zu bearbeiten.

Das Kommando ändert, wird es auf einen symbolischen Link angewandt, die Datei, auf die der Link verweist. Mit der Option "-h" lässt sich **zusätzlich** die Gruppe des Links modifizieren (ab Linux-Kernel 2.2).

```
user@sonne> ls -l bla datei
lrwxrwxrwx   1  user    users         5 May 30 11:43   bla -> datei
-rw-r--r--   1  user    users       120 May 30 11:25   datei

user@sonne> chgrp fibel bla
user@sonne> ls -l bla datei
lrwxrwxrwx   1  user    users         5 May 30 11:43   bla -> datei
-rw-r--r--   1  user    fibel       120 May 30 11:25   datei

user@sonne> chgrp users datei
user@sonne> chgrp -h fibel bla
user@sonne> ls -l bla datei
lrwxrwxrwx   1  user    fibel         5 May 30 11:43   bla -> datei
-rw-r--r--   1  user    fibel       120 May 30 11:25   datei
```

Mit **chown** kann sowohl der Besitzer einer Datei als auch deren Gruppenzugehörigkeit gesetzt werden.

Aufruf: chown [OPTIONEN]... BESITZER[.[GRUPPE]] DATEI...

Die wichtigen Optionen arbeiten analog zu "chgrp"; also kann "-R" verwendet werden, um bei Verzeichnissen rekursiv deren Besitzer / besitzende Gruppe zu modifzieren. Im Unterschied zu "chgrp" bewirkt die Anwendung auf symbolische Links die Änderung deren Werte.

Soll nun nur der Besitzer geändert werden, wird dieser einfach angegeben:

```
root@sonne> chown newbie datei
```

Soll die Gruppe gesetzt werden, ist dem Gruppennamen ein (Doppel)Punkt voranzustellen:

```
root@sonne> chown .fibel datei
```

Wünscht man sowohl den Besitzer als auch die Gruppe zu wechseln, sind beide Namen durch einen Punkt oder Doppelpunkt voneinander zu trennen:

```
root@sonne> chown newbie:fibel datei
```

Gibt man den neuen Besitzer an, gefolgt von einem (Doppel)Punkt, so wird als neue Gruppe dessen Defaultgruppe aus der Datei /etc/passwd verwendet.

Dateiattribute

Die Anwendung von Attributen funktioniert nur auf dem Linux-Dateisystem "ext2".

Selbst der erfahrene Linuxer ist vor Fehlern nicht gefeit. Man stelle sich nur vor, man wollte alle Dateien aus einem Unterverzeichnis löschen:

```
user@sonne> rm -rf ~/ DirToDelete/ *
```

Das Beispiel sieht harmlos aus, birgt aber einen fatalen Fehler in sich: das Leerzeichen zwischen Pfadangabe und dem *. Verschwunden sind nicht die gewünschten Daten, sondern der komplette Inhalt des aktuellen Verzeichnisses (falls man die Berechtigung dazu besaß).

Linux bietet bekannterweise keine zuverlässige Methode des Wiederherstellens gelöschter Daten. Dafür ermöglicht das Linux-Dateisystem ext2 die Vergabe von Attributen für Dateien und Verzeichnisse. Sicher wird man aus Bequemlichkeit meist auf diese verzichten, aber essentielle Daten lassen sich somit zuverlässig schützen. Zum Setzen von Attributen dient das Kommando chattr, anzeigen lassen sie sich mittels lsattr:

a

Die Datei kann weder gelöscht noch deren bisheriger Inhalt verändert werden. Allerdings ist das Anhängen Daten möglich. Dieses Attribut darf nur Root setzen.

```
user@sonne> lsattr test.txt; ls test.txt
-------- test.txt
-rw-r--r--   1 user   users        32 Dec 19 19:52   test.txt
root@sonne> chattr +a test.txt
user@sonne> rm test.txt
rm: Entfernen von ¯test.txt½ nicht möglich: Die Operation ist nicht erlaubt
user@sonne> cat < test2.txt >> test.txt; ls -l test.txt
-rw-r--r--   1 user   users        64 Dec 19 19:53   test.txt
```

d

Eine so gekennzeichnete Datei wird beim Sichern mit dump nicht berücksichtigt.

i

Die Datei kann weder umbenannt noch modifiziert noch gelinkt werden. Nur Root darf das Attribut setzen. Wird dieses Attribut für Verzeichnisse gesetzt, darf in diesem Verzeichnis keine Datei gelöscht werden, jedo ist eine Änderung dieser und das Erzeugen neuer Dateien möglich.

s

Eine mit diesem Attribut versehene Datei wird beim Löschen mit 0-Bytes überschrieben.

S

Wird die Datei modifiziert, so wird sie unmittelbar auf die Festplatte zurückgeschrieben, also ohne Zwischenpufferung.

Es existieren weitere Attribute, die aber in der momentanen Implementierung des ext2 nicht unterstützt werden.

Das X Window System

Überblick

Apple revolutionierte den Desktop. Microsoft machte ihn bekannt. Aber der Ruhm der ersten einsatzfähigen grafischen Oberfläche gebührt X Window.

Nicht zuletzt dank der in den letzten Jahren beschleunigten Bestrebungen um moderne Desktop-Umgebungen gewinnt Linux mehr und mehr als Arbeitsumgebung für den »gewöhnlichen« Anwender an Bedeutung. Galt das System lange Zeit als »schwer konfigurier- und bedienbar«, so überraschen den Neuling die Benutzerfreundlichkeit aktueller Distributionen.

Auch wenn die Oberflächen sich ändern, so steckt hinter all den bunten Fenstern immer noch das X-Protokoll, erweitert zwar um neuere Möglichkeiten, aber im Grunde genommen arbeitet es immer noch nach gleichem Prinzip. Damit ist das Protokoll zwar erprobt, quält sich allerdings auch mit allerlei Ballast herum, der die Unterstützung moderner Grafikalgorithmen erschwert oder ganz und gar verhindert. Aber auch hier brüten schlaue Köpfe über neue Protokolle...

Was ist X? Dieser Frage gehen wir in den nächsten Abschnitten nach. Ein Ausflug in die Vergangenheit gehört ebenso dazu wie eine Diskussion der Eigenschaften und der Blick hinter die Kulissen.

Wobei wir bei der Konfiguration wären. Nicht das Einrichten des für die Darstellung benötigten Servers soll behandelt sein (diese finden Sie im Abschnitt Systemadministration), sondern die speziellen Möglichkeiten zum grafischen Login, die Darstellung grafischer Anwendungen über Rechnergrenzen hinweg usw.

Alles Bisherige betrifft genau genommen die »nicht sichtbare« Ebene. Die Art und Weise der Darstellung selbst organisiert ein so genannter Windowmanager. Aktuell mehren sich die Anzeichen, dass die GNOME-Architektur einmal der Standard für Unix-Systeme wird. Aber noch läuft die Entwicklung des KDE auf vollen Touren und ziert wohl aktuell die meisten Linux-Bildschirme. Und solange Linux wegen seiner bescheidenen Hardware-Anforderungen gern auf »älteren« Computern oder vermehrt auch auf embedded Systemen eingesetzt wird, werden auch die »schlanken« Windowmanager, wie der »fvwm2« seine Berechtigung haben. Und genau die drei benannten Oberflächen sollen dieses Kapitel beschließen.

Ziel des Kapitels

Die ersten Abschnitte werden wohl eher den technisch interessierten Leser ansprechen, anschließend lernen Sie kennen

- wie Sie den Start von X beeinflussen können
- wie Sie die Ausgaben grafischer Anwendungen auf andere Rechner umleiten
- wie Sie die Login-Manager xdm, kdm und gdm konfigurieren
- wie Sie den Windowmanager »fvwm2« einrichten
- wie Sie die Desktop-Umgebungen KDE und GNOME konfigurieren

Inhalt des Kapitels

X Window System - Geschichte

Der Ursprung

Das X Window System basiert auf dem **X-Protokoll**, dessen Grundlagen zunächst am MIT (Massachusetts Institute of Technology) entwickelt wurden und dessen Veröffentlichung auf Mitte der 80er Jahre datiert. Ziel war eine Richtlinie für grafische Nutzerschnittstellen, die noch dazu für einen Anwender transparent im Netzwerk arbeiten sollen.

Die erste Implementierung des Protokolls resultiert aus dem Athena Projekt des MIT und Digital Equipment Corporation (X10R4, 1986), dessen Nachfolger (X11R1) bereits im September 1987 erschien. Zu jener Zeit gründete sich am MIT das **X Consortium**, das sich der weiteren Entwicklung und Betreuung des X Window System widmete. Unter Federführung dieses Gremiums erschienen die nächsten Releases (bis X11R5) im Ein-Jahres-Rhythmus.

1992 entschlossen sich MIT und die Mitglieder des Konsortiums zur Ausgliederung dessen aus dem MIT. Im Folgejahr entstand das **X Consortium, Inc.**, das ab Januar 1994 über alle Rechte am X Window System verfügte. Das Konsortium selbst verstand sich als unabhängige, nicht-profit-orientierte Organisation. Noch im selben Jahr wurde X11R6 heraus gegeben, das auch heute noch den Stand der Dinge widerspiegelt.

Am 31.Dezember 1996 stellte das X Consortium seine Tätigkeit ein und übertrug die Rechte des Projekts an die **Open Software Foundation**.

XFree86

Die zunächst einzige freie Implementierung des X Window System war **XFree86** (1995). Ursprünglich von X11R5 abgeleitet, ermöglicht XFree86 den Zugriff auf alle Funktionen, die auch X11R6 implementiert.
Von Anfang an zielte das XFree86-Projekt auf die Unterstützung vor allem der Intel-Plattform ab. Daher auch der Vermerk **86** im Namen des Projekts. Unterdessen ist XFree86 u.a. auf folgenden Systemen verfügbar:

- System V Release 3.2, 4.0 und 4.2
- Solaris
- FreeBSD, OpenBSD
- Linux
- Mach
- Amöba

Ein weiterer Vorteil von XFree86 gegenüber kommerziellen Produkten ist die breite Palette an unterstützten Grafikkarten.

Die Zukunft von X?

Bei all den Vorteilen, die die Netzwerkfähigkeit des X-Protokolls mit sich bringt, lässt dessen Geschwindigkeit bei der Arbeit über das Netz doch einiges zu wünschen übrig.

Eine Ursache lässt sich sicher in den existierenden Implementierungen finden, der wesentliche Grund liegt aber in dem teils veralteten Protokoll. So fehlen einige der modernen Grafikalgorithmen, die zur Zeit der Protokollentwicklung noch nicht bekannt waren.

Bislang halten sich die Entwickler mit einem Ausblick auf ihr weiteres Vorgehens sehr bedeckt, doch ist ein neues Protokoll oder zumindest eine Anpassung an die aktuellen Erfordernisse nur eine Frage der Zeit...

X Window System - Das Client Server Modell

Das X Protokoll

Das X Protokoll definiert eine Client-Server-Beziehung zwischen der Anwendung und seiner Darstellung. Die Anwendung als Client spezifiziert **was** dargestellt werden soll. Der Server kümmert sich um das **wie** der Darstellung.

Diese klare Trennung ermöglicht den Austausch sowohl des Clients als auch des Servers, ohne dass der jeweils andere Teil einer Anpassung bedarf. Außerdem ermöglicht diese Charakteristik, eine grafische **Anwendung** auf einem entfernten Rechner **im Netz** zu starten und ihre **Ausgaben** durch die Dienste des lokalen X-Servers **lokal** darzustellen.

Des Weiteren unterteilt X den X Server in einen geräteabhängigen und einen geräteunabhängigen Teil. Somit verdeckt X zum einen die hardwarespezifischen Details und zum anderen die Besonderheiten des Betriebssystems.

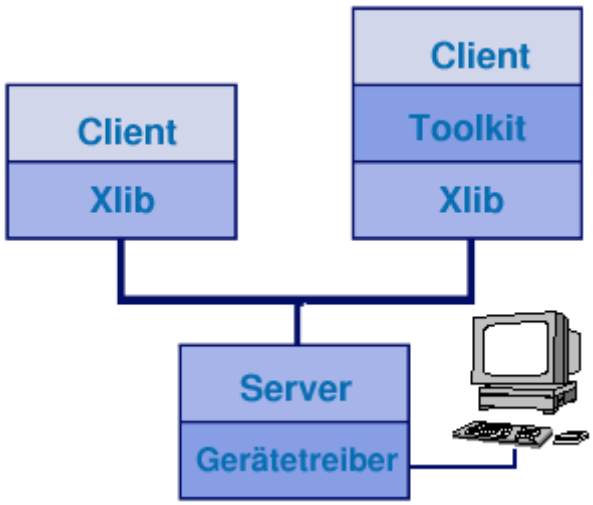


Abbildung 1: X Client Server Architektur

Die Abbildung veranschaulicht das Zusammenwirken von Clients und Server. Der Server implementiert Funktionen der Xlib-Bibliothek, auf deren Schnittstelle alle Clients basieren. Unter einem X-Client versteht man dabei jede grafische Anwendung, die einen X-Server zur Darstellung benötigt. X-Clients sind also xterm, xclock, xman, ...
Ein einzelnes X-Terminal ist kaum als grafische Benutzeroberfläche zu bezeichnen. Um einen einheitlichen Desktop zu präsentieren, setzen Toolkits - Windowmanager oder integrierte Umgebungen wie KDE - auf die Grafikbibliothek auf. Die eigentlichen X-Clients arbeiten nun mit einem Windowmanager zusammen, der die Anwendungen mit einem einheitlichen Rahmen und identischen Bedienelementen versieht...

Der X Server

Der X-Server ist das zur Steuerung des **Displays** zuständige Programm. Ein Display umfasst dabei - analog zum Terminal der Konsole - eine Kombination von Ein- und Ausgabegeräten. Typische Eingabegeräte sind die Tastatur ("keyboard") und die Maus ("pointer"); die Ausgaben erfolgen auf einem *oder mehreren* Bildschirm*en* ("screen").

Der X-Server ist auf der lokalen Maschine aktiv und bearbeitet die Anforderungen, die von lokalen oder auch entfernten Clients eintreffen. Das Format der Kommunikation zwischen Clients und Server ist durch das X-Protokoll festgelegt. Netzwerkverbindungen werden über TCP/IP und DECnet unterstützt.

Alle Datenobjekte werden in der X-Window-Terminologie als **Resources** bezeichnet. Hieruntern zählen die

Cursors, die Fonts, die Windows, ... Jede Resource besitzt eine eindeutige Identifikation und mehrere Clients können gleichzeitig auf eine Resource zugreifen. Es ist nun Aufgabe des X-Servers bei einer Benutzereingabe herauszufinden, welches Fenster gerade aktiv war, welcher Client dessen Eigentümer ist und damit, an welchen Client die Eingaben weiter zu leiten sind.

Das X-Protokoll unterscheidet drei Arten von Nachrichten, die der X-Server an einen Client übermitteln kann:

- **Replay** ist die Antwort des Servers auf die Anfrage eines Clients (z.B. "Fenster ist sichtbar")
- **Event** ist die Benachrichtigung des Clients, z.B. weil ein Benutzer eine Eingabe getätigt hat, oder ein Fenster neu gezeichnet werden muss
- **Error** ist die Mitteilung über einen Fehler

Die Daten der Objekte eines Clients werden soweit möglich auf dem Server gehalten. Der Client bezieht sich nachfolgend auf ein Objekt nur noch über die Resource-ID. Hierin liegt der Grund, warum der Start einer X-Anwendung über eine Netzverbindung zunächst recht langsam vonstatten geht, nachfolgend die Geschwindigkeit aber kaum einen Unterschied zwischen lokalen und entfernten Clients offenbart. Der Datentransfer wird stark reduziert; allerdings sollte ein Server auch über entsprechend viel Speicherkapazität verfügen.

Der X Client

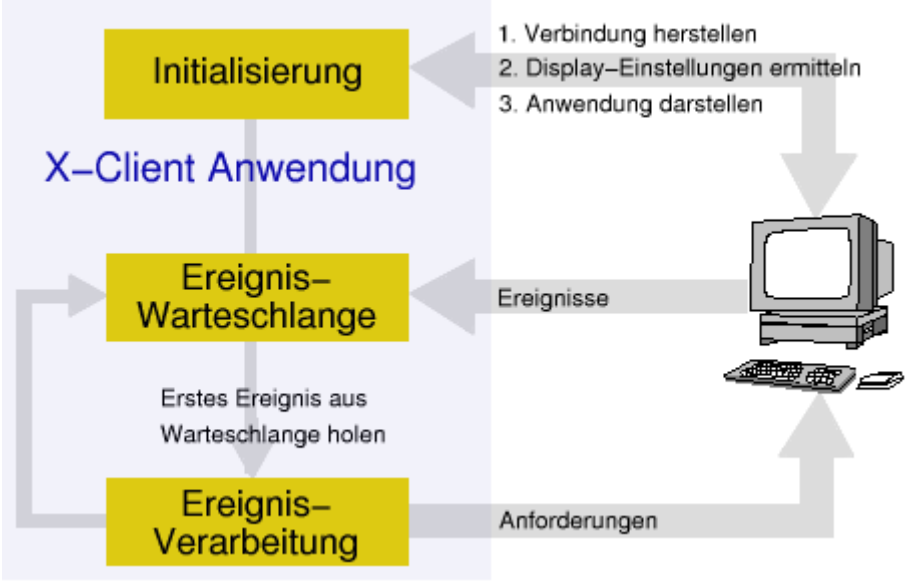


Abbildung 2: Aufbau einer X-Client Anwendung

XFree86 4 - Die neue Architektur

Das X Window System - Steuerung

Übersicht

Abläufe beim Start

xinitrc

xserverrc

xinit

Kommandozeilenparameter

Das X Window System - Login Manager

Übersicht

Der Standard Login Manager xdm

Der KDE Login Manager kdm

Der Gnome Login Manager gdm

Der Windowmanager Fvwm2

Übersicht

Dem originalen Entwickler Rob Nation ist leider die Bedeutung des »f« in seiner Wortschöpfung entfallen. Die gewitzten Betreuer des Projektes benennen den **fvwm** so:
»Fill_in_the_blank_with_whatever_f_word_you_like_at_the_time Virtual Window Manager«.

Der Windowmanager **fvwm2** als Nachfolger des »fvwm« war vor den Zeiten von Desktopumgebungen wie K Desktop Environment oder GNU Network Object Model Environment der verbreiteste Manager unter Linux. Auch heute liegt dieser Windowmanager jeder Distribution bei.

Fvwm wurde mit dem Ziel ins Leben gerufen, einen Windowmanager mit geringen Hardwareanforderungen zu präsentieren (Speicherverbrauch < 1 MByte), der die in ihm laufenden Anwendungen mit einem 3D-Outfit - analog zu Motif's mwm - austattet. Der Fvwm2 stellt in seiner Grundfunktionalität nur einen einfachen virtuellen Desktop bereit. Die Vielgestaltigkeit des Managers ergibt sich erst durch die Verwendung der im Paket enthaltenen Module.

Der »fvmw2« ist hochgradig konfigurierbar, wenngleich das Einrichten der Ressourcen-Datei wohl eher dem Konsolenfreund entgegen kommt.

Die aktuell stabile Version 2.2.4 kennzeichnet folgende Eigenschaften:

- Dynamische Erweiterung mittels Modulen
- Wahlweise mit Tastatur und/oder Maus bedienbar
- Echt-Farben Icons
- Möglichkeit des Erzeugens einer Konfigurationsdatei mit dem Makroprozessor M4
- Unterstützung mehrerer Bildschirme
- Animation von Fensterbewegungen
- Themen, Sounds,...

Die Elemente des Fvwm2

Den typischen Aufbau eines Desktops mit dem Windowmanager Fvwm2 zeigt folgende Abbildung.

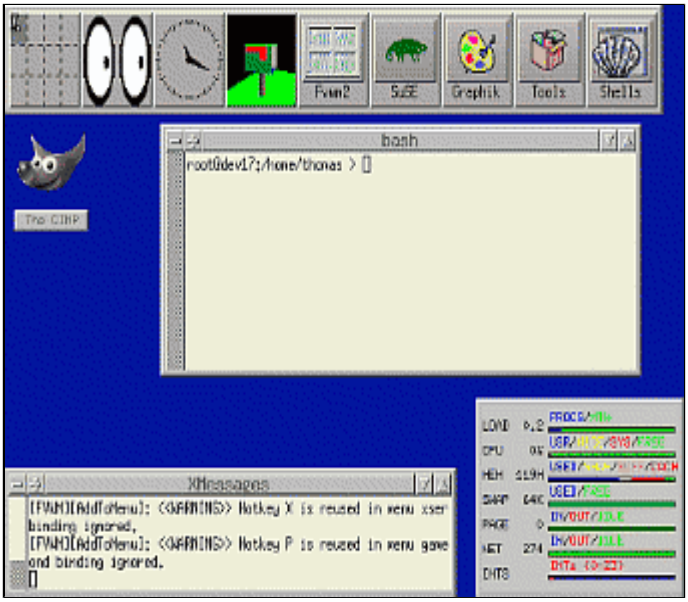


Abbildung 1: Desktop des Fvwm2

Im oberen Teil ist das Modul **FvwmButton** zu sehen, in ihm sind integriert (von links nach rechts):

- Das Modul **FvwmPager**, das die verschiedenen virtuellen Desktops symbolisiert. Der aktive Desktop ist dunkel hinterlegt. Durch Klick auf das Symbol wird zu einem anderen Desktop gewechselt.
- Die Anwendungen »xeyes«, »xclock« und »xbiff«
- Zugänge zu Menüs »Fvwm2«, »SuSE«, »Graphic«, »Tools« und »Shells«. Bei Anklicken eines der Symbole öffnen sich Untermenüs, die weitere Menüs oder Anwendungen enthalten.

Des Weiteren sind auf dem Desktop die Anwendungen »xterm» (Mitte), »xmessages» (links unten), »xosview« (rechts unten) und das Icon der Anwendung »gimp« zu sehen.

Klickt man mit der rechten Maustaste auf den Desktophintergrund, öffnet sich das »Alles«-Menü. Bei der Standardkonfiguration ist dieses auch über die Tastenkombination [Alt]-[F1] erreichbar.

Abbildung 2: Fvwm2: Menü der rechten Maustaste

Die mittlere Maustaste (bei 2-Tasten-Mäusen wird diese durch simultanes Drücken beider Tasten emuliert) oder die Tastenkombination [Alt]-[F2] führt zu einer Liste der aktiven Fenster.

Abbildung 3: Fvwm2: Menü der mittleren Maustaste

Mittels der linken Maustaste oder der Tastenkombination [Alt]-[F3] öffnet sich das Arbeitsmenü.

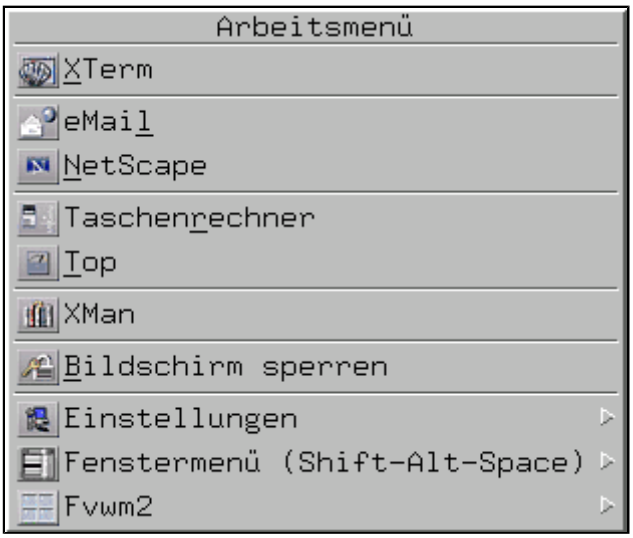


Abbildung 4: Fvwm2: Menü der linken Maustaste

Konfiguration des Fvwm2

Allein das Manual zum »fvwm2« umfasst knapp 3000 Zeilen. Nicht ganz so umfangreich, dafür umso zahlreicher sind die Manuals zu den einzelnen Modulen. Hier alle Möglichkeiten der Konfiguration zu diskutieren, würden den Leser langweilen und den Autor überfordern... Ich beschränke mich deshalb auf die Vorstellung der wesentlichen Elemente des Desktops:

- Die Suchpfade für Icons, Module, Pixmaps
- Das Einstellen der Schriften
- Die Gestaltung der Umrahmungen der Anwendungen
- Die Gestaltung der Schalter der Fenster
- Die Aktionen beim (Neu)Start des Windowmanagers
- Die Belegung der Maustasten
- Die Belegung von Funktionstasten
- Das Erstellen eines Menüs
- Das Gestalten des FvwmButtons

Den **fvmw2** konfiguriert man über eine ASCII-Datei. Die globale Datei befindet sich unter »/usr/X11R6/lib/X11/fvwm2/system.fvwm2rc«. Normalerweise wird man diese unverändert lassen, und eine lokale Kopie mit dem Namen **~/.fvwm2rc** bearbeiten.

Konfiguration - Globale Einstellungen

Die Suchpfade

Der »fvwm2« benötigt zunächst Kenntnis, wo er seine Module, die Pixmaps (kleine Bildchen) und Icons (Symbole für verkleinerte Anwendungen) zu suchen hat:

```
ModulePath /usr/lib/X11/fvwm2
PixmapPath /usr/include/X11/pixmaps
IconPath   /usr/include/X11/bitmaps
```

Die Schriftarten

Weitere Einstellungen legen die verwendeten Schriftarten für die Fenstertitel und Icon-Unterschriften fest. Die meisten Leser wird die Angabe der Schriften verwirren, wer das Geheimnis zu lüften wünscht, sollte einmal das Programm »xfontsel« aufrufen.

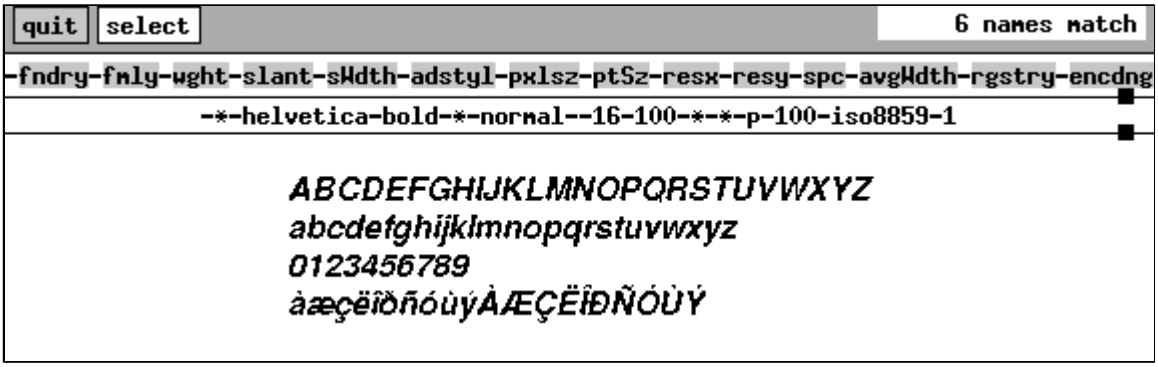


Abbildung 5: Schriften konfigurieren mit »xfontsel«

```
WindowFont -adobe-times-bold-r-*-*-18-*-*-*-*-*-*-*
IconFont -adobe-helvetica-bold-r-*-*-10-*-*-*-*-*-*-*
```

Konfiguration - Die Fenster

Mit **Style** kann das Aussehen jedes einzelnen Fensters und der Module gesteuert werden. Die Syntax der Styles besitzt folgendes Format:

```
Style "<Fenstertitel>" <Definitionen>
```

»Fenstertitel« steht dabei für den konkreten Programmnamen (z.B. »xosview«), alle Programme (»*«) oder einer Kombination aus beiden (»Fvwm*«). Als Definitionen sind u.a. zulässig (mehrere Definitionen werden durch Kommas getrennt):

BackColor

Hintergrundfarbe der Fensterdekoration

```
BackColor #00f0e4
```

BorderWidth

Rahmenbreite in Pixel

```
BorderWidth 5
```

ClickToFocus

Das Fenster wird erst fokussiert, sobald ein Mausklick in dieses erfolgt

```
ClickToFocus
```

Icon

Icon, das beim Verkleinern des Fensters dargestellt werden soll. Das Icon muss im Icon-Suchpfad liegen!

```
Icon Terminal.xpm
```

IconBox

IconFill

Mehrere Icons werden in einem bestimmten Gitter, ausgehend von der ersten Position (IconBox) dargestell
Ein neues Icon wird dabei in das erste freie Gitterfeld platziert. Mit IconFill kann die Ausrichtung der Darste
beeinflusst werden. Voreingestellt ist: erst von links nach rechts, dann von oben nach unten.

IconFill Bottom Right

ForeColor

Farbe der Fensterüberschrift

ForeColor Black

MouseFocus

Das Fenster erhält den Fokus, sobald der Mauszeiger über diesem steht

ForeColor MouseFocus

NoTitle

Unterdrückt die Darstellung der Fenstertitelleiste (Voreinstellung ist »Title«)

NoTitle

SloppyFocus

Wie »MouseFocus«, jedoch verliert das Fenster den Fokus erst, wenn die Maus in eine anderes »MouseFocu
Fenster wechselt oder ein »ClickToFocus«-Fenster aktiviert wird

SloppyFocus

StaysOnTop

Das Fenster wird auf jedem Desktop dargestellt

StaysOnTop

Sticky

Das Fenster ist auf allen virtuellen Desktops vorhanden (Voreinstellung ist »NoSticky«)

Sticky

UseDecor

Zur Verwendung eines mit AddToDecor erzeugten Aussehens

UseDecor DemoDecor

WindowListSkip

Das Fenster erscheint nicht in der Fensterliste (z.B. im Menü der mittleren Maustaste)

```
WindowListSkip
```

Beispielhaft sollen die Gestaltung des FvmwButtons und der Anwendung »xterm« sowie einige Einstellungen, die alle Anwendungen betreffen, skizziert werden:

```
Style  "FvwmButtons"      NoTitle, Sticky, WindowListSkip
Style  "xterm"            Icon Terminal.xpm

Style  "*"                BackColor Grey
Style  "*"                ForeColor DimGrey
```

Konfiguration - Die Schalter der Fenster

Die Fensterleiste enthält meist mehrere Buttons, z.B. zum Minimieren, Maximieren oder Schließen des Fensters. Die Konfiguration der Buttons erfolgt mit dem Kommando **ButtonStyle**:

```
ButtonStyle button [ state ] [ style ] [ -- [!]flag ... ]
```

button kann eine Nummer zwischen 0 und 9 enthalten. Die Reihenfolge der Nummerierung ist dabei sehr seltsam: **1 3 5 7 9 0 8 6 4 2**, wobei die Buttons mit ungerader Nummer links- und die anderen rechtsbündig angeordnet werden. Steht im Feld »button« »ALL«, werden alle Buttons durch das Kommando konfiguriert. »Left« und »Right« sprechen den linken bzw. rechten Schalter an. Mit »Reset« werden alle Schalter in die Voreinstellung versetzt.

state kann entfallen, dann gelten die Einstellungen für den Schalter in jedem Status. Steht dort »ActiveUp«, bezieht sich die Konfiguration auf einen nicht-gedrückten Schalter, »ActiveDown« bezeichnet den gedrückten Schalter. »Inactive« konfiguriert den Schalter eines inaktiven Fensters.

style betrifft nun das tatsächliche Aussehen des Buttons. Eine kleine Auswahl an Styles soll die Mannigfaltigkeit der Konfiguration andeuten:

Default

Setzt voreingestellte Werte, eine optionale Nummer setzt anstelle des durch »button« bezeichneten Schalte die Werte des Schalters dieser Nummer. So setzt folgende Zeile die Aussehen der Schalter 1 und 3 gleich:

```
ButtonStyle 1 default 3
```

Solid

Setzt die Farbe des Buttons:

```
ButtonStyle Left solid blue
```

Vector

Zeichnet Linien in einen Schalter. Dem Schlüsselwort folgen die Anzahl der Punkte und anschließend die Koordinaten für jeden Punkt in der Form »x-Koordinate**x**y-Koordinate@Farbe«. »Farbe« kann dabei "1" ode "0" sein, je nachdem, ob mit der Vorder- oder Hintergrundfarbe gezeichnet werden soll:

```
ButtonStyle 4 Vector 4 50x74@1 25x24@1 75x24@1 50x74@0
```

Pixmaps

MiniIcon

Das Mini-Symbol der Anwendung wird im Schalter dargestellt. Die Verbindungen von Anwendungen und MiniIcon-Dateien muss zuvor mit dem Style-Kommando definiert werden:

```
Style "xterm" MiniIcon mini-term.xpm
ButtonStyle 1 MiniIcon
```

flag letztlich bestimmt das Verhalten des Schalters, wenn er gedrückt wird. Mehrere Flags können, durch Leerzeichen voneinander getrennt, angegeben werden. Die Auswahl kann durch ein vorangestelltes Ausrufezeichen negiert werden.

Abbildung 6: Mögliches Aussehen von Schaltern

Die Schalter der Abbildung werden durch folgende Konfiguration erzeugt:

```
DestroyDecor DecorDemo
AddToDecor DecorDemo
+ TitleStyle Height 30
+ TitleStyle (Solid gray -- Flat)
+ ButtonStyle 1 8 40x80@1 40x50@1 20x50@1 50x20@1 80x50@0 60x50@0 60x80@0 40x80@0
+ ButtonStyle 3 8 40x20@1 40x50@1 20x50@1 50x80@1 80x50@0 60x50@0 60x20@0 40x20@1
+ ButtonStyle 5 8 80x40@1 50x40@1 50x20@1 20x50@1 50x80@0 50x60@0 80x60@0 80x40@0
+ ButtonStyle 7 8 20x40@1 50x40@1 50x20@1 80x50@1 50x80@0 50x60@0 20x60@0 20x40@1
+ ButtonStyle 9 12 10x50@1 35x25@1 35x40@1 65x40@1 65x25@1 90x50@1 65x75@0 65x60@0 35x60@0 35x75@0
10x50@0 10x50@1
+ ButtonStyle 0 12 50x10@1 25x35@1 40x35@1 40x60@1 25x60@1 50x85@1 75x60@0 60x60@0 60x35@0 75x35@0
50x10@0 50x10@1
+ ButtonStyle 8 11 80x20@1 45x20@1 55x30@1 30x55@1 20x45@1 20x80@1 55x80@0 45x70@0 70x45@0 80x55@0
80x20@0
+ ButtonStyle 6 11 20x20@1 55x20@0 45x30@0 70x57@0 80x45@0 80x80@0 45x80@1 57x70@1 30x45@1 20x55@1
20x20@1
+ ButtonStyle 4 14 20x20@1 20x70@1 70x70@0 70x20@0 20x20@1 60x60@0 60x50@0 60x60@0 50x60@0 60x60@0
60x80@0 80x80@0 80x60@0 60x60@0
+ ButtonStyle 2 12 10x23@0 90x23@0 90x28@0 10x28@1 10x47@1 90x47@0 90x52@0 10x52@1 10x70@2 90x70@0
90x76@0 10x76@1
```

Konfiguration - (Re)Starteinstellungen

Der Windowmanager »fvwm2« kennt zwei spezielle Funktionen. **InitFunction** wird beim ersten Aufruf des Managers aufgerufen und **RestartFunction** ist der Ansprungspunkt nach einem Neustart. Beide Funktionen sind identisch aufgebaut und dienen dazu, den »fvwm2« beim (Re)Start weitere Programme aufrufen zu lassen. Wir widmen uns nur der »InitFunction«.

```
AddToFunc InitFunction
  +     "I"     Module FvwmBanner
  +     "I"     Exec susewmif xmessages
  +     "I"     Exec xterm -ls -geometry +150+85
  +     "I"     Exec xosview -geometry 175x135-10-10
  +     "I"     Module FvwmButtons
  +     "I"     Exec xpmroot /usr/X11R6/include/X11/icons/private.xpm
```

AddToFunc gelangt stets zum Einsatz, wenn es um das Hinzufügen einer Aktion zu einer Funktion geht. Jede Zeile, die noch zur Funktionsdefinition gehört, muss mit einem Plus **+** eingeleitet werden. **"I"** besagt, dass die Aktion unmittelbar auszuführen ist. Mit **Module** wird der Start eines fvwm-Modules eingeleitet, alle anderen Programme sind hinter **Exec** anzugeben, wobei das erste Argument als Programmname und alle weiteren bis Zeilenende als Optionen für das Programm verwendet werden.

Die Module des »fvwm2« sollten unbedingt in die RestartFunction aufgenommen werden, da diese im Gegensatz zu fvwm2-unabhängigen Programmen einen Neustart nicht überleben.

Konfiguration - Aktionen mit der Maus

Die Maus ist das wichtigste Eingabegerät unter X. Prinzipiell kann alles so konfiguriert werden, dass sämtliche Aktionen per Tastenkombinationen erreichbar sind, aber wer kann sich schon all die Tastenkürzel merken?

Das Aussehen des Mauszeigers

In Abhängigkeit von der Umgebung kann der Mauszeiger seine Optik verändern. Hierzu ist die Option **CursorStyle** zu verwenden, die als Argument den Bezeichner des Kontexts und die (gerade) Nummer des Stils erhält. Einige der Kontexte sind:

title

Wenn sich der Mauszeiger über einen Fenstertitel bewegt

DEFAULT

Überall dort, wo kein eigener Stil definiert wurde

SYS

Schalter der Fensterleiste

MOVE

Beim Bewegen, Ändern der Größe eines Fensters

WAIT

Während ein Programm ausgeführt wird

MENU

In Menüs

DESTROY

Wenn ein Fenster geschlossen/zerstört wird

LEFT | RIGHT

Linker/rechter Fensterrahmen

BOTTOM_LEFT

In der linken unteren Fensterecke (TOP_LEFT... analog)

Das Aussehen des Mauszeigers wird als gerader numerischer Wert angegeben. Welcher Wert für welchen Zeigertyp steht, kann der Datei »/usr/X11R6/include/X11/cursorfont.h« entnommen werden.

```
CursorStyle title 58
CursorStyle DEFAULT 68
CursorStyle DESTROY 124
```

Aktionen beim Mausklick

Das Binden einer Aktion an eine bestimmte Maustaste folgt immer dem selben Schema:

```
Mouse <Taste> <Kontext> <Modifizierer> <Funktion>
```

Taste entspricht der Tastennummer (0 alle, 1 links, 2 Mitte, 3 rechts), **Kontext** der Umgebung, für die die Definition zutrifft. Hier steht **R** für das Root-Fenster (Desktop), **W** für ein Anwendungsfenster. Weiterhin betrifft **T** den Fenstertitel, **S** den Rahmen, **F** die Ecken eines Fensters und **A** steht stellvertretend für jeden Kontext. **0** bis **9** binden Aktionen an die gleichnamigen Schalter der Fensterleiste.

Mit den **Modifizieren** kann gesteuert werden, ob die Mausaktion erst in Verbindung mit einem Tastendruck ausgelöst wird. Hierbei gilt **N** für »keine« Tastenbetätigung, **A** für »egal welche«, **C** für [Ctrl], **M** für [Alt] und **S** für [Shift]. Kombinationen sind zugelassen, dann müssen neben der Maustaste auch noch die angegebenen Tasten der Tastatur gedrückt sein. Schließlich ist **Funktion** eine der fvmw-Funktionen oder auch eine selbst definierte. Wird an ihrer Stelle ein Minus eingesetzt, so wird eine bestehende Bindung aufgehoben.

Das folgende Beispiel startet das Programm »gimp«, sobald auf dem Desktop die linke Maustaste zusammen mit [Ctrl]-[Shift] gedrückt wird:

```
AddToFunc StartGimp
 +   "I" Exec gimp

Mouse 1 R CS StartGimp
```

Obiges Beispiel funktioniert sogar, wenn zuvor die Mausaktion für das Rootfenster mit dem Kontext "A" (jede Taste) verbunden wurde. In diesem Fall überschreibt die Konfiguration nur die Aktion für diese eine Tastenkombination. Die übliche Konfiguration verbindet die Mausklicks auf dem Desktop mit dem Eröffnen von Menüs:

```
Mouse  1    R    A   Menu    Work_menu316popup   Nop
Mouse  2    R    A   Module  FvwmWinList  Transient
Mouse  3    R    A   Menu    SuSE0popup   Nop
Mouse  2    FST  A   Menu    windowops    Nop
```

Konfiguration - Aktionen mit der Tastatur

Das Binden bestimmter Tastaturereignisse an bestimmte Aktionen erfolgt ähnlich dem Vorgehen bei der Maus. Der Aufruf lautet:

```
Key <Tastenname> <Kontext> <Modifizierer> <Funktion>
```

Der **Tastenname** kann der Datei »/usr/include/X11/keysymdef.h« entnommen werden, wobei das führende »XK_« zu streichen ist. **Kontext**, **Modifizierer** und **Funktion** sind genauso zu verwenden, wie unter Aktionen mit der Maus beschrieben.

Das Beispiel ermöglicht das Bewegen des Mauszeigers mit den Navigationstasten der Tastatur bei gleichzeitigem Drücken der [Shift]-Taste. Hierbei gelangt die Funktion **CursorMove** zum Einsatz, die den Mauszeiger um den angegebenen Wert in x,y-Richtung verschiebt:

```
Key  Left   A S CursorMove -1 +0
Key  Right  A S CursorMove +1 +0
Key  Up     A S CursorMove +0 -1
```

```
Key  Down  A S CursorMove +0 +1
```

Konfiguration - Ein eigenes Menü

Menüs gehören zum guten Stil eines jeden Windowmanagers. In ihnen sammelt man alle Programme, die über einen Mausklick erreichbar sein sollen. Während bspw. KDE seine Programme während der Installation (fast immer) automatisch in die Menüstruktur integriert, ist bei den meisten Windowmanagern ein manueller Eingriff erforderlich. Da helfen auch nicht solche Skripte wie sie SuSE-Linux beiliegen, die nach jeder Installation eines neuen Paketes für die wichtigsten Manager neue Konfigurationen erstellen. Das Programm, das ich eigentlich benötige, wurde garantiert nicht aufgenommen!

Im Falle des **fvwm2** wird ein Menü mit **AddToMenu** erzeugt:

```
AddToMenu <Menüname> [<Menüüberschrift> [Title]]
```

Menüname ist ein eindeutiger Bezeichner, über den in anderen Stellen der Konfiguration auf das Menü Bezug genommen werden kann. Die optionale **Menüüberschrift** wird als als erster Eintrag des Menüs dargestellt und das optionale **Title** zentriert diese Überschrift und trennt die nachfolgenden Einträge durch eine Linie ab.

Um nun die Einträge einem Menü hinzuzufügen, bedient man sich folgender Syntax:

```
+ "&Bezeichner%Iconname%" <Aktion>
+ "B&ezeichner*Iconname*" <Aktion>
```

Bezeichner ist ein (leerer) Text, der im Eintrag erscheinen soll. Das **&** kennzeichnet den nachfolgenden Buchstaben als »Shortcut«, über den der Menüeintrag per Tastatur angewählt werden kann. **Iconname** ist der Dateiname des Pixmaps, das als Grafik im Eintrag platziert wird. Die Datei muss im Pfad der Pixmaps (siehe Globale Einstellungen) zu finden sein. Der Iconname wird entweder in zwei ***** oder **%** eingeschlossen. In ersten Fall wird der Bezeichner unter der Grafik dargestellt, im zweiten rechts dieser. **Aktion** ist die zu rufende Funktion, wenn der Menüeintrag selektiert wurde.

Beispiel: Ein Menü soll den Zugang zu den Grafikverarbeitungsprogrammen **gimp** und **xfig** realisieren:

```
AddToMenu StartGrafik "Grafikprogramme" Title
+  "&Gimp%xgrab.xpm%" Exec gimp
+  "" Nop
+  "&Xfig%xfig.xpm%" Exec xfig
```

Erläuterung: Als Titelzeile soll im Menü »Grafikprogramme« stehen, Das Programm **gimp** soll neben der Maus auch mit der Taste [G] gestartet werden können. Als Grafik gelangt »xgrab.xpm« zum Einsatz. Die zweite Zeile erzeugt eine Linie, **nop** besagt, dass damit keine Aktion verbunden ist. Der dritte Eintrag schließlich fügt **xfig** hinzu, das über die Taste [X] aufgerufen werden kann. Folgende Abbildung zeigt das fertige Menü:

Abbildung 7: Ein eigenes Menü

Jetzt haben wir zwar ein Menü aber noch keine Möglichkeit, auf dieses zuzugreifen. Beispielhaft soll das Menü erscheinen, wenn wir die Tasten [F1]-[Shift]-[Ctrl] (unabhängig vom Kontext) drücken und bei Klick mit der linken Maustaste auf den Desktop bei gleichzeitigem Betätigen der Tasten [Shift]-[Ctrl]:

```
Key F1 A CS Menu StartGrafik
Mouse 1 R CS Menu StartGrafik
```

Erläuterung: Wichtig ist **Menu**, um die Bindung mit einem Menü zu veranlassen.

Ein Menü kann selbst weitere Untermenüs enthalten, ein entsprechender Eintrag in AddToMenu sieht wie folgt aus:

```
+ "&Bezeichner%Iconname%" Popup < Menüname>
```

Die Definition des Untermenüs folgt exakt den Regeln eines »normalen« Menüs.

Konfiguration - Das Modul FvwmButtons

Abbildung 8: Das Modul FvwmButtons

Die Module sind eine Erweiterung des fvwm2, die zur eigenen Arbeit allerdings einen aktiven Windowmanager benötigen. Die Module selbst werden mit dem Schlüsselwort **Module** geladen und verwenden den Pipemechanismus zur Kommunikation mit dem **fvwm**. Stellvertretend für den Umgang mit Modulen soll die Konfiguration des **FvwmButtons** dienen, also des Modules, das in vielen anderen Managern unter dem Begriff "Startleiste" existiert.

Das Modul FvmwButtons kann sowohl Icons beherbergen, über die bestimmte Funktionen zugänglich werden, als auch Programme, die innerhalb des Buttons ausgeführt werden. Im links abgebildeten Beispiel ist das Modul **FvwmPager** innerhalb des Buttons aktiv (ganz unten), der das Umschalten zwischen den einzelnen virtuellen Desktops ermöglicht.. Werfen wir kurz einen Blick auf dessen Konfiguration:

```
DeskTopSize 2x2

*FvwmPagerRows 1
*FvwmPagerColumns 1
*FvwmPagerGeometry -0+0
*FvwmPagerFore Black
*FvwmPagerBack grey67
*FvwmPagerHilight grey50
*FvwmPagerSmallFont -misc-fixed-medium-r-normal--7-70-75-75-c-50-iso8859-1
```

```
*FvwmPagerLabel *  Desktop
```

Erläuterung: DeskTopSize bedingt nicht die Existenz des FvwmPagers, sondern ist eine Option des **fvwm** und legt die Anzahl der virtuellen Desktops fest (hier 2 in horizontaler und 2 in vertikaler Richtung). Der Pager selbst mappt die Desktops in seinen Bereich. Die beiden folgenden Zeilen definieren die Ausdehnung (1 Zeile, 1 Spalte) des Pagers, die nächste Zeile bestimmt die relative Position der Darstellung. Weitere Optionen legen die Farben für Vorder- und Hintergrund, sowie für den gerade aktiven Desktop fest. Der letzte Eintrag setzt »Desktop« als Überschrift des Pagers (***** weist den Titel allen Desktops zu; hier kann für jeden Desktop ein eigener Titel definiert werden).

Bedienung des FvwmPagers: Indem Sie mit der rechten Maustaste auf das Symbol eines Desktops klicken, wird zu diesem gewechselt. Klicken Sie gezielt auf das Symbol einer Anwendung, erhält diese zugleich den Fokus. Auch lassen sich die Anwendungen zwischen den verschiedenen Deskops verschieben, indem Sie mit der mittleren Maustaste das Symbol fixieren und auf den neuen Desktop ziehen.

Der »FvwmButton« wurde durch folgende Einstellungen erzeugt:

```
*FvwmButtonsFont      6x13
*FvwmButtonsFore      Black
*FvwmButtonsBack      grey67
*FvwmButtonsGeometry  +0+0
*FvwmButtonsColumns   2
*FvwmButtons - - Swallow "FvwmXeyes" Exec xeyes -name "FvwmXeyes" -geometry +0+0 -bg grey67 &
*FvwmButtons - - Swallow "FvwmXclock" Exec xclock -name "FvwmXclock" -geometry +0+0 -padding 1 -bg grey67 &
*FvwmButtons - - Swallow "coolmail" Exec coolmail -geometry +0+0 -vol 100 -int 12 -e "xterm -g 80x36 -e pine" &
*FvwmButtons Fvwm2 window3d.xpm Menu WindowManager324popup
*FvwmButtons SuSE logo_suse_3d.xpm Menu barsusepopup
*FvwmButtons Graphics palette3_3d.xpm Menu bargraphicspopup
*FvwmButtons Tools box_full_3d.xpm Menu bartoolspopup
*FvwmButtons Shells Shell.xpm Menu barshellspopup
*FvwmButtons (2x1 Frame 0 Swallow(UseOld) "FvwmPager" "Module FvwmPager 0 0")
```

Kwm - Der KDE Window Manager

Übersicht

Übersicht

Enlightenment - Der Gnome Window Manager

Übersicht

Übersicht

Der Kernel

Überblick

Ziel des Kapitels

Inhalt des Kapitels

Der Kernel - Geschichte

Übersicht

Am 17. September 1991 postete Linus Torvalds die Ergebnisse seiner Studien zu einem Unix-ähnlichen Betriebssystem in die Newsgruppe »Minix«. Sicherlich hatte er etwas Aufmerksamkeit erhofft, aber ob er mit einer solchen Resonanz gerechnet hatte?

Diese Version 0.01 implementierte gerade mal einen rudimentären Kern, wurde aber von zahlreichen Minix-Anhängern binnen kurzer Zeit soweit ergänzt, dass er bald seinem unmittelbaren Vorgänger, dem Minix-System, den Rang ablief.

Linux kam genau zur richtigen Zeit. Schon seit mehreren Jahren arbeitete man bei der **Free Software Foundation** an einem freien Betriebssystem. Diese von Richard Stallman initiierte Vereinigung hatte es sich zum Ziel gesetzt, kostenlose und im Quelltext erhältliche Software für jederman zu erstellen. Die Anzahl verfügbarer Programme wuchs stetig, jedoch ließ man sich mit dem Kern der Sache, einem an Unix angelehnten Betriebssystem, ungewöhnlich viel Zeit. Man legte das Augenmerk auf ein ausgereiftes Konzept, testete verschiedene Ansätze, aber die Implementierung kam einfach nicht auf Touren.

Just in dieser Phase betrat Linux das Feld. Es erfüllte genau die Anforderung der FSF, es war frei, kostenlos und die existierende Software der FSF, allgemein bekannt als GNU-Software, lief ohne wesentliche Änderungen auf dem neuen System. Viele FSF-Anhänger sahen in der Verwendung von Linux den Stein der Weisen.

Fazit der ganzen Geschehnisse war, dass mit Linux bald ein ausgereiftes Unix für den PC, also den Privatbereich, zur Verfügung stand, ebenso wie eine Fülle von Anwendungen, die zumindest aus funktioneller Sicht sich nicht hinter dem Standard kommerzieller Produkte zu verstecken brauchten.

Und das ureigene Betriebssystem des FSF? Dies liegt unterdessen vor. **HURD** verkörpert den Stand der Forschung, es besitzt eine saubere Struktur, einen Mikrokernel und arbeitet sehr schnell. Aus Sicht des Theoretikers sollte es sowohl Windows als auch Linux in den Schatten stellen. Aber HURD hat ein Problem, es kam einfach ein paar Jahre zu spät...

Aber wir möchten den Werdegang von Linux Revue passieren lassen. Beginnend mit dem »Ur-Kernel«, der Version 0.01 streifen wir durch die wichtigsten Stationen und blicken auch voraus, was die Entwickler für die Zukunft für Pläne mit Linux hegen...

Methodik der Versionsnummern

Was hat es mit den Nummern auf sich? Warum gab es die Kernel 1.3 und 2.0, nicht jedoch die Versionen 1.4-1.9? Wer bestimmt, welche Nummer ein Kernel erhält?

Die Nummerierung von Softwareversionen im Linuxumfeld unterscheidet sich häufig etwas von den sonstigen Gewohnheiten. Das liegt in erster Linie daran, dass man schon während der Entwicklungsphase an einem breiten Publikum interessiert ist, das die Software auf Herz und Nieren überprüft. Üblich ist, dass ein produktionsreifes Paket mit der Version 1 herausgegeben wird. Bei freier Software heißt das allerdings nicht, dass Versionen < 1 nicht schon eingesetzt werden können. Oftmals laufen diese schon vollkommen stabil. Dass sie dennoch in ihrer Version den Entwicklerstatus kennzeichnen, liegt zum einen an den noch nicht hinreichenden Tests oder daran, dass noch nicht alle Funktionen enthalten sind, die die Entwickler vorgesehen haben. Und eine »stabile Version« schließt nicht aus, dass das Programm doch einmal den ungewollten Abgang nimmt (wer legt schon seine Hand für sein Produkt ins Feuer?).

Der Linuxkernel folgte bis zur Version 1 ebenso diesem Schema. Stabil waren die meisten der 0.x-Versionen, aber keinesfalls ausgereift genug, um ein einsatzbereites Betriebssystem zu formen. Die erste offiziell stabile Version mit den von anderen Unixen bekannten Eigenschaften war deshalb die Verion 1.0.

Während an den ersten Versionen nur ein kleiner Kreis Entwickler strickte, vermehrte sich die Zahl der Aktivisten stetig. Man suchte nun nach einem Schema, mit dem man einen Entwicklerkernel von einem stabilen Kernel unterscheiden konnte (»Entwicklerkernel« sagt nichts über die tatsächliche Robustheit dessen aus, sondern kennzeichnet diesen nur als »mit Vorsicht zu genießen«).

Schließlich setzte sich für den Kernel eine Versionsbezeichnung bestehend aus **drei Nummern** durch.

Kernel x.y.z

Die **erste Nummer** kennzeichnet eine "epochale" Version und wird nur erhöht, wenn sich Grundlegendes geändert hat.

Kernel x.y.z

Die **zweite Nummer** bezeichnet »wichtige« Neuerungen. I.A. sollte ein Kernel 2.x durch eine Version 2.y beliebig ausgetauscht werden können, ohne dass es das installierte System berührt (leider ändern sich doch und wieder einzelne Schnittstellen, so dass die eine oder andere Komponente den Dienst versagt). Wichtig die zweite Nummer als **Kennzeichnung des Status**. Bei einer **geraden 2. Nummer** handelt es sich um einen **stabilen Kernel**; die **ungerade Zahl** steht einer **Entwicklerversion** voran.

Kernel x.y.z

Die **dritte Nummer** beschreibt schließlich den »Patch« des Kernels. D.h., dass sich an dem Kernel nur »Kleinigkeiten« geändert haben. Meist handelt es sich um ausgemerzte Fehler oder den Ersatz eines Algorithmus durch eine ausgefeiltere Variante.

Und die Vergabe der Nummern?

Diese obliegt den »Verwaltern«, einem engeren Team von Entwicklern rund um Linus Torvald und Alan Cox. Die Diskussion über Kernelneuerungen läuft im Wesentlichen über Mailinglisten ab. Dort werden Vorschläge eingebracht und diskutiert. Absolute Neuerungen wie neueste Treiber liegen zunächst als »inoffizielle« Patches vor. Erlangt eine solche Erweiterung »Produktionsreife«, so wird sie gemeinsam mit anderen als »sinnvoll« erachtete Neuerung zunächst in einen Entwicklerkernel aufgenommen, dessen Versionsnummern (dritte Nummer) fortlaufend hochgezählt werden.

Irgendwann wird es Zeit für einen neue stabile Version. Der derzeitige Entwicklungsstand wird »eingefroren« und in Form von »Prereleases« frei gegeben. Nach einer gewissen Testphase ist es schließlich Zeit, die neue stabile Version zu veröffentlichen. Welche Nummer dieser verpasst wird, hängt letztlich vom Umfang der Neuerungen ab und dies liegt im Ermessen der Projektleiter der Kernelentwicklung.

Der Anfang Version 0.01

Die erste Version von Linux, die Linus Torvalds am 17. September 1991 in die Newsgruppe comp.os.minix postete, enthielt gerade einmal die notwendigsten Funktionen eines Betriebssystems.

Als einzige Plattform lief der Code auf der i386er Architektur; an Hardware wurden einzig IDE-Festplatten, Bildschirm, Tastatur und die serielle Schnittstellen unterstützt.

Als Dateisystem verwendete Linux Version 0.01 Minix in einer multithraeded-Variante, die internen Strukturen waren ansonsten mehr auf Funktionalität als auf Effizienz ausgelegt. So realisierten nur simple Methoden den für Multiprocessing notwendigen Taskwechsel; eine Prioritätensteuerung und Nachrichtenverwaltung fehlte vollkommen. Die frühe Realisierung einer Speicherverwaltung kannte immerhin schon die Verwendung von virtuellem Speicher durch Paging und Segmentierung.

Das gesamte Kernelpaket hatte eine komprimierte Größe von ca. 70 kByte.

Die 0.x Versionen

Die nächsten Versionen 0.10 und 0.11 erschienen binnen weniger Tage noch im Dezember 1991 und im Januar 1992 wurde schließlich mit der Version 0.12 der erste »stabile« Kernel veröffentlicht. Die Änderungen der ersten Versionen waren marginal, rein funktionell waren die Unterstützung von **Ramdisk**, **Floppy** (0.10) und **Swap** (0.12) noch die herausragenden Erweiterungen. In diesen ersten Versionen trimmte man den Kernel zur Zusammenarbeit mit einigen wichtigen GNU-Programmen, so liefen bereits der Compiler »gcc«, die Bourne Again Shell und der Emacs unter Linux Version 0.12.

Der Versionssprung auf 0.95 im März 1992 lässt vermuten, dass man an der schnellen Veröffentlichung einer »fertigen Version« interessiert war. Vielleicht erhoffte man auch hierdurch weitere Entwickler für das Projekt zu begeistern. Zumindest stärkt die folgende Versionsflut eine solche Annahme.

Mit Version 0.96 spendierte man Linux ein eigenes, im Vergleich zu Minix wesentlich effizienteres Dateisystem - das **ext**. Auch kamen die Unterstützung von SCSI, Parallelport, Unix Sockets und dem MSDOS-Dateisystem hinzu.

Alle Kernel der 0.97er Reihe waren als instabil gekennzeichnet. Damalige Neuerungen waren das proc-Filesystem und die Unterstützung der Maus.

Die rasantesten Veränderungen erfuhren offensichtlich die Kernel 0.99.x. Die Versionen 0.99.13 und 0.99.14 wurden sogar in 0.99.13a, 0.99.13b bis 0.99.14z untergliedert, auch die anderen "Zwischenversionen" zierten nicht wenige Buchstaben.

Welche Eigenschaft tatsächlich mit welcher Revision Einzug in den Kernel fand, ist teils unmöglich nachzuvollziehen. Innerhalb der 0.99er Serie enstanden u.a. folgende Erweiterungen:

- als Nachfolger des ext etablierten sich zunächst das **ext2** und das **xiafs**
- **Shared Memory** und **Semaphore** kamen hinzu
- Unterstützung des **TCP/ IP**-Protokollstacks und erster **Ethernet-Karten**
- **Sound**
- **Network File System**
- **Direct Memory Access**
- **HPFS** (OS/2) wurde unterstützt (nur lesender Zugriff)
- **ELF** - das neue Binärformat Executable Linkage Format
- **Virtuelles Dateisystem**

Die 1.x Versionen

1994 hatten die in den 0.99er Kernel eingezogenen Neuerungen einen ausgereiften Stand erreicht und wurden in den im März 1994 erschienenen Kernel Version 1.0 aufgenommen.

Die folgenden Versionen glänzten weniger mit funktionalen Erweiterungen sondern mehr durch interne Umstrukturierungen. Zahlreiche Mechanismen wurden überarbeitet und teils durch effektivere Algorithmen ersetzt. In den Entwicklerversionen konzentrierte man sich auf die Unterstützung des PCI-Busses. In den 1.1er Versionen tauchten die ersten Realisierungen von Modulen auf.

Mit Kernelversion 1.2 wurde im März 1995 Linux auch auf den Architekturen Alpha und Sparc lauffähig. Die PCI-Erweiterungen erreichten die Marktreife und die ersten Treiber konnten als Module realisiert werden, die allerdings noch per Hand zu laden und zu entladen waren.

In der 1.3er Serie schuf man die Grundlagen für den »Quantensprung« auf Kernel 2.

Version 2.0

Im Juni 1996 deutete die Veröffentlichung der Version 2.0 an, dass sich am Design des Kernels eine Menge getan hatte. Der wesentliche Unterschied, der zu Inkompatibilitäten zu den Vorgängerversionen führte, lag in der Multiprozessor-Unterstützung, die die internen Abläufe und Strukturen bisheriger Kernel über den Haufen geworfen

hatte.

Das allein rechtfertigt vielleicht noch keine Erhöhung der Hauptnummer, aber die weiteren Neuerungen waren nicht von minderer Qualität.

So war der neue Kernel auf den Plattformen alpha, i386, mips, ppc, sparc, M68K (Amiga) und Atari lauffähig. Die Module erlebten durch den »kerneld« (Kernel Deamon) eine Aufwertung, da sie nun dynamisch vom Kernel lad- und entladbar waren.

Zum bisherigen Umfang unterstützter Dateisysteme kamen u.a. vfat (Windows95/98), umsdos (Linux auf einem Windows-Laufwerk installieren), Netware Core Protocol (Novell) und SMB (Samba) hinzu. Quotas erlaubten nun das Beschränken von Plattenplatz für einzelne Benutzer.

Neben Überarbeitung der TCP/IP-Implementierung wurden Firewalls, IP-Tunneling, -Masquerading und Multicast-Routing nun direkt vom Kernel unterstützt und verbesserten die Netzwerkfähigkeiten von Linux enorm. Für den Privatanwender war die ISDN-Anbindung ein wesentlicher Aspekt.

Starke Erweiterung erfuhr die Hardwareunterstützung; auch tauchten das Advanced Power Management (APM) und - in späteren Patchversionen - RAID0 im Kernel auf.

Ach ja - fast hätte ich es vergessen - Tux wurde zum offiziellen Linux-Maskottchen erklärt (»Da Pinguine nicht fliegen können, können sie auch nicht abstürzen.«).

Version 2.2

Im Januar 1999 erschien das 11 MByte große Paket des Kernels 2.2.

Konnten bisherige Kernelversionen noch im ursprünglichen Format "a.out" erzeugt werden, müssen die neueren Versionen zwingend im Format Executable Linkage Format (ELF) vorliegen. Zwar können ältere "a.out"-Programme weiterhin verwendet werden, aber die Umstellung auf "glibc2" (neue Version der C-Bibliothek) deutet auf das Aussterben des alten Binaryformats schon hin.

Wesentlich verbessert wurde die Speicherverwaltung, so dass Linux stärker von einem großen Hauptspeicherausbau profitiert. Multiprozessorsysteme skalieren bei bis zu 12 Prozessoren gut, insgesamt werden 64 Prozessoren unterstützt.

Die Behandlung der Module wurde dem "kerneld" entzogen und in einem Kernelthread, den "kmod" realisiert. Hierdurch erreichte man eine beschleunigte Integration eines Moduls in den laufenden Kernel. Gleichsam wurde ein kernelbasierter NFS-Dämon geschaffen, der allerdings an Stabilität nicht dem bisherigen User-Space-NFS-Dämon das Wasser reichen konnte.

Den bisherigen Firewall-Mechanismus "ipfwadm" ersetzte man durch "ipchain"; eine Schnittstelle im proc-Dateisystem erlaubt seitdem das (De)Aktivieren einiger Netzwerkdienste (IP-Forwarding, IPv6...). Eine Überarbeitung erfuhr auch das TCP-Protokoll, das nun "Large Windows" beherrscht. TCP über Satellitenverbindungen ist nun möglich.

Linux spricht nun die Sprachen des Dateisystems CODA und vermag die von Acorn, Apple Mac und WindowsNT zu lesen.

Version 2.4

Nicht grundlos verzögerte sich die Veröffentlichung des neuen Kernels 2.4 mehrfach. Die Umstellung des 32-Bit Dateisystems auf 64 Bit brachte gehörige Probleme mit sich, da zahlreiche Algorithmen einer Neufassung bedurften. Nicht minder weite Kreise zog die Unterstützung von bis zu 64 GByte Hauptspeicher.

Die Liste unterstützter Hardware wurde vor allem mit USB-Geräten weiter angereichert. ISA-Plug&Play-Karten werden nun innerhalb des Kernels behandelt.

Auf Netzwerkebene hat sich für das mit Version 2.2 eingeführte ipchain-Firewall schon wieder eine neue Vorgehensweise eingeschlichen. Dieses Network Packet Filtering erweitert die bisherigen Methoden, stellt allerdings auch die Kompatibilität zu seinen Vorgängern sicher. Intern ist eine Eigenschaft in den Kernel eingeflossen, die es erlaubt, bei einem auf der Netzwerkkarte eintreffenden Paket, einzig den Prozess zu wecken, der den entsprechenden Socket geöffnet hält. Bislang mussten alle Prozesse, die an einem Socket lauschten, aus dem Schlaf gerissen werden, da nur sie entscheiden konnten, an wessen Tür der Postmann klopfte.

Gespannt kann man auch auf den Kernel-Web-Dämon sein, der nach ersten Versuchen einem Webserver einen gravierend gesteigerten Durchsatz ermöglicht.

Ein kleines Manko hat er dann doch, denn das lang ersehnte Journaling Filesystem hat noch immer keinen Platz im Kernelkode erhalten. So ist die einzig relevante Erweiterung im Dateisystem (neben den 64 Bit) der schreibende Zugriff auf OS/2-Partitionen.

Aussichten

Die Version 2.4 ist noch nicht einmal veröffentlicht, da kursieren im Internet schon wieder die Wunschlisten für die kommenden Revisionen. Inwiefern die Vorstellungen tatsächlich Einzug in die Sourcen finden, ist unmöglich abzuschätzen. Dennoch sollen die - meiner Meinung nach - interessantesten Vorschläge diesen Abschnitt ausklingen lassen:

- Kernelunterstützung der Dateisysteme ReiserFS (ab Kernel 2.4.1 enthalten), ext3, XFS und JFS (vergleiche Journaling Filesystems)
- Nanosekunden-Zeitgeber im Kernel (wichtig für Real-Time-Anwendungen)
- Portierung auf NUMA-Architekturen
- Verwendung von SecureRPC

Der Kernel - Architektur

Übersicht

Seitenblick

Der harte Kern

Die Prozessverwaltung

Die Speicherverwaltung

Das Dateisystem

Die Gerätedateien

Das Netzwerk

Die Module

Der Kernel - Module

Übersicht

Mit der Kernel-Version 1.2 hielt mit den Modulen ein Konzept in Linux Einzug, das man in der Art bislang nur von kommerziellen Unix-Systemen kannte.

Musste ein Kernel früherer Version sämtliche Treiber der von ihm zu verwaltenden Hardware fest einkompiliert haben, so wurde es nun möglich, Teile, die während des Bootens nicht zwingend notwendig sind, in separate Einheiten - die Module - auszulagern. Module lassen sich zur Laufzeit, wenn der Kernel die Eigenschaft benötigt, zum Kernel hinzufügen oder, wenn ein Treiber ausgedient hat, auch wieder aus der aktuellen Kernelkonfiguration entfernen.

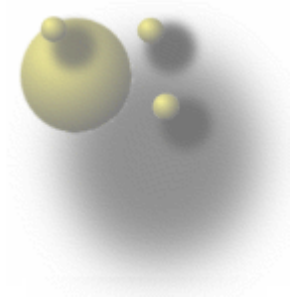
Module mildern einige Probleme.

So konnte man ohne sie keinen »universellen« Kernel erzeugen, wie er z.B. bei einer Installation benötigt wird, um alle erdenklichen Hardwarekonstellationen abzudecken. Eine Installation ging fast immer mit einer anschließenden Kernelkompilation einher, eine Tatsache, die inbesondere bei Anfängern den Ruf von Linux als ein schwer zu konfigurierendes System schürte. Das modulare Design erlaubt nun automatisches Erkennen von Hardware und das dynamische Hinzufügen der Treiber.

Nicht selten schmücken heutige Rechner Soundkarte, TV-Karte, Modem, Netzwerkkarten, Adapterkarten... Nun benötigt jede davon Ressourcen, aber diese sind, vor allem in Form von Interrupts und IO-Adressen, nur begrenzt verfügbar. Nicht selten verweigerte ein Gerät die Arbeit, weil »seine« Ressourcen von einem anderen genutzt wurden. Module ermöglichen einen Ausweg. Zwar können zwei Geräte, die partout auf denselben Einstellungen beharren, nicht zur gleichzeitigen Arbeit bewegt werden, aber eine wechselseitige Verwendung ist durch Laden und Entladen der jeweiligen Treiber durchaus möglich.

Und schließlich lässt sich der Treiber in Form eines Moduls separat kompilieren. Ein neuer Kernel ist nicht nötig.

Statisch vs. modular



Bei all den einführenden Lobeshymnen auf den modularen Kernel fragt man sich, ob die Kerneldiät auch Grenzen kennt? Ja, die gibt es!

Module müssen vor ihrer Verwendung geladen werden. Und woher nimmt der Kernel die Module? Aus einem erreichbaren Verzeichnis. Das Verzeichnis liegt auf einem Speichermedium. Der Kernel benötigt einen Treiber, um auf dieses zugreifen zu können. Also muss dieser fest in den Kernel integriert sein.

Es lassen sich weitere Beispiele von Treibern finden, die der Kernel bereits während des Ladevorgangs benötigt. Sollten Sie in die Situation gelangen, einen eigenen Kernel zu erzeugen, so stehen Sie vielfach vor der Qual der Wahl. Statisch oder modular?

Häufig nimmt das Konfigurationstool Ihnen die Entscheidung ab, indem die modulare Realisierung eines Treibers deaktiviert ist. Wo Sie die Wahl haben, sollten Sie sich fragen, ob der Kernel diesen Treiber schon beim Booten benötigt. Mounten Sie z.B. das Rootverzeichnis via NFS, so darf dieser Treiber keinesfalls als Modul vorliegen.

Alle notwendigen Eigenschaften statisch in den Kernel einzufügen, scheitert meist an der resultierende Größe dessen. Wenn der Kernel in den Hauptspeicher geladen wird, befindet sich der Prozessor noch im sogenannten Real-Modus. Somit kann nur RAM unterhalb der 1MB Grenze adressiert werden. Der Kernel muss in diesen passen!

Auch die Vermutung, ein als Modul arbeitender Treiber verrichtet seine Tätigkeit weniger effizient als sein »fest angestellter Kollege«, ist falsch. Nur der erste Zugriff auf das Gerät wird, bedingt durch den jetzt anstehenden Ladevorgang, eine minimale Verzögerung mit sich bringen. Ist er einmal drin, ist kein Unterschied mehr festzustellen.

Manueller Umgang mit Modulen

Während der Kernel ein benötigtes Modul unverzüglich automatisch nachlädt, erfolgt das Entladen erst nach expliziter Aufforderung. D.h. entweder entledigt man sich des unnötigen Ballasts manuell oder man lässt den cron regelmäßig die Maßnahmen anstoßen.

Zum Entladen eines Moduls ist also echte Handarbeit seitens des Administrators gefragt. Aber auch das Laden kann rein manuell erfolgen. Die benötigten Werkzeuge findet man im Paket »modutils«.

Zeige her, was Du geladen hast!

Zunächst dient das Kommando **lsmod** zur Anzeige aller momentan geladenen Module:

```
user@sonne> /sbin/lsmod
Module                  Size  Used by
ppp_deflate            40300   0 (autoclean)
bsd_comp                4052   0 (autoclean)
ppp                    20876   0 (autoclean) [ppp_deflate bsd_comp]
slhc                    4440   0 (autoclean) [ppp]
snd-pcm-oss            16872   0 (autoclean)
snd-mixer-oss           4308   2 (autoclean) [snd-pcm-oss]
snd-card-cs461x         2008   2
snd-timer               8064   0 [snd-pcm]
snd-rawmidi             9112   0 [snd-card-cs461x]
soundcore               2564   4 [snd]
parport_pc              7568   0 (unused)
parport                 7464   0 [parport_pc]
rtl8139                12224   1 (autoclean)
memstat                 1476   0 (unused)
serial                 42484   1 (autoclean)
```

lsmod tut nichts anderes, als die Daten aus der Datei /proc/modules auszulesen (vergleiche Prozessdateisystem).

Raus mit dem Modul!

Das Kommando **rmmod** dient zum Entladen eines Moduls aus einem laufenden Kernel. Ein Modul kann nur entladen werden, wenn es zum einen nicht benutzt wird und zum anderen auch nicht von einem anderen Modul benötigt wird.

In Bezug auf obiges Beispiel (Ausgabe von lsmod) kann ein Modul nur entladen werden, wenn im Feld »Used by« **unused** steht:

```
root@sonne> /sbin/rmmod snd-timer
snd-timer: Das Gerät oder die Ressource ist belegt
root@sonne> /sbin/rmmod serial
snd-timer: Das Gerät oder die Ressource ist belegt
root@sonne> /sbin/rmmod parport_pc
root@sonne>
```

Lade das Modul!

Eine Möglichkeit, ein Modul per Hand zu laden, bietet das Kommando **insmod**. Als Argument übergibt man dem Kommando mindestens den Modulnamen. Und wie heißt es nun? Alle Module werden unterhalb des Verzeichnisses /lib/modules/*Kernel-Version*, geordnet nach ihrer Verwendung, abgelegt. Der Bezug auf eine konkrete Kernelversion ermöglicht die gleichzeitige Unterstützung mehrerer Kernel. Sicher ist das eher für den Kernelentwickler von Bedeutung... der normale Anwender wird zwar verschiedene Kernel, selten aber Kernel unterschiedlicher Versionen verwenden.

```
user@sonne> ls / lib/ modules/ 2.2.17
block  fs       ipv4  misc        net     scsi  video
cdrom  ieee1394 ipv6  modules.dep pcmcia  usb
user@sonne> ls lib/ modules/ 2.2.17/ block
DAC960.o  cpqarray.o  nbd.o  xd.o
```

Doch zunächst **zur Arbeitsweise von insmod**. Module liegen in Form von Objektcode vor (Endung .o). Dieser Objektcode enthält genau jene Symbole, die der Kernel in seiner Symboltabelle zum Zugriff auf das jeweilige Device verwendet.

insmod muss nun das Modul finden und befolgt dabei eine bestimmte Suchreihenfolge. Zunächst beachtet es den Inhalt der Shellvariablen **MODULECONF**, die den Pfad zu einer Konfigurationsdatei enthält. Ist die Variable nicht gesetzt, versucht das Kommando, die Informationen der Datei /etc/module.conf zu entnehmen (dazu später mehr). Ist diese Datei nicht vorhanden, sucht **insmod** in dem durch die Shellvariablen **MODULEPATH** vorgegebenen Verzeichnis. Ist diese Variable ebenso nicht existent, werden Standardverzeichnisse durchsucht. Welche das konkret sind, ist fest in das Programm einkompiliert und kann nur durch Änderung in den Quellen mit anschließender Neuübersetzung beeinflusst werden.

Der Name des Moduls muss ohne den Suffix ».o« angegeben werden, da »insmod« diesen selbsttätig ergänzt.

Von den zahlreichen Optionen seien nur die - unserer Meinung nach - wichtigen vorgestellt:

-f

Es wird versucht ein Modul zu laden, selbst wenn seine Version von der des Kernels abweicht.

-k

Das »autoclean«-Flag wird gesetzt. Näheres dazu unter Automatisiertes Entladen.

-p

Das Laden wird nur simuliert; somit kann getestet werden, ob das Modul geladen werden kann.

-q

Fehlermeldungen werden unterdrückt. Der Erfolg oder Misserfolg des Kommandos kann anhand seines Rückgabewertes verifiziert werden.

-r

Lässt auch das Laden eines Moduls zu, wenn dieses nicht Root gehört. Aus Sicherheitsgründen sollte ein Mo immer Root gehören.

-s

Alle Ausgaben gehen an den syslogd.

-v

Erweiterte Ausgabe

Des Weiteren lassen manche Module die Übergabe von Parametern zu. Verbreitet sind die Angabe von IO-Port-Adressen und Interrupt-Nummern. Als Beispiel lädt der nachfolgende Aufruf das Modul für eine Realtek-Netzwerkkarte, wobei explizit der Interrupt 11 zugewiesen wird:

```
root@sonne> insmod rtl8139 irq=11
Using /lib/modules/2.2.16/net/rtl8139.o
```

Eine solche Zuweisung überschreibt eventuelle Einstellungen in der Datei modules.conf. Stimmen die Parameter

der Kommandozeile nicht, ist häufig die Ausgabe *Device or resource busy* zu lesen. Leider trifft die Fehlermitteilung selten den Kern der Sache...

Teste und Lade!

insmod wird seine Aufgabe verweigern, wenn das zu ladende Modul die Existenz anderer Module bedingt, jene aber noch nicht geladen wurden. Per Hand ließe sich das Sorgenkind schnell hinzufügen, doch bedeutet das zum einen unnötigen Aufwand und zum anderen verbietet es die Verwendung von **insmod** in Skripten, da Fehler quasi vorprogrammiert sind.

Der Ausweg kommt in Form von **modprobe** daher, das für sich ein intelligentes **insmod** darstellt, da es Abhängigkeiten zwischen den Modulen entdecken und auch auflösen kann (tatsächlich ruft es insmod zum eigentlichen Laden auf; daher können Sie - analog zu insmod - Parameter an modprobe übergeben). Natürlich spekuliert modprobe nicht ins Blaue hinein, sondern konsultiert eine kleine Datenbank **modules.dep**, die (hoffentlich) die notwendigen Informationen bereit hält. Zu dieser Datenbank folgt im Anschluss mehr...

Stellt also **modprobe** fest, dass ein Modul weitere Module für seine Funktionsfähigkeit bedingt, so wird es diese in der durch die Datenbank vorgegebenen Art und Weise vor dem »eigentlichen« Modul laden. Jetzt kann es aber sein, dass die Aufnahme eines der Module scheitert (z.B. weil die angeforderten Ressourcen belegt sind). **modprobe** wird nun alle in den bisherigen Schritten geladenen Module entfernen, denn diese sind ja nun nutzlos geworden, da nicht alle Abhängigkeiten des Zielmoduls erfüllt sind.

Eine nützliche Eigenschaft, die vor allem während des Bootvorgangs angewandt wird, ist die Möglichkeit, alle oder auch nur das erste funktionierende Modul aus einem Verzeichnis zu laden. Da die Module nach ihrem Einsatz gruppiert sind (Netzwerkmodule liegen in einem Verzeichnis, die der Soundkarte in einem anderen...), genügt die Angabe eines Verzeichnisnamens und **modprobe** wird der Reihe nach die Module testen, bis das Laden eines Moduls gelingt. Liegen für alle nur erdenklichen Hardwareelemente die Module vor, so kann für jede Hardware quasi automatisch der richtige Treiber eingebunden werden.

Wiederum sei die Beschreibung der Optionen auf die in der Praxis relevanten beschränkt:

-a

Alle funktionsfähigen Module (aus einem Verzeichnis) werden geladen (anstatt nach dem ersten zu stoppen

-c

Die aktuelle Konfiguration wie Aliasnamen für Module, Parameter usw. wird angezeigt.

-k

Das »auotclean«-Flag wird gesetzt. Näheres dazu unter Automatisiertes Entladen.

-n

Simuliert den Ladevorgang.

-r

Entfernt ein Modul und alle, von denen es abhängt.

-s

Lenkt die Ausgaben zum syslogd.

-t Typ

Nur Module des angegebenem Typs werden betrachtet.

Als Beispiel soll ein Modul für die Netzwerkkarte geladen werden (zunächst eines; dann alle):

```
root@sonne> modprobe -t net \*
root@sonne> modprobe -a -t net \* 2>&1| head -5
/lib/modules/2.2.16/net/eepro100.o: init_module: Das Gerät oder die Ressource ist belegt
/lib/modules/2.2.16/net/eepro100.o: insmod /lib/modules/2.2.16/net/eepro100.o failed
/lib/modules/2.2.16/net/tulip.o: init_module: Das Gerät oder die Ressource ist belegt
/lib/modules/2.2.16/net/tulip.o: insmod /lib/modules/2.2.16/net/tulip.o failed
/lib/modules/2.2.16/net/olympic.o: init_module: Das Gerät oder die Ressource ist belegt
```

Abhängigkeiten

modprobe benötigt für seine Arbeit eine Datenbank **modprobe.dep**. Diese sollte bei den gängigen Distributionen im Verzeichnis »/lib/modules/*<Kernelversion>*« liegen.

Für jedes Modul sind dort die Abhängigkeiten aufgeführt. Da die Datenbank im ASCII-Format vorliegt, lässt sie sich sogar von Hand anpassen. Allerdings sollte das besser das Kommando **depmod** übernehmen, denn genau das ist dessen Berufung.

In den meisten Fällen genügt ein einfacher Aufruf:

```
root@sonne> depmod
depmod: *** Unresolved symbols in /lib/modules/2.2.16/pcmcia/mpsuni_cs.o
```

Eine neue Datei modules.dep sollte nun existieren (die produzierte Fehlermeldung ist unwichtig, da auf dem System kein PCMCIA vorhanden ist).

In die Versuchung **depmod** zu rufen, werden Sie ohnehin nur gelangen, falls Sie dem System ein neues Modul spendieren (indem Sie eines einspielen oder ein existierendes neu übersetzen). Nahezu jede Linuxdistribution hat den Aufruf in einem der Bootskripte versteckt, um die Datenbank bei jedem Bootvorgang automatisch zu aktualisieren. Dennoch sollen einige der gängigen Optionen das Thema beschließen:

-a

Die Module werden in den durch die Datei /etc/modules.conf benannten Verzeichnissen gesucht (ansonster default-Verzeichnissen).

-A

modules.dep wird nur aktualisiert, wenn sich tatsächlich etwas geändert hat (Kriterium ist der Zeitstempel).

-n

Anstatt modules.dep zu erstellen, wird das Ergebnis auf die Standardausgabe geschrieben.

-s

Die Fehlerausgaben landen beim syslogd.

Die Datei modules.conf

Vorab: Sie sollten sich nicht wundern, falls diese Datei in Ihrem System nicht existiert. Der frühere Name dieser Konfigurationsdatei lautete conf.modules und obwohl die Benennung als veraltet verschrien ist, findet man sie noch bei zahlreichen Distributionen.

modules.conf ist sowohl für insmod, modprobe als auch für depmod interessant, da hier u.a. die Parameter enthalten sind, mit denen ein Modul zu laden ist. Sollten Sie einmal neue Hardware ins System integrieren, so sollten Sie zunächst diese Datei konsultieren und durch Aufnahme einiger Zeilen das System zur Zusammenarbeit mit ihrer Hardware überreden. Mitunter finden Sie bereits die passenden Einträge vor, die Sie nur noch »entkommentieren« müssen.

Wie in nahezu allen Konfigurationsdateien leitet auch in modules.conf das Doppelkreuz # einen Kommentar ein; lange Zeilen können mit einem Backslash »umgebrochen« werden. Die resultierenden Zeilen besitzen folgende Struktur:

Parameter= Wert

Einige Variablen werden hiermit belegt. Dabei handelt es sich um **depfile= ...**, das den vollständigen Pfad z Datei mit den Abhängigkeiten »modules.dep« enthält (benötigt von depmod und modprobe). Über **insmod_opt= ...** lassen sich Kommandozeilenoptionen spezifizieren, die dem Kommando insmod mitgegeb werden.

Etwas aufwändiger sind dagegen die Angaben zur Variablen **Path= ...**, die einen Pfad enthält, wo die Modu suchen sind. Mit der Angabe eines optionalen »Tags« kann der Pfad für einen Modultyp explizit verändert werden. So setzt **Path[net]= ...** die Pfade zu Modules, die die Netzwerkfunktionen betreffen. Ohne den Ta Path gleichbedeutend mit Path[misc].

Die Pfadangaben dürfen sogar Metazeichen der Shell enthalten; so ist die nachfolgende Zuweisung legitim:

```
path[boot]=/lib/modules/$(uname -r)
path[misc]=/lib/modules/2.2.1?
```

Alle Parameter dürfen auch wiederholt in der Datei definiert werden; die originalen Einstellungen werden da verworfen.

keep

Ab der Zeile, in der dieses Schlüsselwort erscheint, ergänzen alle Zuweisungen zu **Path= ...** bzw. **Path[...]=** den alten Inhalt der Variablen. Ohne »keep« werden die Inhalte überschrieben.

[add] options < Modulname> < Symbol> = < Wert>

Sollen einem Modul beim Laden besondere Optionen mitgeteilt werden, so sind diese anzugeben. Ein einleitendes **add** fügt die neuen Optionen zu schon bestehenden hinzu; ohne »add« wird der alte Inhalt ersetzt. **Symbol** und **Wert** sind modulabhängig und müssen der entsprechenden Kerneldokumentation entnommen werden. Als Beispiel werden Interrupt und IO-Port-Adresse der Ethernetkarte gesetzt:

```
options eth0 irq=11
add options eth0 io=0x378
```

Die beiden Anweisungen im Beispiel hätten auch auf eine Zeile geschrieben werden können:

```
options eth0 irq=11 io=0x378
```

alias < Aliasname> < Modulname>

Oft ist es einleuchtender, ein Modul unter einem Aliasnamen anzusprechen. Ein Modul kann auch unter mehreren Aliasnamen angesprochen werden. Beachten Sie, dass Optionen, die unter einen Aliasnamen defi wurden, auch nur beim Laden des Moduls unter diesem Namen zur Geltung kommen.

```
alias eth0 rtl8139
```

Steht als letztes Wort einer mit alias beginnenden Zeile **off**, so werden Versuche, die benannten Module zu laden, immer fehlschlagen. Steht als letzter Eintrag **null**, so wird der Ladeversuch mit einer Erfolgsmeldung (Rückgabewert) enden, obwohl tatsächlich kein Modul geladen wurde. Dieses Vorgehen kann notwendig werden, wenn sich Module gegenseitig bedingen.

install < Modulname> < Kommando>
remove < Modulname> < Kommando>

pre-install <Modulname> <Kommando>
pre-remove <Modulname> <Kommando>
post-install <Modulname> <Kommando>
post-remove <Modulname> <Kommando>

Diese vier Einträge ermöglichen die eigentliche Definition der Abhängigkeiten. Je nach einleitendem Schlüsselwort wird in Bezug auf das angegebene Modul das Kommando vor/nach dem Laden/Entladen ausgeführt. Bedingt bspw. ein Modul ein anderes, so kann dieses in einer **pre_install**-Zeile wie folgt formu werden:

```
pre-install wavefront modprobe "-k" "cs4232"
```

Zusätzlich lassen sich Teile der Datei in Abhängigkeit von einer Bedingung ausgeführt werden, indem folgende **if**-Syntax angewandt wird:

```
if <Bedingung>
  ...
[elseif <Bedingung> ]
  ...
[else]
  ...
endif
```

Solche Bedingungen können der Test auf Existenz einer Datei »-f Datei«, auf ein gesetztes »autoclean«-Flag »-k« oder ein Vergleich von Werten sein. Wert ist hierbei das Resultat einer Expansion analog zum Vorgehen in der Bash. In der Voreinstellung arbeitet ein Vergleich auf Zeichenkettenbasis; durch ein vorangestelltes »-n« kann aber eine numerische Auswertung erzwungen und mit »!« das Resultat negiert werden.

Automatisches (Ent)Laden von Modulen

Laden bei Bedarf

Mit dem Kernel Version 2.0 wurde ein »user space« Dämon geliefert, der bei Bedarf vom Kernel angeforderte Module nachladen konnte. Dieser **kerneld** wurde ab Kernel 2.2 durch den nun direkt im Kernel integrierten **kmod** ersetzt. Die Vorteile der Realisierung als Kernelthread sind:

- die Interprozesskommunikation (SysV IPC) kann nun als Modul realisiert werden, da der **kmod** diese - im Gegensatz zum kerneld - nicht verwendet
- Der Kommunikationsfluss »Kernel<=>kerneld<=>modprobe« wurde auf »Kernel<=>modprobe« verkürzt (dadurch Beschleunigung des Ladevorgangs)
- der Quellkode von IPC wurde einfacher
- **kmod** protokolliert Fehler über den **klogd**

Die Arbeitsweise von **kerneld** und **kmod** sind identisch; beide werden, wenn der Kernel ein Modul benötigt, von diesem geweckt und versuchen ihrerseits das angeforderte Modul mit Hilfe von **modprobe** zu laden.

Um die Automatisierung des Modulladens verwenden zu können, ist einzig die Unterstützung durch den Kernel erforderlich. Bejahen Sie bei der Konfiguration dessen die Optionen »Enable loadable module support« und »Kernel module loader support« so wie in folgender Abbildung dargestellt.

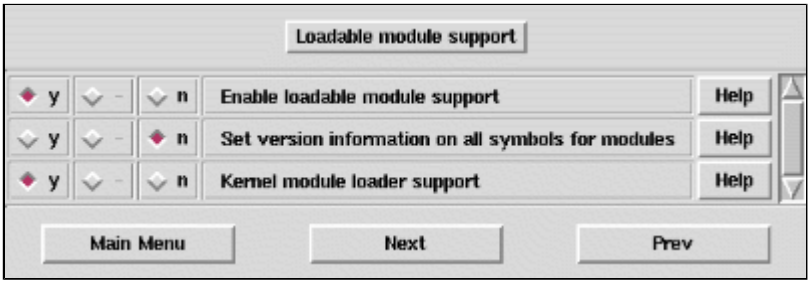


Abbildung 1: Modul-Unterstützung im Kernel

Automatisiertes Entladen

Von sich aus entlädt der Kernel Module niemals, allerdings genügt ein Eintrag im Terminkalender des cron, um das Vorgehen zu automatisieren.

Zunächst sollte klar sein, dass ein Modul nur entladen werden kann, wenn es den Status **unused** aufweist (Ausgabe von "lsmod"). Des Weiteren darf kein weiteres Modul dieses Modul bedingen (keine Abhängigkeiten). Und als letzte Voraussetzung, muss ein automatisch entladbares Modul als **autoclean** gekennzeichnet sein. Letzteres erreicht man durch Laden eines Modules mittels:

```
root@sonne> insmod -k <Modulname>
# oder:
root@sonne> modprobe -k <Modulname>
```

Das Entladen aller "autoclean unused" Module veranlasst das Kommando **rmmod -a**. Zumindest laut Manual Page sollte ein Aufruf genügen. Er tut es aber nicht!

Selbst wenn ein Modul als »autoclean unused« markiert ist, muss es zuvor auch verwendet worden sein, bevor »rmmod -a« dieses überhaupt in Betracht zieht! Des Weiteren sind **zwei** Aufrufe von »rmmod - a« zwingend erforderlich. Der erste ist der Richter: »Verurteilt zum Entladen!«; der zweite der Vollstrecker: »Raus damit!«.

Um letztlich ein nicht mehr verwendetes Modul per Cronjob zu verbannen, sähe ein Eintrag in die »crontab« wie folgt aus:

```
# Eintrag in /etc/crontab, der unbenutzte Module nach 10 Minuten rauswirft (nach 5 Minuten markieren, nach 10 entladen):
*/5 * * * * root /sbin/rmmod -a
```

Der Kernel - Das Prozessdateisystem

Übersicht

Beim Prozessdateisystem handelt es sich um ein virtuelles Dateisystem. D.h. es existiert nur während der Laufzeit von Linux, es beansprucht keinen Speicherplatz auf der Festplatte, es liegt im Hauptspeicher und besitzt mit dem Verzeichnis »/proc« einen definierten Eintrittspunkt.

Das Prozessdateisystem ist ein Spiegel des Systems und ordnet die Informationen des Kernels in einer hierarchischen Struktur an. Aus ihm lassen sich etliche Informationen gewinnen und - in gewissen Grenzen - die Arbeit des Kernels modifizieren.

Das Prozessdateisystem wurde speziell für die x86er-Architektur entwickelt und steht deswegen unter Linux-Implementierungen auf anderen Hardwaretypen nicht zur Verfügung.

Informationen über das System

Nicht nur dem Administrator stehen mit dem Prozessdateisystem allerlei Informationen über den Zustand des Systems zur Verfügung:

- Zu jedem aktiven Prozess des Systems sammelt der Kernel im Prozessdateisystem alle relevanten Informationen, wie Rechendauer, aktueller Zustand, Speicherverbrauch, seine Kommandozeilenoptionen,...
- Die Einstellungen der Hardware (IO-Basisadressen, Interrupts,...)
- Die Parameter des Netzwerks (Statistiken)
- Speicherauslastung, Systemauslastung
- Geöffnete Dateien, Dateisperren, ...

Etliche Kommandos, die Informationen aus den oben genannten Bereichen darstellen, gewinnen diese aus dem Prozessdateisystem. Andere Kommandos wiederum kommunizieren direkt mit dem Kernel.

Der Inhalt des Prozessdateisystems

Informationen zu einem Prozess

Ein Blick in das Verzeichnis »/proc« bringt eine Reihe von Verzeichnissen, die mit Ziffern benannt sind, zu Tage. Eine Ziffer entspricht dabei genau der Nummer (PID) eines Prozesses. Und jeder aktive Prozess ist durch sein eigenes Verzeichnis vertreten. Jedes der Unterverzeichnisse enthält diegleichen Einträge:

```
user@sonne> ls -l /proc/1
ls: /proc/1/exe: Keine Berechtigung
ls: /proc/1/root: Keine Berechtigung
ls: /proc/1/cwd: Keine Berechtigung
insgesamt 0
-r--r--r--   1 root     root            0 Jul  5 15:58 cmdline
lrwx------   1 root     root            0 Jul  5 15:58 cwd
-r--------   1 root     root            0 Jul  5 15:58 environ
lrwx------   1 root     root            0 Jul  5 15:58 exe
dr-x------   2 root     root            0 Jul  5 15:58 fd
pr--r--r--   1 root     root            0 Jul  5 15:58 maps
-rw-------   1 root     root            0 Jul  5 15:58 mem
lrwx------   1 root     root            0 Jul  5 15:58 root
-r--r--r--   1 root     root            0 Jul  5 15:58 stat
-r--r--r--   1 root     root            0 Jul  5 15:58 statm
-r--r--r--   1 root     root            0 Jul  5 15:58 status
```

Auch wenn das Kommando **ls** jedem Eintrag die Speichergröße »0« bescheinigt, sind die Dateien keineswegs leer:

```
user@sonne> cat / proc/ 1/ status
Name:   init
State:  S (sleeping)
Pid:    1
PPid:   0
Uid:    0       0       0       0
Gid:    0       0       0       0
Groups:
VmSize:       372 kB
VmLck:          0 kB
VmRSS:        204 kB
VmData:        24 kB
VmStk:          4 kB
VmExe:        332 kB
VmLib:          0 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 7fffffffd7f0d8fc
SigCgt: 00000000280b2603
CapInh: 00000000fffffeff
CapPrm: 00000000ffffffff
CapEff: 00000000fffffeff
```

Die Einträge enthalten im Einzelnen:

cmdline

Argumente der Kommandozeile

cwd

Link zum Arbeitsverzeichnis des Prozesses

environ

Werte der Umgebungsvariablen

exe

Link zum Programm

fd

Verzeichnis, dass die offenen Dateideskriptoren des Prozesses beinhaltet

maps

Speichermapping

mem

Speicherbedarf des Prozesses

root

Link zum Wurzelverzeichnis

stat

Prozessstatus

statm

Informationen zum belegten Speicher

status

Wie »stat«, nur in tabellarischer Form

Die Kerneldaten

Direkt unterhalb von »/proc« finden sich allerlei Dateien, die teilweise identische Namen haben, wie sie bei den einzelnen Prozessen auftauchten. Etwas oberflächlich betrachtet, ist der Kernel ja auch nichts anderes als ein Programm und da er aktiv ist, weiß er allerlei zu berichten:

apm

Die Statistiken des Power Managements

cmdline

Die Kommandozeilenargumente des Kernels

cpuinfo

Informationen zum Herz des Rechners

devices

Dem Kernel bekannte Devices

dma

Benutzte DMA-Kanäle

filesystems

Unterstützte Dateisysteme

interrupts

Belegte Interrupts

ioports

Verwendete IO-Ports

kcore

Zugang zum Arbeitsspeicher

kmsg

Protokolldatei des Kernels (siehe Protokollierung - klogd)

ksyms

Die Symboltabelle des Kernels (Zuordung von Adressen zu Funktionsnamen)

loadavg

Systemlast

locks

Sperren des Kernels

meminfo

Auslastung des Speichers

misc

Statistiken, die anderen Bereichen nicht zuzuordnen sind

modules

Alle geladenen Module

mounts

Gemountete Dateisysteme

partitions

Alle bekannten Partitionen

pci

Aktivierte PCI-Geräte

rtc

Real Time Clock

slabinfo

Slab Pool Informationen

stat

Allgemeine Statistiken

swaps

Swap-Auslastung

uptime

Zeitdauer, die das System aktiv ist u.a.

version

Kernelversion und Übersetzungsdatum

Viele der Informationen stehen auch dem normalen Benutzer zur Verfügung. Als Beispiel lesen wir die Daten zur CPU aus:

```
user@sonne> cat / proc/ cpuinfo
processor       : 0
vendor_id       : AuthenticAMD
cpu family      : 5
model           : 8
model name      : AMD-K6(tm) 3D processor
stepping        : 12
```

```
cpu MHz         : 451.035904
fdiv_bug        : no
hlt_bug         : no
sep_bug         : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr mce cx8 sep mtrr pge mmx 3dnow
bogomips        : 897.84
```

IDE-Geräte

Die Informationen zu den IDE-Geräten befinden sich im Verzeichnis "/proc/ide". Zunächst enthält die dortige Datei "drivers" die dem Kernel bekannten IDE-Treiber:

```
user@sonne> cat / proc/ ide/ drivers
ide-floppy version 0.9
ide-tape version 1.13
ide-cdrom version 4.54
ide-disk version 1.09
```

Weiterhin existieren Unterverzeichnisse, deren Namensgebungen der Nummerierung der IDE-Geräte im System entsprechen und es existieren Links auf das jeweilige Verzeichnis, deren Namen wie das zugehörige Device lauten. Zum Beispiel ist die erste IDE-Festplatte das erste IDE-Gerät im System und wird durch das Verzeichnis »ide0« repräsentiert. Auf dieses existiert der Link »hda«. Wechseln wir in das Verzeichnis der ersten IDE-Festplatte:

```
user@sonne> ls / proc/ ide/ ide0/
channel  config  hda  mate  model
```

channel ist der verwendete IDE-Kanal (0 oder 1), **config** enthält bei PCI-Bridges die Konfigurationsdaten, **mate** beinhaltet Namen verwandter IDE-Geräte und **model** ist der Typ des Gerätes (z.B. »pci«). Weiterhin existiert ein Unterverzeichnis **hda**, das die Details zum Gerät kennt:

cache

Cache-Speicher (nur bei Festplatten)

capacity

Speicherkapazität des Gerätes

driver

Treibername und -version

geometry

Plattengeometrie (nur bei Fetsplatten)

identify

Block, anhand dessen das Gerät erkannt wird

media

Mediumtyp (disk, cdrom, ...)

model

Kennung des Gerätes

settings

Einstellungen des Gerätes

smart_thresholds

Schranken des IDE-Disk Managements

smart_values

Werte des IDE-Disk Managements

Netzwerk

Im Unterverzeichnis »/proc/net« finden Sie die Statistiken und Informationen zum Netzwerk:

```
user@sonne> ls / proc/ net/
arp        igmp        ip_masquerade  rarp   rt_acct   tcp
dev        ip_fwchains  netlink       raw    rt_cache  tr_rif
dev_mcast  ip_fwnames   netstat       route  snmp      udp
dev_stat   ip_masq      psched        rpc    sockstat  unix
```

arp

ARP-Tabelle des Kernels (Zuordnung von IP-Adressen zur Hardwareadresse)

dev

Statistiken der Netzwerk-Devices

dev_mcast

Multicast-Gruppen, die an einem Device »lauschen«

dev_stat

Status der Netzwerk-Devices

igmp

Tabelle des Internet Group Management Protocol (für Routing)

ip_fwchains

Firewall IP-Forwarding Regeln

ip_fwnames

Namen der Firewall-Listen

ip_masq

Unterverzeichnis mit Tabellen für das Masquerading (Maskieren einer privaten IP-Adresse durch eine öffentliche), aktive »Maskierungen«

ip_masquerade

netlink

Liste der Netlink Sockets

netstat

Netzwerk Statistiken

psched

Parameter des Paket Schedulers, der die Pakete filtert, um eine dedizierte Auswertung des Datenaufkomme vornehmen zu können

rarp

Kerneltabelle des Reverse ARP (nur für plattenlose Rechner relevant, die ihre IP-Adresse von einem Server anhand ihrer Hardwareadresse anfordern)

raw

Statistiken des raw-Devices

route

Routing Tabelle des Kernels

rpc

Verzeichnis mit Informationen aktiver RPC-Dienste

rt_cache

Cache mit Routing-Informationen

snmp

Daten (Managements Information Bases) des Simple Network Management Protocols

sockstat

Statistiken aller Sockets

tcp

Tabelle der TCP Sockets

tr_rif

Token Ring Routing Tabelle

udp

Tabelle der UDP Sockets

unix

Tabelle der geöffneten Unix Domain Sockets mit Refernzzähler, Protokoll, Flags, Typ, Status, Inodenummer Pfad

Parallele Schnittstellen

Informationen zu den parallelen Schnittstellen des System werden im Verzeichnis »/proc/parports« gehalten. Zu jeder existierenden Schnittstelle existiert ein Unterverzeichnis (Benennung »0«, »1«,...),

autoprobe

Informationen, die automatisch erworben wurden

devices

Liste der Devices, die den Port verwenden

hardware

IO-Port-Adresse, (default) Interruptnummer, DMA-Kanal und Modus

irq

Interruptnummer (überschreibt die Nummer in "hardware")

SCSI-Geräte

Alle im System aktiven SCSI-Adapter erhalten ein eigenes Unterverzeichnis mit allen angeschlossenen Geräten. Diese Dateien liefern Informationen (Parameter) und Statistiken zum jeweiligen SCSI-Gerät.

Die Datei »/proc/scsi/scsi« beinhaltet eine Liste aller erkannten SCSI-Geräte des Systems.

System

Im Unterverzeichnis «/proc/sys» befinden sich verschiedene Kernelinformationen. Die Einordnung dieser in einen eigenen Verzeichnisbaum könnte durch die Möglichkeit motiviert worden sein, durch manuelle Änderung dieser Parameter Einfluss auf den aktiven Kernel nehmen zu können.

Dieser »heiklen« Methode der Konfiguration werden wir uns im nächsten Abschnitt annehmen. Hier soll nur eine knappe Beschreibung der enthaltenen Dateien und Verzeichnisse erfolgen (diese können sich von Kernelversion zu Kernelversion teils stark unterscheiden).

dev

Die Parameter des CDROM-Treibers stehen hier (andere Treiber sollen in späteren Kernelversionen Unterstützung erfahren)

fs

Informationen zu einigen unterstützten Dateisystemen findet man hier

kernel

Hier kann direkt am Kernel geschraubt werden, so lassen sich Rechnername, NIS-Domainname, die Menge Debugausgaben u.v.m. manipulieren

net

Hierunter finden sich Dateien, über die sich einige Netzwerkparameter abstimmen lassen

sunrpc

Das Debuggen einer RPC-Dientse kann an- bzw. abgestellt werden

vm

Die hier enthaltenen Dateien steuern das Verhalten der virtuellen Speicherverwaltung

Terminals

Die aktiven und verfügbaren Terminals (tty's) sind im Verzeichnis »/proc/tty« enthalten.

driver

Verzeichnis, das die Datei »serial« enthält, in der wiederum die Zugriffsstatistiken stehen

drivers

Liste der Terminal-Treiber

ldiscs

???

Manuelle Eingriffe

Stellen Sie sich vor, Sie müssten ein Linux-System auf einen Webserver aufspielen. Die Forderung seitens Ihres Chefs lautet, dass gleichzeitig 1000 Zugriffe bedient werden müssen. Ihre Anwort gipfelt in einer Aufzählung der erforderlichen Hardware: SMP-System mit Gigahertz-Prozessoren, ultra flinke SCSI-Platten, maximaler Hauptspeicheraufbau... Und dennoch werden Sie nach verrichteter Arbeit bald bemerken, dass die Zugriffszahlen allmählich schwinden. Ihre Nachforschungen werden ergeben, dass die schlechten Antwortzeiten Ihres Systems die Benutzer vergraulten. Und dabei werkelten die Prozessoren doch niemals am Limit?

Der Grund identifiziert sich bald in Form einiger Schranken, die Linux sich selbst auferlegt, weil diese Schranken dem »Normalfall« allemal genügen und das gesamte System so schneller läuft.

Aber Ihr Fall ist kein Normalfall. Und so sollten Sie ein wenig an den Schrauben der Begrenzung drehen...

Vorsicht: Ein unüberlegter Eingriff kann Ihr System abstürzen lassen!

Schranken in »/proc/sys/fs/«

```
user@sonne> ls / proc/ sys/ fs
dentry-state  dquot-nr  file-nr    inode-nr     super-max
dquot-max     file-max  inode-max  inode-state  super-nr
```

dentry-state

Statusinformation zum Verzeichniscache, wobei der zweite Eintrag die Anzahl freier Cachezeilen angibt und dritte die Verweildauer, bis ein Eintrag, auf den nicht mehr zugegriffen wurde automatisch gelöscht wird

dquot-max

Maximale Anzahl von Quota-Einträgen im Cache (mit Quotas kann der Verbrauch an Festplattenspeicher für einzelne Benutzer begrenzt werden), für Systeme auf denen sehr viele Benutzer zur selben Zeit angemeldet sind, kann eine Erhöhung des Wertes notwendig werden:

```
root@sonne> cat / proc/ sys/ fs/ dquot-max
1024
root@sonne> echo "2048" > / proc/ sys/ fs/ dquot-max
root@sonne> cat / proc/ sys/ fs/ dquot-max
2048
```

dquot-nr

Anzahl belegter und freier Einträge im Quota-Cache

file-max

Maximale Anzahl gleichzeitig geöffneter Dateien, der übliche Wert wird für einige Serveranwendungen nicht ausreichen und sollte erhöht werden (spätestens wenn Fehlermeldungen erscheinen, dass keine neuen Dat mehr geöffnet werden können:

```
root@sonne> echo "8192" > /proc/sys/fs/file-max
```

Ein einzelner Prozess darf allerdings nur eine beschränkte Anzahl Dateien öffnen (meist 1024), um diesen V zu erhöhen, muss ein neuer Kernel mit vorherigem Eingriff in den Datei limits.h und fs.h generiert werden.

file-nr

Die drei Werte bezeichnen von links nach rechts die Anzahl insgesamt reservierter Dateihandle, die Anzahl (davon) belegter Dateihandle sowie maximal verfügbare Anzahl Dateihandles.

inode-max

Die Inodes geöffnter Dateien werden zum schnelleren Zugriff in einer Kernelstruktur gehalten. Wieviele Eint so zwischengespeichert werden, kann durch den Wert "inode-max" beeinflusst werden. Er sollte "(3-4)*file-max" betragen.

inode-nr

Zwei statistische Werte: Zum einen die Anzahl tatsächlich allokierter Einträge für Inodes (er kann etwas grö sein als »inode-max«, da die Anforderung neuen Speichers seitenweise erfolgt und die gesamte Seite für ne Inodes zur Verfügung steht), zum zweiten die Anzahl freier Einträge

inode-state

Von den sieben Werten werden die letzten derzeit nicht verwendet, die beiden ersten decken sich mit den Informationen aus »inode-nr«, der dritte Wert ist verschieden von Null, um anzuzeigen, dass für neue Inod kein weitere Speicher angefordert werden kann, sondern andere (ältere) Inode-Einträge überschrieben wer müssen

super-max

Auch die Superblöcke der gemounteten Dateisysteme werden in einer Kernelstruktur gehalten, werden sehr viele Mounts notwendig, kann die Erhöhung des Wertes angemessen sein

super-nr

Anzahl derzeit verwendeter Superblock-Einträge

Schranken in »/proc/sys/kernel/«

```
user@sonne> ls /proc/sys/kernel
acct            hostname        panic           rtsig-nr
cap-bound       modprobe        printk          shmall
ctrl-alt-del    osrelease       real-root-dev   shmmax
domainname      ostype          rtsig-max       version
```

acct

Betrifft die Kontrolle der Prozesserzeugung nach BSD, wo diese von der Belegung des Dateisystems (dasjer wo die Prozessdaten abgelegt werden) abhägig gemacht wird. Die drei Werte bestimmen die Grenze an fre

Speicherplatz (1.Wert in Prozent), ab der die Erzeugung neuer Prozesse verhindert wird, die Schranke (in Prozent), ab welcher die Erzeugung wieder zugelassen wird und das Zeitintervall in Sekunden, nachdem die Speicherbelegung neu bewertet wird.

ctrl-alt-del

Der Wert sollte immer »0« sein, da diese Tastenkombination den Prozess **init** zu einem Reboot überredet. hier ein von »0« verschiedener Wert eingegeben, so bootet **init** ohne das Sichern von irgendwelchen Puffe

domainname

Der NIS-Domainname des Rechners

hostname

Der Rechnername

modprobe

Zugriffspfad zum Programm, das die Module nachlädt (normal »/sbin/modprobe«)

osrelease

Version des Kernels

ostype

Name des Betriebssystems, sollte immer "Linux" sein

panic

panic ist eine Meldung höchster Priorität und deutet einen schwerwiegenden Fehler im System an (siehe a Syslog-Dämon), der normalerweise einen Neustart erfordert. Steht in dieser Datei »0«, so hält der Kernel d System an (so dass die letzten Konsolenausgaben nicht sichtbar sind). Bei einem Server, der ohne Aufsicht ist solch ein Verhalten allerdings nicht erwünscht. Ein solcher sollte selbsttätig booten. Indem in "panic" ein Wert >0 eingetragen wird, rebootet der Kernel das System nach Ablauf der durch Wert angegebenen Sekunden.

printk

Die Datei beeinflusst die Arbeit der Kernelfunktion »printk()«, die die Kernelmeldungen ausgibt. Der erste V ist das Level, ab welcher Priorität Kernelmeldungen auf der Konsole ausgegeben werden (siehe klogd). Der zweite Wert ist das Prioritätslevel, mit dem Meldungen, die keine Prioritätsangabe enthalten, versehen werd Wert Nr. 3 ist das Minimum, worauf der erste Eintrag gesetzt werden darf und der vierte Eintrag ist das voreingestellte Level, ab dem Meldungen auf der Konsole landen.

real-root-dev

Enthält die Devicenummer, wo sich das Root-Dateisystems befindet., für NFS steht hier die Pseudonummer

rtsig-max

Anzahl Signale, die gleichzeitig verwaltet werden können (bis zur Behandlung eines Signals wird dieses in e Kernlstruktur zwischengespeichert)

rtsig-nr

Anzahl aktueller Signale

shmall

Maximale Anzahl Shared Memory Segmente

shmmax

Maximale Größe eines Shared Memory Segments in Bytes

version

Übersetzungsdatum des Kernel

Schranken in »/proc/sys/net/«

```
user@sonne> ls / proc/ sys/ net
802  core  ethernet  ipv4  token-ring  unix
```

802

In diesem Unterverzeichnis befinden sich die Daten zum E802-Protokoll (AX.25), allerdings macht Linux selt hiervon Gebrauch

core

Einige generelle Einstellungen zum Netzwerk lassen sich hier vornehmen. Interessant ist **message_burst**, die Zeitspanne in zehntel Sekunden enthält, innerhalb derer maximal eine Warnung von Netzwerktreibern protokolliert wird. Damit wird sicher gestellt, dass durch Fehlermeldungen (die durchaus »von außen« mutv provoziert werden könnten) binnen kurzer Zeit die Protokolldateien anschwellen. Mit **message_cost** kann durch einen hohen Wert die im Netzwerk generierte Anzahl der Meldungen verringert werden.
Empfangene Nachrichten werden zwischen gespeichert und sobald als möglich an den Kernel weitergereich Wieviele Pakete maximal gespeichert werden könen, steht in **netdev_max_backlog**. **optmem_max** beschränkt die Puffer (Platz zur Aufnahme der Köpfe ohne Daten der Pakete), die für Sockets zur Verfügung stehen. Wieviel Speicher einem Socket bei dessen Erzeugung für zu empfangende Daten insgesamt zur Verfügung steht, sagt **rmem_default**. Der Socket selbst kann weiteren Speicher anfordern, insgesamt jed nicht mehr als **rmem_max** besagt. Die zu sendenden Datenpuffer werden durch **wmem_default** bzw. **wmem_max** beschränkt.

ethernet

Das Unterverzeichnis kann Daten zum Ethernet Protokoll enthalten

ipv4

Unter diesem Verzeichnis verbirgt sich der komplexeste Teil des Netzwerks.

conf

In diesem Unterverzeichnis finden Sie weitere Verzeichnisse zu jedem im System konfigurierten Netzwerkdevice. Zusätzlich steht noch ein Verzeichnis **all** zur Verfügung, um Konfigurationen vorzu die alle Devices betreffen. Jedes Unterverzeichnis beinhaltet die gleichen Dateien:

accept_redirects enthält »1« oder »0«, je nachdem, ob der Kernel ICMP-Pakete annehmen s nicht. Fungiert der Rechner als Router, sollte »0« eingetragen werden.
accept_source_route legt fest, ob »Source routing« Pakete vermittelt oder verworfen werde »Source Routing« wird der Weg eines Paketes nicht dynamisch ermittelt, sondern durch den Ab vorgegeben. Der Wert sollte im Falle von Routern auf »1« stehen und»0« sonst.
bootp_relay wird von aktuellen Kerneln nicht unterstützt
forwarding erlaubt oder verhindert das Weiterleiten von Paketen zu einer IP-Adresse über die Device.
log_martians Pakete, deren Route nicht ermittelt werden kann, können protokolliert werden.
mc_forwarding erlaubt oder unterbindet Multicast-Routing, mit dem ein Paket gleichzeitig an Empfänger geleitet werden kann
proxy_arp unterstützt bzw. unterbindet Proxy ARP (der lokale Rechner liefert stellvertretend fü Rechner deren Hardwareadresse, falls eine ARP-Anfrage gestellt wird)
rp_filter schaltet die Überprüfung der Sender-IP an bzw. ab

secure_redirects akzeptiert ICMP-Pakete nur von bekannten Gateway-Rechnern
send_redirects stellt ein, ob ICMP-Pakete vermittelt werden
shared_media erlaubt den direkten Datenaustausch zwischen verschiedenen Netzwerken (»1«
untersagt diesen »0«.

icmp_echo_ignore_all und icmp_echo_ignore_broadcasts

Die Werte sollten immer auf »0« stehen, um Anforderungen eines ICMP ECHO's zu bearbeiten, die
Datei betrifft nur Multicast- und Broadcastadressen (indem der Wert von icmp_echo_ignore_all auf
gesetzt wird, kann der Rechner vor einer »ping«-Anfrage verborgen werden)

icmp_destunreach_rate, icmp_echoreply_rate, icmp_paramprob_rate und icmp_timeexeed_rate

Hier wird angegeben, in welchem Zeitintervall maximal ein ICMP-Paket gesendet werden darf (Ang
1/100 Sekunden), »0« hebt die Beschränkung auf. Die einzelnen Dateien stehen für den Grund, wa
Paket verworfen und eine ICMP-Nachricht erzeugt wurde.

ip_autoconfig

Enthält eine »1«, falls der Rechner seine IP-Adresse über ARP, BOOTP oder DHCP erhält, sonst »0«

ip_default_ttl

Maximale Anzahl Stationen, die ein Paket durchläuft, bevor es als unzustellbar gilt

ip_dynaddr

Die dynamische Vergabe von IP-Adressen kann an- bzw. abgeschaltet werden. Rechner mit Modem
die IP für dieses Device meist vom Provider zugewiesen.

ip_forward

Erlaubt oder untersagt das Routen von Paketen zwischen verschiedenen Netzwerkinterfacen.

ip_local_port_range

Bereich der Portnummern, die von TCP oder UDP als lokale Ports verwendet werden können. Die A
erfolgt als Bereich <von_Portnummer>-<bis_Portnummer>.

ip_masq_debug

Schaltet die Debugmeldungen über das Masquerading ein und aus.

ip_no_pmtu_disc

Schaltet die MTU (maximal transfer unit), die Beschränkung der Paketlängen ein oder aus.

ipfrag_high_trash, ipfrag_low_trash

Puffergröße zum Zusammenfügen fragmentierter IP-Pakete. Wenn der Puffer die Größe
»ipfrag_high_trash«, erreicht, werden alle Pakete verworfen, bis er den Wert »ipfrag_low_trash«
unterschreitet.

ipfrag_time

Zeit in Sekunden, nach der ein ungenutztes Fragment eines IP-Paketes verworfen wird (falls nicht a
Fragmente ihr Ziel erreichen, wird sichergestellt, dass unvollständige Pakete nicht ewig im Speiche
verharren)

neigh

Das Verzeichnis enthält eine Reihe von Dateien mit Routing-Informationen:

error_burst Faktor, der die Anzahl von Fehlermeldungen vom Routingsystem limitiert
error_cost Bestimmt, alle wieviel Sekunden (Angabe 1/100 Sekunden) maximal eine Meldung protokolliert werden darf
flush Beim Leeren des Routing-Caches wird der Inhalt in diese Datei geschrieben
gc_elasticity, gc_interval, gc_min_interval, gc_thresh, gc_timeout Steuerung der Zeits und Art, wie der Routing-Cache aufgeräumt wird
max_delay, min_delay Minimale und maximale Zeit zwischen dem Leeren des Caches (damit Routen nicht ewig ihre Gültigkeit behalten)
max_size Maximale Anzahl Einträge im Routing Cache, ist er voll, überschreibt ein neue Route ältesten nicht verwendeten Eintrag
mtu_expires Maximale Zeit, die eine unbenutzte Route im Cache verweilen kann
redirect_load Dies ist ein Faktor der bestimmt, ab welcher Belastung des Netzwerksystems IC Nachrichten nicht gesendet werden.
redirect_number Anzahl Versuche, eine ICMP-Nachricht zuzustellen
redirect_silence Zeitspanne zwischen zwei Versuchen, eine ICMP-Nachricht zuzustellen

Wenn bei einer TCP-Verbindung die Gegenseite dem Wunsch zum Abbau der Verbindung nicht nac wird diese nach Ablauf dieser Zeitspanne (Sekunden) gekappt.

Anzahl Testpakete, die ausgesendet werden, um festzustellen, ob der Partner aktiv ist. Danach gilt Verbindung als unterbrochen.

Zeit des regelmäßigen Aussendens von Testpaketen, um eine Verbindung offen zu halten. Die Uhr neu, sobald ein »reguläres« Paket über die Verbindung geht.

Wie viele Pakete zum Überprüfen sollen gesendet werden, während ein Timer die Zeitspanne bis zu Beenden der Verbindung stoppt?

tcp_max_syn_backlog

Länge der Warteschlange für eingehende TCP-Verbindungsanforderungen für einen Socket. Treffer Verbindungswünsche ein, werden die Pakete einfach verworfen, außer »syncookies« ist eingeschal letzterem Fall wird die Länge der Warteschlange ignoriert.

tcp_retrans_collapse

Es soll Netzwerkdrucker mit einem fehlerhaften TCP-Stack geben. Durch eine »1« wird versucht, ei Drucker abgelehntes Paket diesem doch noch unterzuschieben, indem die Größe des Pakets erhöht somit der Stack-Fehler (hoffentlich) nicht mehr Tragen kommt.

tcp_retries1, tcp_retries2

Der erste Wert bestimmt, wie oft eine Antwort auf ein TCP-Paket gesendet wird (jedes Paket muss Partner quittiert werden), bevor aufgegeben wird. Der zweite Wert bestimmt die maximale Anzahl Versuchen, ein TCP-Paket zu senden

tcp_stdurg

Einschalten der Funktionalität des Urgent Pointers (Vorrangdaten) nach den RFC793-Vorgaben, Voreinstellung ist die Interpretation nach BSD. Solche Datenpakete werden beim Empfänger sofort betreffenden Anwendung durchgestellt und nicht erst durch den Empfangspuffer geschleust.

tcp_syn_retries

Anzahl Versuche eines Verbindungsaufbaus (ausgehende Verbindungen)

tcp_timestamps

Die Verwendung von Zeitstempeln wird an- oder abgeschaltet

tcp_window_scaling

Das dynamische Anpassen des Übertragunsgfensters wird aktiviert. Hierdurch kann allmählich vers werden, die Übertragung zu beschleunigen. Treten dabei Paketverluste auf (z.B. weil der Empfänge Pakete nicht schnell genug verarbeitet), wird das Fenster dynamisch verkleinert.

token-ring

Daten zum Token Ring von IBM

unix

Im Unterverzeichnis existieren drei Dateien. **delete_delay** gibt die Zeit in Sekunden an, nachdem ein geschlossener Socket aus der Tabelle entfernt wird. Sein reservierter Speicher wird nach der in **destroy_d** festgelegten Spanne frei gegeben. **max_dgram_qlen** gibt die Länge der Warteschlange an, die eintreffen Daten am Socket zwischenspeichern kann.

Der Kernel - Konfiguration

Übersicht

Wer sich nicht gerade intensiv mit Computern beschäftigt, der wird der Möglichkeit, einen Kernel nach Maß zu generieren, wenig Bedeutung zumessen. Zu komplex mutet die Konfiguration an. Und spätestens nach dem dritten Versuch, der mit einer kryptischen Fehlermeldung endete, landet die Flinte dann im Korn.

Tatsächlich ist es selten notwendig, sich in die Tiefen des Betriebssystems herabzulassen. Die Zeiten, da man durch Entrümpelung den Kernel noch ein paar Maschinenzyklen effizienter trimmte, gehören spätestens seit Athlon und Pentium III der Vergangenheit an. Den Unterschied merken bestenfalls die Benchmark-Programme. Und dort differiert auch nur noch die Zahlenkolonne hinter dem Komma.

Alle (von mir getesteten) Distributoren legen ihren Paketen heutzutage einen stark modularisierten Kernel bei. Nahezu jede verfügbare Eigenschaft liegt zumindest als Modul vor und kann bei Bedarf geladen werden.

Das Backen eines eigenen Kerns ist quasi einzig erforderlich, wenn - aus welchen Gründen auch immer - unbedingt die aktuellsten Kernelversionen verwenden werden müssen. Klar, dass die Aktualität der Distributionen nicht mit dem rasanten Erscheinen neuester (Entwickler-) Versionen Schritt halten kann. Wer nicht warten will, muss nun selbst Hand anlegen und aus den Quellen einen einsatzbereiten Kernel formen.

Andererseits macht es wenig Sinn, jede Distributionsversion mitzunehmen. Meist ändern sich an diesen doch nur wenige Teile, die man sich womöglich schon längst aus dem Netz gezogen hat. Um stets auf der Höhe der Zeit zu bleiben, genügt die Aktualisierung des Kernels. Doch der liegt nur als Quellkode vor, womit man um die Generierung nicht umhin kommt...

Wer übrigens den Kernel nicht übers Modem auf die heimische Platte holen möchte, der sollte mal in linuxspezifischen Zeitschriften schmökern. Regelmäßig enthalten diese CDROM mit den neuesten Programmen und Kernelversionen.

Anmerkung: Nachfolgend beziehen wir uns in den Beispielen immer auf das Verzeichnis **/ usr/ src/ linux**. Wenn Sie die Kernelquellen unter einem anderen Pfad installiert haben, dann funktionieren die folgenen Aufrufe nur in diesem Stammverzeichnis! Die beschriebene **Kernelversion** ist **2.4.17**.

Die Werkzeuge zur Konfiguration

Der harte Eingriff in die Quellen bleibt den eingefleischten Kernel-Entwicklern vorbehalten. Es sind einfach zu viele Abhängigkeiten zwischen den einzelnen Kodeteilen, als dass Otto Normalverbraucher auch nur den Hauch eine Chance hätte, dort durchzusehen. Glücklicherweise existieren verschiedene Konfigurationshilfen, die menügesteuert etwas Struktur ins Chaos bringen.

make config

Überliefert aus alten Zeiten, steht gar ein textbasiertes Frage-Antwort-Spiel zur Verfügung:

```
root@sonne> cd / usr/ src/ linux
root@sonne> make config
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in .config
#
*
* Code maturity level options
```

```
*
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n/?]
...
```

Die konfigurierbaren Punkte sind dieselben, wie in den nachstehend vorgestellten Hilfen. Die Konfiguration erfolgt in vorgegebener Reihenfolge. Die nachträgliche Änderung einer einmal getroffenen Auswahl ist nur durch einen erneuten Programmdurchlauf möglich. Eine Option kann aktiviert (Eingabe von [y] bzw. [Y]) oder deaktiviert werden ([n] bzw. [N]). Manche Eigenschaften lassen sich als Modul realisieren. In der Auswahl symbolisiert ein [M] diese Möglichkeit. Hinweise zum Sinn einer Option erhalten Sie durch Eingabe von [?].

make menuconfig

Wenn kein X-Server zur Verfügung steht, ist das auf Dialogboxen aufbauende Werkzeug eine komfortable Hilfe zur Bändigung des Kernels:

```
root@sonne> cd /usr/src/linux
root@sonne> make menuconfig
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
make -C scripts/lxdialog all
make[1]: Wechsel in das Verzeichnis Verzeichnis »/root/linux/scripts/lxdialog«
gcc -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -DLOCALE -I/usr/include/ncurses -DCURSES_LOC="<ncurses.h>" -c -o
checklist.o checklist.c
...
```

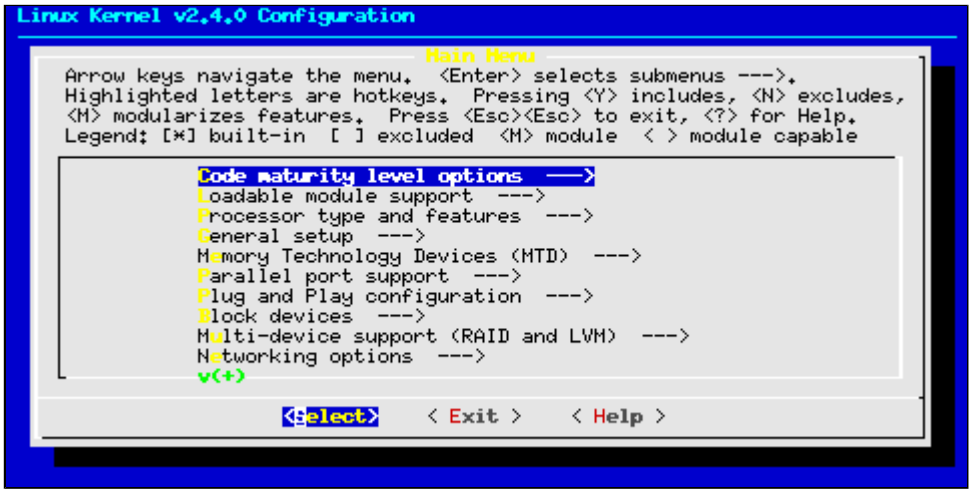


Abbildung 1: 'make menuconfig'

Die Navigation erfolgt mit den Cursortasten. Die Anwahl einer Option geschieht durch Eingabe von [y], die Abwahl mittels [n]. Steht die Möglichkeit der Realisierung als Modul zur Verfügung, so setzt [M] diese Variante. Das Wechseln zwischen den Möglichkeiten kann ebenso durch wiederholtes Drücken von [Space] geschehen.

make xconfig

Unter X steht die bequemste Möglichkeit der Kernelkonfiguration (Tcl/Tk) zur Verfügung:

```
root@sonne> cd /usr/src/linux
root@sonne> make xconfig
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
make -C scripts kconfig.tk
make[1]: Wechsel in das Verzeichnis Verzeichnis »/root/linux/scripts«
cat header.tk >> ./kconfig.tk
./tkparse < ../arch/i386/config.in >> kconfig.tk
...
```

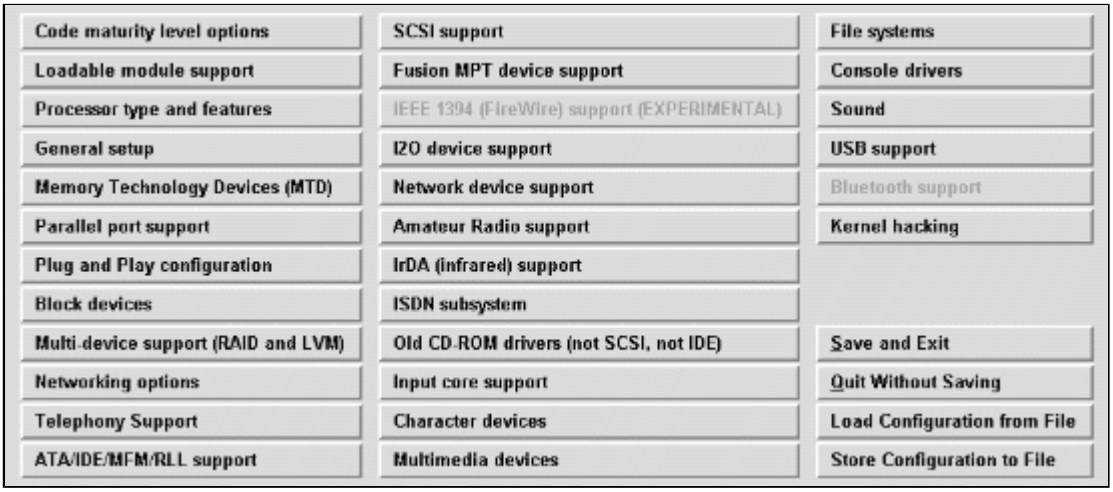


Abbildung 2: 'make xconfig' (offizieller Kernel 2.4.17)

Wir beschreiben die wesentlichen Teile der Konfiguration anhand der durch das X-fähige Werkzeug vorgegebenen Reihenfolge.

make oldconfig

Die zuletzt vollzogene Kernelkonfiguration wird jeweils in der Datei **.config** im Kernel-Stammverzeichnis gespeichert. Nach einem Kernelpatch oder der Installation neuer Kernelquellen (vergessen Sie nicht, .config zu sichern) besteht berechtigtes Interesse, den neuen Kernel mit identischen Einstellungen zu konfigurieren. Wenn die Datei .config existiert, kann mit:

```
root@sonne> cd / usr/ src/ linux
root@sonne> make oldconfig
```

die automatische Konfiguration angeschoben und das ganze Prozedere der manuellen Menüpunkteauswahl umgangen werden. Manche Distributionen (bspw. SuSE) speichern die aktuelle Kernelkonfiguration in einer Datei »/boot/vmlinuz.config«. Um dessen Konfiguration für eine Kernel-Neuerzeugung zu verwenden, bringen sie das make-Ziel »cloneconfig« mit. Abgesehen von den unterschiedlichen Dateien mit den Kernelinformationen arbeiten oldconfig und cloneconfig identisch.

Aufräumen

Während ein »make clean« nur die Objektdateien entfernt und somit eine spätere Neuübersetzung aller Kernelteile erzwingt, sorgt ein »make mrproper« für ein Rücksetzen der Kernelquellen in den Originalzustand. Im Wesentlichen handelt es sich um das zusätzliche Entfernen der Statusdateien.

Schließlich existiert »make distclean«, das neben »make mrproper« auch noch die Rückstände von Patchvorgängen entsorgt.

Die Optionen - Schritt für Schritt

Konfiguration zur Laufzeit

Etliche Parameter des Kernels können per Kommandozeilenoptionen übergeben werden, was allerdings dessen Start mittels eines Bootmanagers bedingt. Ein Kernel, der als Rettungsanker auf einer Diskette untergebracht wird, wird jedoch i.d.R. zum Booten seinen eigenen Startcode, der im Kernelimage zuvorderst steht, verwenden (u.a. weil ein Bootloader selbst wiederum einen Teil des knappen Diskettenplatzes belegen würde). Bei einem solchen Kernel müssen die »wichtigsten« Parameter passen, damit er bspw. das Root-Dateisystem finden und mounten kann.

Konkret lassen sich folgende Parameter fest in das Kernelimage einbrennen:

- Die Lage des Root-Dateisystems
- Die Parameter einer RAM-Disk
- Den zu verwenden Video-Modus
- Wie das Root-Dateisystem zu mounten ist (»nur lesend« oder »schreibend und lesend«
- Die Lage der Swap-Partition

Möglich wird das Modifizieren dieser Einträge, weil jene jedes Kernelimage an exakt festgelegten Adressen (beginnend bei Offset 504 Bytes) enthält.

Ein Kommando für alle

Zum Auslesen und Setzen dieser Parameter existieren mehrere Kommandos, wobei **rdev** (Root Device) in Zusammenhang mit **Optionen** identisch zu den weiteren Kommandos wirkt:

-r

rdev arbeitet wie **ramsize**

-R

rdev arbeitet wie **rootflags**

-v

rdev arbeitet wie **vidmode**

-s

rdev arbeitet wie **swapdev**

-h

Ein kurzer Hilfetext wird ausgegeben

Es ist also egal, ob Sie bspw. die Lage der Swap-Partition mittels **swapdev** oder mittels **rdev -s** einstellen. Wir beschränken uns in den Beispielen auf **rdev**.

Kommandoaufruf

Wird **rdev** ohne Argumente gerufen, so gibt es einzig die Lage des aktuellen Root-Dateisystems preis. Es befragt hierzu nicht erst den aktiven Kernel, sondern schaut in der Datei /etc/mtab nach:

```
root@sonne> rdev
/dev/hda9 /
```

Um weitere Informationen zu gewinnen, ist das zu betrachtende Kernelimage anzugeben:

```
# Welches Rootdevice ist in /boot/vmlinuz eingetragen?
root@sonne> rdev / boot/ vmlinuz
Root device /dev/sda5
# Welches Rootflag ist in /boot/vmlinuz gesetzt?
root@sonne> rdev -R / boot/ vmlinuz
Root flags 1
```

Liegt der Kernel auf einer Diskette, so ist er über das entsprechende Device erreichbar (das gilt nur, wenn der Kernel per **dd** auf das Medium kopiert wurde, was der übliche Weg ist, eine bootbare Diskette zu erzeugen):

```
# Kernel liegt auf Diskette /dev/fd0:
root@sonne> rdev -v / dev/ fd0
Video mode 65535
```

Ändern des Rootdevices

Um das Rootdevice permanent im Kernelimage zu ändern, ist das neu zu verwendende Device dem Imagenamen nachzustellen:

```
root@sonne> rdev / boot/ vmlinuz
Root device /dev/sda5
root@sonne> rdev / boot/ vmlinuz / dev/ hda2
root@sonne> rdev / boot/ vmlinuz
Root device /dev/hda2
```

Ändern der RAM-Disk-Parameter

Die wenigsten Leser werden mit Ram-Disk tatsächlich in Berührung kommen. Dennoch soll auch diese Thematik angesprochen werden. Eine Ram-Disk verwenden i.d.R. die Distributoren, um die Installationskernel einerseits relativ klein und andererseits umfangreich genug zu halten, um auf jeder Standard-Hardware zu booten. Der Kernel enthält nun im Wesentlichen die Fähigkeiten, um eine solche (komprimierte) Ram-Disk zu laden und das enthaltene Skript zu starten. Dieses Skript kümmert sich dann zumeist um die Erkennung der Hardware und die Einbindung der benötigten Treiber. Eine Einführung in die Thematik ist im Abschnitt Booten (Systemadministration) zu finden.

Was muss der Kernel nun wissen? Zum einen muss ihm mitgeteilt werden, dass er überhaupt eine Ramdisk laden soll. Das, was als Kernelargument **load_ramdisk** veranlasst, regelt intern das Bit Nummer 14 in einem 2 Byte großen Wert (das »Ramdisk-Wort«). Steht es auf "1", lädt der Kernel die Ramdisk; steht es auf "0", ignoriert er sie. Des Weiteren muss die Ramdisk nicht zwangsläufig auf derselben Diskette liegen, wie der Kernel selbst. Daher stoppt eine "1" in Bit 15 des obigen Wertes den Kernel, sodass die Diskette ggf. gewechselt und die Arbeit nach Betätigen von [Enter] fortgesetzt werden kann (entspricht dem Kernelargument **prompt_ramdisk**). Die ersten 11 Bytes bestimmen nun den Offset (in 1k Blöcken; Kernelargument **ramdisk_start**), an dem die Ramdisk auf der Diskette liegt (Bit 11-13 werden nicht belegt).

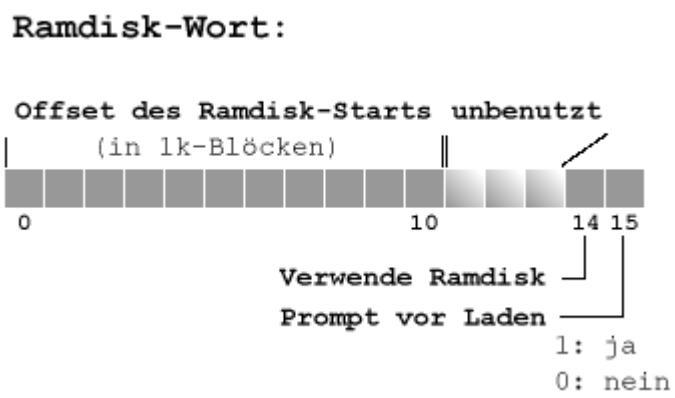


Abbildung 3: Bitbelegung des Ramdisk-Worts

Der korrekte Wert für das Ramdisk-Wort ergibt sich nun aus der Summe von:

- Offset, an dem die Ramdisk beginnt
- 2^{14}, falls Bit 14 gesetzt ist
- 2^{15}, falls Bit 15 gesetzt ist

Als Beispiel nehmen wir den Fall an, dass Kernel und zugehörige Ramdisk hintereinander auf dieselbe Diskette kopiert werden:

```
# Kernel kopieren
root@sonne> dd if=/boot/kernel_image of=/dev/fd0 bs=1k
324+0 records in
324+0 records out
# Ramdisk »hinter« den Kernel kopieren
root@sonne> dd if=/boot/initrd of=/dev/fd0 bs=1k skip=324
514+0 records in
514+0 records out
```

Die Ramdisk beginnt somit am 325. Block. Damit der Kernel sie auch lädt, muss das Bit 14 gesetzt sein, also ergibt sich der Wert für das Ramdisk-Wort zu $325+2^{14}=33093$:

```
root@sonne> rdev -r /dev/fd0 33093
```

Ändern des Video-Modus

Bei heutigen Bildschirmgrößen wirkt die übliche Darstellung mit 80x25-Zeichen etwas unangemessen; etwas »mehr« Text hätte gut und gern auf der Mattscheibe Platz. Um die zu verwendende Auflösung dem Kernel mittzuteilen, ist deren numerische Code zu setzen.

Dabei steht der Wert von **-1** für den Standardmodus (also 80x25 Zeichen), eine **-2** schaltet den erweiterten VGA-Modus ein (80x50 Zeichen). Die weiteren unterstützen Videomodi werden beginnend bei 1 durchnummeriert; jedoch hängt es von der Grafikkarte ab, welche Modi diese bereit stellt. Aus diesem Grund sollte zunächst **-3** gewählt werden, wodurch der Kernel beim Start eine Liste der verfügbaren Modi anzeigt und zur Auswahl auffordert. Entspricht ein Modus Ihren Ansprüchen, können Sie diesen später fest eintragen.

```
root@sonne> rdev -v /boot/vmlinuz -3
```

Ändern des Rootflags

Die zulässigen Werte sind "0", um das Rootdevice mit Schreib- und Leseberechtigung zu mounten unt "1", um das Dateisystem schreibgeschützt einzuhängen:

```
root@sonne> rdev -R /boot/vmlinuz
Root flags 1
root@sonne> rdev -R /boot/vmlinuz 0
root@sonne> rdev -R /boot/vmlinuz
Root flags 0
```

Ändern des Swapdevices

Diese Option ist nur erforderlich, falls der Hauptspeicher stark begrenzt ist (<16..32MB). Der Kernel wird nun bereits während des Startens das eingetragene Swapgerät aktivieren und den verfügbaren Speicher somit um die Swapgröße erweitern. Gerade in Verbindung mit initialen Ramdisks ist ein Minimum an Hauptspeicher Voraussetzung, damit der Kernel booten kann.

```
root@sonne> rdev -s /dev/fd0
Swap device is /dev/hda7
root@sonne> rdev -s /dev/fd0 /dev/hda2
root@sonne> rdev -s /dev/fd0
Swap device is /dev/hda2
```

Der Kernel - Installation

Übersicht

Der rechte Fleck

Die Richtung

Netzwerk Grundlagen - Übersicht

Überblick

Ziel des Kapitels

Inhalt des Kapitels

Die Geschichte des Internets

Strategie der Angst

Der kalte Krieg zwischen Ost- und Westblock erreichte mit dem atomaren Wettrüsten in den 50er Jahren einen unrühmlichen Höhepunkt. Auf Kommunikationsebene nahm das amerikanische Miltär mit seiner Ausrüstung an Computern weltweit einen Spitzenplatz ein, aber man erkannte die Achillesferse bisheriger Netztechnologien: die Netzknoten. Fiel nur ein Knoten aus, bedeutete dies den Stillstand des gesamten Netzes. Und eine zerstörte Kommunikation gilt in militärischen Kreisen als die strategische Niederlage schlechthin. Damit war klar, dass man für die Zukunft eine Technologie benötigte, die auch bei Ausfall mehrerer Elemente die Funktionalität aufrecht erhielt. Die vom amerikanischen Verteidigungsministerium gegründete Forschungsabteilung Advanced Research Projects Agency ARPA (1957) unterstützte daraufhin die Forschungen nach resistenten Netzstrukturen und -protokollen.

1962 wurde von der RAND Corporation der erste Vorschlag eines paketvermittelnden Netzwerkes veröffentlicht, das die Datenübertragung beim Ausfall eines Teils der Vermittlungsrechner garantierte. Wichtigste Erkenntnisse dieser Theorie waren der Verzicht auf zentrale Entscheidungseinrichtungen (keine steuernde Instanz) und die Zerlegung von Nachrichten in Teile konstanter Größe (Pakete). Diese Merkmale kennzeichnen auch die Arbeitsweise des heutigen »Internets«.

Schon damals existierten durch unterschiedliche Hardware Inkompatibilitäten zwischen den im Netzwerk vorgesehenen Rechnern. Man benötigte somit ein einheitliches Datenformat. Die weiteren Forschungen erbrachten die Einführung eines Interface Message Prozessors (IMP), der, an jedem Host angeschlossen, mit jedem IMP kommunizieren konnte. Die Formatumsetzungen mussten somit im Protokoll zwischen Rechner und IMP erfolgen. Den Interface Message Prozessor könnte man als Vorgänger der Netzwerkkarte bezeichnen.

Das ARPANET

Wieder war die ARPA federführend und initiierte eine Ausschreibung zur Entwicklung der Interface Message Prozessoren und eines Netzwerkes, das bis zu 16 Standorte miteinander verbinden können sollte. Als Pilotprojekt wurde Ende 1969 ein Netzwerk zwischen den Universitäten Los Angeles, Santa Barbara, Stanford und Utah errichtet. Dieses Referenznetzwerk gilt als Anfang des ARPAnet.

Überraschend gebührt der Ruhm des ersten einsatzfähigen Netzwerkes auf paketvermittelnder Basis den Briten, die dieses bereits 1968 in Großbritanien in Betrieb nahmen. Auch in Frankreich hatten die Forschungen zu paketvermittelnden Netzen begonnen.

Vermutlich aufgrund der unterschiedlichen Hardware-Plattformen und dem damit verbundenen Aufwand bei der Implementierung der Protokolle für jede IMP-Rechner-Kombination wurde das angestrebte Ziel mit 23 Rechnern, die an 15 Knoten angeschlossen waren, erst 1971 erreicht. In den selben Zeitraum fielen auch die ersten Vorschläge für die Protokolle TELNET und FTP.

Zwei Jahre später (1973) kamen zu den nun 33 Knoten innerhalb der Vereinigten Staaten die ersten beiden Anbindungen aus Übersee hinzu (Großbritannien, Norwegen). Erstmals sah man sich mit dem Problem der inkompatiblen Protokolle konfrontiert, da nur innerhalb des ARPAnet mit dem Network Control Protocol (NCP) eine einheitliche Richtlinie existierte, die nun hinzukommenden Netzwerke aber eigene Protokolle verwendeten.

TCP/IP

Noch im September 1973 lag der erste Entwurf eines Transmission Control Protocols (TCP) vor, das auch heute noch - unter diesem Namen - einen Eckpfeiler des Internet-Protokoll-Stacks markiert. Die anfänglichen Eigenschaften beinhalteten sowohl die Sicherung der Übertragung als auch deren Adressierung.

1975 brachten Bolt, Beranek and Newman (BBN) mit dem Telnet das erste kommerzielle paketvermittelnde Netzwerk auf den Markt, im folgenden Jahr nahmen das auf dem Amateurfunkdienst basierende Packet Radio Network (PRNET) und das Atlantic Packet Satellite Network (SATNET) ihren Betrieb auf.

1977 erkannte man bei Versuchen mit der Übertragung von Sprache die Unzulänglichkeiten der Kapselung von Datensicherung und Adressierung in einem Protokoll, da im Falle eines Übertragungsfehlers das erneute Senden zu einer verminderten Qualität der Sprachausgabe - bedingt durch die Verzögerung - führte. Man benötigte ein »unsicheres« Protokoll, das die Daten einfach ins Netz sendete, ohne auf eventuelle Probleme der Übertragung anzusprechen. Die Überlegungen führten zur Trennung der vom bisherigen TCP erledigten Aufgaben in ein Protokoll, das einzig für die Adressierung zuständig ist Internet Protocol (IP) und eines, das den Datenfluss steuert (TCP). Als »unsicheres« Protokoll konnte anstelle von TCP auf das User Datagram Protocol (UDP) zurück gegriffen werden.

1978 schließlich lagen die Protokolle IP, TCP und UDP im Wesentlichen in der noch heute verwendeten Fassung vor.

Neue Netze

Zwei Studenten verbanden 1979 zwei Unix-Rechner über eine Telefonverbindung miteinander. Zur Datenübertragung nutzten sie Unix to Unix Copy (UUCP). Das Netzwerk funktionierte, indem eine Nachricht von einem Rechner auf einen Server übertragen wurde, welche diese speicherte und bei nächster Gelegenheit (Telefonverbindung) diese an den nächsten Server weiter gab, bis alle Rechner eine Kopie der Nachricht besaßen. Eine weitere Neuerung war die hierarchische Anordnung der Nachrichten nach thematischen Gesichtspunkten. Dieses Netzwerk hat bis heute als USENET überlebt.

Im gleichen Jahr datierte die Grundsteinlegung für das auf TCP/IP basierende Computer Scienes Network (CSNET), zu dem sich vor allem die nicht im ARPAnet integrierten Universitäten der USA zusammenschlossen. Allerdings dauerte es bis zur praktischen Realisierung noch zwei Jahre, während zum preiswerteren USENET binnen Jahresfrist bereits 15 Rechner zählten.

Nach Verabschiedung der Standards für TCP/IP erfolgte im Jahre 1982 die Umstellung des gesamten ARPAnets. Das NCP hatte damit ausgedient. In Europa wurde indes das EUnet installiert, das im Wesentlichen auf Basis des USENET arbeitete.

Ebenfalls 1982 wurde der Standard für ein Email-Protokoll verabschiedet.

1983 wurde das ARPAnet in ARPA Internet umbenannt und alle militärischen Einrichtungen der USA in ein neues Netzwerk (MILNET) integriert.

Durch die Aufnahme des TCP/IP-Codes in die Berkeley UNIX-Implementierung (Version 4.2) erhielt der Protokollstack eine starke Aufwertung. Auch MSDOS-Rechner erhielten mit den FidoNet Zugang zum sich entwickelnden Internet.

1984 wurde die Grenze von 1000 vernetzten Rechnern überschritten. Das Problem der Adressierung neu hinzukommender Rechner führte zur Einführung des Domain Name Services (DNS). Bis dato musste jeder integrierte Rechner lokal eine Datei speichern, die sämtliche Adressen der erreichbaren Rechner im Netz enthielt. Schwer zu handhaben war schon seit längerem die Größe der »Datenbank« (man bedenke die damaligen Speicherdimensionen und -kosten); mit fortschreitender Ausbreitung des Netzwerks kam nun noch die Frage nach der Aktualität einer solchen Datei hinzu.

Geschwindigkeit

Das neue Netzwerk (NSFNET) der National Science Foundation (NSF) verband 1986 5 Supercomputerstandorte der USA miteinander. Die Übtragungsrate der Daten betrug immerhin 56 kBit/s. Die NSF ermöglichte außenstehenden Institutionen, ihre Netzwerke an das NFSNET anzukoppeln. Da das Angebot reges Interesse fand, erhöhte man die Leistung der Verbindungen auf 1.5 Megabit/s, um dem gestiegenen Datenaufkommen Herr zu werden. Etwa zur gleichen Zeit entstand das Network News Transfer Protocol (NNTP), das die Daten des USENET über TCP/IP-Verbindungen transportieren konnte, auch führte man Mail Exchanger ein, um Rechnern den Nachrichtenaustausch

zu ermöglichen, die über keinen (permanenten) Zugang zum Internet verfügten. Erste europäische Länder wurden 1988 ins NFSNET integriert.

1988 nahmen ca. 60000 Rechner am weltweiten Verbund teil, als der »Internet-Wurm« zahlreiche Computer blockierte und erstmals die Problematik der Sicherheit der Netzwerke verdeutlichte.

1989 erhielt u.a. Deutschland Anschluss an das NFSNET.

Den Überblick bewahren...

Einher mit dem anhaltenden rasanten Wachstum des Internets (ca. 300000 Rechner 1990) ging die Flut an Informationen, die auf den weltweit verteilten FTP-Servern lagerte. »Dass es das Dokument irgendwo gibt«, war nicht die Frage, es galt die Information zu finden.

In Montreal wurde 1990 »archie« entwickelt, ein Programm das regelmäßig einen Schnappschuss von in seiner Region befindlichen FTP-Servern generierte, und diese mit »Inhaltslisten« mit anderen Archie-Servern austauschte. Gewissermaßen handelt es sich um die erste »Suchmaschine« des Internet.

Eine komfortablere Navigation durch die Datenbestände des Netzes ermöglichte das 1991 veröffentlichte GOPHER, das erstmals eine grafische Menüführung präsentierte.

Archie und Gopher ermöglichten somit die Suche nach Dateien mit bestimmten Namen, aber erst das Wide Area Information System (WAIS, 1991) ermöglichte eine Volltextsuche, also das Durchsuchen von Dokumenten nach enthaltenen Textpassagen.

Die Zahl der im Internet verbundenen Rechner näherte sich Ende 1991 700000, im Jahr darauf wurde die Millionengrenze durchbrochen. Die wichtigen Neuerungen im Jahr 1992 waren die Einführung einer Multicast-Übertragungstechnik und der Multipurpose Internet Mail Extensions (MIME) als Erweiterung des bisherigen Mailprotokolls, womit auch das Senden von Bildern u.a.m. ermöglicht wurde.

Mit dem Rückzug der NFS 1993 wurde die zentrale Verwaltung des Internet aufgegeben. Es besteht seitdem aus zahlreichen unabhängigen Netzwerken, die über verschiedene Punkte miteinander gekoppelt sind. Gleichzeitig wurde der inzwischen standardisierte Vorschlag zur Aufweitung des IP-Adressbereichs von damals 32 Bit auf 128 Bit eingebracht, um den schon zu jener Zeit absehbaren Zuwachs gerecht zu werden (4 Millionen 1994). Da die Adressvergabe allerdings nur von einer zentralen Institution vorgenommen werden kann, wurde noch 1993 das Internet Network Information Center (InterNIC) ins Leben gerufen.

Die Anzahl heute am Internet teilnehmenden Rechner ist schwerlich abzuschätzen, da zum einen hinter manchen offiziellen IP-Adressen sich zahlreiche Rechner verbergen, die über Proxi-Mechanismen mit der »Außenwelt« kommunizieren und zum anderen die meisten privaten Rechner nur temporär eine Verbindung unterhalten. Prognosen gehen daher von bis zu 1 Milliarde verbundener Rechner aus.

Protokolle

Wozu Protokolle? Eine nicht ganz ernst zu nehmende Ausarbeitung.

Kommunikationsprobleme sind allgegenwärtig. Sei es der juristische Kauderwelsch der aktuellen Fassung der Steuererklärung, der selbst gestandenen Juristen den Schweiß auf die Stirn treibt. Seien es die Nuancen des englischen Humors, die es dem Außenstehenden schier unmöglich machen, Ernst von Ironie zu unterscheiden. Oder die Sprachbarriere allgemein, die die Verständigung zwischen den Nationalitäten ungemein erschwert...

Mensch weiß sich oft zu helfen. Er engagiert einen Steuerberater, schmunzelt verständnisvoll bei einer Backgammon-Runde in feiner englischer Gesellschaft oder gestikuliert, wenn der magere Wortschatz eine Beschreibung seines Herzenswunsches nicht zulässt.

Bei der Kommunikation zwischen Computern ist das ungleich problematischer. Hier gibt es (noch) keine intelligenten Programme, die fehlende oder zweifelhafte Informationen in Eigenregie recherchieren. Hier gibt es keine einheitliche Sprache und es gibt für jede Anforderung dutzende Client-Programme, die diese formulieren und ebenso dutzende Server, die sie erfüllen. Und jeder Client sollte mit jedem Server zusammen arbeiten und jeder Server muss jedem Client antworten können.

Die Lösung für derartige Probleme ist die Schaffung allgemein gültiger Regeln, so genannter Protokolle.

OSI Referenzmodell - Spielzeug der Theoretiker

Der Nebentitel *Spielzeug der Theoretiker* soll verdeutlichen, dass dieses Modell praktisch kaum konsequente Anwendung findet. Dazu sind manche Definitionen zu »schwammig«, um verbindliche Strukturen zu schaffen. Und dennoch verhalf es, das Verständnis für die Prinzipien des Datenaustauschs zu schärfen.

Beginnen wir mit einem Beispiel und betrachten die Kommunikation zwischen einem NIS-Server und einem Client. Um die Bedeutung der Schichten des 7-Schichten-OSI-Modells zu veranschaulichen, nehmen wir an, dass der Server auf einem 64-Bit-Rechner und der Client auf einem 32-Bit-Rechner läuft, d.h. die Daten liegen auf den beteiligten Rechnern in unterschiedlichen Formaten vor.

NIS–Server		NIS–Client
Daten formatieren	Darstellung	Daten formatieren
RPC–Aufruf	Sitzung	RPC–Aufruf
Pakete zusammenfügen	Transport	Pakete formen
Pakete empfangen	Vermittlung	Ziel adressieren
Übertragung sichern	Sicherung	Übertragung sichern
Empfangen	Übertragung	Senden

Abbildung 1: Schichten des OSI-Modells

Ablauf der Kommunikation: Der NIS-Client sendet eine Anfrage. Da er das vom Server verstandene Datenformat nicht kennen kann, bringt er diese in ein allgemeines Format (im Zusammenhang mit RPC kommt meist *External Data Representation* XDR zum Einsatz). Der *Remote Procedure Call* organisiert den »entfernten Prozeduraufruf«, indem er die Daten zusammen mit Informationen zum erwünschten Dienst verpackt. Auf Transportebene werden u.a. die Daten in Blöcke bestimmter Größe zerlegt, die Vermittlung übernimmt die Adressierung des Zielrechners auf IP-Ebene. Die Sicherungsschicht ist schon recht Hardware spezifisch und

kümmert sich um die fehlerfreie Übertragung auf dem eigentlichen Medium. Schließlich erfolgt die Übertragung über ein physisches Medium, dessen Charakteristika durch das Protokoll der Übertragungsschicht spezifiziert ist. Auf Serverseite wird das Paket der Schicht 2 übergeben. Stellt diese die Fehlerfreiheit fest, erreicht es die Vermittlungsschicht. Auf Transportebene werden die vom Sender erzeugten Pakete zu der kompletten Nachricht zusammen gefügt. Der RPC-Server-Prozess wertet die Anforderungen aus. Sind sie erfüllbar, werden die Daten ins benötigte Format übertragen und der NIS-Server konsultiert. Dieser generiert eine Antwort und das Spiel wiederholt sich mit verteilten Rollen...

Widmen wir uns den einzelnen Schichten und ihren Aufgaben.

Übertragungsschicht

Diese Schicht korrespondiert mit der zugrunde liegenden Hardware. Protokolle dieser Schicht legen die Eigenschaften der Schnittstellen fest wie Anschlusseigenschaften, zulässige Übertragungsraten, Signalpegel und elektrische Kodierung der einzelnen Bits (z.B. Modulationsverfahren bei Modems)...

Verbreitete Vertreter dieser Protokolle sind RS-232 (serielle Schnittstelle) und X.21.

Sicherungsschicht

Protokolle dieser Schicht gewährleisten die Unversehrtheit der übertragenen Daten. Dazu verpacken sie die Daten in für das Medium zulässige Einheiten, steuern den Fluss der Übertragung und fügen Prüfsummen an die eigentlichen Datenpakete an, anhand derer der Empfänger den Zustand der Daten überprüfen kann. Im Fehlerfall kümmern sich die Protokolle dieser Schicht automatisch um eine Wiederholung der Übertragung.

Hier offenbart sich eine Schwäche des OSI-Modells, das die Grenzen der Schichten nicht allzu konsequent gezogen hat. So unterteilen die Realisierungen lokaler Netzwerke diese Schicht nochmals in *Media Access Control (MAC)* und *Logical Link Control (LLC)*. So organisiert z.B. Ethernet den Zugang zum Übertragungsmedium mittels einer MAC-Schicht (oft CSMA/CD - Carrier Sense Multiple Access with Collision Detect), während LLC eine Schnittstelle zu übergordneten Diensten darstellt.

Weitere wichtige Protokolle der Sicherungsschicht sind High-Level Data Link Control HDLC, das darauf basierende Point-to-Point-Protocol PPP, und das Serial Line Protocol SLIP.

Vermittlungsschicht

Auch als Netzwerkschicht bezeichnet, ist diese Schicht für den Aufbau eines virtuellen Kommunikationskanals verantwortlich. Dazu zählen das Auffinden eines Weges zum Zielrechner, die Vermittlung der Nachrichten und Pakete (eine »lange« Nachricht wird ggf. in einzelne Pakete unterteilt).

Verfolgt man den Weg eines Paketes vom eigenen Rechner hin zu einem Rechner »am anderen Ende der Welt«, so stellt man fest, dass die Route durch zahlreiche Teilnetze führt, denen mitunter vollkommen unterschiedliche Technologien (Lichtwellenleiter, Funk, Ethernet, Modem...) zugrunde liegen. Für jeden Übergang in ein neues Teilnetz wird ein Protokollwechsel in den »unteren« Schichten notwendig. An welchen Rechner eines solchen Teilnetzes das Paket als nächstes zu »routen« ist, bleibt Angelegenheit der Vermittlungsschicht.

Den bekanntesten Vertretern dieser Schicht, IP, ICMP und ARP, wenden wir uns nachfolgend zu. Weitere Protokolle sind X.25, Exterior Gateway Protcol EGP, Border Gateway Proctocol BGP, Open Shortest Path First OSPF und Routing Information Protocol RIP.

Transportschicht

Diese Schicht stellt den »anwendungsorientierten« Schichten einen logischen Übertragungskanal zur Verfügung, so dass diese ihre Daten sequentiell an die Schnittstelle senden. Protokolle der Transportschicht steuern die Blocklängen, die Geschwindigkeit, mit der Pakete an die unteren Schichten weiter gegeben werden und realisieren (oft) eine Fehlersicherung.

Damit sich Sender und Empfänger auch verstehen, muss auf beiden Seiten dasselbe Transportprotokoll zum Einsatz gelangen, d.h. beide Kommunikationspartner müssen »dieselbe Sprache sprechen«.

Bekannte Protokolle sind TCP und UDP.

Sitzungsschicht

Protokolle der Sitzungsschicht realisieren die logische Adressierung, d.h. sie sind u.a. zuständig, einen Zielrechner zu finden, der die geforderte Anforderung erfüllen kann. Weitere Funktionen sind Nutzeridentifikationen, die erneute Verbindungsaufnahme nach einem Abbruch, Wechsel der Kommunikationsrichtung usw...

Zu den Protokollen zählen der Remote Procedure Call und LU6.2.

Darstellungsschicht

Ein Protokoll der Darstellungssicht organisiert die Umwandlung von Daten und Texten in ein bestimmtes Format. Auch die Terminalemulationen ordnen sich in diese Schicht ein.

XDR und ASN.1 zählen zu dieser Gruppe.

Anwendungsschicht

Letztlich bilden die Protokolle der Anwendungsschicht die Schnittstelle zu den Anwendungsprogrammen.

Mit FTP und HTTP seien nur zwei Vertreter erwähnt.

Der TCP/IP-Protokollstack - Die Praxis

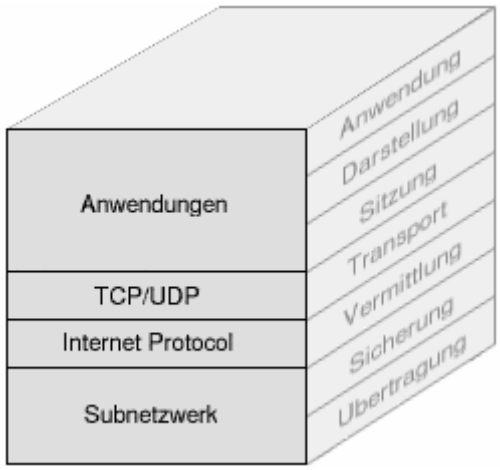


Abbildung 2: TCP/IP vs. OSI-Modell

Die TCP/IP-Architektur hatte sich bereits durchgesetzt, als die *International Standardisation Organisation* (ISO) ihr Modell eines Internet-Protokollstacks *Open System Interconnection* (OSI) veröffentlichte. Letztlich basierte das OSI-Modell auf den Erfahrungen von TCP/IP, aber es konnte sich bis heute in der Praxis nicht durchsetzen.

Der Grund für den Erfolg des TCP/IP-Ansatzes ist nicht allein in der BSD-Implementierung zu suchen. Es hat sich auch gezeigt, dass die gewählte Struktur den Zusammenhängen zwischen den Komponenten der Hardwareebene wie auch der Anwendungsebene besser gerecht wird. So liefern Hersteller der Übertragungstechnik die Software der Ebenen 1 und 2 mit dieser aus, während der Anwendungsentwickler seine Applikation mit Eigenschaften der »oberen« OSI-Schichten ausstattet.

Nicht zuletzt verfügt jedes Betriebssystem, das den Zugang zum Internet ermöglicht, über eine Implementierung

des TCP/IP-Protokollstacks.

Im weiteren Verlauf geben wir zu den beschriebenen Protokollen deren Einordnung in das OSI-Referenz-Modell an.

Sicherungsschicht: Das Point to Point Protocol

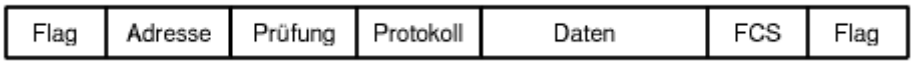


Abbildung 3: Point to Point Protocol

Das **Point-to-Point-Protokoll** ist eine Erweiterung von HDLC und ermöglicht permanente Punkt-zu-Punkt-Verbindungen. Häufigster Einsatzbereich sind Wählverbindungen (Modem).

Das enthaltene Protokollfeld ermöglicht die Unterstützung beliebiger Vermittlungsprotokolle; die Prüfsumme erlaubt die Verifizierung des Dateninhalts.

Als optionale Eigenschaften können Implementierungen Folgendes enthalten:

- Einwahl nach Bedarf, Verbindungsabbau nach Zeit- oder Gebührenüberschreitung
- Parallele Verwendung mehrerer Leitungen
- Einfache Filtermechanismen
- Komprimierung der Daten und/oder des Headers

Das PPP handelt mit der Gegenseite eine Maximale Empfangspaketgröße aus, das ist der Grund für den verzögerten Verbindungsaufbau über analoge Telefonleitungen.

Sicherungsschicht: Das Serial LIne Protocol

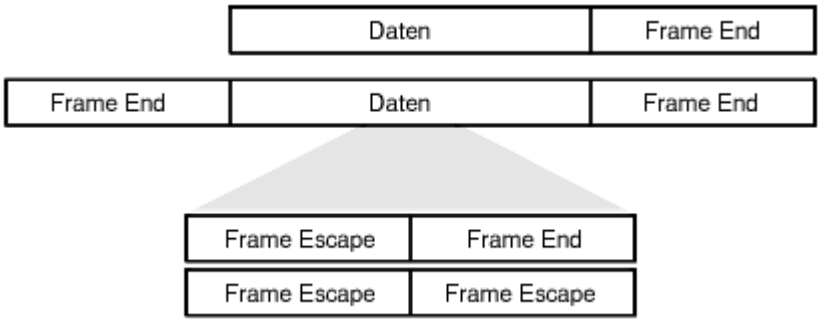


Abbildung 4: Serial Line Protocol

Das **Serial Line Protokoll** ist das zweite Protokoll der Sicherungsschicht, das bei Modemverbindungen zum Einsatz gelangt. Wie der Name bereits besagt, dient es der Verbindung zweier Rechner über eine serielle Leitung. Als Protokoll der Schicht 3 kommt allerdings einzig IP in Frage; auch beinhaltet das Protokoll keinerlei Sicherungsmaßnahmen.

Mögliche Einsatzgebiete sind sichere Leitungen, bspw. ein Nullmodemkabel.

Auf SLIP baut das **CSLIP** (Compressed SLIP) auf, das den Header von TCP-Paketen nach einem Verfahren von Van Jacobsen komprimiert.

Vermittlungsschicht: Das Address Resolution Protocol

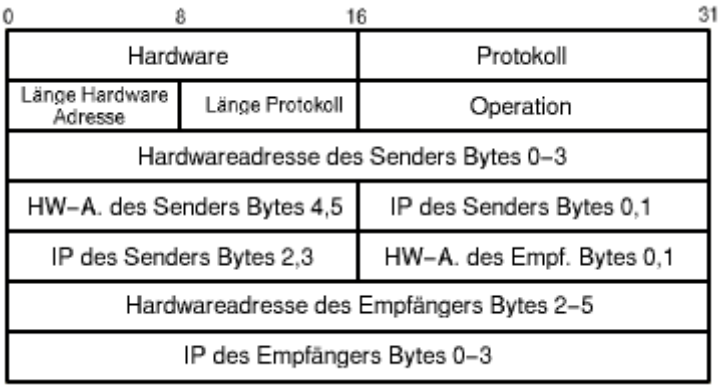


Abbildung 5: Address Resolution Protocol

Während im Internet Pakete anhand ihrer IP-Adresse vermittelt werden, erfolgt in lokalen Netzen die Adressierung mittels Hardwareadressen. So besitzt jede Ethernetnetzwerkkarte eine (weltweit) eindeutige 48 Bit lange Adresse.

Um nun ein IP-Paket vermitteln zu können, muss ein Rechner die Hardwareadresse des Rechners kennen, an den das Paket als nächstes zu senden ist (Bis ein Paket seinen Bestimmungsort erreicht, durchläuft es mitunter zahlreiche Rechner. In jedem dieser Zwischenknoten muss entschieden werden, wohin das Datenpaket im nächsten Schritt zu senden ist. Auf dieses als **Routing** bezeichnete Verfahren werden wir später eingehen).

Für die Adressermittlung bieten sich mehrere Vorgehensweisen an:

- Umsetzung von IP- in Hardwareadressen mittels statischer Tabellen
- Verwendung eines Algorithmus zur Berechnung der Hardwareadresse aus der IP-Adresse
- Verwendung dynamischer Tabellen, die regelmäßig aktualisiert werden

In der großen, weiten Welt des Internets ist nichts statisch. Rechner kommen hinzu oder gehen vom Netz, Rechner erhalten eine neue IP-Adresse oder eine andere Netzwerkkarte... die Zuordnung mittels statischer Methoden zu versuchen, würde jeden Administrator zum Ausdauerathleten ausbilden.

Ein Algorithmus, der 48 Bit auf derer 32 abbildet, kann niemals ein eindeutiges Ergebnis liefern, zumal eine Gruppierung von Rechnern eines bestimmten Bereiches zu einem Subnetz unmöglich realisierbar wäre.

Bleibt der dynamische Aufbau von Tabellen. Ein Rechner, der ein IP-Paket senden möchte, schaut nun erst einmal in seiner ARP-Tabelle nach, ob ein zugehöriger Eintrag existiert. Wenn ja, verwendet er die gespeicherte Hardwareadresse bereits. Wenn nein, kommt das Address Resolution Protocol zum Zuge. Der Rechner sendet nun einen Broadcast (eine Nachricht, die gleichzeitig an mehrere Rechner eines zumeist lokalen Netzwerkes geht) aus, mit der Bitte, dass der Rechner, zu dem die IP-Adresse gehört, seine Hardwareadresse übermitteln möge. Genau jener Rechner sendet darauf hin eine Antwort. Erst jetzt kann der Rechner das IP-Paket zu seinem nächsten Bestimmungsort schicken. Die soeben ermittelte Hardwareadresse bekommt ihren Platz in der ARP-Tabelle (»Caching«), um im sehr wahrscheinlichen Falle einer folgenden Übermittlung sofort den Adressat parat zu haben.

Diese ARP-Tabelle ist allerdings in ihrer Kapazität beschränkt, somit werden ältere Einträge zyklisch entfernt (meist nach 10 Minuten des Nichtgebrauchs). Das Verfahren hat den Vorteil, dass die Tabelle auch auf Veränderungen im Netz reagiert, weil bspw. die IP-Adresse einem anderen Rechner zugewiesen wurde...

Vermittlungsschicht: Das Internet Protocol

Die heute gebräuchlichen Adressen des Internet-Protokolls sind 32 Bit lang, die häufigste Notation erfolgt byteweise als Dezimalzahl, bspw. »127.211.7.9«. Der Mathematiker errechnet rasch, dass mit 32 Bit 2^{32} = 4.294.967.292 Rechner adressierbar wären. Der Praktiker interveniert, dass sich nicht alle Adressen nutzen lassen, da etliche Adressen und Adressbereiche für bestimmte Funktionen reserviert sind. Auch existieren genügend Lücken im Adressraum, da IP's im Block an lokale Netzwerke vergeben werden, diese aber nur selten ihr Kontingent voll ausschöpfen.

Fazit ist, dass schon heute die Zahl der verfügbaren Adressen den Bedarf bei weitem nicht mehr decken kann und eine Aufweitung der Adressen erforderlich ist. Aus diesem Grund steht der designierte Nachfolger des korrekt als IPv4 bezeichneten Protokolls seit Jahren (erste Spezifikation 1995) in den Startlöchern. Anliegen dieses IPv6 ist nun die Definition eines Protokolls der Vermittlungsschicht, das mit 128 Bit Adressen arbeitet. Die damit erzielte Größe erscheint übertrieben (jedem Quadratmillimeter auf dem Globus ließen sich hiermit ca. 667 Billiarden IP's zuweisen), jedoch ist man auf weite Sicht auf der sicheren Seite.

Mit dem Adressformat nach IPv4 befasst sich der Abschnitt Netzwerkstrukturen, IP-Adressen.

Eigenschaften des IPv4

Bei der immensen Bedeutung, welche gerade dieses Protokoll für die Paketvermittlung im Internet erlangt hat, lohnt sich ein tieferer Einblick in dessen Fähigkeiten.

Zunächst gilt zu vermerken, dass es sich um ein **verbindungsloses** Protokoll handelt, d.h. es arbeitet, ohne dass eine Verbindung zum Partner zuvor aufgebaut wurde (analog zum Unterschied zwischen einem Telegramm und einem Telefonat). Die maximale **Paketgröße beträgt 65535 Bytes**. Ein Paket durchläuft auf seinem Weg zum Empfänger meist verschiedenste Subnetze, die ihrerseits nur eine kleinere Paketgröße unterstützen. Das IP beinhaltet deswegen einen Mechanismus zur **Fragmentierung** von Paketen, d.h. dass ein für ein zu durchlaufendes Subnetz zu großes Datenpaket zerlegt wird und nun mehrere IP-Pakete ihren Weg zum Empfänger suchen. Der Zusammenbau der Paketteile erfolgt erst beim endgültigen Empfänger, da die Teilpakete durchaus auf unterschiedlichen Routen ihr Ziel finden.

Eine **Prüfsumme** stellt die Unversehrtheit des IP-Kopfes sicher, nicht jedoch die der enthaltenen Daten. Deren Überwachung obliegt dem Protokoll der nächsthöheren Schicht; ein Code im IP-Kopf (**Protokoll**-Feld) verweist auf den Typ des Protokolls (bspw. steht eine 6 bei TCP und eine 17 bei UDP).

Die letzte erwähnte Eigenschaft, die die **Lebensdauer** eines IP-Pakets begrenzt, garantiert, dass ein nicht vermittelbares Paket nicht endlos im Netz kursiert, sondern nach Ablauf seiner Lebenszeit verworfen wird. Früher wurde hier tatsächlich mit einer Zeiteinheit gearbeitet, jedoch wich diese bald der maximalen Anzahl Stationen (»Hops«), die ein Paket maximal durchlaufen darf. Jeder Rechner, der das Paket weiterleitet, verringert diesen Wert um 1. Erreicht er in einem Rechner den Wert 0 und handelt es sich nicht um den Zielrechner, so sendet dieser Rechner ein ICMP-Protokoll an den Absender und verwirft das eigentliche Paket.

Wichtige Felder des IPv4

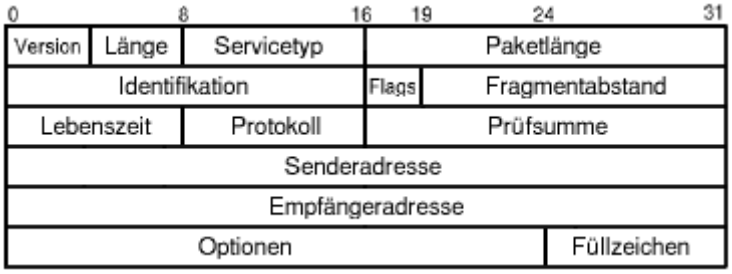


Abbildung 6: Internet Protocol Version 4 (IPv4)

Die Bedeutung mancher Felder ergibt sich schon aus deren Namen, diejenige, deren Bedeutung nicht sofort ersichtlich ist, sollen nun erläutert werden:

Versionsnummer

Version des Protokolls, also 4 bei IPv4 und 6 bei IPv6

Länge

Größe des IP-Kopfes in Worten (32 Bit = 1 Wort); hierdurch ist die Angabe von Optionen variabel

Servicetyp

In der Linux-Implementierung wird dieses Feld nicht berücksichtigt. Es dient dazu, ein Paket mit unterschiedlicher Priorität oder erhöhter Zuverlässigkeit... zu vermitteln

Paketlänge

Länge inklusive der IP-Kopfes in Worten

Identifikation

Vom Absender vergebene eindeutige Nummer, anhand derer einzelne Fragmente im Zielrechner in der richt Reihenfolge zusammen gesetzt werden können

Flags

Das erste Bit wird nicht benutzt, das zweite Bit gibt an, dass ein Paket nicht fragmentiert werden darf. Ist e solches Paket zu groß für ein Teilnetzwerk, muss es verworfen werden. Das dritte Bit gibt an, ob dem Pake noch weitere Teile folgen

Fragmentabstand

Relative Lage des Paketes, wenn dieses Teil eines zuvor größeren Paketes war (Fragmentierung)

Lebenszeit und Protokoll

Siehe unter Eigenschaften des IPv4

Optionen

In erster Linie werden diese Optionen von Netzwerkadminstrationswerkzeugen genutzt. Sie können bspw. verwendet werden, um jeden durchlaufenden Rechner anzuweisen, seine IP-Adresse und/oder einen Zeitstempel zu hinterlegen. Ebenso kann eine Route, die ein Paket zurücklegen soll, festgeschrieben werde

IPv6 - Das neue Format

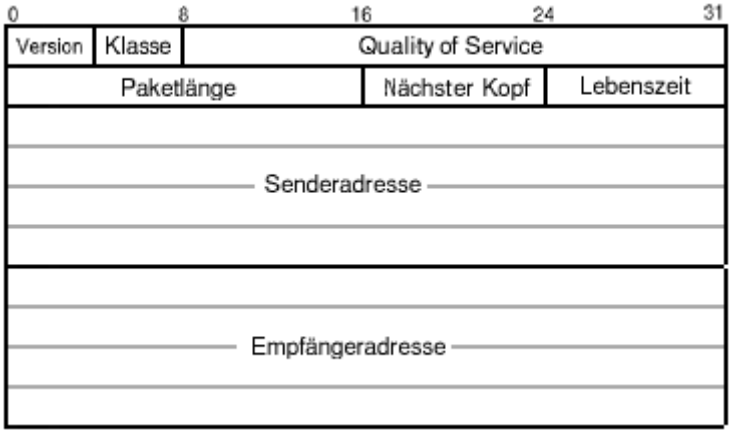


Abbildung 7: Internet Protocol Version 6 (IPv6)

Versionsnummer

Version des Protokolls (6 bei ipv6); die Lage des Feldes entspricht exakt dem des IPv4-Protokolls, sodass Rechner sofort den Typ des Protokolls erkennen können.

Klasse

Kann vom Absender gesetzt werden, um dem Paket eine bestimmte Priorität einzuräumen, sodass es

»bevorzugt« (bei höherer Priorität) vermittelt wird. Vorgesehen sind derzeit:

0	Unspezifizierte Daten	**1**	News und Ähnliches
2	Mail	**3**	Reserve
4	Datentransfer (FTP,...)	**5**	Reserve
6	Interaktive Anwendungen	**7**	Netzwerkkonfiguration (SNTP)

Qualitiy of Service

Kennzeichnet die Daten, um sie beim Empfänger einer speziellen Behandlung zuzuführen. Wichtigstes Beisp sind Echtzeitanwendungen, wo die Daten unmittelbar nach dem Empfang zur Anwendung weiterzureichen s (bei mehreren eingetroffenen Pakten also unabhängig von der Reihenfolge des Empfangs).

Paketlänge

Im Unterschied zum IPv4 steht hier die Paketlänge ausschließlich des Protokollkopfes (da die Länge des Ko bei IPv6 konstant ist). Enthält das Feld eine "0", steht die Paketlänge in einem speziellen »Erweiterungs«-Protokollkopf (siehe »Nächster Kopf«), sodass Längen > 64kByte übertragen werden können

Nächster Kopf

Kennzeichnet den Typ des eingebetteten Protokolls, der unmittelbar auf den IP-Kopf folgt. Derzeit existiere verschiedene Erweiterungs-Header, die bspw. für die Verschlüsselung der Daten oder eine Authentifizierung zuständig sind. Jeder dieser Header beinhaltet selbst ein »Nächster Kopf«-Feld, sodass für ein Paket auch mehrere Erweiterungen zulässig sind. Der »letzte« der Erweiterungsheader eines Pakets trägt im »Nächster Kopf«-Feld den Typ des Protokolls der Transportebene (bspw. Tcp).

Lebenszeit

Anzahl Stationen (Hops), die das Paket maximal durchlaufen kann

Senderadresse

IP des Absenders (128 Bit)

Empfängeradresse

IP des Empfängers (128 Bit)

Eine Prüfsumme wie bei IPv4 ist nicht mehr vorgesehen, da die Neuberechnung auf jeder Zwischenstation recht teuer ist (geändertes »Lebenszeit«-Feld) und unter- bzw. übergeordnete Protokolle i.d.R. ohnehin eine Fehlerbehandlung beinhalten. Auch entfällt eine Fragmentierung. Ist ein Datenpaket zu groß für eine Übertragungsstrecke, wird es verworfen und der Absender per ICMP darüber in Kenntnis gesetzt. Der Absender sollte daraufhin das Paket in kleineren Einheiten erneut versenden.

Vermittlungsschicht: Das ICMP-Protokoll

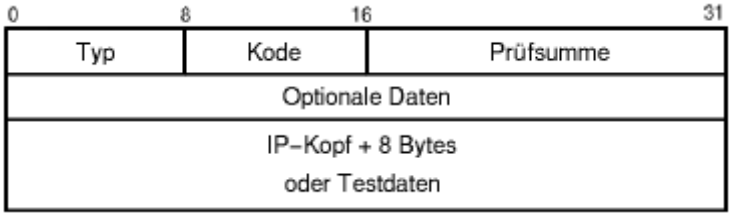


Abbildung 8: Internet Control Message Protocol

Wenn Fehler bei der Vermittlung eines Pakets auftreten, wird i.d.R. eine ICMP-Nachricht an den Absender geschickt. Aber auch zu Diagnosezwecken dient dieses Protokoll, dessen **Typ**-Feld seine eigentliche Aufgabe offenbart:

Typ = 0

Antwort auf eine Echo-Anfrage

Typ = 3

Zielrechner nicht erreichbar (i.d.R. existiert keine Route)

Typ = 4

Paket gelöscht (es wurde von einem Rechner verworfen, weil dessen Kapazität erschöpft war)

Typ = 5

Wechsel der Route, wird von einem Vermittlungrechner gemeldet, wenn er feststellt, dass das Paket eine kürzere Route wählen könnte

Typ = 8

Echo-Anfrage

Typ = 11

Lebenszeit abgelaufen

Typ = 12

Probleme im IP-Kopf

Typ = 13

Zeitstempel-Anforderung

Typ = 14

Zeitstempel-Antwort

Der **Kode** unterteilt den Pakettyp ggf. in weitere Unterfunktionen. Die **Prüfsumme** wird über das gesamte Paket gebildet. Das Feld **Optionale Daten** kann vom jeweiligen Kode abhängige Informationen enthalten.

Schließlich wird im Falle einer aus einem verworfenen IP-Paket resultierenden ICMP-Nachricht der Anfang des IP-Paketes als Daten übermittelt bzw. Testdaten, wenn es sich um ein Diagnosepaket handelte.

Transportschicht: Das Transmission Control Protocol

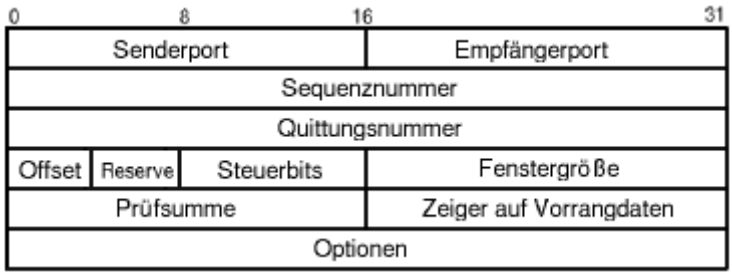


Abbildung 9: Transfer Control Protocol

Aus Sicht einer Anwendung eröffnet das *Transmission Control Protocol* einen bidirektionalen, virtuellen Datenkanal zwischen den beiden Kommunikationsendpunkten. Die Daten werden scheinbar in einem Fluss übertragen. Intern gehen diese natürlich blockweise übers Netz, wobei die Blockgröße dynamisch anhand von Parametern wie der

Netzauslastung, der Fenstergröße oder der Empfangs- bzw. Sendepuffer angepasst wird. Im Unterschied zum nachfolgend erwähnten *User Datagram Protocol* kümmert sich TCP selbst um die sichere Übertragung. Es verwendet hierzu Sequenznummern, Prüfsummen, Quittungen und Wiederholung des Transfers bei einer Zeitüberschreitung. Andere wesentliche Eigenschaften sind das Sliding-Window-Verfahren und die Kennzeichnung von Vorrangdaten.

Die Felder des Protokollkopfes bedeuten:

Senderport, Empfängerport

Analog zum Telefonat spielt der Sender einen aktiven und der Empfänger einen passiven Teil. Der Sender adressiert den Partner über IP-Adresse des Zielrechners und eine 16-Bit lange Portnummer. Beide zusamme bezeichnet man unter Unix als **Socket**. Um den Empfänger adressieren zu können, muss der Sender desse Portnummer kennen. Der Sender wiederum kann (meist) eine beliebige freie Portnummer wählen, da er sei eigene Nummer dem Kommunikationspartner mitteilt. Für die Standarddienste stehen die Portnummern in d Datei /etc/services. Des Weiteren ist anzumerken, dass UDP einen eigenen Adressraum verwendet und glei Portnummern sich somit nicht überschneiden.

Sequenznummer

Dieser 32-Bit Wert kennzeichnet eindeutig die Stellung eines Pakets innerhalb des Datenstroms in Senderichtung. Die initiale Sequenznummer wird zu Beginn des Verbindungsaufbaus von jedem Kommunikationspartner festgelegt, wobei gilt, dass sie für die maximal mögliche Lebensdauer des Pakets (TimetoLive des Internet Protokolls) bez. der verbundenen Rechner eindeutig ist.

Die Sequenznummer eines folgenden Pakets berechnet sich aus der initialen Sequenznummer und der Anza bisher gesendeter Bytes. Somit ist es möglich, bei Verlust oder Beschädigung eines Pakets gezielt dieses wiederholt zu senden.

Quittungsnummer

Die Quittungsnummer sendet der Empfänger eines Pakets als Bestätigung für den Empfang. Sie gibt an, wi viele Bytes bislang beim Partner unversehrt eingetroffen sind. Sollten Sequenznummer oder Quittungsnumr im Laufe einer Sitzung einmal überlaufen, so wird bei 0 fort gefahren.

Offset

Das Feld enthält die Länge des TCP-Kopfes in 32-Bit Worten. Anhand dieser wird der Beginn der enthaltene Daten ermittelt.

Reserve

Wird nicht verwendet.

Steuerbits

Die 6 Steuerbits bedeuten:

URG

Die Daten im Feld »Vorrangdaten« sind gültig

ACK

Die Quittungsnummer ist gültig

PSH

Die Daten sollten sofort der Anwendung übergeben werden

RES

Wunsch nach Aufbau einer Verbindung

Beenden der Verbindung. Ein Partner, der dieses Bit setzt, muss seinerseits die Verbindung offenha
auch das Gegenüber das FIN-Bit sendet. Er selbst darf aber keine weiteren Daten senden (Ausnahr
die Quittungen auf eintreffende Pakete).

Fenstergröße

Momentane Kapazität des Empfangspuffers auf Absenderseite. Sein Gegenüber darf maximal so viele Dater
(auch aufgeteilt auf mehrere Pakete) senden, wie durch die Fenstergröße angegeben ist. TCP arbeitet nun
dass es versucht, die Fenstergröße automatisch an die Kapazität des Übertragungsmediums anzupassen. D
wird das Fenster allmählich vergrößert, bis Pakete aufgrund des zu hohen Datenaufkommens verworfen we
müssen. Treten nun vermehrt solche Übertragungsfehler auf, wird das Fenster wieder verkleinert, um es
anschließend erneut mit einer Erhöhung zu versuchen. Dieses Sliding-Window-Prinzip lässt sich sehr gut be
Download von Dateien beobachten, wobei die Datentransferrate ständig schwankt.

Prüfsumme

Prüfsumme über das gesamte Paket.

Zeiger auf Vorrangdaten

Der Zeiger gibt einen Offset innerhalb der Daten im Paket an. Die dem Zeiger folgenden Daten werden som
als besonders wichtig deklariert. Eine Anwendung wird beim Eintreffen solcher Daten unterrichtet. Sie sollte
ihre bisherige Arbeit unterbrechen und die dringliche Nachricht bearbeiten. Gebrauch von diesem Mechanis
macht wohl nur Telnet.

Optionen

Beim Verbindungsaufbau wird meist "MaximumSegmentSize" gesendet, um dem Partner mitzuteilen, dass
größere Pakete empfangen werden können. Die weiteren Optionen sind "EndOfOptionList" und "NoOperatio

Das Zusammenspiel von Sequenz- und Quittungsnummer wird in den meisten Fällen die Unversehrtheit der übertragenen Daten garantieren. Jedoch verlangt eine ausstehende Quittung das Warten auf diese. Ist nun der Partner ausgefallen, würde ein Sender bis in alle Ewigkeit auf die Bestätigung des Empfangs seines Pakets lauern. Um einen solchen "Hänger" zu verhindern, werden beim Versand eines Pakets gleich mehrere Zeitgeber gestartet.

Der wichtigste Ticker stoppt die seit dem Senden vergangene Zeit. Läuft er ab, ohne dass eine Quittung eintraf, muss das Paket erneut auf die Reise geschickt werden. Diese Zeitspanne wird allerdings dynamisch berechnet (aus dem Mittelwert der bisherigen Paketlaufzeiten), sodass sie sich an veränderte Situationen (hohe Netzlast, alternative Route) allmählich anpasst.

Ein weiterer Wecker wird verwendet, um die Bereitschaft des Empfängers zu überprüfen. Dieser Zeitgeber garantiert, dass ein Datentransfer nicht blockiert, weil dessen Fenstergröße auf 0 steht, das Paket zum Öffnen des Empfangsfensters aber verloren ging.

Der letzte hier vorgestellte Timer hält einen Port noch eine gewisse Zeit geschlossen, nachdem die Verbindung schon abgebaut wurde. Die Zeitspanne entspricht in etwa der maximalen Lebensdauer (TimeToLive) eines Datenpakets und ist nützlich, um die nächste auf dem selben Port eröffnete Verbindung nicht durch alte irrgeleitete Pakete durcheinander zu bringen.

Transportschicht: Das User Datagram Protocol

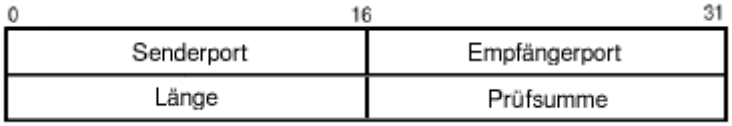


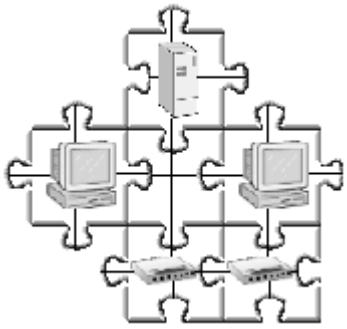
Abbildung 10: User Datagram Protocol

Neben TCP spielt das *User Datagram Protocol* eine bedeutende Rolle als Transportprotokoll. Es arbeitet verbindungslos und garantiert keinen Erfolg der Übertragung. Es beinhaltet einzig eine Prüfsumme über die Daten, um beim Empfänger deren Unversehrtheit kontrollieren zu können. Dienste, die UDP verwenden, implementieren häufig eigene Routinen zur Fehlersicherung.

Der schlanke Protokollkopf und der Verzicht auf jegliche Sicherungsmechanismen oder eine Flusssteuerung prädistinieren das Protokoll für zeitkitische Übertragungen auf sicheren Medien (wo ein Datenverlust ziemlich unwahrscheinlich ist). Auch verwenden die meisten RPC-basierten Dienste UDP, da eine solche Abfrage eher einen Telegrammcharakter trägt. (einmalige, kurze Nachrichten).

Struktur des Netzwerkes

Übersicht



Die Komplexität eines Netzwerks wächst mit der Anzahl zu verbindender Komponenten. Zwei Rechner lassen sich einfach mittels eines gekreuzten Kabels verbinden. Kommt jedoch noch ein dritter hinzu, so bieten sich als Lösungen der Einsatz eines **Hubs**, eines **Switches** oder aber die Bestückung eines Rechner mit zwei Netzwerkkarten an.

Hängen nun gar tausend Rechner an einem Strang, spielen plötzlich Faktoren wie Netzwerkbandbreite oder Ausfallsicherheit eine Rolle, die im privaten Versuchsnetzwerk niemals zu einem Problem erwachsen. Schon stellt sich die Frage, ob es überhaupt sinnvoll ist, eine größere Anzahl Computer in einem einzigen Netzwerk zu vereinen. Aber ja doch! Das Internet ist doch auch nur ein Netzwerk und da tummeln sich schließlich Millionen von Rechnern! Wie sollten da die 1000 Rechner in meinem Netz ein Problem darstellen?

Stellen Sie sich vor, Sie säßen mit 999 anderen Leuten in einem großen Raum. Allmählich bilden sich kleine Diskussionrunden zu je zwei Teilnehmern. Je mehr Gespräche in Gang kommen, desto höher wird die Lautstärke. Bald wird der Geräuschpegel eine Schwelle überschreiten, bei dem Sie ihrem Gegenüber nur noch unvollständig folgen können. »Wie bitte?« wird Ihre wichtigste Botschaft.

Ebenso verhält es sich in der Computerwelt. Drängeln sich zu viele Rechner um die Bandbreite eines Netzwerkes, wird es irgendwann zu einer Sättigung kommen, wo ein beachtlicher Teil der Kommunikation einzig dazu dient, fehlerhafte Daten erneut auszutauschen, weil einzelne Pakete durch gleichzeitiges Senden mehrerer Rechner kollidierten und damit unbrauchbar wurden. Die wiederholte Datensendung bedeutet allerdings zusätzliche Netzwerklast; eine Verschärfung der Situation ist vorprogrammiert.

Jetzt ist es unmöglich, eine pauschale Grenze anzugeben, wie viele Rechner in einem Netzwerk integriert sein sollten. Das hängt ebenso stark vom Nutzungsprofil ab, wie auch von der Bandbreite des Übertragungsmediums und von weiteren Faktoren.

Dieser Abschnitt soll deshalb zunächst einen Abriss vermitteln, welche Technik heute in welchem Umfeld zum Einsatz gelangt. Er soll Begriffe erläutern, eingesetzte Techniken benennen und auch Alternativen präsentieren. Aber neben all der Theorie spannen Fakten zu IP-Adressen, Subnetzen und zur Paketvermittlung den Bogen zur Praxis.

LAN, WAN und andere Begriffe

LAN (Local Area Network) hat sich im Sprachgebrauch als Begriff für das **lokale Netz** etabliert. Was aber bedeutet *lokal*?

Lokal hat zunächst einmal etwas mit der räumlichen Ausdehnung zu tun. Genauso wie sich mancher Mitbürger kaum mehr vorstellen kann, dass Distanzen über 5 Kilometer durchaus mit den eigenen Füßen überbrückt werden können, während der Leistungswanderer erst jenseits der 50km eine Beanspruchung verspürt, genauso weit gestreut ist die Definition von der Ausdehnung eines lokales Netzwerks. Es reicht von wenigen Metern bis zu 10km, wobei die Grenzen fließend sind.

Wohl eindeutiger hinsichtlich der Abgrenzung zu **WAN**s (Wide Area Networks) ist der Erklärungsversuch bez. der Übertragungsgeschwindigkeit. Hier geht man von 10..100 (1000) Mbit/Sekunde aus, wobei der geklammerte Ausdruck die noch gering verbreitete Technik des Gigabit-Ethernets repräsentiert.

Drittes Kriterium eines lokalen Netzes ist sein homogener Aufbau, da (fast immer) alle Rechner durch ein und dieselbe Technologie miteinander verbunden sind. Wichtigster Vertreter der LAN-Technik ist das Ethernet, aber auch Token Ring, Fiber Distributed Data Interface (FDDI), Asynchron Transfer Mode (ATM), Funknetze, Myrinet oder Scalable Coherent Interface (SCI) gliedern sich in diese Sparte ein.

Die Weitverkehrsnetze (WAN) hingegen verbinden mehrere lokale Netze. Der Wirkungsbereich der dahinter stehenden Betreiber beschränkt sich zumeist auf ein Land, häufig existieren mehrere WANs parallel und überschneiden sich in ihrer Ausdehnung.

Aus Kostengründen werden im WAN nicht annähernd solche Übertragungsraten angeboten wie es in lokalen Netzen der Fall ist. Typische Raten liegen bei unter 2 MBit/Sekunde. Durch Bündelung von Kapazitäten findet man aber auch Angebote von wesentlich mehr als 100 MBit/Sekunde (ISDN H12 = 140 MBit/Sekunde).

Als Techniken zur Datenübertragung dominieren nach wie vor Telefonnetz und ISDN. Das deutsche Forschungsnetz (DFN) basiert auf X.25. Aber auch Funklösungen sind im Einsatz.

I.A. muss für die Benutzung von WAN-Angeboten eine Gebühr entrichtet werden. Entweder in Form einer Leitungsmiete oder durch Verrechnung des produzierten Datenaufkommens.

Ein **GAN** (Global Area Network) ist eher ein abstrakter Begriff. Hinter ihm steht nicht wirklich eine Technik, sondern es bezeichnet nur die Verbindung mehrere LANs und WANs. Das Internet ist der bekannteste Vertreter dieser Zunft.

Topologien

Die Topologie beschreibt die logische Verbindungsstruktur eines Netzwerks. In der Theorie scheint die Klassifizierung eindeutig. Praktisch finden sich allerdings häufig Mischformen, deren eindeutige Zuordnung zu einer Topologie eine Frage der Betrachtungsweise ist.

Die klassischen Topologien im LAN

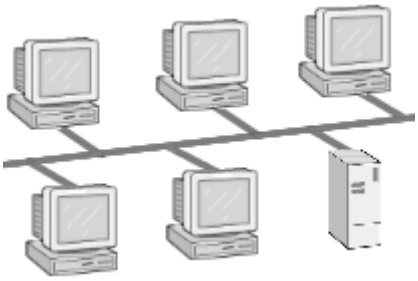


Abbildung 1: Die Busstruktur

Fakt ist, dass die **Busstruktur** (Abbildung 1) das verbreitetste Schema der Rechnerverbindung darstellt. Alle Rechner eines Netzwerks teilen sich denselben Bus, genauso wie es rechnerintern Prozessor, Speicher und Peripherie handhaben. Analog zum Systembus eines Computers ist auch die mögliche Anzahl der Rechner an einem Netzwerk-Bussystem begrenzt. Kritisch bei Bussystemen ist die Zugangssteuerung zum Verbindungsmedium, da zu einem Zeitpunkt stets nur eine Verbindung aktiv sein kann. Die Verfahren reichen von vorsorglicher Vermeidung konkurrierender Zugriffe bis hin zur Erkennung von Kollissionen. Der typische Vertreter bez. LANs ist das Ethernet.

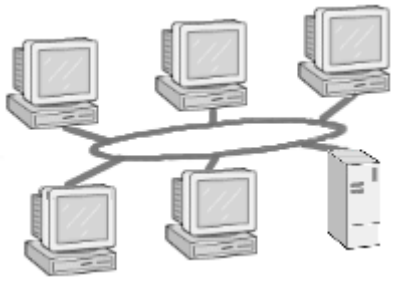


Abbildung 2: Die Ringstruktur

Vor allem mit Token Ring hat bei lokalen Netzwerken die **Ring-Topologie** als Vertreter der so genannten Punkt-zu-Punkt-Verbindungen eine gewisse Verbreitung erlangt. Ein Rechner kann (theoretisch) nur mit seinen unmittelbaren Nachbarn kommunizieren. Eine Verbindung zu weiteren Rechnern im Ring gelingt nur unter Verwendung der Zwischenrechner als »Vermittlerstationen«. »Theoretisch« deutet bereits eine weitere Einschränkung an, da in praktischen Vermittlungsverfahren in Ringstrukturen die Kommunikation nur in eine Richtung funktioniert, d.h. ein Rechner kann bspw. direkt zu seinem rechten Nachbarn senden, benötigt aber alle weiteren Rechner, um dem linken Nachbarn ein Paket zukommen zu lassen. Ring-Netzwerke arbeiten ausschließlich mit »Token«, einem Rahmen, der fortwährend im Ring kreist und in den ein Rechner - insofern das Token nicht belegt ist - seine Nachricht platzieren kann. Der Zielrechner entnimmt dem Token die Daten und markiert dieses wieder als frei, sodass ein anderer sendewilliger Rechner das Token nun belegen kann. Dieses Token-Verfahren kann ebenso auf reinen Bussystemen angewendet werden.

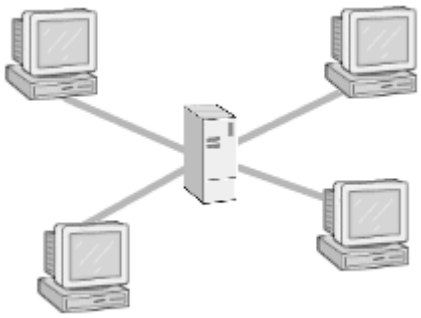


Abbildung 3: Die Sternstruktur

Die Beschreibung der **Sternstruktur** ist simple: Jeder Rechner im Netzwerk ist direkt mit der zentralen Komponente verbunden, welche im einfachsten Fall auf einer Leitung eintreffender Daten diese auf alle anderen Leitungen durchreicht. Betrachtet man die Struktur realer Ethernet- und Token-Ring-Netzwerke, wird man immer wieder auf Analogien zur Sternstruktur stoßen.

Ein auf Ethernet basierendes LAN größeren Umfangs besteht nur selten aus einem einzelnen Bussystem. I.d.R. werden mehrerer solcher Busse gekoppelt, häufig gar in einer zentralen Komponente (allgemein als »Sternkoppler« bezeichnet) , sodass tatsächlich eine Sternstruktur resultiert. Je nach »Intelligenz« der Komponente fügt sie die einzelnen Stränge des Netzes zu einem großen Bussystem zusammen, indem sie alle Daten ohne Rücksicht auf deren Zieladresse in jeden Anschluss einspeist oder aber sie »filtert« die Pakete und reicht sie nur in den Teil des Netzwerks weiter, in dem der Empfängerrechner liegt.

Auch bei Token-Ring-Netzwerken mit mehreren Teilnehmern werden keine kilometerlangen Leitungen verlegt. Der Ring selbst ist in einem einzelnen Hardwarebaustein (»Ringleitungsverteiler«) realisiert, von welchem aus Stränge zu den einzelnen Rechnern gehen. Rein optisch gleicht es somit einem Stern.

Hinterm Horizont

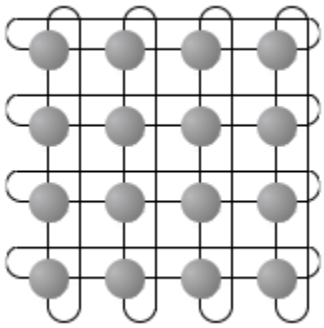


Abbildung 4: 2D-Torus

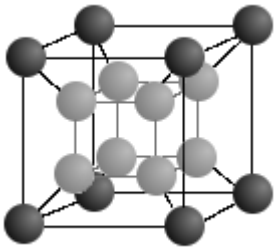


Abbildung 5: Hypercube (3. Dimension)

Sind hiermit alle Topologien genannt? Mitnichten! Einzig die für lokale Netzwerke relevanten Strukturen wurden erwähnt. Verbindungsnetzwerke gänzlich anderer Art finden sich in zahlreichen Parallelrechnern. Gerade dort kommt es auf kürzeste Signalwege bei maximaler Bandbreite an. Während Bus- und Ringstrukturen nur bei kleineren Modellen (< 64 Prozessoren) von praktischen Nutzen sind, gelangen bei so genannten massiv-parallelen Rechnern Gitterstrukturen, Tori (Gitter, deren Außenknoten miteinander gekoppelt sind), Hypercubes oder Baumstrukturen u.a.m. zum Einsatz. Auch dynamisch erzeugte Verbindungen (Kreuzschienenverteiler, Delta-Netzwerke) sind eine Domäne der Parallelrechentechnik.

Die Elemente eines Netzwerks

Um die einzelnen Endgeräte (Computer, Terminals, Netzwerkdrucker etc.) miteinander zu koppeln, gelangen verschiedenste Komponenten zum Einsatz. Eine grobe Unterteilung erfolgt anhand der Signalbehandlung. Für Bauteile, die Signale unverändert weiterreichen, wird oft der Begriff der **passiven Komponente** angewandt. Demzufolge sind **aktive Komponenten** Elemente, die eine Signalaufbereitung vornehmen. Den Aufgaben der wichtigsten Verteter beider Gruppen soll sich die folgende Abhandlung widmen.

Passive Komponenten

Abbildung 6: Verbreitete Steckverbindungen

Zu den passiven Bauelementen zählen die **Steckverbindungen** und **Kabel**. Erstere werden in lokalen Netzwerken in den Ausführungen **BNC** und **RJ45** angeboten.

BNC wird im Zusammenhang mit 10base2 (»Ethernet-Jargon« für Koaxial-Kabel) benutzt. Die Anbindung einer Station erfolgt entweder durch Auftrennung des Kabels und Einfügen eines BNC-T-Stücks (Abbildung 6) oder auch ohne Unterbrechung des Leiters (»Vampirstecker«). Im Falle dieser klassischen Bus-Topologie müssen alle freien Enden durch einen **Abschlusswiderstand** »geschlossen« werden, da Pakete ansonsten an diesen reflektiert werden und nachfolgende Pakete damit verwischen würden (Abbildung 7).

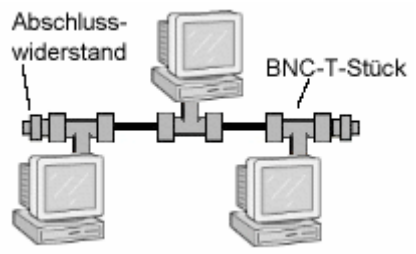


Abbildung 7: Koax-Netz

RJ45 ist die zum 10BaseT ([Un]shieled Twisted Pair) Anschluss passende Verbindung. Im Zusammenhang mit Ethernet bedingen Twisted-Pair-Kabel den Einsatz von aktiven Verteilerelementen (Bridges), um die einzelnen Stationen miteinander zu verbinden (daraus resultiert eine Sternstruktur).
Die Leistungsdaten eines Netzwerks werden durch die Kabel bestimmt, die Ausführung der Steckverbindungen ist letztlich irrelevant.

Abbildung 8: Verkabelungstechnik

Die einzusetzende **Kabeltechnik** hängt vor allem von zwei Faktoren ab. Zum einen von der Anzahl der Teilnehmer im Netz und somit von der benötigten Bandbreite und zum zweiten vom finanziellen Budget, das der Chef zur Verkabelung zur Verfügung stellt.

In beider Hinsicht die Höchstnoten verdienen sich **Lichtwellenleiter**, wobei Monomode-Leiter die Daten schneller übertragen und die Kosten rasanter in die Höhe katapultieren als die technisch einfacher herzustellenden - weil dickeren - Multimode-Leiter. Je dünner das Medium, desto geradliniger muss ein Lichtstrahl hindurch. Und die Gerade ist bekanntlich die kürzeste Verbindung zwischen zwei Punkten. Ein nicht unwesentlicher Kostenfaktor sind die notwendigen Umsetzer, die die elektrischen Signale in Lichtimpulse und umgekehrt wandeln. Wohl wegen der Kosten finden sich Lichtwellenleiter vorwiegend in so genannten **Backbones**, also Hochgeschwindigkeits-Verbindungen zwischen räumlich getrennten Teilen eines Netzwerks. Ein anderes Anwendungsfeld erschließt sich durch dessen Unempfindlichkeit gegenüber elektromagnetischer Einflüsse.

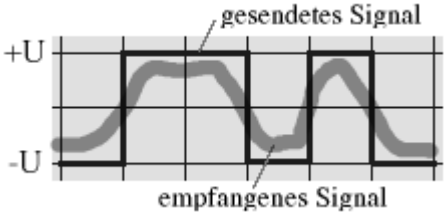


Abbildung 9: Störeinflüsse auf die Signalübertragung

Wegen fehlender Unterstützung von Hochgeschwindigkeitstechnologien nimmt die Verbreitung des Einsatzes von Koaxial-Kabeln weiterhin ab. Die »dickere« Ausführung (Thick Koax) ist gar noch seltener anzutreffen als das »schlankere« Thin Koax. Thick Koax ermöglicht theoretisch den Anschluss von mehr Stationen (ca. 100) pro Kabelsegment als Thin Koax (32) und auch größere Segmentlängen (bis zu 500m gegenüber 185m). Allerdings bedingt Thick Koax auch eine aufwändigere Abschirmung, um das Signal über die weiten Wege stabil zu halten und das spiegelt sich im Preis wider. Auch zeigte die Praxis, dass der Aufbau von Netzwerken, die aus vielen »kleinen« Segmenten bestehen, sowohl den möglichen Durchsatz als auch die Ausfallsicherheit erhöht (vergleiche: Entkopplung durch Bridges). Bei beiden Typen von Koaxialkabeln ist die Übertragungsgeschwindigkeit auf maximal 16 MBit/s begrenzt.

Obwohl die Eigenschaften der Koaxial-Kabel prinzipiell höhere Übertragungsgeschwindigkeiten als Twisted-Pair-Kabel (»verdrillte Vierdrahtleitung«) zulassen, genügt letztere Verkabelungstechnik vollkommen zum Aufbau eines 100 MBit-Ethernets. Und auch hier erwies sich, dass die teureren geschirmten Twisted-Pair-Kabel den günstigeren ungeschirmten gegenüber kaum Vorteile bringen.

Aktive Komponenten

Netzwerkkarte

Die wichtigste Komponente, um einen Rechner in ein Netzwerk zu integrieren, ist die **Netzwerkkarte** (oft als Network Interface Card *NIC* bezeichnet). Für jeden Netzwerktyp existieren eigene Karten, die den Zugang zum Medium ermöglichen; die Karte muss also zum Netzwerk »passen«. Die Aufgaben der Netzwerkkarte bestehen im Senden und Empfangen von Daten.

Repeater

In Netzen größerer Ausdehnung (wobei »größer« von Netzwerktyp, Verkabelung usw. abhängig ist) ist ggf. eine Signalverstärkung notwendig, um die Daten auch über weite Distanzen unverfälscht übertragen zu können. **Repeater** verbinden hierzu zwei Kabelstränge miteinander und leiten die elektrischen Impulse verstärkt vom jeweils einen Kabelsegment in das andere. Um Routingprobleme zu verhindern, sind beim Einsatz von Reaptern einige Regeln zu beachten. So darf es zwischen zwei Stationen nur **exakt einen Weg** durch das Netz geben. Um ein stabiles Signal zu gewährleisten, dürfen zwischen zwei Empfangsstationen maximal 4 Repeater geschaltet sein; auch dürfen maximal 3 der 5 Segmente mit Koax-Kabeln realisiert werden.

Hub

Für Repeater mit mehr als zwei angeschlossenen Segmenten hat sich der Begriff des **Hub** etabliert. Ein Hub leitet das in einem Anschluss ankommende Signal verstärkt auf alle anderen Anschlüsse (oft 5, 8 oder 16) weiter. Ein Port des Hubs ist als so genannter Uplink-Port ausgelegt, der zum Anschluss weiterer Hubs dient. Auf diese Art und Weise lassen sich große Netze aufbauen.

Hubs gibt es in Ausführungen mit Bandbreiten zu 10 MBit, 100 MBit und 10/100 MBit. Nur bei letzterer Variante können in einem Netz Rechner mit 10 MBit-Netzwerkkarten und solche mit 100 MBit-Karten kombiniert eingesetzt werden, wobei allerdings die Bandbreite für alle Stationen auf 10 MBit sinkt.

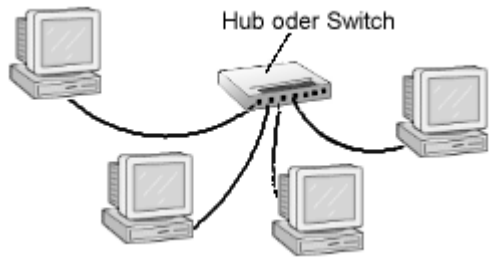


Abbildung 10: Hubs oder Switches verbinden Teilnetze

Bridge

Quasi »intelligente Repeater« stellen **Bridges** (Brücken) dar, indem sie das Signal nicht nur verstärken, sondern für eine Lastentkopplung zwischen den beiden angeschlossenen Segmenten sorgen. Brigdes speichern hierzu die ankommenden Pakete, werten sie aus und leiten sie erst anschließend weiter (»Store&Forward«) und zwar nur, wenn der Empfänger im anderen Segment liegt. Intern halten sich Brücken hierzu Tabellen mit Hardwareadresse und zugehörigem Segment (nicht bei Token Ring); zum Standard gehört es unterdessen, dass die Brücken diese Tabelle dynamisch anpassen um auch auf Änderungen in der Netzkonfiguration reagieren können. Neben selbstlernenden Brücken lassen sich die »besseren« zusätzlich manuell konfigurieren (bspw. als einfacher Adressfilter).

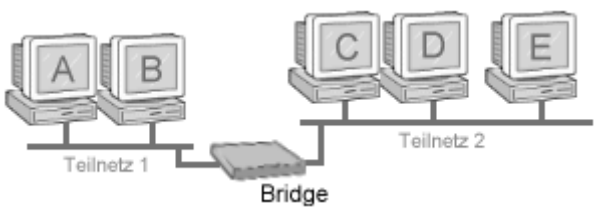


Abbildung 11: Brücke

Beispiel: *Die Bridge aus Abbildung 11 wird alle Pakete aus Teilnetz 1 nach Teilnetz 2 leiten, wenn ihr Ziel der Rechner C, D oder E ist. Ein Paket von A an B bzw. B an A hingegen wird von der Bridge ignoriert. Pakete werden vom Teilnetz 2 in Teilnetz 1 vermittelt, wenn und nur wenn der Empfänger Rechner A oder B ist.*

Durch den Einsatz von Brücken entfällt auch die Einschränkung herkömmlicher Repeater, dass es nur eine Verbindung zwischen zwei Empfängerstationen geben darf. Hierzu fungiert bei redundanten Strukturen eine Brücke als Master. Um welche es sich handelt, klären die Brücken in gegenseitiger Kommunikation. Ausgehend von der Masterbrücke werden Verbindungen gesucht, die zu weiteren Brücken führen. Ausgehend von jenen Brücken erfolgt wiederum die Suche nach weiteren bislang nicht erfassten Brücken. Der Algorithmus endet, wenn alle Brücken gefunden wurden. Per Software werden nachfolgend alle Verbindungen eliminiert, die im Suchergebnis zum wiederholten Male zu einer entfernten Brücke führten, sodass letztlich zwischen zwei Stationen nur genau ein Weg für den Datentransport übrig bleibt. Das Verfahren wird als *Spanning Tree Algorithmus* bezeichnet.

Des Weiteren führen moderne Brücken zusätzlich eine Überprüfung der Pakete auf Korrektheit durch. Da notwendige Zwischenspeicherung und Prüfung Zeit beanspruchen, gehört der Durchsatz »Datenpakete pro Sekunde« zu den Kenndaten einer Bridge. Arbeitet eine Brücke mindestens so schnell wie das angeschlossene Interface, wird sie als **wirespeed** bezeichnet.

Switch

Bei **Switches** handelt es sich um Multiport-Brücken. Äußerlich gleichen Switches somit den Hubs, jedoch bieten sie die Funktionalität einer Bridge. Die herausragende Eigenschaft von Switches ist die Bereitstellung der vollen Bandbreite unabhängig von der Zahl der angeschlossenen Rechner.

Indem der Hub ein einkommendes Signal an einem Port auf alle ausgehenden Ports legt, erscheinen alle angebundenen Stationen in einem einzigen großen Netz. Diese Stationen müssen sich die verfügbare Übertragungskapazität teilen.

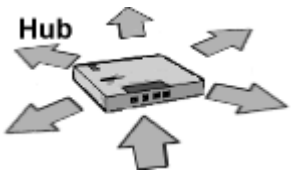


Abbildung 12: Datentransport im Hub

Abbildung 13: Datentransport im Switch (nach Lernphase)

Ein Switch hingegen leitet das Paket nur an den Port weiter, in dessen Segment sich auch der Empfänger befindet. Switches mit entsprechender Leistung können parallel mehrere »virtuelle« Verbindungen zwischen verschiedenen Ports gleichzeitig aufbauen, wobei für jede Verbindung die volle Übertragungsbandbreite zur Verfügung steht.

Wie auch Hubs lassen sich Switches über einen Uplink-Port kaskadieren.

Router

Während alle bisher genannten Komponenten typisch für ein lokales Netzwerk sind, dient ein **Router** vorrangig zur Verbindung mehrerer unabhängiger Netzwerke. Ein Router entscheidet anhand der in einem IP-Paket enthaltenen Empfängeradresse und seiner (dynamisch aktualisierten) Routing-Tabelle, in welches der angeschlossenen Netzwerke die Daten weiterzuleiten sind. Dabei ist ein Router auch in der Lage, Netzwerke unterschiedlicher Topologie miteinander zu koppeln. Während vor einigen Jahren ausschließlich teure Spezialhardware als Router zum Einsatz gelangte, wird heute auch gern ein (Linux)Rechner mit dieser Aufgabe betraut. Bei geringem zu erwartenden Datenaufkommen genügt ein älterer Prozessor (Pentium I, K5) durchaus; für stark frequentierten Knotenpunkte ist selbst ein schneller Bolide deutlich preiswerter als ebenbürtige Spezialhardware.

LAN-Router gleichen ihre internen Tabellen mit denen benachbarter Router über das *Routing Information Protocol* (RIP) oder mittels des neueren *Open Shortest Path First* (OSPF) ab. WAN-Router verständigen sich über *Exterior Gateway Protocol* (EGP) oder das *Border Gateway Protocol* (EGP) . Eine spezielle Ausführung - so genannte B-Router - vereinen die Funktionen von Bridge und Router und arbeiten, je nach eintreffendem Paket, als das eine oder das andere. In Netzwerken größerer Dimension kann sich der Einsatz von Routern anstatt von Brücken

auszahlen, da Router die Pakete über den »best möglichen« Weg weiter leiten. Während Bridges zu große Pakete verwerfen, fragmentieren Router diese bei Bedarf. Auch bieten Router einen einfachen Schutz gegen »Broadcast-Stürme«.

Gateway

Verbindet ein Rechner gar Netzwerke unterschiedlicher Technologie (bspw. IP und IPX), so wird eine Protokollumsetzung (des IP-Protkollstacks) notwendig. Für diese über die Aufgaben eines normalen Routers hinausgehenden Aufgaben gelangen **Gateways** zum Einsatz. Dass Linux auch als Gateway betrieben werden kann, sollte kaum mehr verwundern...

Ethernet

1972 startete *Xerox* in Pala Alto die Experimente mit einem Vorläufer des heutigen Ethernets, dem *Alto Aloah Network*, deren erste Ergebnisse erst 1976 der Öffentlichkeit vorgestellt wurden. Auf Basis dieser Arbeit entwickelte eine Gruppe aus DEC, Intel und Xerox, die sich 1979 zusammengefunden hatte, eine erste Spezifikation *Ethernet V 1.0* (1980). Der Name *Ethernet* - ein Kunstgefüge aus *Äther* und *Netz* - wurde gewählt, um die prinzipielle Unterstützung jedes Computertyps durch die Technologie zu untermauern.

Der IEEE diente diese Spezifikation maßgeblich als Vorlage für einen Standard für LAN's, der 1982 als 802.3-Standard (10Base5) verabschiedet wurde. Die Weiterentwicklung *Ethernet V. 2.0* (1985) wurde schließlich in den ISO-8802.3-Standard überführt.

Weitere Standards legten die Richtlinien für 10Base2 und das kaum verbreitete 10BroadT (1985) fest. 1988 wurde Ethernet in Verbindung mit Twisted-Pair eingeführt, was den noch heute weit verbreiteten Standard 10BaseT (1991) formulierte. Schließlich hielt 1995 mit 100BaseT ein Standard Einzug, der wohl eher unter dem Begriff *Fast Ethernet* geläufig ist.

Während Produkte des Gigabit-Ethernets über Glasfaserkabel (802.3z, 1998) und über Kupferleitungen (802.3ab, 1999) trotz der langen Verfügbarkeit nur selten anzutreffen sind, steht der nächste Standard des 10 Gigabit-Ethernets bereits zur Debatte.

Und was hat es mit den Bezeichnungen 10Base5... auf sich? Die erste Zahl gibt die Bandbreite in MBit/s an, *Base* steht für Basisband- und *Broad* für Breitbandübertragung. "2" bzw. "5" geben in Zusammenhang mit Koaxialkabel die maximale Segmentlänge in 100m-Einheiten an; ein "T" kennzeichnet eine Twisted-Pair-Verkabelung. Seltener wird man ein "F" vorfinden, was auf ein Ethernet auf Basis von Lichtwellenleitern hinweist.

Zugang zum Medium

Jede Station in einem Ethernet arbeitet unabhängig von den anderen; es existiert keine zentrale Instanz, die die Kommunikation auf dem gemeinsamen Medium regelt (*Medium* soll hier allgemein für das Verbindungssystem stehen, bspw. Twisted-Pair-Kabel).

Signale werden bitweise übertragen, wobei jede am Medium angeschlossene Station jedes Signal empfängt. Wünscht eine Station zu senden, so »lauscht« sie zunächst auf dem Medium, ob zur Zeit eine Übertragung im Gange ist. Trifft dies zu, wartet die Station eine zufällige Zeitspanne und lauscht dann erneut. Ist die Leitung frei, schickt die Station das Paket, verpackt in einen definierten Rahmen (*Ethernet Frame*) auf die Reise. Nach Abschluss einer jeden Übertragung müssen sich stets alle sendewilligen Stationen neu um den Zugang zum Medium bewerben. So ist sichergestellt, dass der Zugang zum Medium fair abläuft und keine Station »verhungert«, weil andere permanent Daten durch das Netz schicken.

CSMA/CD

Nun kostet jede Übertragung bekanntlich Zeit. Und weil eine Station soeben ihr Paket versendet, da sie das Medium als frei erkannte, könnte zur gleichen Zeit eine andere Station genau zu demselben Schluss gelangt sein und schickt ihrerseits Daten auf die Strecke. Eine Kollision der Pakete und damit die Unbrauchbarkeit ihrer Inhalte sind die Folge. Um für solche Situationen gewappnet zu sein, definiert das **CSMA/ CD-Protokoll** (*Carrier Sense Multiple Access with Collision Detection*) exakte Regeln für den Medienzugriff.

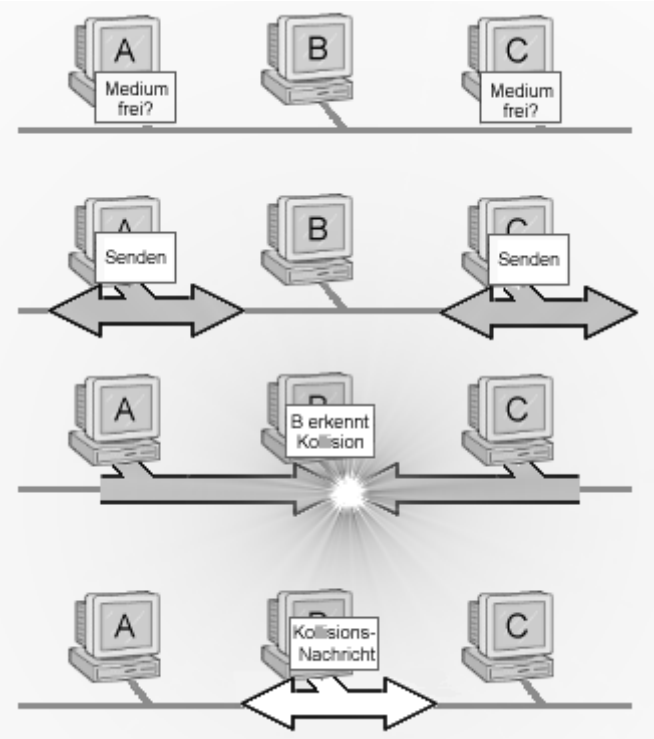


Abbildung 14: Prinzip des CSMA/CD-Verfahrens

Nach CSMA/CD lauschen alle beteiligten Stationen permanent auf dem Medium. Das müssen sie ja auch, um für sie bestimmte Daten erkennen zu können.

Gedenkt eine Station nun zu senden, so wartet sie, bis das Medium frei ist (Carrier Sense). Nach einer Zeitspanne von 9,6 Mikrosekunden beginnt die Station mit der Übertragung. Diese Zeitspanne entspricht genau dem vorgeschriebenen Mindestabstand zwischen zwei Ethernetpaketen.

Mit Beginn des Sendevorgangs hört die Station weiterhin das Medium ab, um eine eventuelle Kollision mitzubekommen (Collision Detection). Trat keine Kollision auf, erkennt die Zielstation anhand der im Paket vermerkten Adresse die für sie bestimmten Daten, empfängt sie und die Übertragung ist somit beendet.

Da aber der Zugang zum Medium im Ethernet für alle Stationen gleichberechtigt ist (Multiple Access), sind Kollisionen nicht ausgeschlossen, weil zwei oder mehrere Stationen nahezu zeitgleich den Sendevorgang anstrengten und das Medium als frei erkannten. In dem Fall wird durch Überlagerung der Signale der Inhalte der Pakete zerstört. Die Station, die die Kollision zuerst registrierte, schickt deshalb ein spezielles »Überlagerungssignal« aus, worauf die sendenden Stationen mit sofortiger Einstellung ihrer Übertragung reagieren.

Die Stationen, deren Senden mit einer Kollision endete, wiederholen den Sendevorgang nach Ablauf einer (relativ) zufälligen Wartezeit. Kommt es wiederholt zu einer Kollision, so wird die »Auszeit« stetig erhöht.

CSMA/CD vermeidet also keine Kollisionen, minimiert jedoch durch die interne Verfahrensweise (flexible Wartezeiten) die Wahrscheinlichkeit ihres Auftretens.

Der Ethernet Frame und Adressen

Daten, die über ein Ethernet übertragen werden, werden nochmals in einen Rahmen verpackt. Die Rahmen können geringfügig zwischen den verschiedenen Ethernet-Standards differieren; über eine Präambel, die Ziel- und Absenderadresse sowie über eine Prüfsumme verfügen sie jedoch alle, sodass die Darstellung aus Abbildung 15 (Ethernet V2.0) durchaus repräsentativen Charakter besitzt.

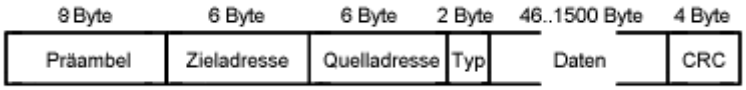


Abbildung 15: Ethernet-Frame

Die **Präambel** enthält die Bitfolge "010101...01". Sie ermöglicht den Stationen, den Beginn eines Pakets zu erkennen. Bei Ethernet nach IEEE 802.3 wird das 8. Byte »Start Frame Delimiter« genannt; wobei dessen Wert sich nicht vom 8.Byte des Ethernet-Frames nach V2.0 unterscheidet.

Die **Adressen** des Empfängers und Absenders umfassen jeweils 48 Bit. Die »oberen« 24 Bit dieser so genannten **MAC-Adresse** werden einem potentiellen Kartenhersteller vom IEEE zugewiesen; die »unteren« 24 Bit vergibt der Hersteller, wobei er jede Adresse nur einmalig an eine Karte vergibt. Die gesamte Hardwareadresse ist fest in die Firmware der Karte »eingebrannt«. Bei manchen Karten kann sie umkonfiguriert werden, wobei beachtet werden muss, dass sie im lokalen Netz stets eindeutig ist.

Der **Typ** kennzeichnet das eingebettete Protokoll; zumeist wird hier 2048 als Kennzeichnung des IP-Protokolls stehen. Nach 802.3-Standard steht hier stattdessen die Länge des Pakets ohne Präambel.

Zwischen 46 und 1500 Bytes Nutzdaten können in den Rahmen eingebettet werden, sodass ein Ethernet-Paket eine Gesamtlänge von maximal 1526 Bytes besitzt.

Eine über das Gesamtpaket gebildete Prüfsumme (CRC) ermöglicht dem Empfänger, Übertragungsfehler zu erkennen.

Wichtige Ethernet-Vertreter

10Base2

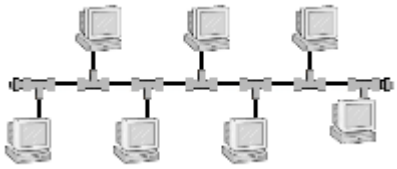


Abbildung 16: Busstruktur des 10Base2

Als Vertreter der klassischen Busstruktur des Ethernets hat sich einzig 10Base2 bis in heutige Zeit gehalten. Das Argument für den Einsatz der Technik - die Ersparnis der Kosten für eine zentrale Kopplerkomponente - hat sich jedoch mit sinkenden Hardwarepreisen verflüchtigt, was kurz über lang wohl auch für 10Base2 selbst zutreffen wird.

Als Verbindungsmedium gelangt ausschließlich Koaxial-Kabel zum Einsatz, an das mittels BNC-T-Steckern die Stationen angeschlossen werden. Die Enden des Kabel müssen mit einem Endwiderstand (»Terminator«) abgeschlossen werden. Maximal 32 Stationen lassen sich an einem Kabelsegment betreiben, wobei der Mindestabstand zweier Anschlüsse 0.5m und der maximale Abstand 185m beträgt.

10BaseT

Abbildung 17: Sternstruktur des 10BaseT

Die physische Struktur heutiger Ethernet-Installationen gleicht zumeist einem Stern. 10BaseT war die erste Realisierung eines Ethernets, die zum Anschluss der Stationen auf eine zentrale Komponente (Hub oder Switch) setzte. Zur Verbindung dient zumeist Twisted-Pair-Kabel, wobei die Auslegung des zentralen Koppelelements die maximale Anzahl zu verbindender Stationen bestimmt. Typische Hubs verfügen über 5, 8 oder 16 Anschlüsse; aber auch Ausführungen mit 150 und mehr Anschlüssen sind erhältlich. Die maximale Kabellänge zwischen Station und Koppelkomponente darf 100m nicht übersteigen.

Zum Aufbau größerer Netzwerke lassen sich mehrere Koppelelemente miteinander verbinden.

Fast- und Gigabit Ethernet

Beide Typen unterscheiden sich vom 10BaseT einzig hinsichtlich des Übertragungsmediums und der Schnittstellenadapter; d.h. auch ihre Grundstruktur formt einen Stern.

Um die Übertragungsraten von 100 MBit zu erzielen, gelangt bei 100BaseT ein aufwändiger verarbeitetes Twisted-Pair-Kabel (»Typ 5«; ein Typ legt die elektrischen Eigenschaften fest) zum Einsatz, als es für 10Base2 (»Typ 3«) erforderlich ist; auch die Netzwerkkarte muss für die hohen Datenraten gewappnet sein. Aktuelle Karten, die 100 MBit unterstützen, beherrschen i.d.R. ebenso den 10 MBit-Datentransfer; so dass sie ggf. die Geschwindigkeit anpassen, falls ihr »Gegenüber« nur die gemächlichere Korrespondenz beherrscht.

Datenraten von 1000 MBit/Sekunde im Gigabit-Ethernet werden entweder über Glasfaserleitungen oder über geschirmte Twisted-Pair-Kabel erzielt. Vor allem wegen der extrem teuren Netzhardware (Hubs, Repeater) sind Gigabit-Lösungen heute fast ausschließlich im Backbone-Bereich zur Kopplung separater Netzwerke zu finden.

Token Ring

Neben Ethernet hat sich lange Zeit einzig IBM's Token Ring als LAN-Technologie durchsetzen können, wenn auch bei weitem nicht mit dem durchschlagenden Erfolg, der dem Ethernet beschieden war. Dabei entstand das Konzept des »kreisenden« Tokens bereits Anfand der 70er Jahre (Willemjin) bei IBM. Vielleicht ließ sich IBM mit der Vorstellung eines ersten Prototypen (1983) einfach zu viel Zeit; als IEEE802.5 endlich als Standard anerkannt wurde (1985), dominierte Ethernet schon längst den Markt.

Die ersten Realisierung mit ihrer »IBM Typ 1«-Verkabelung (Shield Twisted Pair) arbeitete gerade mal mit einer maximalen Datenrate von 4MBit/Sekunde. Einigermaßen Verbreitung hat die Unshield-Twisted-Pair-Verkabelung mit 16MBit/Sekunde Übertragungsrate gefunden. Standards für »Fast Token Ring« (100 MBit/Sekunde) und »Gigabit Token Ring« existieren zwar seit Jahren, haben sich aber nicht durchsetzen können. Die Vermutung liegt nahe, dass Token Ring in naher Zukunft gänzlich an Bedeutung verliert.

In einer Hinsicht ist Token Ring dennoch dem Ethernet überlegen. Es hat weit weniger mit Reichweitenbeschränkungen zu kämpfen; selbst Netzwerke mit mehreren Kilometern Ausdehnung *ohne* zusätzliche Koppelhardware sind möglich.

Zugang zum Medium

Warum wir es dennoch erwähnen, liegt an dem besonderen Verfahren der Regelung des Medienzugangs, das eine »faire« Behandlung aller Stationen bei Vermeidung von Kollisionen garantiert.

Die logische Struktur eines jeden Token-Ring-Netzwerkes ist der Ring; physisch wird sogar ein Doppelring eingesetzt (Abbildung 19). Zwar könnte jeder der Ringe für die Datenübertragung herhalten, jedoch wird in der Praxis der zweite Ring nur als Backuplösung verwendet, der sich selbsttätig zuschaltet, wenn der andere Ring unterbrochen wird.

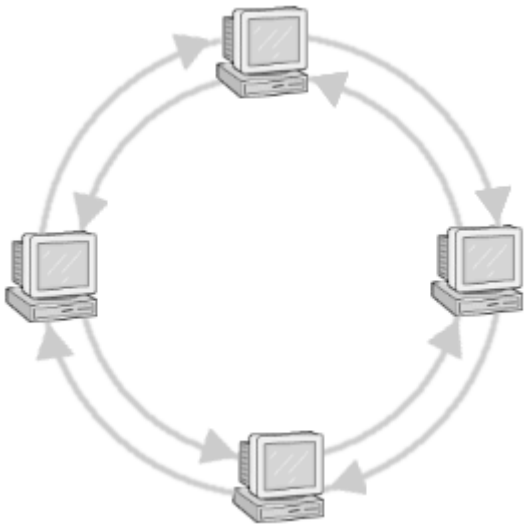


Abbildung 19: Typisch für Token Ring ist der Doppelring

Für die Praxis hat obige Anordnung jedoch entscheidende Nachteile. Da jeder Rechner doppelt mit seinen beiden Nachbarn verkabelt wäre, ließe sich ein solches Netzwerk nur aufwändig erweitern. Auch wären enorm lange Kabel zur Verbindung erforderlich. Deshalb ähneln Token-Ring-Installationen eher einer Sterntopologie. Der Ring ist in einem so genannten Ringverteilungsleiter realisiert, von dem aus sternförmig die Stränge zu den Client verlaufen.

Die Steckverbindungen im Ringverteilungsleiter zum Anschluss der Clients sind so konzipiert, dass sie den Ring selbstständig schließen, sollte der Client abgeschlossen oder die Verbindung zu diesem gestört sein. Damit wird der zweite Strang, der formals als BAckuplösung angedach war frei. Er dient nun der Verbindung mehrerer Ringverteilungsleiter untereinander, um auch größere Netzwerke realisieren zu können.

Im Ring kreist ständig eine spezielle Signalfolge - das **Token** - dessen Aufbau Abbildung 20 zeigt.

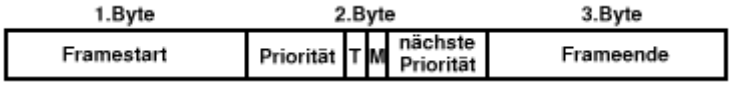


Abbildung 20: Tokenformat

Framestart und **Frameende** sind dabei Bitmuster, die von der eigentlichen elektrischen Bitkodierung des Token Rings abweichen (indem der Pegelsprung bewusst verlegt wird) und somit das Token eindeutig als solches identifizieren. Das Token selbst beinhaltet einen 3-stelligen **Priorität**swert. Ein sendewillige Station darf das Token nur dann vom Ring nehmen und durch ein Datenpaket ersetzen, wenn ihre Daten eine gleichwertige oder höhere Priorität besitzen. Die **nächste Priorität** kann von einer Station gesetzt werden, um die Priorität der übernächsten Runde zu bestimmen. Die übernächste deshalb, weil nach einer Datenübertragung stets eine Runde mit Priorität 0 folgt. Somit wird sichergestellt, dass auch Stationen mit »weniger wichtigen Daten« zum Zuge kommen.

Das mit **T** gekennzeichnete Bit gibt an, ob es sich um ein Token oder um ein Datenpaket handelt (bei letzterem unterscheiden sich die folgenden Bytes). **M** ist das Monitorbit (siehe Abschnitt »Was ist bei Verlust des Tokens?«).

In den beiden letzten Bits des **Frameende**-Bytes sind der Erkennung gesplitteter Datenpakete (Bit ist gesetzt, wenn das Gesamtpaket aus mehreren Teilen besteht; bei leerem Token ist das Bit stets 0) und die Fehleranzeige kodiert.

Die drei beschriebenen Bytes tauchen mit identischer Bedeutung auch bei einem Datenpaket auf, dessen Aufbau Abbilungs 21 zeigt.

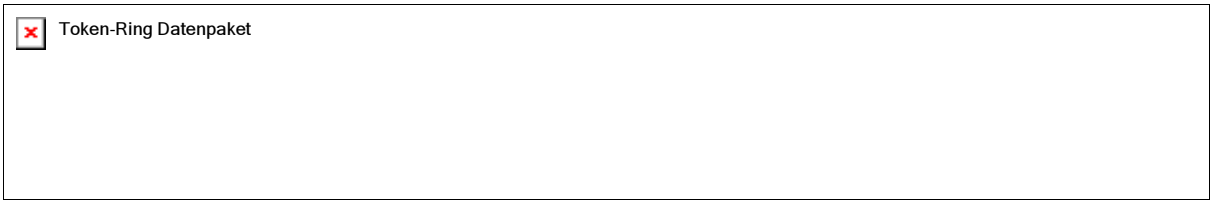


Abbildung 21: Token-Ring Datenpaket

Quell- und **Zieladresse** sind letzlich Hardwareadressen, die i.d.R. einer Karte fest zugeordnet und, da sie vom Hersteller vergeben wurden, eindeutig sind. Der Adresstyp (Geräte-, Broadcast-, ...Adresse) ist in den ersten Bits einer jeden Adresse kodiert.

Die beiden ersten Bits der **Framekontrolle** kennzeichnen den Typ des Pakets (Token-Ring-Steuerpaket oder Datenpaket). Zwei weitere Bits werden nicht verwendet, während die letzten Bits Informationen zur Pufferung der Daten enthalten.

Über das gesamte Paket mit Ausnahme des **Frame Status** wird eine **Prüfsumme** gebildet und im entsprechenden Feld abgelegt.

Im Byte **Framestatus** hinterlegt der Empfänger eine Quittung. Erkennt er, das ein Paket für ihn bestimmt ist, setzte der Empfänger die Bits 0 und 4. War der Empfang korrekt, setzt er ebenso die Bits 1 und 5. Die Doppelung soll die Fehleranfälligkeit verringern, da Framestatus nicht durch die Prüfsumme erfasst ist.

Was ist bei Verlust des Tokens?

Der Verlust des Tokens wäre wohl der schlimmeste offensichtliche Fehlerfall. Aber er ist bei weitem nicht der Einzige. Die Erkennung und BehebungFolgender Fehler muss im Ring gewährleistet sein:

- Verlust des Tokens
- Endlos kreisendes Datenpaket
- Endlos kreisende Pakete mit hoher Priorität

Um solcher Fehler zu erkennen, wird eine Station zum so genannten **Monitor** ernannt. Diese Station ist fortan für das (Rück)Setzen des Monitorbits im Frame verantwortlich.

Ein Auswahlverfahren im Token Ring stellt sicher, dass zu einem Zeitpunkt nur eine Station die Monitorfunktion ausführt.

Eine sendewillige Station, die das Token erhält, initialisiert das Monitorbit mit 0. Die Station, die die Monitorfunktion ausübt, setzt beim Durchlauf des Paketes dessen Monitorbit auf 1. Kommt das Paket erneut beim Sender an, muss dieser dieses vom Ring nehmen (selbst wenn der Empfäger es nicht angenommen hat). Ist der Sender aber ausgefallen, erreicht das Paket die Monitorstation erneut. Sie erkennt anhand des gesetzten Monitorbits den wiederholten Durchlauf, beginnt mit der Neuinitialisierung des Token Rings und erzeugt anschließend ein neues Token. Um den Verlust des Tokens selbst zu erkennen, startet die Monitorstation bei jedem Durchlauf einen Timer. Läuft dieser ab, gilt das Token als verloren und die erneute Initialisierung startet.

Asynchron Transfer Mode

Andere Technologien

Wireless Lan

Firewire

Bluetooth

Bei Bluetooth handelt es sich um eine Funktechnologie, um verschiedenste Geräte über Distanzen bis zu maximal 100m zu vernetzen. Bluetooth verwendet hierfür ein frei zugängliches Frequenzband (2.4GHz).

Während der verschlüsselten Übertragung erfolgt 1600 mal pro Sekunde eine Frequenzsprung. Das garantiert zum Einen eine relativ verlässliche Abhörsicherheit und zum Anderen eine Konfliktvermeidung mit anderen Bluetoothgeräten der näheren Umgebung, da die Wahrscheinlichkeit, dass diese dieselbe Frequenz benutzen, gegen Null geht.

Übertragungsraten bis reichlich 700 kByte/s können mit der Technologie erreicht werden.

Sonstiges

IP-Adressen

Um am Datenaustausch in TCP/IP-basierten Netzwerken teilzunehmen, benötigt der Rechner eine in seinem Netzwerk eindeutige IP-Adresse. Für jeden Teilnehmer im Internet folgt damit, dass seine Adresse weltweit eindeutig sein muss. Dass dies bei dem knappen 32-Bit Adressraum schon heute nicht mehr praktikabel ist, führte zu Techniken der Maskierung, wobei ein Rechner, der über eine offizielle IP-Adresse verfügt, stellvertretend für andere Rechner aus seinem Verwaltungsbereich im Internet auftritt. Aber derartige Belange sollen uns erst an anderer Stelle interessieren.

IP-Adressen nach IPv4

Mit 32 Bit Adressen ließen sich theoretisch 2^{32} (4294967296) Rechner ansprechen. Praktisch ist es allerdings kaum zu realisieren, dass ein Rechner wahllos aus dem Pool der Adressen eine freie zugewiesen bekommt (zur Paketvermittlung müssten dann zumindest einige Rechner im Netz mit definiertem Standort die Adressen aller Teilnehmer verwalten). Ähnlich zu einer Telefonnummer, bei der das Ziel durch Länderkode, Ortswahl und Teilnehmernummer lokalisiert wird, werden IP-Adressen in Netzwerk- und Rechnernummer unterteilt. Die (klassische) Interpretation der 32 Bit ist abhängig vom ersten auf 0 gesetzten Bit (von links):

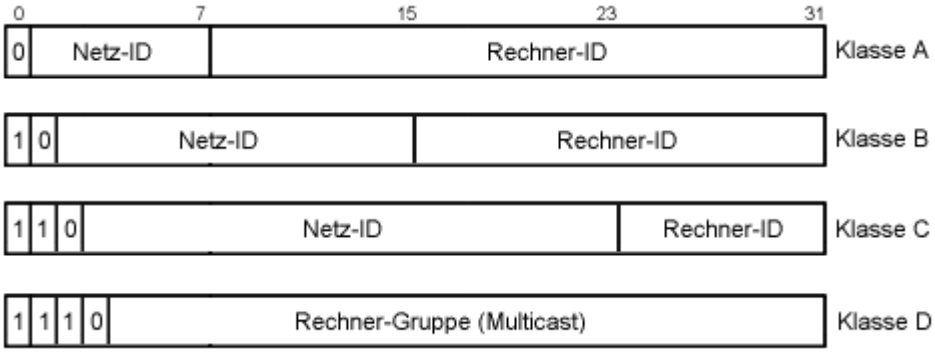


Abbildung 19: Adressklassen nach IPv4

Klasse A

Eine Adresse der Klasse A liegt vor, wenn das erste Bit (Bit 0) der IP-Adresse auf 0 gesetzt ist. Bit 1 bis 7 werden als Netzwerkadresse behandelt, womit 2^7 (128) Netzwerke mit je 2^{24} (16777216) Hosts adressierbar sind.

Klasse B

Eine IP-Adresse der Klasse B ist durch die ersten beiden Bits 10 gekennzeichnet. Die folgenden 14 Bits bilde die Netzwerkadresse (16348). Je Netzwerk stehen 2^{16} (65536) Adressen für die Hosts zur Verfügung.

Klasse C

Startet die IP-Adresse mit den Bits 110, wird sie als Klasse C interpretiert, d.h. 21 Bits (2^{21}=2097152) bilden Netzwerk- und 8 Bit (256) die Hostadressen.

Klasse D

Adressen der Klasse D beginnen mit der Bitfolge 1110 und sind als **Multicastadressen** bekannt. Die gesam 28 »Restbits« bilden eine Multicastadresse; hinter der sich letztlich eine Gruppe von Rechnern verbirgt, wob die zugehörigen Rechner nicht zu einem einzigen lokalen Netzwerk gehören. Zur Vermittlung solcher Adress sind spezielle Multicast-Router erforderlich. Wichtigste Anwendung sind Videokonferenzen.

Klasse E

Adressen, die mit vier 1-Bits starten, ist keine konkrete Bedeutung zugeordnet. Sie dienen in erster Linie zu Forschungszwecken, weshalb auch von einer »experimental class« gesprochen wird.

Die 32 Bit einer IP-Adresse werden als Quadrupel zu je 8 Bits angegeben, wobei die dezimale Schreibweise bevorzugt angewendet wird. Bezugnehmend auf die ersten 8 Bit bedeutet dies für eine Adresse:

- Beginnt sie mit 1-128, so handelt es sich um eine Klasse-A-Adresse
- Beginnt sie mit 129-191, so handelt es sich um eine Klasse-B-Adresse
- Beginnt sie mit 192-223, so handelt es sich um eine Klasse-C-Adresse
- Beginnt sie mit 224-239, so handelt es sich um eine Multicast-Adresse

Die Klassenstruktur der IP-Adressen ist historisch bedingt, schien doch zu jener Zeit der Adressraum von 32 Bit als ungemein groß. Um die Suche nach dem Empfänger eines Pakets in vertretbarem (Zeit)Rahmen zu realisieren, bewertete ein Router eine Adresse nur anhand des Netzwerkteils; anstatt sich bspw. die 65535 Adressen eines beliebigen Klasse-B-Netzwerks zu merken, genügt dem Router somit ein einziger Eintrag für das Netz selbst, um den folgenden Empfänger eines Datenpakets auf dem Weg zu dessen Ziel zu identifizieren. Bei dem damaligen Stand der Rechentechnik war dies eine signifikante Erleichterung.

Aus heutiger Sicht erweisen sich die 32 Bit der IP-Adresse nicht nur als viel zu klein (und Abhilfe ist mit der 128 Bit-Adresse aus IPv6 bereits geschaffen worden), sondern das 3-Klassen-Schema zur allgemeinen Adressierung als zu starr. Die wenigen Klasse-A-Netzwerke beinhalten eigentlich zu viele Hostadressen, um sie direkt auf eine Netzwerkstruktur abzubilden; dementgegen sind die 256 Rechner aus den unzähligen Klasse-C-Netzen schon für die kleineren Organisationen oft zu wenig, sodass vor allem Klasse-B-Adressen lange Zeit heiß umworben waren. Doch auch ihre Zahl war schon bald erschöpft, womit neue Möglichkeiten der Adressinterpretation gefunden werden mussten.

Die Vergabe von Netzwerkadressen erfolgt heutzutage in zusammenhängenden Adressblöcken an Internet Service Provider (ISP). Die ISP reichen aus diesem Fundus Adressen an ihre Kunden weiter. Ein Router im Internet speichert nun für einen solchen Adressbereich nur noch eine einzige Adresse; an die alle Pakete mit Adressen aus dem Bereich geleitet werden. Die Router im Verantwortungsbereich des ISP nehmen dann die weitere Vermittlung vor. Technisch wird eine solche Paketvermittlung mittels einer Bitmaske realisiert, anhand derer die 32 Bit in Netzwerk- und Hostteil unterteilt werden. Das Prinzip gleicht somit dem nachfolgend beschriebenen Verfahren der Subnetz-Bildung.

Spezielle Adressen

Die obigen Angaben zu adressierbaren Netzwerken und Rechnern sind etwas zu optimistisch, denn so mancher Adresse eines bestimmten Schemas kommt eine besondere Bedeutung zu.

Zunächst einmal stehen die Adressen mit erstem Byte größer als 223 (Multicast und Experimental) nicht für die eigentliche IP-Adressierung zur Verfügung.

Aus jedem der Klassen A, B und C ist ein Adressbereich als so genannter **privater Bereich** ausgewiesen. Solche Adressen werden vom keinem Router im Internet vermittelt, womit sie sich zum Aufbau lokaler Netzwerke eignen (theoretisch kann jede Adresse hierfür heran gezogen werden, jedoch nur, wenn das lokale Netz über keine direkte Verbindung zum Internet verfügt). Die Bereiche sind 10.0.0.0 - 10.255.255.255 (Klasse A), 172.16.0.0 -

172.31.255.255 (Klasse B) und 192.168.0.0 - 192.168.255.255 (Klasse C).

Die Adresse 0.0.0.0 wird als **Standardroute** bezeichnet und wird für die Weiterleitung von Adressen mit unbekannter Route verwendet.

Das **Loopback**-Netzwerk umfasst die Adressen 127.0.0.0-127.255.255.255. Über derartige Adressen ist es Netzwerkprogrammen möglich, den lokalen Rechner auf gleiche Art und Weise wie entfernte Rechner zu adressieren.

Jede IP-Adresse, deren Hostteil nur aus Nullen besteht, bezeichnet das Netzwerk selbst (**Netzwerkadresse**). Rechner, die ihre IP-Adresse zum Bootzeitpunkt nicht kennen (können), verwenden bspw. diese Adresse, um via DHCP oder BOOTP diese von einem Server zu erfahren.

Ein komplett aus Einsen bestehender Rechnerteil adressiert alle im Netzwerk befindlichen Rechner auf einmal und wird als **Broadcastadresse** bezeichnet.

Subnetze

Wozu?

Die Administration von Netzwerken mit nahezu 64000 (Klasse B) oder gar 16 Millionen von Rechnern (Klasse A) erweist sich als nicht praktikabel. Zum einen kann keine heutige Technologie einen befriedigenden Durchsatz bei einer solchen Menge von Teilnehmern in einem einzigen Netz garantieren. Zum anderen wäre wohl jeder Administrator mit einem solch gigantischen Verantwortungsbereich überfordert. Aus diesem Grund wird die Struktur der IP-Adresse - genau genommen, deren Hostteil - zumindest in Klasse-A- und Klasse-B-Netzwerken - in selteneren Fällen auch bei Klasse-C-Netzwerken - lokal »geeignet interpretiert«.

Subnetzwerke ermöglichen die Strukturierung großer Netzwerke nach topologischen oder auch organisatorischen Gesichtspunkten. So könnten die Rechner eines Gebäudes in einem Teilnetzwerk organisiert und gleichzeitig die Verantwortung für einen solchen abgegrenzten Bereich an eine andere Zuständigkeit delegiert sein. Manchmal erfordert bereits die räumliche Ausdehnung der zu vernetzenden Teilnehmer die Unterteilung in Teilnetze, da alle Netzwerktechnologien eine maximal zulässige Entfernung zwischen den Stationen vorschreiben. Ein Nebeneffekt der Strukturierung, bedingt duch »intelligente« Koppelelemente, ist eine Datenfilterung zwischen den Subnetzen, sodass die Last im gesamten Netzwerk merklich sinkt. Subnetzbildung ist aber auch die einzige Möglichkeit, verschiedene Netzwerktechnologien (bspw. Ethernet und Token Ring) in einem einzigen IP-Netzwerk zu vereinen.

Wie?

Die Bits des Hostteils einer IP-Adresse werden bei der Subnetzbildung nochmals in zwei Teile untergliedert. Die führenden Bits repräsentieren lokal die Adresse des **Subnetzwerks**; die verbleibenden Bits adressieren einen Rechner eindeutig innerhalb eines solchen Teilnetzes.

Realisiert wird die Abbildung durch Bitmasken. Bei einem gesetzten Bit dieser Maske wird das korrespondierende Bit der IP-Adresse als Netzwerkbit interpretiert; bei einem nicht gesetzten Bit gehört das entsprechende Bit der IP-Adresse zur Hostadresse. Da letztlich einzig die Anzahl gesetzter Bits und nicht deren relative Lage innerhalb der Maske das Verhältnis von Anzahl Subnetzen zu Anzahl Hosts bestimmt, werden die gesetzten Bits stets »linksbündig« angeordnet.

Beispiel: Die Netzwerkmaske zu einem Klasse-B-Netz ist 255.255.0.0 (in Bitdarstellung entspräche dies 16 gesetzten Bits, gefolgt von 16 nicht gesetzten Bits). Zu einem Klasse-B-Netz gehören maximal 65536 Hosts (wir lassen die speziellen IP-Adressen einmal außen vor). Der Administrator entscheidet nun, diese Rechner lokal in 64 Subnetzen zu je 1024 Hosts zu organisieren. Dazu muss er den Hostteil der IP-Adresse so splitten, dass die höherwertigen 6 Bits ($2^6=64$) das Subnetzwerk und die »restlichen« 10 Bits die Rechneradresse repräsentieren. In der Subnetzmaske müssen daher die ersten 22 Bits auf "1" stehen, was in dezimaler Darstellung 255.255.252.0 entspricht.

Zur einfachen Lesbarkeit hat sich eine abweichende Schreibweise für IP-Adressen in Subnetzwerken etabliert. Anstatt die Subnetzmaske getrennt anzugeben, werden die relevanten (also die gesetzten) Bits der IP-Adresse

nachgestellt. Liegt bspw. 162.168.100.12 in einem Subnetz mit der Maske 255.255.240.0, so ist die Angabe von 162.168.100.12/20 gebräuchlich.

Eine Subnetzmaske wirkt nur im lokalen Netz, d.h. »außerhalb« unseres Netzes wird die IP-Adresse »ganz normal« interpretiert und anhand der durch die Adressklasse vorgegebenen Netzwerkadresse vermittelt. Erst die Rechner im lokalen Netz betrachten die Subnetzmaske und verteilen Pakete anhand der berechneten Subnetzadresse.

Beispiel

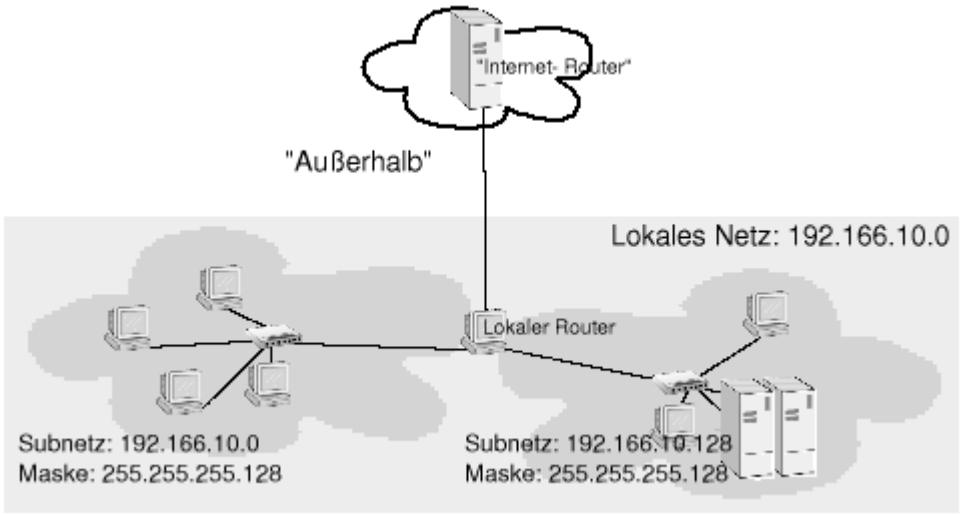


Abbildung 20: Paketvermittlung im Subnetz

Unser lokales Netz mit der Netzwerkadresse 192.166.10.0 (Klasse-C) soll in zwei Teilnetze gesplittet werden. Hierfür genügt ein Bit des Hostteils, woraus sich als Subnetzmaske 255.255.255.128 ergibt. Durch diese Maske gehören alle Rechner mit IP-Adressen 192.166.10.1/25 bis 102.166.10.127/25 zu dem einen und alle Rechner mit IP-Adressen 192.166.10.129/25 bis 102.166.10.254/25 zum anderen Subnetz. Beachten Sie, dass es sich bei 192.166.10.0 und 192.166.10.128 um die Subnetzwerkadressen selbst und bei 192.166.10.128 bzw. 192.166.10.255 um die Broadcastadressen handelt.

Jeder Rechner im gesamten lokalen Netzwerk kennt nun sowohl seine eigene Adresse als auch die Subnetzmaske. Um herauszufinden, ob der Empfänger eines Datenpakets sich im selben Subnetz befindet, muss der Rechner sowohl seine eigene IP als auch die des Adressaten bitweise UND verknüpfen. Sind beide Ergebnisse gleich, befindet sich der Empfänger im selben Teilnetz und das Paket kann direkt an diesen gesendet werden. Andererseits muss das Paket in das andere Subnetz geleitet werden.

Das Koppelglied (Router oder Gateway) zwischen den einzelnen Subnetzen ist Mitglied beider Netze; es besitzt also mehrere Netzwerkschnittstellen mit unterschiedlichen IPs. In der Beispielabbildung stellt ein Router gleichzeitig die Verbindung zur »Außenwelt« her; die auf dieser Schnittstelle verwendete IP-Adresse kann allerdings im Subnetzwerk nicht nochmals vergeben werden.

»Subnetting« bedingt also gleichzeitig ein »Routing«. Um die Problematik der Paketvermittlung soll es im nächsten Abschnitt gehen.

Routing

Allgemein bezeichnet **Routing** das Verfahren der Ermittlung des Weges, den ein Paket vom Quell- zum Zielrechner zurückzulegen hat. Routing-Entscheidungen werden einzig anhand von Netzwerkadressen getroffen, indem die Netzwerkmaske auf die eigene IP-Adresse angewendet wird. Im lokalen Netzwerk sind damit die Regeln der Routenfindung für die einzelnen Rechner sehr einfach:

- Ist das Zielnetzwerk gleich dem eigenen Netzwerk, so werden die Daten direkt an den Empfänger gesendet.
- Ist das Zielnetzwerk nicht das eigene Netzwerk, so werden die Daten an ein Gateway des lokalen Netzwerks

gesendet. Es ist nun Aufgabe dieses Gateway-Rechners, das Datenpaket weiter zu vermitteln.

Für die meisten Netzwerkadministratoren hört hier die Arbeit eigentlich schon auf, da das Gateway meist nur eine Route zu einem Gateway eines ISP's (Internet Services Provider) besitzt und die komplexeren Mechanismen der Routenfindung dem Einflussbereich des ISP obliegen.

Große Netzwerke mit mehreren Subnetzwerken - und somit auch mehreren integrierten Gatewayrechnern - können jedoch auch im lokalen Netz den Aufwand des Einrichtens eines Routing-Protokolls rechtfertigen. Fast schon notwendig werden dynamische Routingverfahren, wenn mehrere Gateways des lokalen Netzes eine Anbindung an die »Außenwelt« realisieren und somit unterschiedliche Wege zu einunddemselben Ziel existieren. Auch hier könnte eine statische Route Pakete stets über einundasselbe Gateway leiten, jedoch würde ein solches Verfahren nicht auf Änderungen im Netz (bspw. Ausfall eines Gateways) reagieren können und dem Sinn mehrerer Routen zuwider laufen.

Die Routingtabelle des Kernels

Bevor wir die verschiedenen Möglichkeiten zum Setzen von Routen kennenlernen, betrachten wir den Aufbau der internen Routingtabellen des Linux-Kernels. Ein geeignetes Kommando ist netstat:

```
user@sonne> netstat -rn
Kernel IP Routentabelle
Ziel          Router          Genmask         Flags   MSS Fenster irtt Iface
192.168.100.0   0.0.0.0         255.255.255.0   U         0 0          0 eth0
0.0.0.0         192.168.100.2   0.0.0.0         UG        0 0          0 eth0
```

Unsere Beispieltabelle umfasst zwar nur Einträge, aber sie genügt zur Erläuterung vollauf.

Anhand der Zieladresse wird entschieden, ob ein Eintrag für das aktuelle Routing in Frage kommt. Die Suche in der Tabelle endet nach dem ersten passenden Eintrag!

Bei den **Ziel**adressen kann es sich sowohl um Adressen einzelner Rechner (bspw. ein Rechner am Ende einer Punkt-zu-Punkt-Verbindung) als auch um Netzwerkadressen handeln. Ein mit einer Netzwerkadresse beginnender Eintrag der Routingtabelle passt somit auf alle Adressen aus diesem Netzwerk. Schließlich gilt eine so genannten **Defaultadresse** (»0.0.0.0«) für jede Adresse.

Router (in manchen Versionen von *netstat* steht hierfür auch *Gateway*) ist die Adresse, an die das Paket als nächstes zu senden ist. »0.0.0.0« meint hier, dass die Zieladresse direkt angesprochen wird. Im Beispiel werden somit alle Pakete an Rechner aus dem Netz »168.129.100.0« direkt über die Schnittstelle *eth0* an die Rechner gesendet, während Pakete mit anderen Zielen zum Rechner »192.168.100.2« weiter geleitet werden (in der Hoffnung, dass dieser sich darum kümmert).

Statisches Routing

Der Begriff *statisch* deutet die Natur dieser Art der Routingkonfiguration bereits an: sie wird vom Systemadministrator vorgegeben und passt sich Änderungen im Netzwerk auch nicht an.

route

Routing-Protokolle im Intranet

Routing Information Protokoll (RIP)

Open Shortest Path First (OSPF)

Routing-Protokolle im Internet

Exterior Gateway Protokoll (EGP)

Border Gateway Protokoll (BGP)

Allgemeine Dienste

Übersicht

Der physische Anschluss der Hardwarekomponenten wurde im Abschnitt Systemadministration, Hardware-Installation erörtet. Das entsprechende Gerät sollte also bereits verfügbar, d.h.der entsprechende Treiber geladen worden sein.

Natürlich ist es damit allein nicht getan; die einzelnen Komponenten bedürfen einer Konfiguration, um im Netzwerk ihre Dienste verrichten zu können. So muss einer Netzwerkkarte eine IP-Adresse zugewiesen, das Modem muss zum Wählen einer Telefonnummer animiert, oder die ISDN-Karte zur Simulation eines Anrufbeantworters eingerichtet werden... Um solche und andere Konfigurationsschritte mehr soll es uns im folgenden Abschnitt gehen.

Netzwerkkarte

Modem - Initialisierung

Wie ein Modem physisch an den Rechner angeschlossen wird, wurde im Kapitel Systemadministration Hardware-Installation besprochen. Der Schwerpunkt im folgenden Text soll nun auf die Einrichtung eines Zugangs über einen Internet Service Provider (ISP) liegen, also die Schritte der Konfiguration, um mit einem simplen Befehl oder Mausklick den automatischen Zugang zum Internet aufzubauen.

Debian: pppconfig

PPP? ProviderName, DNS, Protokoll, dann USername bei ISP, Passwort, Baudrate , Pulse oder Tonwahl, Nummer des ISP, Automatischer Test, wo Modem hängt. Nochmals Zusammenfassung und Möglichkeit der Änderung der einzelnen Einträge

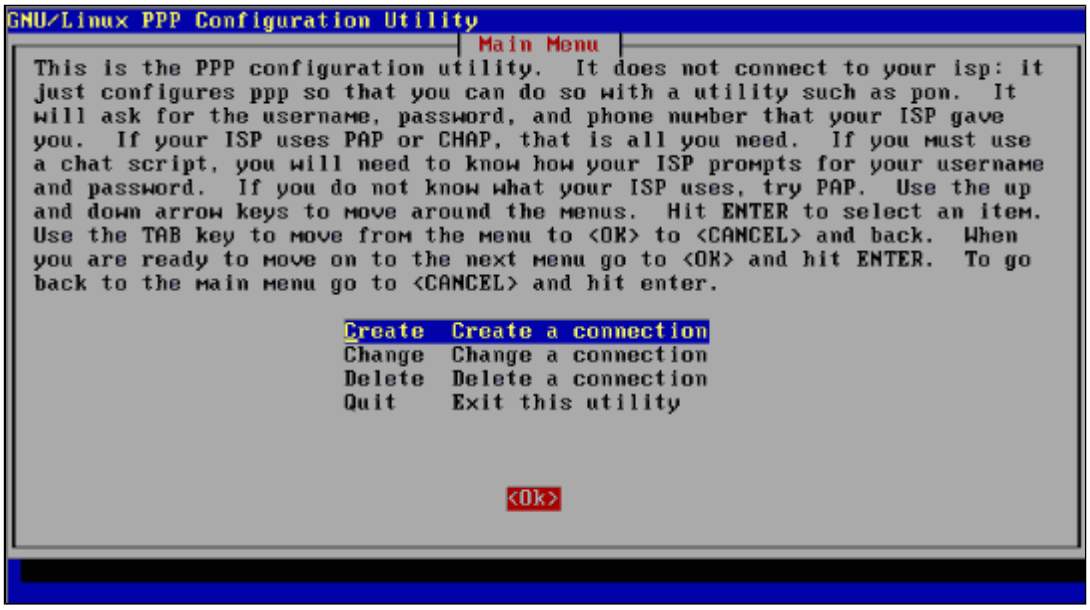


Abbildung 1:

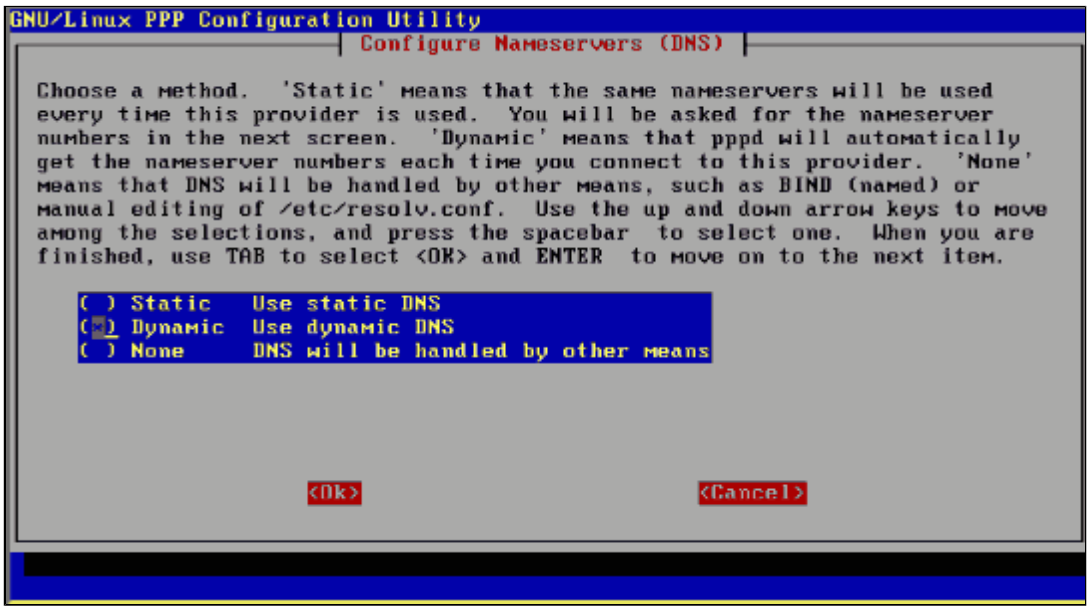


Abbildung 2:

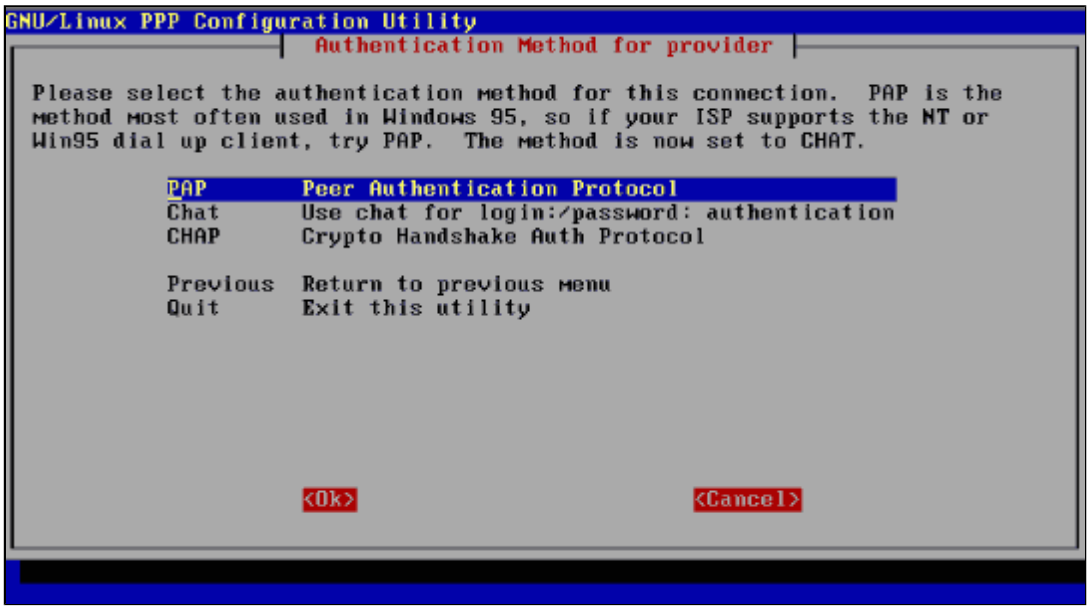


Abbildung 3:

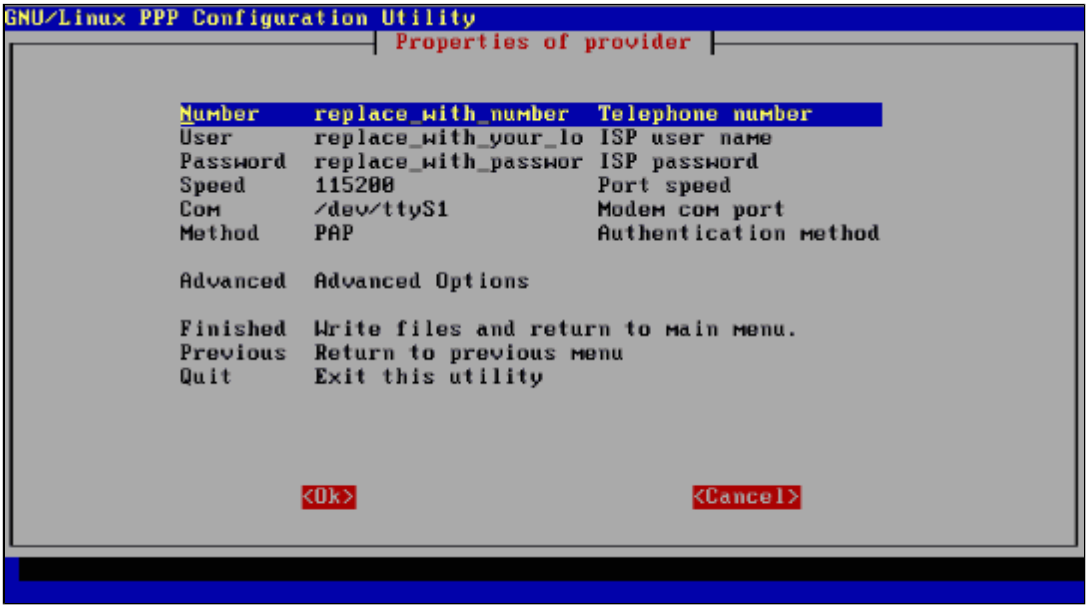


Abbildung 4:

SuSE - Yast

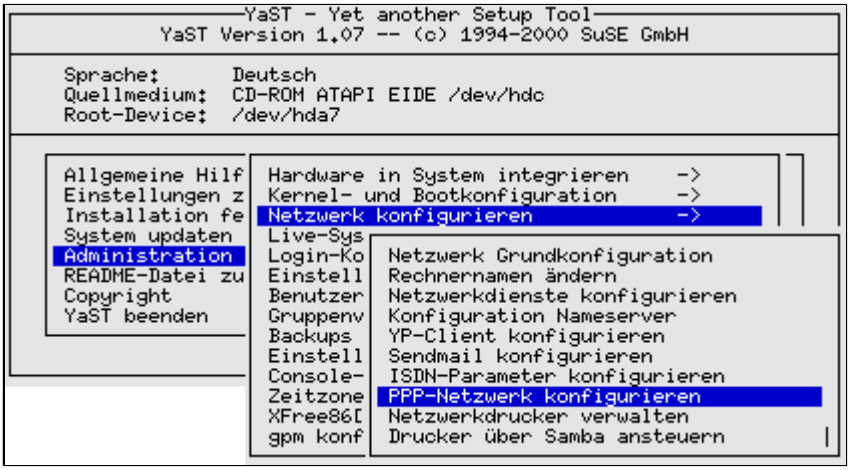


Abbildung 5:

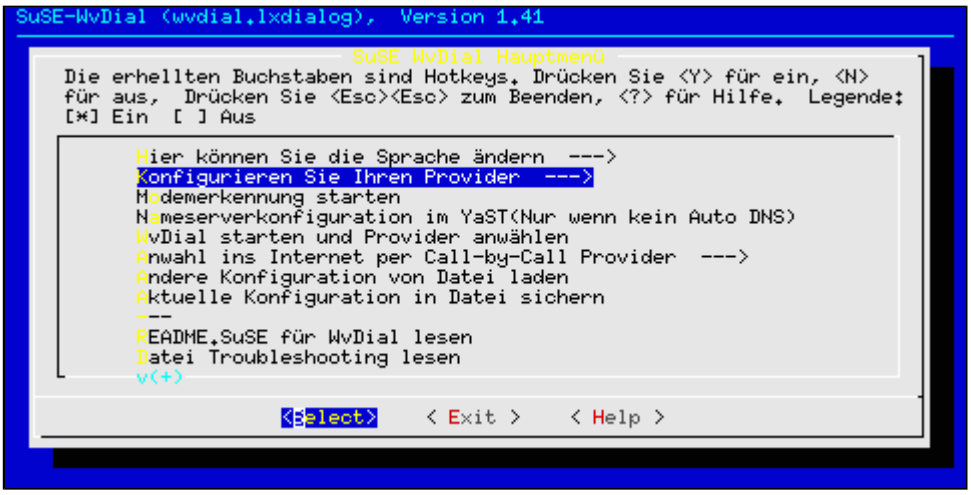


Abbildung 6:

Abbildung 7:

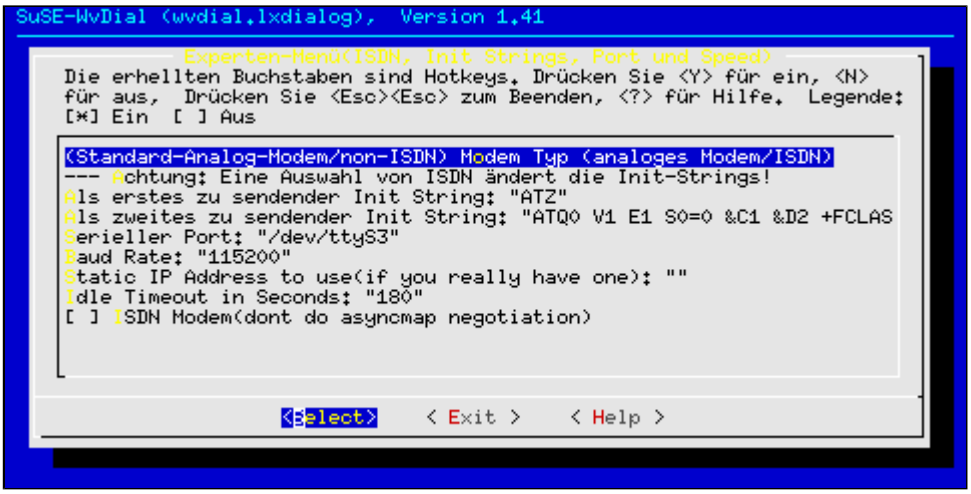


Abbildung 8:

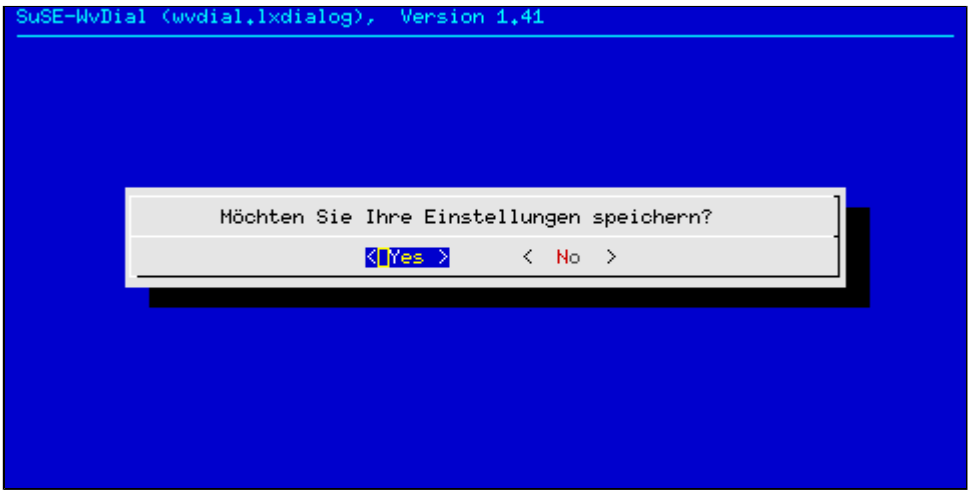


Abbildung 9:

Modem - Einwahlserver

»Einwahl« meint, dass der sich verbindende Rechner zu einem vollwertigen Mitglied des Netzes des Einwahlservers wird. Sie kennen das Verfahren von ihrem Internet-Service-Provider, zudem Sie sich via Modem verbinden und ihr Rechner in dem Augenblick zu einem Knoten des Internets wird. Wir spielen jetzt quasi ISP, indem wir unseren Rechner so konfigurieren, das dieser bei einem eingehenden Verbindungswunsch via Modem die Sitzung

akzeptiert. Vorausgesetzt, der sich einwählende Benutzer verfügt über die entsprechenden Rechte.

»Mgetty« hat Telefondienst...

Wie immer, wenn es um den Zugang zu einem Rechner geht, bedarf es eines Gettys, das die jeweiligen Gerätedateien auf eingehende Verbindungen überwacht und Schritte zur Authentifzierung einleitet. Auch wenn sich bspw. mit agetty weitere Kandidaten um den Posten bemühen, hat sich in Verbindung mit analogen Modems doch das für Modemverbindungen optimierte »mgetty« aus dem Paket »mgetty&sendfax« durchgesetzt.

»mgetty« beim Booten starten

Hierfür genügt ein simpler Eintrag in der Datei /etc/inittab. Angenommen, das analoge Modem hängt an der dritten seriellen Schnittstelle (/dev/ttyS2), so lautet ein passender Eintrag wir folgt:

```
root@sonne> vi / etc/ inittab
...
S2:35:respawn:/usr/sbin/mgetty -s 115200 /dev/ttyS2
...
```

Mit obiger Zeile würde das Modem allerdings sofort das Gespräch an sich reißen. Fügen Sie obige Zeile die Option **-n *Anzahl_Klingelzeichen*** hinzu und das Modem wird erst nach vereinbarter Anzahl abheben.

Wer darf sich einwählen?

Zeitliche Beschränkungen

Modem - Anrufbeantworter

ISDN - Initialisierung

ISDN - Einwahlserver

ISDN - Anrufbeantworter

Mit den Programmen aus dem Paket **i4l** lässt sich unter Linux ein Anrufbeantworter einrichten. Einzige Voraussetzung ist eine funktionierende Anbindung an das Telefonnetz über ISDN.

Eine solche »Voice box« besitzt gegenüber herkömmlichen Anrufbeantwortern einige Vorteile. So lässt sich für jeden Benutzer ein eigener Anrufbeantworter konfigurieren. Notwendig sind hierfür allerdings ausreichend Rufnummern, die entweder eine TK-Anlage oder Ihre Telefongesellschaft bereitstellt. Wahlweise kann ein Benutzer per Mail über eintreffende Anrufe unterrichtet oder aber der Anruf als Sounddatei direkt versendet werden. Auch eine Webabfrage des Status des Anrufbeantworters ist über Zusatzprogramme möglich.

Die Konfiguration von **vbox** gliedert sich in zwei Abschnitte. Zum einen betrifft dies die Einrichtung des Servers, der die eingehenden Anrufe entgegennimmt, einen Ansagetext abspielt und den Anruf aufzeichnet. Dieser Teil der Konfiguration wie auch das obligatorische Einrichten des Netzwerkzugriffs auf gespeicherte Nachrichten obliegt dem Systemverwalter.

Jeder Benutzer hingegen ist selbst gefordert, die Abfrage seiner gespeicherten Anrufe zu konfigurieren.

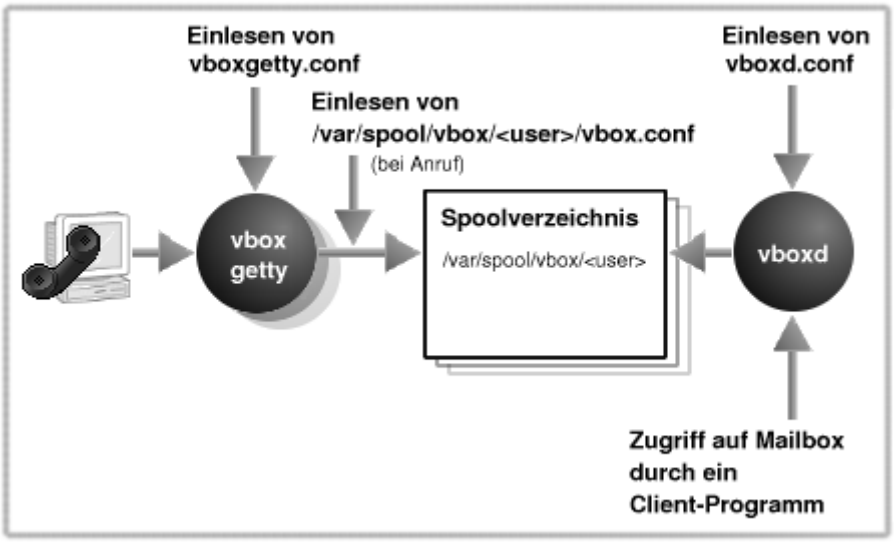


Abbildung 1: Bestandteile des ISDN-Anrufbeantworters

Der Serverdienst vboxgetty

In die Zuständigkeit des Programms **vboxgetty** fällt die Entgegennahme von Anrufen, das Abspielen eines Textes sowie die Speicherung des Anrufs (genau genommen ruft **vboxgetty** ein Tcl-Script, das die beiden letzteren Aufgaben übernimmt). Sinnvoll ist der Start von **vboxgetty** durch den init-Prozess, da das Programm, wie jedes Getty, ein bestimmtes Terminal überwacht und es neu gestartet werden muss, nachdem ein Anruf bearbeitet und das Programm beendet wurde.

Automatischer Serverstart

Für jeden Benutzer mit einem eigenen Anrufbeantworter ist ein eigenes **vboxgetty** zu starten. Jedes überwacht einen anderen Port. Die Zuordnung von Port zu Benutzer erfolgt in der im Anschluss beschriebenen Konfigurationsdatei »vboxgetty.conf«.

Im Ausschnitt der folgenden Datei /etc/inittab überwachen zwei »vboxgetty« die Terminals »ttyI6« und »ttyI7«:

```
root@sonne> vi /etc/inittab
...
I6:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI6
I7:35:respawn:/usr/sbin/vboxgetty -d /dev/ttyI7
...
```

Mit der zwingend erforderlichen Option **-d *Port*** ist das zu überwachende Terminal anzugeben. Die Voreinstellung der von **vboxgetty** zu verwendenden Konfigurationsdatei kann mittels **-f *Datei*** überschrieben werden.

Konfiguration mittels vboxgetty.conf

Die Ablage der Konfigurationsdatei unterscheidet sich bei den verschiedenen Distributionen, sollte aber entweder »/etc/isdn/vboxgetty.conf« oder »/etc/vbox/vboxgetty.conf« lauten.

»vboxgetty.conf« besteht aus einem *globalen* und mindestens einem *lokalen* Teil. Als *globale* Einstellungen gelten dabei alle Werte, die *vor* dem ersten *port*-Eintrag stehen. Dementsprechend beginnt mit jedem weiteren *port*-Eintrag ein neuer *lokaler* Teil. Globale Konfigurationen gelten dabei für alle Ports - insofern sie dort nicht überschrieben wurden - während lokale Konfigurationen nur dem entsprechenden Port zugeordnet sind. Die möglichen Optionen sind:

alivetimeout *Anzahl*

Anzahl Sekunden, nach der **vboxgetty** das Modem auf Aktivität testet

badinitsexit ***Anzahl***

Maximale Anzahl der Wiederholungen von **modeminit** (siehe unten)

commandtimeout ***Anzahl***

Anzahl Sekunden, die auf die Antwort eines Modemkommandos gewartet werden soll

compression ***Format***

Format der Ablage der Anruftexte (komprimierte Audiodateien); zulässig sind »adpcm-1«, »adpcm-2«, »adp
3«, »adpcm-4«, und »ulaw«

debuglevel ***Stufe***

Steuert die Menge der Statusinformationen, die **vboxgetty** zum Besten gibt. Die Werte reichen von fatalen
Fehlern (F) über behebbare Fehler (E), Warnungen (W), Informationen (I) und reine Debug-Ausgaben (D)
hin zu sehr detaillierten Auskünften (J).

dropdtrtime ***Anzahl***

Anzahl Millisekunden, die Dtr-Line aktiv bleiben soll, um das Modem zurück zu setzen

echotimeout ***Anzahl***

Anzahl Sekunden, die auf ein Echo eines Modemkommandos gewartet werden soll

freespace ***Anzahl***

Anzahl Bytes, die auf der Partition mit dem Spoolverzeichnis mindestens frei sein müssen, damit **vboxgetty**
Anrufe entgegen nimmt

group ***Gruppenname***

Der vboxgetty-Prozess wird mit den Rechten dieser existierenden Gruppe ausgestattet

initpause ***Anzahl***

Anzahl Millisekunden, die nach der Modem-Initialisierung zu warten ist

modeminit ***Intialisierung***

Eine Zeichenkette mit AT-Befehlen zur Initialisierung des Modem(treiber)s (siehe im Anschluss)

port ***Device***

Das ISDN-Device, für das die folgenden Einstellungen gelten sollen

ringtimeout ***Anzahl***

Anzahl Sekunden, die auf ein Freizeichen gewartet werden soll

spooldir ***Verzeichnis***

Verzeichnis zur Ablage von aufgezeichneten Anrufen, weiteren Konfigurationsdateien...

umask ***Maske***

Rechte der neu erzeugten Audiodateien; siehe Zugriffsrechte, umask

user ***Benutzername***

vboxconfig *Datei*

Konfigurationsdatei von vbox, Voreinstellung ist <spooldir>/vbox.conf

Für zahlreiche Optionen gelten Voreinstellungen, insofern sie nicht in der Konfigurationsdatei aufgeführt werden. Sie können davon ausgehen, dass diese Default-Werte in den allermeisten Fällen sinnvoll belegt sind, sodass nachfolgende Beispielkonfiguration für zwei Benutzer quasi das Minimum darstellt:

```
root@sonne> vi /etc/isdn/vboxgetty.conf
# Globale Einstellungen
group       users
debuglevel  FEW
# Nur für Port »ttyI6«
port        /dev/ttyI6
modeminit   ATZ&B512&E08150815
user        user
spooldir    /var/spool/vbox/user
# Nur für Port »ttyI7«
port        /dev/ttyI7
modeminit   ATZ&B512&E08150816
user        tux
spooldir    /var/spool/vbox/tux
```

Modemtreiber-Initialisierung

Im System emulieren die Gerätedateien »/dev/ttyI*X*« (X=0..15) ISDN-Modems. Jedes der konfigurierten virtuellen Modems überwacht den Steuerkanal am »echten« Modem und erkennt hieran für ihn bestimmte Anrufe.

Die Konfiguration der virtuellen Gerätedatei wird durch den Eintrag **modeminit** in »vboxgetty.conf« vorgenommen. Wie bei »echten« Modems gelangt der so genannte AT-Befehlssatz (weil die Befehle mit AT beginnen) aus dem Hayes-Kode zum Einsatz (die Erläuterung der wichtigsten Befehle finden Sie im Abschnitt System-Administration ➔ Hardware-Installation ➔ Modem).

Mit *ATZ*, was die Vorbelegung für **modeminit** ist, wird das Modem angewiesen, die internen Register auf »werkseitige« Standardeinstellungen zurückzusetzen. *AT&B512* stellt die maximale Größe ausgehender Pakete ein. Für die meisten Modemtreiber ist 512 ein gängiger Wert. Beachten Sie, dass die meisten Treiber »zu große« Pakete einfach ohne Warnung ignorieren. Mit *AT&E08150816* weisen Sie das mit dem Modem verbundene Gerät an, auf Anrufe der angegebenen *Multiple Subscribe Number* zu reagieren. Diese MSN-Nummer ist eine der Endgerätenummern (ohne Vorwahl!), die Ihnen für Ihren Anschluss zur Verfügung stehen.

Anlegen der Spool-Verzeichnisse

Die in »vboxgetty.conf« mittels **spooldir** angeführten Verzeichnisse müssen erzeugt und mit entsprechenden Attributen versehen werden. Jedes Verzeichnis muss dem im entsprechenden Abschnitt unter **user** deklarierten Benutzer und der jeweiligen Gruppe (**group**) gehören. Des Weiteren sind **incoming** und **messages** als Unterverzeichnisse anzulegen.

Bezogen auf obige Beispielkonfiguration hat der Administrator Folgendes zu tun:

```
root@sonne> mkdir -p /var/spool/vbox/{tux,user}/{incoming,messages}
root@sonne> chown -R tux:users /var/spool/vbox/tux
root@sonne> chown -R user:users /var/spool/vbox/user
```

Jeder Benutzer mit eigenem Anrufbeantworter benötigt in seinem Spoolverzeichnis die beiden Dateien **vbox.conf** und **standard.tcl**, die die Konfiguration der Abfrage des Anrufbeantworters enthalten. Um eine brauchbare Anfangskonfiguration bereitzustellen, sollte der Administrator zwei dem i4l-Paket beiliegende Beispieldateien kopieren:

```
root@sonne> cp /usr/share/doc/packages/vbox/examples/vbox.conf.example /var/spool/vbox/
```

```
{tux,user}/vbox.conf
root@sonne> cp /usr/share/doc/packages/vbox/examples/standard.tcl.example /var/spool/vbox/
{tux,user}/standard.tcl
root@sonne> chown -vR tux.users /var/spool/vbox/tux
root@sonne> chown -vR user.users /var/spool/vbox/user
root@sonne> chmod -vR o-rwx,g-rwx /var/spool/vbox/{tux,user]
```

Die endgültige Anpassung beider Dateien obliegt dem jeweiligen Benutzer (siehe nachfolgend).

Start und Test

Nach obigen Schritten muss **init** angewiesen werden, die Datei /etc/inittab neu einzulesen:

```
root@sonne> init q
```

Eventuelle Fehlkonfigurationen erkennen Sie anhand der Dateien */var/log/vbox/vboxgetty-ttyI**x**.log*. Nach angepasster Konfiguration genügt es nun, die betreffenden Gettys neu zu starten:

```
root@sonne> killall -1 vboxgetty
```

Den Eingang eines Anrufs kann der Systemverwalter außerdem anhand eines Eintrags in der Datei /var/log/messages überprüfen:

```
root@sonne> cat /var/log/messages | grep ttyI6
## Jan 16 18:38:34 sonne kernel: isdn_tty: call from 0, -> RING on ttyI6
```

Konfiguration des Anrufbeantworters

Erkennt **vboxgetty** einen einkommenden Anruf, so lädt es die der Rufnummer zugeordnete Datei **vbox.conf** (»vboxconfig«-Eintrag aus vboxgetty.conf). Diese Konfigurationsdatei bestimmt das weitere Verfahren von **vboxgetty**. I.d.R. wartet **vboxgetty** eine gewisse Anzahl Rufzeichen ab, bevor es ein Script **standard.tcl** startet, das die weitere Bearbeitung des Anrufs übernimmt.

Die Datei »vbox.conf«

vbox.conf sollte vom Benutzer angepasst werden, denn Anrufe werden nur von konfigurierten Rufnummern entgegen genommen.

Die Datei besteht i.d.R aus den drei notwendigen Sektionen »[CALLERIDS]«, »[RINGS]« und »[STANDARD]« sowie den optionalen nutzerspezifischen Sektionen. Die Namen eigener Sektionen sind frei wählbar. Sie werden in eckige Klammern eingeschlossen; Groß- und Kleinschreibung wird nicht unterschieden. Alle einem Sektionsbezeichner folgenden Einträge zählen zu dieser Sektion und enden mit Beginn einer weiteren Sektion oder dem Dateiende. Zeilen, die mit dem Kommentarzeichen # beginnen, werden ebenso ignoriert wie Leerzeilen.

Die Sektion **[CALLERIDS]** ordnet Rufnummern Personen(-gruppen) zu und ermöglicht somit den gezielten Ansprung einer nutzereigenen Sektion. Jeder Eintrag in [CALLERIDS] besteht aus drei durch Leerzeichen/Tabulatoren getrennten Werten:

```
Rufnummer Sektion Real-Name
```

Anstelle einer konkreten **Rufnummer** können Sie auch die shellüblichen Wildcards verwenden. **Sektion** ist entweder der Name einer existierenden Sektion oder "-", was gleichbedeutend mit [STANDARD] ist, oder aber '*', womit der Realname gleichzeitig als Sektionsbezeichner verwendet wird. Realname wird zumeist zur Beschreibung der unter dieser Nummer erreichbaren Person verwendet. Hier sind Leerzeichen gestattet, da alles dem zweiten Leerzeichen Folgende dem Realnamen zugeordnet wird.

```
[CALLERIDS]
    08151234[0-4]   BEKANNT   Tux der Pinguin
    98765432        BEKANNT   Alf
    *               -         *** Unknown ***
```

Im Beispiel verwenden wir die letzte Zeile, um Anrufe uns »unbekannter« Rufnummern zu behandeln. Wir weisen ihnen die Standard-Sektion (-) zu.

[RINGS] steuert das Verhalten beim Eintreffen eines Anrufs: Zu welcher Tageszeit sollen Anrufe entgegengenommen werden? Nach wie vielen Klingelzeichen soll der Anrufbeantworter abheben?

Jeder Eintrag in der Sektion [RINGS] enthält drei Werte:

```
Zeit Tag Anzahl_Klingelzeichen
```

Die **Zeit** wird als »Startzeit-Endzeit« angegeben. Die Angabe kann sekundengenau erfolgen (12:00:05-14:01:59). Fehlen Minuten und/oder Sekunden, so wird intern im Falle der Startzeit auf den kleinstmöglichen (also 12 auf 12:00:00) bzw. im Falle der Endzeit auf den maximalen Wert ergänzt (12 wird zu 12:59:59). Wird auf Angabe der Endzeit verzichtet, so wird die Startzeit ggf. »gespreizt« (12 wird zu 12:00:00-12:59:59). »Jede Zeit« wird durch * und »niemals« durch ! repräsentiert. Mehrere Zeitspannen lassen sich durch Kommata getrennt angeben (keine Leerzeichen!).

Als **Tag** sind sowohl die beiden ersten Buchstaben der deutschen Wochentagsnamen (Samstag statt Sonnabend) als auch die drei führenden Buchstaben der englischen Namen zulässig. Mehrere Werte lassen sich durch Kommata getrennt angeben (keine Leerzeichen!).

Anzahl Klingelzeichen ist als Vielfaches der Zeitspanne von ca. 4-5 Sekunden zu verstehen.

```
[RINGS]
    12:00:00-12:45:00,18-06:00:00   MO,DI,MI,DO,FR   5
    *                               SA,SO            2
```

Insofern Sie Anrufe unbekannter Nummern entgegennehmen, sollte zumindest eine allgemeine Sektion (vorgesehen ist [STANDARD]) enthalten sein. Auch ist für jede unter [CALLERIDS] benannte Sektion selbige zu konfigurieren. Mit diesen Sektionen können Sie gezielt auf konkrete Anrufe reagieren.

Ein Eintrag in der nutzerspezifischen Sektion besteht aus folgenden Werten:

```
Zeit Tag Ansagetext Aufzeichnungszeit [Flag]
```

Mit **Zeit** und **Tag** können Sie analog zum Vorgehen in [RINGS] die Zeiten einschränken, in denen nachfolgende Einstellungen anzuwenden sind. **Ansagetext** ist der Dateiname mit der Nachricht, die Sie abzuspielen wünschen. Entweder geben Sie den vollständigen Pfad an oder die Datei muss in /var/spool/vbox/*Benutzer*//messages liegen. Die **Aufzeichnungszeit** (in Sekunden) beschränkt die maximal zulässige Länge des empfangenen Anrufs.

Durch Kombination verschiedener **Flags** kann eine von der Voreinstellung abweichende Reaktion konfiguriert werden. Mit **NOANSWER** wird auf einen Anruf nicht geantwortet. **NORECORD** unterdrückt das Abspielen des Ansagetextes. **NOTIMEOUTMSG** verhindert die Benachrichtigung zur Überschreitung der Aufzeichnungslänge und **NOBEEPMSG** schaltet das Startzeichen der Aufzeichnung ab. Mit **NOStdMSG** wird auf einen Ansagetext verzichtet. Alle genannten Flags schalten die Voreinstellungen ab.

Des Weiteren kann mittels **RINGS=** ***xxx*** die Voreinstellung in [RINGS] überschrieben werden. **TOLLRINGS=** ***xxx*** wirkt analog, jedoch erst nach Eintreffen der ersten neuen Nachricht (also für jeden weiteren Anruf).

Mit **TOLLCHECK=** ***Verzeichnis*** kann ein von /var/spool/vbox/*Benutzer*/incoming abweichendes Verzeichnis angegeben werden, indem nach neuen Nachrichten gesucht wird. Ein vom

Standardskript /var/spool/vbox/*Benutzer*/standard.tcl abweichendes Skript kann per **SCRIPT=*Skript*** spezifiziert werden.

```
[STANDARD]
    *   *   standard.msg   60   RINGS=8
[BEKANNT]
    0-6   * guten_morgen.msg  120  RINGS=4
    7-23  * guten_tag.msg     120  RINGS=4
```

Das Skript »standard.tcl«

vboxgetty nimmt Anrufe nur entgegen, initialisiert anschließend anhand der Konfiguration aus vbox.conf interne Statusvariablen und überantwortet das weitere Vorgehen einem Tcl-Skript (in der Voreinstellung »standard.tcl«). Der Fluss in diesem Skript wird im Wesentlichen durch diese Statusvariablen von **vboxgetty** bestimmt. Tcl selbst erinnert an eine Mischung aus C- und Bash-Syntax und soll uns hier nicht näher interessieren. Wir vermitteln nur kurz anhand eines typischen Skriptbeginns den Umgang mit Tcl:

```
user@sonne> vi /var/spool/vbox/user/standard.tcl
# Löschen der Touchtone-Sequenz von vboxgetty und Entfernen aller Einträge von der Anrufer-Warteliste
vbox_init_touchtones
vbox_breaklist rem all

# Soll die Standard-Nachricht abgespielt werden? »vbox_flag_standard« wird von vboxgetty initialisiert. Voreinstellung ist 'true' - außer in vbox.conf ist das Flag NOStdMSG gesetzt.
if { "$vbox_flag_standard" == "TRUE" } {

# »vbox_put_message« spielt die als Argument übergebene Nachricht ab (»vbox_msg_standard« wird wiederum von vboxgetty belegt); der Rückgabewert wird an die Variable RC zugewiesen
  set RC [ vbox_put_message $vbox_msg_standard ]

# 500 ms warten
  vbox_pause 500

# Wurde von Gegenseite aufgelegt, dann wird das Skript beendet
  if { "$RC" == "HANGUP" } {
    return
  }
}
...
```

Sollten Sie in die Verlegenheit gelangen, »standard.tcl« anpassen zu müssen, so finden Sie eine vollständige Auflistung der vbox-relevanten Befehle im Manual »vboxtcl«.

Der VBox-Daemon

In lokalen Netzwerken verfügt nicht jeder Rechner über eine direkte Anbindung an die Außenwelt. Meist regelt ein einzelner Rechner den Zugang und auch nur dieser Rechner wird ein ISDN-Modem integriert haben. Es wäre nun zweckmäßig, wenn die Benutzer an den anderen Rechnern sich nach dem Eingang neuer Anrufe erkundigen könnten. Genau die Beantwortung solcher Anfragen übernimmt ein eigener Hintergrundprozess - der **vboxd**.

Vorarbeiten

Zunächst sollten Sie sich vergewissen, dass **vboxd** in der Datei /etc/services aktiviert ist:

```
root@sonne> grep vbox /etc/services
vboxd       20012/tcp     # for vbox daemon
vboxd       20012/udp
```

vboxd beendet seine Tätigkeit automatisch nach einer gewissen Zeitspanne der Inaktivität. Zwar lässt sich die Zeitspanne von 600 Sekunden per Option ändern, um seine permanente Verfügbarkeit dennoch zu garantieren, sollte er jedoch bei Bedarf vom inetd gestartet werden:

```
root@sonne> grep vbox / etc/ inetd.conf
# vbox (Voice Box)
vboxd   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/vboxd
```

Bevorzugen Sie den sicheren xinetd, dann sieht ein Eintrag in der Datei /etc/xinetd.conf wie folgt aus:

```
root@sonne> vi / etc/ xinetd.conf
service vboxd
{
  socket_type = stream
  wait        = no
  user        = root
}
```

Globale Konfiguration in /etc/vbox/vboxd.conf

Die Datei »vbox.conf« umfasst zwei Arten von Einträgen. Zum Einen die mit 'L' beginnenden Zeilen, die den Zugriff auf **vboxd** auf Rechnerebene steuern und zum Zweiten mit 'A' beginnende Zeilen, die Zugriffsrechte auf Benutzerebene steuern.

```
L:Rechnermuster:Zugriff
A:Rechnermuster:Modus:Name:Passwort:Spoolverzeichnis:Verzeichnis_neuer_Nachrichten
```

Die Angabe des Zugriffs auf Rechnerebene ist einfach: Dem 'L' folgt ein die Rechner beschreibendes Muster gefolgt von einem Zugangsflag. Die drei Werte werden durch Doppelpunkten voneinander getrennt.

```
L:*.galaxis.de:Y
L:*:N
```

Der Eintrag mit dem ersten zutreffenden Muster bestimmt die Rechte für einen Rechner. Im Beispiel sind also alle Rechner aus der Domain »galaxis.de« zum Zugriff auf **vboxd** berechtigt (Y). Allen anderen Rechner wird der Zugriff verweigert (N).

Ein Eintrag auf Nutzerebene besteht aus 7 jeweils durch Doppelpunkt getrennten Werten. Dem anfäglichen 'A' folgt ein Rechnermuster, von dem aus der mit Name spezifizierte Benutzer aus zugreifen darf (4. Wert; natürlich muss dem Rechner selbst der Zugriff gestattet sein). Der Modus beschränkt den Zugriff auf Nur-Lesen (R) oder Nur-Schreiben (W) oder er lässt auch beides (RW) zu. Das Passwort wird zur Zugangskontrolle verwendet. Das Spoolverzeichnis ist dem Benutzer zugäglich. Hier werden Kontrolldateien abgelegt. Das letzte Verzeichnis beschreibt den Ablageort für neue Nachrichten.

```
A:localhost:RW:user:password:/var/spool/vbox/user:incoming
A:sonne.galaxis.de:RW:tux:password:/var/spool/vbox/tux:incoming
A:*:!:!:!:!:!
```

Nutzerspezifische Anpassungen in ~/.vboxrc

Um mit dem Konsolenprogramm **vbox** auf die Voicebox zugreifen zu können, muss die Datei .vboxrc im Heimatverzeichnis des Benutzers existieren. Drei wesentliche Variablen sollten angepasst werden:

USERNAME

Name des Benutzers, unter dem die Anmeldung am **vboxd** vorgenommen wird. Dieser Name muss in der vboxd.conf (4.Wert einer 'A'-Zeile) existieren!

PASSWORD

Passwort zur Anmeldung (5.Wert der 'A'-Zeile aus vboxd.conf)

VOLUME

Die weiteren möglichen Einträge in .vboxrc steuern das Aussehen der Anwendung. Im nachfolgenden Beispiel wird ihre Anwendung gezeigt; der Nutzen sollte aus den Variablennamen ersichtlich sein:

```
user@sonne> vi ~/.vboxrc
USERNAME  user
PASSWORD  GEHEIM

VOLUME    200

# Farbdefinitionen:
Element Vordergrundfarbe:Hintergrundfarbe
C_BACKGROUND    GRAY:BLACK
C_STATUSBAR     GRAY:BLUE
C_STATUSBAR_HL  YELLOW:BLUE
C_POWERLED_ON   GREEN:BLUE
C_POWERLED_OFF  RED:BLUE
C_STATUSLED_ON  YELLOW:BLUE
C_STATUSLED_OFF BLACK:BLUE
C_LIST          GRAY:BLACK
C_LIST_SELECTED GRAY:RED
C_INFOTEXT      GREEN:BLACK
```

Zum Abschluss...

Und wie generieren Sie nun Ihre Ansagetexte? Indem Sie sich selbst anrufen und den Begrüßung auf den Anrufbeantworter sprechen. Kopieren Sie die Nachricht anschließend entsprechend um... und fertig!

ISDN - Kanalbündelung

Faxserver

Bei **faxgetty** handelt es sich um den *Hylafax*-Server-Prozess, der das Modem auf einkommende Anrufe überwacht und diese entweder selbst behandelt oder sie an ein geeignetes Programm deligiert.

Nullmodem

Wie schaufelt man eine Menge Daten zwischen zwei Rechnern hin und her? Mit zwei Netzwerkkarten kein Problem, aber extra solche besorgen, nur wegen eines sporadischen Datenaustauschs? Auch über ein ZIP-Laufwerk verfügen die wenigsten Computer und die Transaktion via Disketten ist wirklich nur dem gedultigen Benutzer zuzumuten.

Worüber aber (fast) jeder Rechner verfügt, ist mindestens eine serielle Schnittstelle. Zu einer Verbindung dieser ist einzig ein preiswertes Kabel erforderlich und mit den heute üblichen Schnittstellenbausteinen ist eine Übertragungsrate nahe derer von langsamen Netzwerkkarten zu erwarten. Das ganze nennt sich **Nullmodem** und ist eine Low-Cost-Variante zur gelegentlichen Verbindung zweier Rechner.

Sicher ist ein Datentransfer der häufigste Verwendungszweck, aber auch eine Fernadministration eines Rechners über dessen serielle Schnittstelle ist hier und da von Nutzen.

Konfigurationsdateien

Übersicht

Der vorliegende Abschnitt beschreibt die Syntax der Konfigurationsdateien in /etc mit netzwerkrelevantem Inhalt. Die alphabetische Anordnung unterstreicht unser Ansinnen, diese Darstellung nicht vorrangig als Lehrstoff sondern als Referenz zu verstehen.

So manche Konfigurationsdatei wird von zahlreichen Programmen benötigt. Andere wiederum sind nur für einen einzigen Dienst relevant. Die »allgemeineren« Dateien werden nachfolgend vollständig beschrieben. Wir werden uns bei der Besprechung der Dienste auf die hier getroffenen Aussagen berufen, ohne die Details zu wiederholen. Dem entgegen sollen die Feinheiten der »spezifischen« Konfigurationsdateien erst im Abschnitt zum jeweiligen Dienst die ihnen gebührende Aufmerksamkeit erfahren. Dass wir sie dennoch hier erwähnen, dient der Vollständigkeit der Aufzählung.

Die Datei / etc/ HOSTNAME

In der Startphase des Netzwerks ermitteln einige Programme anhand des Inhalts dieser Datei den Namen des Rechners. Im Unterschied zu SuSE-Linux nennt sich die Datei bei RedHat und Debian **/ etc/ hostname**. Hier sollten Sie den Netzwerknamen des Rechners **ohne** die Domainerweiterung eintragen. Für den Rechner *sonne.galaxis.de* lautet der Eintrag:

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

Genau genommen, weist das Kommando **hostname** einem Rechner seinen Namen zu. Nahezu alle Distributionen lesen den Inhalt von /etc/hostname bzw. /etc/HOSTNAME während des Bootvorgangs ein und füttern damit **hostname**.

Die Datei / etc/ exports

Die Datei /etc/exports legt die von einem Network Filesystem Server bereitgestellten Verzeichnisse fest. Zu jedem Verzeichnis wird eine Liste der Rechner angeführt, die dieses importieren dürfen. Des Weiteren regeln verschiedene Optionen die Art des Zugriffs wie auch die Verfahrensweise mit bestimmten Benutzerkennungen (bspw. Root). Ein Eintrag besitzt folgende Gestalt:

```
<Verzeichnis> <Rechner>([Option(en)]) [<Rechner>([Option(en)])]
```

Rechner kann hierbei sowohl ein konkreter Rechnername ("sonne", "sonne.galaxis.de") sein, als auch ein durch Wildcards angegebenes Muster für Rechnernamen ("*.galaxis.de") oder aber eine Netzgruppe ("@LocalAdmins") sein. Letztere muss in der Datei /etc/netgroup erscheinen.

Die Besprechung der vielfältigen Optionen soll dem Abschnitt zum NFS-Server vorbehalten bleiben, an dieser Stelle dient einzig ein Beispiel der Demonstration des grundsätzlichen Aufbaus dieser Datei:

```
user@sonne> cat / etc/ exports
# Verzeichnis darf von jedem Rechner »nur-lesend« importiert werden
/opt/export/              *(ro)

# Root bleibt Root auf diesem Verzeichnis
/usr/src/linux            @developer(no_root_squash,rw)

# Einige Home-Verzeichnisse
/home/user                *.galaxis.de(rw) @outside(rw)
/home/tux                 *.galaxis.de(rw) @outside(rw)

# Für »diskless« Clients:
/tftpboot/nfsroot         *(rw,no_root_squash)
```

Die Dateien / etc/ ftpaccess, / etc/ ftpconversions, / etc/ ftpmsg.dead, / etc/ ftpusers

Die Dateien / etc/ gated, / etc/ gated.conf, / etc/ gated.version

Die Datei / etc/ host.conf

Die Datei steuert den so genannten **Resolver**, also den Mechanismus, der für die Auflösung von symbolischen Rechnernamen in IP-Adressen und umgekehrt zuständig ist. Allerdings greifen einzig Programme auf »/etc/host.conf« zurück, die gegen die (veralteten) Bibliotheken »libc4« bzw. »libc5« gelinkt sind. Aktuelle Programme nutzen die »glibc2« (diese Bezeichnung ist gebräuchlicher als »libc6«), wo die durch »/etc/host.conf« bereit gestellte Funktionalität von der Datei /etc/nsswitch.conf übernommen wird.

Um aus einem symbolischen Rechnernamen die zugehörige IP-Adresse zu ermitteln, stehen prinzipiell 3 Möglichkeiten zur Verfügung:

1. Nachschauen in der Datei /etc/hosts
2. Befragen eines Domain Name Servers
3. Befragen eines NIS (oder NIS+) Servers

Damit ergibt sich einmal die Frage, auf welches Vorgehen Sie zurückgreifen möchten und, im Falle der Verwendung mehrerer Schemata, in welcher Reihenfolge eine Auflösung erfolgen soll.

```
user@sonne> cat / etc/ host.conf
order hosts nis bind
multi on
```

In der Datei »/etc/host.conf« bestimmt die mit **order** beginnende Zeile die Namensauflösung. »hosts« besagt, dass zunächst in der lokalen Datei /etc/hosts nachgesehen werden soll (die lokale Auflösung sollte immer zuerst stehen), anschließend soll ein NIS-Server befragt werden (nis) und, falls immer noch kein Ergebnis erzielt wurde, soll nun die Adresse über DNS ermittelt werden (»bind«, hier kann auch »dns« stehen). **multi on** besagt, dass ein Rechner durchaus mehrere IP-Adressen haben kann (Gateways, Router,...), mit »multi off« führt eine Anfrage, die verschiedene IP-Adressen ermittelt, zu einem Fehler.

Die Datei / etc/ hosts

In dieser Datei findet die Zuordnung von Rechnernamen zu entsprechenden IP-Adressen statt. Selbstverständlich kann hier nicht jeder Rechner des Internets aufgeführt werden (in dessen Anfängen war es aber so), weshalb diese Datei nur für die unmittelbaren, »wichtigen« Nachbarrechner (z.B. Nameserver, DNS-Server, Proxy) genutzt wird.

```
user@sonne> cat / etc/ hosts
# Typischer Aufbau einer /etc/hosts
# <IP-Adresse> <vollständiger Rechnername> <Rechnername> [<Alias>]
#
127.0.0.0       localhost            # loopback
192.168.10.101  sonne.galaxis.de     sonne   mail
192.168.85.1    tuxhausen.outside.all   tuxhausen

# Darstellung der Adresse in oktaler Form
0340.023.0222.03        vulcan.outsite.org

# Darstellung der Adresse in hexadezimaler Form
0xA0.0x77.0x02.0x02     melmac.outsite.com
```

Die Dateien / etc/ hosts.allow, / etc/ hosts.deny, / etc/ hosts.equiv

Die Datei **hosts.allow** dient der Zugangskontrolle von Nutzern/Diensten anderer Rechner. Für bestimmte Hosts/Netzwerke kann hier der Zugriff auf bestimmte lokale Dienste explizit gestattet werden.

```
user@sonne> cat / etc/ hosts.allow
# 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 tuxhausen erlaubt.
#
telnetd, ftpd: LOCAL, tuxhausen.outside.all

# Finger ist jedem erlaubt, aber root wird per Mail darüber informiert
#
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

In **hosts.deny** kann der Zugang zu bestimmten Diensten des Rechners für bestimmte Hosts/Netzwerke explizit untersagt werden.

```
user@sonne> cat / etc/ hosts.deny
# Aufbau von /etc/hosts.deny analog zu /etc/hosts.deny
#
ALL: outside.all

# Die remote-Shell ist eine bekannte Sicherheitslücke
#
rsh: ALL
```

hosts.allow und **hosts.deny** werden u.a. vom TCP-Wrapper ausgewertet. Dabei wird zuerst die **hosts.allow** und dann die **hosts.deny** betrachtet. Es gilt: *Wurde etwas in der "hosts.allow" explizit gestattet, darf es in "hosts.deny" nicht mehr verboten werden.*. Wenn ein entsprechender Eintrag in "hosts.deny" dennoch steht, wird er ignoriert.

Der Vollständigkeit halber soll auch der Aufbau der Datei **hosts.equiv** erwähnt werden. Alle von den darin aufgeführten Rechnern zugreifenden Benutzer haben dieselben Rechte wie lokale Nutzer ("trusted hosts") und können ohne explizite Angabe eines Passwortes über entfernte Dienste auf den Rechner zugreifen. Voraussetzung ist - neben einem entsprechenden Eintrag - die Existenz derselben Nutzerkennung auf dem Zielrechner.

Aus Sicherheitsgründen sollte auf die Möglichkeiten dieser Konfiguration verzichtet werden.

```
user@sonne> cat / etc/ hosts.allow
# Aufbau einer /etc/hosts.equiv
```

```
#     [+|-] <Rechnername> [<username>]
#
# Die Nutzer "user" und "newbie" haben von Rechner "erde" aus Zugang
erde.galaxis.de user
erde.galaxis.de newbie

# alle Nutzer des Rechners tuxhausen
tuxhausen.outside.all

# Mitglieder dieser Netzgruppe sind vom passwortfreien Zugang ausgeschlossen
tuxhausen.outside.all    -@LocalAdmins
```

Anmerkungen:

- Ein einzelnes + erlaubt den Zugang von sämtlichen Rechnern aus
- Wird ein Rechner oder Nutzer vom Zugang ausgeschlossen (führendes -), so heißt das nur, dass ein Anmelden immer die Angabe des Passwortes erfordert
- Root wird niemals der passwortfreie Zugang gestattet

Die Datei / etc/ hosts.lpd

Die Datei / etc/ inetd.conf

Die Datei / etc/ lmhosts

Die Datei / etc/ named.conf

Die Datei / etc/ netgroup

Bei der Konfiguration der Zugriffsberechtigungen für die Dienste **rlogin** (Remote Login), **rsh** (Remote Shell) und dem Export von Verzeichnissen (NFS) möchte man den Zugang häufig ein und derselben Gruppe von Rechern/Nutzern gestatten. Anstatt z.B. jedes zu exportierende NFS-Verzeichnis mit der vollständigen Liste der (nicht) berechtigten Rechner/Nutzer zu versehen, kann durch Definition von Netzgruppen der Schreibaufwand erheblich reduziert werden.

```
<Name der Netzgruppe>  (<Rechnername>,<Benutzername>,<Domänname>)
```

Jede Netzgruppe wird auf einer eigenen Zeile beschrieben. Dabei folgt dem Namen der Netzgruppe eine Liste von (Rechner,Nutzer,Domän)-Einträgen oder der Name einer zugehörigen Netzgruppe. Eine fehlende Angabe steht für "alle", ein Minus (-) bedeutet "kein gültiger Wert". Das Domän-Feld beschreibt nicht die vertrauenswürdigen Rechner, sondern nur den Geltungsbereich der Netzgruppe. Dieses Feld kann entweder leer sein oder es muss den Namen der lokalen Domän enthalten.

```
user@sonne> cat / etc/ netgroup
# /etc/netgroup
#
LocalAdmins (,root,)

NetAdmins (sonne,user1,) (sonne,user2,) LocalAdmins

Gateways (rechner1,,) (rechner7,,)
```

Die Datei / etc/ networks

Netzwerknamen werden hier in Netzwerkadressen umgesetzt. Diese Informationen dienen manchen Programmen, um anstelle von Nummern aussagekräftige Namen anzuzeigen. Notwendig sind die Angaben jedoch nicht.

```
user@sonne> cat / etc/ hosts.allow
# Aufbau einer /etc/networks
# <Netzwerkname> <Netzwerkadresse>
#
loopback  127.0.0.0
localnet  192.168.10.0
edu-net   192.168.85.0
```

Die Datei / etc/ nntpserver

Diese Datei enthält eine einzige Zeile mit dem Namen eines Newsservers. Einige Newsclients (tin, leafnode) kontaktieren bei Aufruf den eingetragenen Server. Hat man einen lokalen Newsserver konfiguriert, steht hier einfach "localhost":

```
user@sonne> cat / etc/ nntpserver
# bei lokalem Newsserver...
#
localhost
```

Andere Newsserver oder -clients erwarten den Namen des Servers in der Shellvariablen NNTPSERVER. Nahezu jede Distribution belegt diese Variable anhand der Datei /etc/nntpserver.

Die Datei / etc/ nsswitch.conf

Die Datei / etc/ protocols

Die Protokollnummern der Transportprotokolle sind hier so eingetragen, wie sie im IP-Protokollkopf erscheinen. Diese Datei sollte nach der Installation existieren und muss nicht verändert werden.

```
# 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
...
```

Die Datei / etc/ resolv.conf

```
search galaxis.de subdomain.galaxis.de
nameserver 127.0.0.1
nameserver 192.168.100.3
```

Die Datei / etc/ services

Da mehrere Dienste (»Services«) gleichzeitig auf einem Rechner aktiv sein können, muss ihre Adressierung differenzierter erfolgen, als es allein durch die Rechneradresse möglich wäre. Hier gelangen die **Portnummern** ins Spiel. Nun gibt es reichlich »Standarddienste«, deren Portnummern fest vorgeschrieben sind (Wie sollte sonst ein Client in Erfahrung bringen, an welchem Port der Webserver auf www.linuxfibel.de wartet?). Genau jene Nummern sind in der Datei /etc/services ausgeführt.

Die Vergabe der Portnummern (und auch der Domainnamen) obliegt der **ICANN** (Internet Corporation for Assigned Names and Numbers), einer nicht-profit-orientierten Vereinigung, der die Organisation der Namensvergabe im Internet obliegt. Die Portnummern sind in drei Bereiche unterteilt:

1. **Well Known Ports** (0..1023) Sie werden vom ICANN zugewiesen und sollten niemals geändert werden.
2. **Registered Ports** (1024..49151) Sie werden vom ICANN registriert und dienen zumeist »großen« Firmen dazu, um ihren Client-Server-Anwendungen konkrete Ports zu reservieren. Ihre Verwendung zu eigenen Zwecken sollte keine negativen Konsequenzen haben.
3. **Private Ports** (49152..65536) Stehen zur freien Verfügung...

Kurzum: Die /etc/services sollte nur derjenige modifizieren, der eigene Client-Server-Anwendungen entwickelt oder auf solche zugreift.

```
#               0/tcp    Reserved
#               0/udp    Reserved
tcpmux          1/tcp               # TCP Port Service Multiplexer
tcpmux          1/udp               # TCP Port Service Multiplexer
compressnet     2/tcp               # Management Utility
compressnet     2/udp               # Management Utility
compressnet     3/tcp               # Compression Process
compressnet     3/udp               # Compression Process
rje             5/tcp               # Remote Job Entry
rje             5/udp               # Remote Job Entry
...
```

Die Datei / etc/ xinetd.conf

Die Dateien / etc/ yp.conf und / etc/ ypserv.conf

```
root@sonne>  cat / etc/ yp.conf
ypserver sonne.galaxis.de
ypserver erde.galaxis.de
ypserver 192.168.100.107
```

Übersicht

Konfigurationstools

Übersicht

Netzwerk-Diagnose

Übersicht

Das eine oder andere hier aufgeführte Kommando finden Sie auch an anderen Stellen des Buches. Der hiesige Schwerpunkt liegt auf Netzwerkdiagnose, d.h. auf der Verwendung der Kommandos zur Analyse von Netzwerkproblemen. Deshalb finden sie zu Kommandos, die auch für andere Zwecke eingesetzt werden können, im folgenden Abschnitt einzig die zur Inspektion der Netzwerkfunktionalität relevanten Optionen.

Die Fehlersuche gestaltet sich zumeist schwierig. Sie erfordert eine gewisse Methodik, um sich aus der Masse der Möglichkeiten zur wahrscheinlichsten Ursache vorzuarbeiten. Die Reihenfolge der Beschreibung der Ihnen als Administrator zur Seite stehenden Werkzeuge spiegelt die von uns bevorzugte Herangehensweise ab. Wir hangeln uns vom Allgemeinen ins Detail, beginnend bei einfacher Funktionsprüfung bis hin zu Maßnahmen der Optimierung.

Das Kommando ping/ ping6

Vorwort

Funktional besteht kein Unterschied zwischen **ping** (IPv4) und **ping6** (IPv6), sodass wir in nachfolgender Beschreibung auf eine Unterscheidung verzichten. Verwenden Sie **ping** in traditionellen IPv4-Netzwerken und **ping6** in Netzwerken, die unter IPv6 laufen.

Sollten Sie dennoch einmal einem der Kommandos die »falsche« IP-Adresse unterschieben, so wird dieses den Dienst mit »unknown host« quittieren:

```
user@sonne> ping ::1
ping: unknown host :1
user@sonne> ping6 ::1
PING ::1 (::1): 56 data bytes
64 bytes from ::1: icmp_seq=0 ttl=64 time=0.054 ms
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.052 ms
...
```

Das Kommando **ping** greift auf Systemressourcen zu, die i.d.R. nur für *root* zugänglich sind. Um dem »normalen« Benutzer dennoch seine Anwendung zu ermöglichen, ist zumeist das setuid-root Flag gesetzt.

Allgemeines

ping dient vorrangig zum Testen, Messen und Administrieren von Netzwerken. Es erhöht die Netzwerklast, womit sein Einsatz in automatischen Skripten in Produktionsumgebungen zu vermeiden ist.

»Ping« wurde nach dem Klang von Sonar-Ortungssystemen benannt, wo niederfrequenter Schall in die Tiefen des Ozeans ausgesandt wird, um anhand von Reflexionen den Standort von Objekten zu kalkulieren. Im übertragenen Sinne arbeitet »Ping« analog zu einem Sonar, da es eine Anforderung ins Netzwerk entlässt und anhand der Art und Weise der Reaktion (»Reflexion«) verschiedene Rückschlüsse zu ziehen vermag. Welche das sein können, werden Sie nachfolgend kennen lernen.

Intern arbeitet **ping** über das ICMP-Protokoll, indem es ein *ICMP ECHO_REQUEST* (Typ 8) an den entfernten Rechner im Netzwerk sendet, um (hoffentlich) ein *ECHO_RESPONSE* (Typ 0) als Antwort zu erhalten.

Wichtigste Erkenntnis aus einem *ECHO_RESPONSE* ist sicherlich, dass die Gegenseite arbeitet und erreichbar ist. Aussagen zur Qualität der Erreichbarbeit lassen sich ebenso ableiten.

Die Anwendung des Kommandos

Zumindest der Zielrechner ist beim Aufruf des Kommandos als Parameter anzugeben. Ohne weitere Optionen werden ständig *ECHO_REQUEST*'s ausgesendet, d.h. der Zielrechner wird ohne Unterlass »angepingt«. Dies können Sie nur durch Eingabe von [Ctrl][C] unterbrechen. **ping** schreibt abschließend eine kurze Zusammenfassung auf die Standardausgabe:

```
user@sonne> ping localhost
PING localhost (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.143 ms
64 bytes from localhost (127.0.0.1): icmp_seq=2 ttl=64 time=0.124 ms
64 bytes from localhost (127.0.0.1): icmp_seq=3 ttl=64 time=0.123 ms
[Ctrl][C]
--- localhost ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 1998ms
rtt min/avg/max/mdev = 0.123/0.130/0.143/0.009 ms
```

[Ctrl][C] bewirkt das Senden des Signals *SIGINT* an den Prozess, das im Falle von **ping** zum Beenden des Programms führt. **ping** reagiert ebenso auf ein Signal *SIGQUIT*, womit die Statistik erscheint, ohne das Programm zu beenden. *SIGQUIT* wird durch keinen Tastencode erzeugt.

Die Statistik

Das Ausgabeformat der Statistik ist zum Großteil selbsterklärend. Eine erste Zeile enthält die Anzahl ausgesendeter und empfangener Pakete sowie den Verlustfaktor (in Prozent). Ein letzter Wert gibt die Gesamtzeit des Programmlaufs an.

Die zweite Zeile umfasst vier Zeitwerte. Erster beschreibt die kürzeste Zeitspanne, nach der eine Antwort auf ein ausgesendetes Paket eintraf. Der zweite Wert enthält die längste gemessener Dauer. Dritter Wert ist die durchschnittliche Zeitdauer und Wert Nummer 4 beschreibt die gemittelte Abweichung vom durchschnittlichen Wert.

Die Optionen

Allgemeine Optionen

Etliche Optionen von **ping** lassen sich nach funktionalen Aspekten gruppieren. Aber eben nicht alle. Die nach unserem Schema nicht »qualifizierbaren« Optionen fassen wir deshalb als »allgemeine« Optionen zusammen. Diese Einteilung ist rein willkürlich!

Die weiter oben beschriebene Anwendung des Kommandos eignet sich nicht zur Verwendung in Skripten, da es zum Beenden eine Interaktion mit dem Benutzer erfordert. Zwei Optionen steuern deshalb das Programmende:

-c *Anzahl*

Begrenzt die Anzahl zu sendender »Pings«. Nach Erreichen dieser endet das Programm selbstätig.

-w *Sekunden*

Nach Verstreichen der angegebenen Sekunden endet das Programm selbstätig unabhängig von der Anzahl gesendeter bzw. erhaltenener Pakete.

In Verbindung mit beiden Optionen steht Ihnen zur Auswertung der Rückgabewert des Kommandos **ping** zur Verfügung. Hierbei bedeutet:

0

Der Zielrechner ist erreichbar; es wurde mindestens ein *ECHO_RESPONSE* empfangen.

1

Es wurde entweder gar kein *ECHO_RESPONSE* empfangen oder, bei gleichzeitiger Verwendung der Optione w« und »-c«, die spezifizierte Anzahl wurde nicht erreicht (weniger »Pongs« als »Pings«).

2

Nicht näher spezifizierter Fehler.

Zwei Beispiele zur Auswertung des Rückgabewertes:

```
user@sonne> ping -c 1 localhost
PING localhost (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.158 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% loss, time 0ms
rtt min/avg/max/mdev = 0.158/0.158/0.158/0.000 ms
user@sonne> echo $?
0
```

```
user@sonne> for host in server1 server1;
> do
> ping -c 1 $host 2>&1 > /dev/null || echo "Rechner $host nicht erreichbar!"
> done
```

Eher formalen Charakter besitzen folgende Optionen:

-a

Für jedes eintreffende *ECHO_REPLY* wird ein Piepton ausgegeben. Die Erreichbarkeit eines Rechners wird d akkustisch signalisiert.

-n

Rechner werden stets mit ihrer IP-Adresse angegeben (und nicht mit ihrem Namen).

-q

Sämtliche Ausgaben mit Ausnahme der ersten Zeile und der Statistik werden unterdrückt.

Optionen zur Steuerung des Zeitverhalten

In der Voreinstellung versendet **ping** seine Anforderungen im Sekundentakt. Die Optionen zur Manipulation des Zeitverhaltens erfordern teils Rootrechte, da sie zu extremer Erhöhung der Netzwerklast missbraucht werden könnten.

-i *Sekunden*

Die Option legt die Zeitspanne in Sekunden zwischen dem Aussenden zweier aufeinander folgender */CMP_REQUEST*'s fest. Nur *Root* darf einen Wert < 0.2 (Sekunden) wählen.

-f

(*flood ping*). Dieses »flutende« Ping entspricht der Option »-i 0«, d.h. die */CMP_REQUEST* werden ohne Verzögerung abgesetzt. Das Setzen der Option bleibt einzig Root vorbehalten. Jedes gesendete */CMP_REQUEST*-Paket wird in der Ausgabe durch einen Punkt symbolisiert, der bei eintreffendem

ICMP_RESPONSE durch einen Backspace »\b« gelöscht wird. Die Anzahl Punkte entspricht somit der Anzah ausstehender Antworten.

-l *Anzahl*

Die Wirkungsweise der Option ist wie »-f« nur dass sie die Anzahl zu sendender *ICMP_REQUEST*'s beschrär Nur Root darf mehr als 3 Pakete gleichzeitig abschicken.

-t *TimeToLive*

Der so genannte *TTL*-Wert (Time To Live) entspricht der Anzahl von Routern, die ein Paket maximal passie kann. Jeder Router auf dem Weg des Pakets verringert dessen TTL-Wert stets um eins. Geht der Wert in ei Router auf Null, so wird dieser das Paket verwerfen und eine Fehlermeldung an den Absender senden. Die Bezeichnung *Time to live* stammt noch aus den Anfängen von TCP/IP, zu denen tatsächlich die Lebensdaue eines Pakets durch einen Zeitwert begrenzt wurde.
Die Voreinstellung für TTL ist 255.

```
# Falls 3 Router auf dem Weg liegen...
user@sonne> ping -c 1 rechner.irgendwo.foo
PING rechner.irgendwo.foo (192.168.99.125) from 192.168.239.100 : 56(84) bytes of data.
64 bytes from rechner.irgendwo.foo (192.168.99.125): icmp_seq=1 ttl=252 time=3.92 ms

--- rechner.irgendwo.foo ping statistics ---
1 packets transmitted, 1 received, 0% loss, time 0ms
rtt min/avg/max/mdev = 3.926/3.926/3.926/0.000 ms
```

Der *TTL*-Wert in der Endstatistik (obiges Beispiel) wird von dem Rechner gesetzt, der »angepingt« wurde. Die Interpretation des Wertes gestaltet sich schwierig, da es keine feste Regeln gibt, nach denen ein Rechner das Feld belegt. Manche Implementierungen schreiben dort einen Wert, der 255 minus der Anzahl der durchlaufenen Router beträgt. Wiederum andere Rechner verändern den Wert auf 255 oder auf einen Wert, der durch ein übergeordnetes Protokoll des TCP/IP-Protollstacks vorgegeben wird (bspw. arbeitet Telnet mit einen TTL-Wert von 60 oder 30).

Theoretisch ist es also möglich, dass ein Rechner per **ping** erreichbar ist, der Kontakt via Telnet aber scheitert (wegen des zu geringen TTL-Wertes von Telnet). Praktisch wird ein solcher Fall kaum auftreten, da eine Vermittlung über mehr als 30 Router ein sicheres Indiz für eine Fehlkonfiguration ist (eine Schleife).

Optionen zur Beeinflussung der Testdaten

Im Abschnitt Netzwerkprotokolle finden Sie eine Beschreibung des Aufbaus eines ICMP-Protokollkopfes. Uns interessiert im Folgenden das in der dortigen Skizze als »Optionale Daten« bezeichnete Feld. Dessen Grösz lig;e findet sich in der Ausgabe des **ping**-Kommandos wieder:

```
user@sonne> ping -c 1 localhost | head -1
PING localhost (127.0.0.1) from 127.0.0.1 : 56(84) bytes of data.
```

Per Voreinstellung verpackt **ping** 56 Bytes Daten in ein Paket. Dieser Wert trägt auch zur zweiten, in Klammern stehenden Angabe bei. Diese 84 Bytes im Beispiel errechnen sich aus den 20 Bytes des IP-Protokollkopfes, aus 8 Bytes, die Laufzeitinformationen des Pakets enthalten und eben den 56 Datenbytes.

Die Paketgröße ist konfigurierbar:

-s *Paketgröße*

Setzt die Größe des Datenpakets auf den angegebenen Wert (in Bytes).

Die Option eignet sich insbesondere zur Diagnose von Netzwerkproblemen, die auf Probleme mit der Paketgröße hindeuten.

```
user@sonne> ping -c 1 -s 0 localhost | head -1
```

```
PING localhost (127.0.0.1) from 127.0.0.1 : 0(28) bytes of data.
root@sonne> ping -c 1 -s 65508 localhost |  head -1
WARNING: packet size 65508 is too large. Maximum is 65507
ping: local error: Message too long, mtu=16436
```

In letztem Beispiel provozierten wir einen Überlauf, indem wir ein für IP zu großes Datenpaket verwendeten (das Maximum ist fest im Programm implementiert, da hilft auch keine Änderung an der MTU).

Selbst das Muster der Testdaten kann per Option eingestellt werden:

-p *Muster*

Muster ist ein 16 Bytes großer Wert, der als Hexadezimalzahl anzugeben ist. »-p FF« führt zu Testdaten, die aus einer Folge von 1-Bits bestehen.

```
user@sonne> ping -p xx localhost
ping: patterns must be specified as hex digits.
user@sonne> ping -p ff -c 1 localhost |  head -1
PATTERN: 0xff
```

Optionen zum Routing

Ein ausbleibendes *ECHO_REQUEST* muss nicht den Ausfall des Zielrechners bedeuten. Ebensogut könnte einer der Vermittlerstellen (Router, Gateway,...) streiken. Die Aufzeichnung der Wegstrecke, die ein Paket passierte, kann unter Umständen weiter helfen:

-R

Aufzeichnen der Hin- und Rückroute, die ein ICMP-Paket durchläuft.

```
user@sonne> ping -c 2 -R lkap1073
PING lkap1073.meine-firma.de (191.0.9.104) from 187.0.34.197 : 56(124) bytes of data.
64 bytes from lkap1073.meine-firma.de (191.0.9.104): icmp_seq=1 ttl=127 time=1.35 ms
NOP
RR:     lkcp38.meine-firma.de (187.0.34.197)
        zsnr2-if17-vl10.meine-firma.de (191.0.100.10)
        lkap1073.meine-firma.de (191.0.9.104)
        lkcr2.meine-firma.de (187.0.34.10)
        lkcp38.meine-firma.de (187.0.34.197)

64 bytes from lkap1073.meine-firma.de (191.0.9.104): icmp_seq=2 ttl=127 time=1.35 ms
NOP     (same route)

--- lkap1073.meine-firma.de ping statistics ---
2 packets transmitted, 2 received, 0% loss, time 1002ms
rtt min/avg/max/mdev = 1.350/1.351/1.352/0.001 ms
```

Die Erreichbarkeit aller Rechner eines Netzwerks kann mit einem Broadcast erreicht werden:

-b

»Broadcast-Ping« an alle Rechner eines angegebenen Netzwerks.

Da alle Rechner einunddasselbe Paket erhalten, entstehen Duplikate bei der Antwort, die **ping** durch ein nachgestellten *DUP!* in der Ausgabe kennzeichnet:

```
user@sonne> ping -b -c 2 187.0.34.0
WARNING: pinging broadcast address
PING 187.0.34.0 (187.0.34.0) from 187.0.34.197 : 56(84) bytes of data.
64 bytes from 187.0.34.197: icmp_seq=1 ttl=64 time=0.049 ms
64 bytes from 187.0.34.236: icmp_seq=1 ttl=255 time=0.163 ms (DUP!)
```

```
64 bytes from 187.0.34.206: icmp_seq=1 ttl=255 time=0.211 ms (DUP!)
64 bytes from 187.0.34.196: icmp_seq=1 ttl=255 time=0.213 ms (DUP!)
64 bytes from 187.0.34.181: icmp_seq=1 ttl=255 time=0.220 ms (DUP!)
64 bytes from 187.0.34.174: icmp_seq=1 ttl=255 time=0.243 ms (DUP!)
64 bytes from 187.0.34.133: icmp_seq=1 ttl=255 time=0.245 ms (DUP!)
64 bytes from 187.0.34.159: icmp_seq=1 ttl=255 time=0.247 ms (DUP!)
64 bytes from 187.0.34.134: icmp_seq=1 ttl=255 time=0.368 ms (DUP!)
64 bytes from 187.0.34.167: icmp_seq=1 ttl=255 time=0.371 ms (DUP!)
64 bytes from 187.0.34.164: icmp_seq=1 ttl=255 time=0.373 ms (DUP!)
64 bytes from 187.0.34.11: icmp_seq=1 ttl=255 time=0.432 ms (DUP!)
64 bytes from 187.0.34.10: icmp_seq=1 ttl=255 time=0.435 ms (DUP!)
64 bytes from 187.0.34.194: icmp_seq=1 ttl=255 time=0.471 ms (DUP!)
64 bytes from 187.0.34.132: icmp_seq=1 ttl=255 time=0.494 ms (DUP!)
64 bytes from 187.0.34.130: icmp_seq=1 ttl=255 time=0.496 ms (DUP!)
64 bytes from 187.0.34.7: icmp_seq=1 ttl=255 time=0.523 ms (DUP!)
64 bytes from 187.0.34.135: icmp_seq=1 ttl=255 time=0.525 ms (DUP!)
64 bytes from 187.0.34.190: icmp_seq=1 ttl=255 time=0.527 ms (DUP!)
64 bytes from 187.0.34.20: icmp_seq=1 ttl=255 time=0.812 ms (DUP!)
64 bytes from 187.0.34.131: icmp_seq=1 ttl=255 time=0.953 ms (DUP!)
64 bytes from 187.0.34.152: icmp_seq=1 ttl=60 time=1.51 ms (DUP!)
64 bytes from 187.0.34.153: icmp_seq=1 ttl=60 time=1.73 ms (DUP!)
64 bytes from 187.0.34.17: icmp_seq=1 ttl=32 time=5.39 ms (DUP!)
64 bytes from 187.0.34.197: icmp_seq=2 ttl=64 time=0.046 ms

--- 187.0.34.0 ping statistics ---
2 packets transmitted, 2 received, +23 duplicates, 0% loss, time 1010ms
rtt min/avg/max/mdev = 0.046/0.682/5.393/1.038 ms
```

Das Kommando traceroute/ tracepath

Traceroute oder Ping?

»Ping« in Verbindung mit der Option »-R« erfüllt scheinbar dieselben Aufgaben wie »traceroute«. Jedoch existieren entscheidende Einschränkungen.

Zunächst vermag »Ping« nur maximal 9 Stationen (»Hops«), die ein Paket durchläuft, aufzuzeichen. Für mehr ist einfach kein Platz in einem ICMP-Request-Paket. Sie können das Verhalten leicht nachvollziehen, wenn Sie nur einen Zielrechner wählen, der »weit genug« entfernt ist:

```
user@sonne> ping -R -c 1 www.heise.de
PING www.heise.de (193.99.144.71) from 145.254.78.131 : 56(124) bytes of data.
64 bytes from www.heise.de (193.99.144.71): icmp_seq=1 ttl=245 time=278 ms
RR:     dialin-145-254-078-131.arcor-ip.net (145.254.78.131)
        dre-145-253-16-118.arcor-ip.net (145.253.16.118)
        dre-145-254-6-14.arcor-ip.net (145.254.6.14)
        dre-145-254-6-34.arcor-ip.net (145.254.6.34)
        dre-145-254-17-10.arcor-ip.net (145.254.17.10)
        bln-145-254-18-58.arcor-ip.net (145.254.18.58)
        ams-ix.arcor-ip.net (193.148.15.123)
        pos-8-0.mpr1.ams1.nl.mfnx.net (208.184.231.181)
        pos1-0.mpr2.ams1.nl.mfnx.net (208.184.231.253)

--- www.heise.de ping statistics ---
1 packets transmitted, 1 received, 0% loss, time 0ms
rtt min/avg/max/mdev = 278.643/278.643/278.643/0.000 ms
```

Erst »traceroute« zeigt die fehlenden Zwischenstationen auf:

```
user@sonne> /usr/sbin/traceroute www.heise.de
traceroute to www.heise.de (193.99.144.71), 30 hops max, 40 byte packets
 1  dre-145-253-1-105.arcor-ip.net (145.253.1.105)  158.395 ms  178.291 ms  188.181 ms
 2  dre-145-253-16-97.arcor-ip.net (145.253.16.97)  208.099 ms  228.014 ms  247.940 ms
 3  dre-145-254-6-9.arcor-ip.net (145.254.6.9)  277.870 ms  307.814 ms  347.706 ms
```

```
 4  amd-145-254-16-238.arcor-ip.net (145.254.16.238)  397.616 ms   417.569 ms   447.458 ms
 5  ge2-0.mpr1.ams1.nl.mfnx.net (193.148.15.122)  477.370 ms   507.330 ms   537.225 ms
 6  pos-0-0.mpr2.ams1.nl.mfnx.net (208.184.231.182)  567.160 ms   587.022 ms   616.952 ms
 7  pos2-0.cr2.ams2.nl.mfnx.net (208.184.231.254)  149.019 ms   178.923 ms   198.843 ms
 8  so-1-1-0.cr2.fra1.de.mfnx.net (64.125.31.142)  228.753 ms   258.670 ms   288.616 ms
 9  pos2-0.er1b.fra1.de.mfnx.net (216.200.116.141)  308.503 ms   328.437 ms   358.360 ms
10  plusline-gw2.fra1.above.net (216.200.116.222)  398.298 ms   428.218 ms   458.146 ms
11  c6.f.de.plusline.net (213.83.57.19)  478.057 ms   507.970 ms   537.914 ms
12  c22.f.de.plusline.net (213.83.19.83)  557.839 ms   587.735 ms   617.674 ms
13  www.heise.de (193.99.144.71)  149.010 ms   168.906 ms *
```

Ist der Zielrechner nicht erreichbar, bleibt Ihnen »ping« eine Antwort schuldig. Mit »traceroute« hingegen können Sie mit einiger Sicherheit den Streckenabschnitt identifizieren, der das Problem verursacht.

Und schließlich ermöglicht erst »traceroute« eine Analyse der Laufzeiten der Pakete und die Identifikation eventueller Leistungsengpässe im Netz.

Die Arbeitsweise von Traceroute

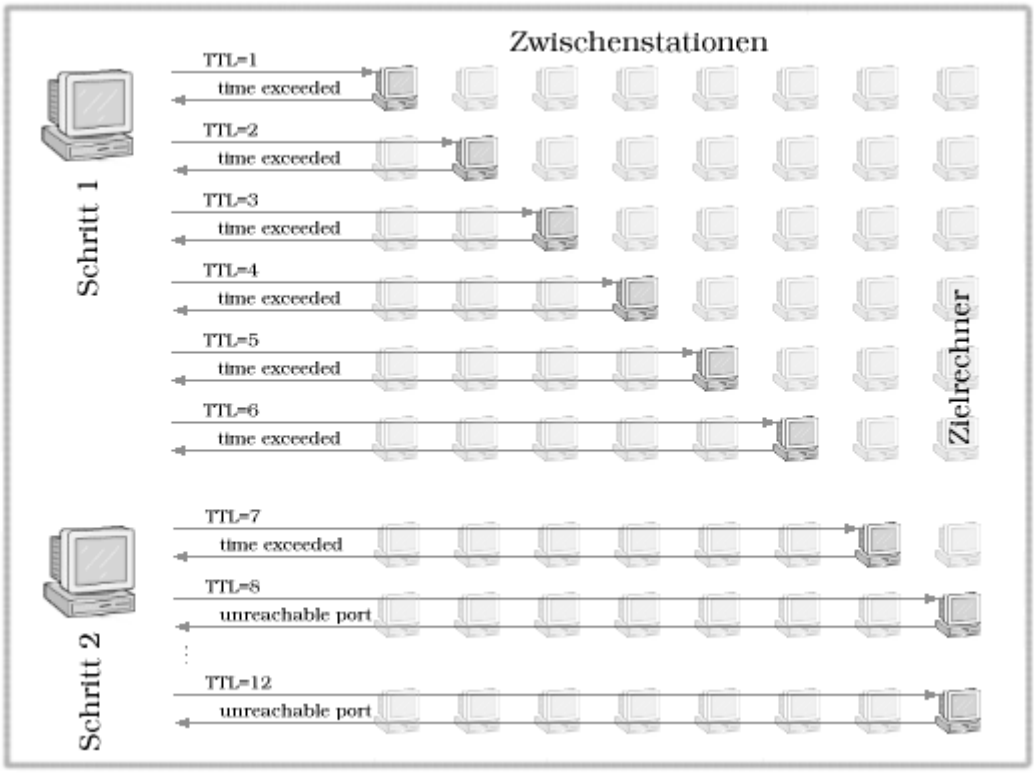


Abbildung 1: Die Arbeitsweise von Traceroute

Zwei Techniken kommen zum Einsatz. Zum einen ist dies die Variation des *Time-to-live*-Feldes der UDP-Testpakete. Erinnern Sie sich an die Aufgabe dieses TTL-Feldes, das verhindern soll, dass fehl geleitete Pakete ewig im Netz kursieren. Jeder Rechner, den ein Paket passiert, inkrementiert hierzu den Wert des Feldes. Sinkt der Wert auf Null, wird der aktuelle Rechner, insofern er nicht das Ziel markiert, das Paket verwerfen und eine Fehlermitteilung an den Absender initiieren (ICMP-Meldung »Time excceded«). »Traceroute« erzwingt eine solche Fehlernachricht von jeder Zwischenstation, die das Paket durchläuft.

Auf dem Zielrechner kontaktiert »Traceroute« den Port 33434. In der Regel wartet niemals ein Programm auf diesem Port, sodass der Rechner den Verbindungswunsch als Fehler interpretiert und seinerseits mir der ICMP-Mitteilung »Unreachable Port« antwortet. Hieran erkennt »Traceroute« das Erreichen des Ziels.

Aus Effizienzgründen sendet »Traceroute« zu einer Zeit gleich mehrere UDP-Pakete aus (im Beispiel aus Abbildung 1 sind es 6). Des Weiteren wird in der Voreinstellung jedes Paket genau drei Mal versandt, um zum Einen eine gewisse Resistenz gegenüber Fehlern zu wahren (bspw. darf ein Router bei Überlastung Pakete verwerfen, ohne entsprechende Fehlerpakete auszusenden) und zum Anderen mehrere Zeitmessungen vorzunehmen (um

»Ausreißer« zu markieren). Fehlversuche kennzeichnet »Traceroute« mit einem Stern (*). Es ist durchaus mölgich, dass »Traceroute« einen Weg zum Ziel findet, obwohl ein Zwischenrechner scheinbar nicht erreichbar ist (nur Sterne in der Ausgabe). Ursachen sind zumeist Fehler in »Traceroute«-Implementierungen mancher Unix-Systeme, die dazu führen, dass die »Time exceeded« Fehlermitteilung den Absender nicht erreichen.

Anmerkung: Manche Netzwerkrouter können so konfiguriert werden, dass sie »Traceroute«-Pakete ignorieren, d.h. sie weiter leiten, ohne das TTL-Feld anzutasten. Deshalb muss die aufgezeichnete Route nicht vollständig sein (eine Option *-R*, um auch solche Router aufzuzeichnen, existiert zwar, ist jedoch in der aktuellen Version nicht implementiert).

Die Optionen

Optionen zur Steuerung des Sendeverhaltens

-q < Anzahl>

In der Voreinstellung versendet »Traceroute« je drei UDP-Pakete je *Time-to-live*-Wert. Mit der Option **-q** ka jeder anderer Wert vorgegeben werden.

```
user@sonne> /usr/sbin/traceroute -q 4 www.heise.de | head -4
traceroute to www.heise.de (193.99.144.71), 30 hops max, 40 byte packets
 1  dre-145-253-1-103.arcor-ip.net (145.253.1.103)  119.121 ms   169.047 ms   188.981 ms   208.920 ms
 2  dre-145-253-16-98.arcor-ip.net (145.253.16.98)  228.861 ms   248.794 ms   268.735 ms   288.668 ms
 3  dre-145-254-6-13.arcor-ip.net (145.254.6.13)  318.605 ms   348.546 ms   368.478 ms   398.416 ms
```

Anmerkung: In einigen (älteren) Programmversionen stürzt »Traceroute« bei hohen Werten mit einen Segmentation fault (Speicherschutzverletzung) ab oder es verweigert das Setzen des TTL-Wertes. Eine Aktualisierung auf die neueste Version behebt die Ursachen.

-w < Anzahl>

»Traceroute« sendet in der Voreinstellung sechs Pakete quasi parallel mit jeweils verändertem Time-to-live- Wert aus (im ersten Schritt mit TTL=1..6, im zweiten Schritt mit TTL=7..12 usw.). Trifft die letzte Antwort e wird die nächste Welle von Paketen ins Rennen geworfen. Mit der Option **-w** kann eine Zeitspanne fest gel werden, die zwischen dem Aussenden zweier Sondierungspakete zu warten ist (Angabe in Sekunden).

-N < Anzahl>

Die Voreinstellung von sechs gleichzeitig zu sendenden Paketen kann hiermit verändert werden. Beachten S dass sehr hohe Werte zwar die Routenfinding durchaus beschleunigen können, gleichzeitig jedoch den Netzwerkverkehr extrem erhöhen. Spätestens wenn Fehlermeldungen der Art »ICMP rate throttling« verme austreten, sollten Sie ein solches Vorgehen vermeiden, um nicht Auml;rger mit Ihren Netzwerkadministrato heraufzubeschwören.

-f < Start-TTL>

Wenn Sie die Route zu einem weit entfernten Ziel verfolgen und genau wissen, dass die ersten Stationen zuverlässig arbeiten, dann können Sie deren Test auslassen, indem Sie mit einem höheren Startwert für das »Time-to-live«-Feld beginnen.

-F

Im IP-Protokollkopf wird das *Don't Fragment*-Bit gesetzt. Router dürfen ein solches Datenpaket nicht weiter splitten, auch wenn es für das folgende Teilnetz zu groß sein sollte. Sie sind gezwungen, dieses zu verwerfe und eine Fehlermeldung an den Absender zu verschicken.

-m < max_hops>

Der maximale Wert für TTL, mit dem Pakete ausgesendet werden, liegt bei 30. Die Option **-m** gestattet das Setzen dieser Grenze. Schon der Wert von 30 erscheint recht hoch; es ist unwahrscheinlich, dass zwischen Rechnern mehr Zwischenstationen liegen, es sei denn, irgendwo im Routing liegt eine Fehlkonfiguration vo

```
user@sonne> /usr/sbin/traceroute -n -m 3 www.heise.de
traceroute to www.heise.de (193.99.144.71), 3 hops max, 40 byte packets
 1  145.253.1.105  119.031 ms   138.962 ms   168.899 ms
 2  145.253.16.97  198.829 ms   228.763 ms   258.696 ms
 3  145.254.6.9  288.624 ms   438.560 ms   448.500 ms
```

Optionen bezüglich der gewählten Route

-I <Interface>

Verfügt ein Rechner über mehrere Netzwerk-Schnittstellen (mehrere Netzwerkkarten, Modem, ISDN-Karte.. so muss die zu verwendende Schnittstelle angegeben werden, insofern es sich nicht um die erste Schnittste handelt (Vergleichen Sie die Reihenfolge der Ausgabe von ifconfig).

-S <Quell-IP>

Für einen Rechner mit mehreren IP-Adressen, kann mit der Option **-S** die als Absender zu verwendende Adresse angegeben werden.

-g <Gateway>

Fügt dem ausgehenden Paket die »IP source routing Option« hinzu, womit eine Route durch das angebene Gateway zu wählen ist. Etliche Route ignorieren jedoch aus Sichereitsgründen diese Option.

-t <Type of Service>

Zahlreiche Implementierungen des IP-Protokoll-Stacks ignorieren diese Option, die das Routing derart beeinflussen soll, dass Routen mit bestimmten Eigenschaften bevorzugt gewählt werden. Mögliche Werte si (Voreinstellung), 1 (Route mit den geringsten Kosten), 2 (Route mit höchster Zuverlässigkeit), 4 (Route mit höchstem Durchsatz), 8 (schnellste Route) und 16 (Route mit höchster Sicherheit.
Der praktische Nutzen dieser Information steigt und fällt mit den den Routern zur Verfügung stehenden Informationen.
Das Setzen der Option ist nur Root möglich.

Weitere Optionen

-n

Verhindert die Namensauflösung von IP-Adressen, d.h. die Zwischenstationen werden stets mit ihrer IP-Adr angegeben.

```
user@sonne> /usr/sbin/traceroute -n www.heise.de
traceroute to www.heise.de (193.99.144.71), 30 hops max, 40 byte packets
 1  145.253.1.105  189.075 ms   198.996 ms   218.929 ms
 2  145.253.16.97  228.873 ms   248.803 ms   258.757 ms
 3  145.254.6.9  288.675 ms   308.602 ms   338.519 ms
 4  145.254.16.238  398.466 ms   428.395 ms   448.348 ms
 5  193.148.15.122  488.293 ms   508.211 ms   528.148 ms
 6  208.184.231.182  558.069 ms   598.005 ms   627.933 ms
 7  208.184.231.254  429.692 ms   459.618 ms   479.550 ms
 8  64.125.31.142  456.574 ms   486.505 ms   516.440 ms
 9  216.200.116.141  469.589 ms   499.526 ms   529.455 ms
10  216.200.116.222  439.595 ms   469.529 ms   509.456 ms
11  213.83.57.19  457.729 ms   477.657 ms   507.585 ms
12  213.83.19.83  429.704 ms   459.633 ms   489.57 ms
13  193.99.144.71  449.649 ms   469.590 ms *
```

-4

Erzwingt die Verwendung von IP-Adressen nach Version 4 (ipv4).

-6

Paketlänge

Der zwischen 1 und 65536 liegende Wert (Voreinstellung 40 Bytes) kann hinter der Angabe des Zielrechner der Kommandozeile folgen. In Verbindung mit der Option **-F** besteht so eine Möglichkeit, die Maximale Transfereinheit (MTU) zu ermitteln.

Anmerkung: Die Optionen *-4* bzw. *-6* erübrigen sich, wenn der Zielrechner als IP-Adresse angegeben wird. In dem Fall entscheidet »Traceroute« anhand des Adressformats, nach welchem Mechanismus vorzugehen ist.

tracepath

Für Fälle, in denen es einzig um den Test des Routenverlaufs geht, ist »Traceroute« mit der Fülle seiner Optionen quasi überqualifiziert, zumal etliche Optionen die Rechte des Administrators bedingen. Oft genügt **tracepath**, das keine Optionen kennt und vom »jedem« Benutzer verwendet werden darf. Neben der Angabe der Zieladresse kann optional der zu verwendende Port vorgegeben werden. Ohne diesen wählt »Tracepath« zufällig einen Port aus einem definierten Pool (i.d.R.) ungültiger Portnummern aus.

```
user@sonne> /usr/sbin/tracepath www.heise.de
 1?: [LOCALHOST]      pmtu 1500
 1:  dre-145-253-1-105.arcor-ip.net (145.253.1.105)       658.744ms
 2:  dre-145-253-16-97.arcor-ip.net (145.253.16.97)       669.516ms
 3:  dre-145-254-6-9.arcor-ip.net (145.254.6.9)           667.342ms
 4:  amd-145-254-16-238.arcor-ip.net (145.254.16.238)     asymm  7 689.634ms
 5:  ge2-0.mpr1.ams1.nl.mfnx.net (193.148.15.122)         asymm  8 695.013ms
 6:  pos-0-0.mpr2.ams1.nl.mfnx.net (208.184.231.182)      asymm  9 689.666ms
 7:  pos2-0.cr2.ams2.nl.mfnx.net (208.184.231.254)        asymm 10 689.717ms
 8:  so-1-1-0.cr2.fra1.de.mfnx.net (64.125.31.142)        asymm 10 688.892ms
 9:  pos2-0.er1b.fra1.de.mfnx.net (216.200.116.141)       asymm 10 690.307ms
10:  plusline-gw2.fra1.above.net (216.200.116.222)        asymm 11 689.343ms
11:  c6.f.de.plusline.net (213.83.57.19)                  asymm  9 689.566ms
12:  c22.f.de.plusline.net (213.83.19.83)                 asymm 10 689.310ms
13:  www.heise.de (193.99.144.71)                         asymm 11 689.836ms reached
Resume: pmtu 1500 hops 13 back 11
```

Das Kommando arp

Der ARP-Cache des Kernels

Während im Internet Pakete anhand ihrer IP-Adresse vermittelt werden, erfolgt in lokalen Netzen die Adressierung mittels Hardwareadressen. So besitzt jede Ethernetnetzwerkkarte eine (weltweit) eindeutige 48 Bit lange Adresse. Das Ermitteln der Hardwareadresse des Rechners zu einer gegebenen IP-Adresse erfolgt durch das Address Resolution Protocol.

Einmal ermittelte Zuordnungen speichert der Linuxkernel in einem internen Zwischenspeicher (»arp-Cache«), um wiederholte Anfragen ohne teure Rechercheoperationen bedienen zu können. In der üblichen Konfiguration stehen 256 Einträge im Cache zur Verfügung. Lange Zeit nicht verwendete Adressen werden nach einer maximalen Verweildauer oder bei Bedarf (voller Cache) entfernt.

Mit dem Kommando **arp** steht dem Administrator ein Werkzeug zur manuellen Inspektion und Manipulation des Arp-Caches zur Verfügung.

Die Optionen

Optionen zum Betrachten des Arp-Caches

Für den Fall, dass ein Rechner via Netzwerk nicht erreichbar ist, kann ein Blick in den Arp-Cache helfen, um Probleme auf Hardwareebene auszuschließen. Vorhandene Einträge (die nicht manuell eingefügt wurden) sind

sichere Anzeichen für eine funktionierende Anbindung.

```
user@sonne> arp
Address              HWtype  HWaddress           Flags Mask         Iface
pluto.galaxsis.de    ether   00:A0:C9:FC:1F:45   C                  eth0
mars.galaxis.de      ether   00:40:05:A1:A6:C4   C                  eth0
erde.galaxis.de      ether   00:B0:15:AA:AA:11   C                  eth0
```

Eine an die arp-Implementierung auf BSD-Systemen angelehnte Form der Ausgabe vermag *arp* ebenso zu erzeugen.

-a

Anzeige gemäß des Formats der BSD-Implementierung.

```
user@sonne> arp -a
pluto.galaxsis.de (192.168.239.1) at 00:A0:C9:FC:1F:45 [ether] on eth0
mars.galaxsis.de (192.168.239.2) at 00:40:05:A1:A6:C4 [ether] on eth0
erde.galaxsis.de (192.168.109.2) at 00:B0:15:AA:AA:11 [ether] on eth0
```

Auch zur Ausgabe erweiterter Informationen und zur Anzeige numerischer Adressen anstatt Rechnernamen stehen Optionen zur Verfügung.

-n

Adressen werden in numerischer Form wiedergegeben.

-v

Ausführlichere Ausgaben.

```
user@sonne> arp -vn
Address              HWtype  HWaddress           Flags Mask         Iface
192.168.239.5        ether   00:02:B3:2E:65:AE   C                  eth0
192.168.239.1        ether   00:A0:C9:FC:1F:45   C                  eth0
192.168.239.2        ether   00:40:05:A1:A6:C4   C                  eth0
192.168.239.12       ether   00:D0:B7:4D:D7:E1   C                  eth0
Entries: 4      Skipped: 0      Found: 4
```

-H Typ

Beschränkung auf Einträge des angegebenen Schnittstellentyps. Voreinstellung ist **ether** (Ethernet). Weiter mögliche Werte sind **arcnet** (ARCnet), **ax25** (AMPR AX.25), **dlci** (Frame Relay DLCI), **fddi** (Fiber Distribut Data Interface), **hippi** (HIPPI), **irda** (IrLAP), **netrom** (AMPR NET/ROM), **pronet** (PROnet), **strip** (Metricor Starmode IP), **tr** (16/4 Mbps Token Ring) und **x25** (generic X.25).

-i < Schnittstelle>

Die Anzeige beschränkt sich auf Einträge der angegebenen Netzwerkschnittstelle (nur bei Rechnern mit mehreren Schnittstellenkarten sinnvoll).

Das Kommando netstat

Das Kommando route

Das Kommando ifconfig

Das Kommando bing

Das Kommando nmap

Internet Service Dämonen

Übersicht

Damit ein Client einen entsprechenden Server findet, benötigt er neben Kenntnis der IP-Adresse des Serverrechners auch noch die Portnummer, an der der Dienst wartet.

In den ersten Unix-Systemen mit TCP/IP-Unterstützung wurden bereits während des Systemstarts sämtliche Serverprozesse aktiviert, die ihre Ports initialisierten und auf eintreffende Anforderungen warteten. Traf ein Verbindungswunsch ein, erzeugte ein Server einen Kindprozess, vererbte diesem die geöffnete Verbindung, schloss seinerseits die Verbindung und begab sich selbst erneut in den Wartezustand, um eintreffende Anfragen entgegenzunehmen.

Bald erkannte man, dass die meisten Serverprozesse vollkommen umsonst gestartet wurden, sie wurden im Laufe der Aktivität des Systems nicht oder nur selten in Anspruch genommen. Mit den von den warteten Prozessen in Anspruch genommenen CPU-Zeiten konnte man noch gut leben, denn da sie nichts zu erledigen hatten, erzeugten sie auch nur geringe Rechenlast, aber der ständig reservierte Hauptspeicher, zumal in jenen Zeiten die heutigen RAM-Dimensionen ins Reich der Fantasie gehörten, entpuppte sich bald als Schwachpunkt.

Die Lösung kam, wie so viele Anreize aus jenen Tagen, aus Berkeley und wurde mit dem **inetd** als erstem »Superserver« mit 4.3BSD veröfentlicht.

Anstelle der Aktivierung sämtlicher Netzwerkdienste wurde nun nur noch der Superserver beim Start des Systems zum Leben erweckt. Dieser entnahm alle zu eröffnenden Portnummern einer Konfigurationsdatei und überwachte diese auf eintreffende Verbindungsanforderungen. Lag irgendwo eine Anfrage an, startete der Superserver den entsprechenden Serverdienst und vermachte ihm die bereits offene Verbindung.

Als Konsequenz hieraus ist die meiste Zeit über stets nur ein Server aktiv und alle weiteren können bei Bedarf nachgeladen und nach Erfüllung der Anforderung auch wieder beendet werden.

Der **inetd** ist der Standard-Superserver für Linux, jedoch lässt er in Sachen Zugriffssteuerung viele Wünsche offen. So hat sich der **xinitd** in vergangenen Jahren zu einer ernsthaften Alternative entwickelt.

Der Dämon inetd

Der **inetd** ist der mit Abstand beliebteste Superdämon. Er ist in der Lage sowohl TCP- als auch UDP-basierte und in aktuellen Versionen sogar RPC-Dienste zu starten. Dazu überwacht er die Ports (oft wird in diesem Zusammenhang auch von Internet Sockets gesprochen) der ihm anvertrauten Server und entscheidet anhand der Portnummer, an der eine Anforderung eintrifft, welcher Netzwerkdienst zu starten ist.

Nicht alle Dienste, die der Rechner im Netzwerk anbieten soll, werden von einem Superserver verwaltet. Einige hochverfügbare Server wird man schon während des Systemstarts aktivieren, man denke nur einen http-Dämonen auf einem frequentierten Webserver. Für welche Dienste sich der **inetd** letztlich verantwortlich zeichnet, muss ihm deshalb in seiner Konfigurationsdatei **/ etc/ inetd.conf** mitgeteilt werden.

Es gibt Dienste, die sind in ihrer Art so einfach, dass es vergebene Mühe wäre, sie in ein eigenes Programm zu packen. Der **inetd** verfügt deswegen über einige interne Dienste, deren Anforderung er höchst persönlich erfüllt:

chargen

Der »Zeichengenerator«. Trifft ein Verbindungswunsch für diesen Dienst ein, wird mit einem ununterbroche Zeichenstrom geantwortet, solange, bis der Client die Verbindung beendet. Der Service kann zur Performan Messung eingesetzt werden.

daytime

Gibt die Rechnerzeit in einem für den Mensch verständlichen Format wieder

discard

Der Dienst basiert auf dem Initial Connection Protocol und anwortet bei eintreffendem Verbindungswunsch dem Aufbau einer Verbindung in die Gegenrichtung. Anschließend wartet discard auf eine Anwort und leitet diese nach /dev/null. Ist die Anwort »verarbeitet«, beendet sich der Prozess. Der Dienst wird zu Testzwecke eingesetzt.

echo

Der Dienst sendet die empfangenen Daten unverändert an den Absender zurück.

time

Gibt die Rechnerzeit in einem maschinenlesbaren Format wieder

»time« und »daytime« werden bei einem Zeitserver benötigt, die anderen Dienste sollten nur temporär für Testzwecke geöffnet werden, da sie von außen zum Erzeugen unnötiger Rechenlast missbraucht werden können.

Die Datei /etc/inetd.conf

Alle Dienste, die der **inetd** starten soll, müssen in dessen Konfigurationsdatei /etc/inetd.conf aufgeführt sein. Beginnt eine Zeile mit dem Doppelkreuz, so handelt es sich um einen Kommentar, alle anderen Zeilen bestehen aus 6 Feldern:

Dienstbezeichnung

Name des Serverdienstes, so wie er in der Datei /etc/services eingetragen ist. Im Falle eines RPC-Dienstes zusätzlich dessen Versionsnummer angegeben werden. Der RPC-Dienstname steht in der Datei /etc/rpc, ein gültiger Eintrag lautet dann <RPC-Dienst>/<Version>.

Sockettyp

Die Art des Sockets muss hier angegeben werden. Als Schlüsselwörter sind zulässig: **stream**, **dgram**, **raw** **rdm** und **seqpacket**. **stream** ist dabei ein Socket, in dem ein Datenfluss (»streaming«) eintrifft, so wie sie Transportprotokoll TCP implementiert. **dgram** ist der »Telegrammtyp« und korrespondiert mit dem UDP-Protokoll. **raw** sind Rohdaten, also Daten, die nicht in einemTransportprotokoll verpackt sind; sie werden d an die Anwendung weitergereicht. **rdm** (reliably delivered message) sind »sicher verteilte« Daten, so dass den oberen Schichten des TCP-Protokollstacks von ihrer Unversehrtheit ausgegangen werden kann. **seqpac** ist eine Sequenz von Paketen.

Protokoll

Ist der Name des Protokolls, über den der Dienst arbeitet, in den meisten Fällen wird es sich um tcp oder u handeln. Im Falle eines RPC-Dienste steht hier z.B. »rpc/tcp« oder »rpc/udp«.

[no]wait

Eines der Schlüsselworte ist immer anzugeben, wobei die Angabe sich einzig auf das UDP-Protokoll auswirk **wait** bewirkt, dass der **inetd** nach dem Verbindungsaufbau neue Anforderungen erst entgegennimmt, wen der UDP-Dienst seine Arbeit beendet hat. Mit **nowait** wird unverzüglich auf neue Anforderungen gewartet.

Benutzerkennung

Name des Benutzers, in dessen Auftrag der Dienst gestartet wird.

Dienstname

Vollständiger Programmname des Dienstes (inklusive Pfad und Optionen).

Eine Datei »/etc/inetd.conf« könnte dann wir folgt aussehen (Auszüge):

```
user@sonne> cat / etc/ inetd.conf
# <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
# chargen       stream  tcp     nowait  root    internal
# chargen       dgram   udp     wait    root    internal
time    stream  tcp     nowait  root    internal
time    dgram   udp     wait    root    internal
#
# These are standard services.
#
# ftp   stream  tcp     nowait  root    /usr/sbin/tcpd  wu.ftpd -a
ftp     stream  tcp     nowait root     /usr/sbin/tcpd  in.ftpd
#
# If you want telnetd not to "keep-alives" (e.g. if it runs over a ISDN
# uplink), add "-n". See 'man telnetd' for more deatails.
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
nntp    stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/leafnode
smtp    stream  tcp     nowait  root    /usr/sbin/sendmail  sendmail -bs
# printer       stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/bin/lpd -i
# Shell, login, exec and talk are BSD protocols.
# The option "-h" permits ``.rhosts'' files for the superuser. Please look at
# man-page of rlogind and rshd to see more configuration possibilities about
# .rhosts files.
# shell  stream  tcp nowait  root    /usr/sbin/tcpd  in.rshd -L
# shell  stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -aL
#
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
# login stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind -a
# exec  stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd

finger  stream  tcp     nowait  nobody  /usr/sbin/tcpd in.fingerd -w
# systat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
# netstat       stream  tcp     nowait  root    /usr/sbin/tcpd  /bin/netstat
# identd is now started at boot time, the following is not longer necessary.
# ident stream  tcp     wait    nobody  /usr/sbin/in.identd  in.identd -w -e

# swat is the Samba Web Administration Tool
swat    stream  tcp     nowait.400      root    /usr/sbin/swat  swat
```

Der einigen Diensten vorgeschaltete TCP-Wrapper »/usr/sbin/tcpd« implementiert eine dedizierte Zugangskontrolle für den jeweiligen Netzwerkdienst. Er ist eine relevante Komponente der Netzwerksicherheit und wird im Kapitel Netzwerk-Sicherheit TCP-Wrapper vorgestellt.

Der Dämon xinetd

Im vergangenen Abschnitt lernten Sie den **inetd** kennen. Die Bemerkungen zur nichtvorhandenen Zugangskontrolle sollten Ihnen ebenso wenig entgangen sein, wie der Kompromiss des TCP-Wrappers, der die Sicherheitsmängel des **inetd** teilweise beheben kann. Dennoch kann die Vorgehensweise des »alles ist erlaubt, solange es nicht explizit verboten wurde« bei halbherziger Konfiguration schnell zu unbemerkten Sicherheitslöchern führen.

Der **xinetd** ist ein vollwertiger Ersatz für den **inetd**. Das »x« deutet hier nicht etwa ein X-Window-Programm an, sondern steht für »extended« (erweitert). Er implementiert dieselben internen Dienste wie der »inetd« (chargen, daytime, discard, echo, time). Des Weiteren ermöglicht der **xinetd** die Protokollierung aller Zugriffe.

Die Datei /etc/xinetd.conf

Konfiguriert wird der **xinetd** mittels der Datei **/ etc/ xinetd.conf**. Er kennt zwei Typen von Einträgen. Zum einen handelt es sich um den optionalen »default«-Eintrag, der auf alle anderen Einträge angewandt wird, insofern diese die Optionen nicht selbst definieren. Dieser default-Eintrag besitzt folgende Struktur:

```
default
  {
    < Schlüssel> < Operator> < Parameter> < Parameter> ...
...
  }
```

Die weiteren Einträge betreffen die einzelnen Dienste:

```
service < Dienstbezeichnung>
  {
    < Schlüssel> < Operator> < Parameter> < Parameter> ...
...
  }
```

Als Operatoren sind zulässig: die normale Zuweisung mit »**=** «, das Hinzufügen eines weiteren Schlüssels mit »**+ =** « und das Entfernen eines Schlüssels mit »**-=** «. Die Verwendung der Operatoren »+ = « und »-= « macht vor allem in Verbindung mit dem »default«-Eintrag Sinn.

Von den nachfolgend beschriebenen Schlüsseln sind im Falle des default-Eintrags nicht alle sinnvoll. Erlaubt sind hier »log_type«, »log_on_success«, »log_on_failure«, »only_from«, »no_access«, »passenv«, »instances« und »disabled«. »disabled« kann nur im »default«-Eintrag stehen und sperrt den Zugang zu den angegebenen Diensten.

id

Jeder Dienst muss eindeutig identifiziert werden können, die Angabe ist für Mulit-Protokoll-Dienste notwend Fehlt sie, so wird der Name des Dienstes (»Dienstbezeichnung«) als ID angenommen.

type

Kombination aus »RPC« (RPC-Dienst), »INTERNAL« (interner Dienst) oder »UNLISTED« (Dienst, der nicht i den Dateien »/etc/rpc« oder »/etc/services« eingetragen ist).

flags

Von den möglichen Einträgen ist »IDONLY« der Interessanteste. Eine Anforderung wird dann nur zugelasse wenn der entfernte Benutzer identifiziert werden kann.

socket_type

Die Typen sind »stream«, »dgram«, »raw« und »seqpacket« und besitzen dieselbe Bedeutung wie beim **in** beschrieben wurde.

protocol

Name des Protokolls, über das der Dienst arbeitet. Fehlt die Angabe, wird das »übliche« Protokoll des Dien angenommen.

wait

Steht hier »yes«, wartet der **xinitd** nach einem Verbindungsaufbau auf das Ende des Servers, bevor er neu Anforderungen entgegen nimmt. Mit »no«, startet er ggf. weitere Serverprozesse.

user

Die Nutzerkennung, unter der der Serverprozess gestartet wird.

group

Die Gruppenkennung, unter der der Serverprozess gestartet wird.

instances

Maximal mögliche Anzahl Prozesse, die den Dienst zu einer Zeit ausführen dürfen. Neben der Zahl darf aucl »UNLIMITED« stehen.

nice

Priorität des Serverprozesses.

server

Programmname des Servers (inklusive Pfad)

server_args

Argumente des Serverprogrammes

only_from

Liste von Rechnern, von denen der Zugang erlaubt ist. Die Angabe kann eine IP-Adresse, ein Rechnername eine Netzwerkadresse oder ein Adressbereich sein (192.168.100.12/100 erlaubt den Zugriff von den Rechne 192.168.100.12 bis 192.168.100.100)

no_access

Rechner, denen der Zugriff verwehrt wird. Die Angaben erfolgen wie unter »only_from«, wobei bei Mehrfachnennung die »bessere« Übereinstimmung gilt (wurde der Zugriff den Rechern eines Netzwerks erl und ist nun ein konkreter Rechner aus jedem Netzwerk in der verbotenen Liste enthalten, so wird ihm der Zugang verwehrt).

access_times

Tageszeit, zu der der Zugriff erlaubt ist

log_type

Wo und wie soll protokolliert werden? Mit »SYSLOG Herkunft [Level]« wird die Protokollierung an den syslo weitergereicht; mit »FILE Datei [soft_limit [hardlimit]]« erfolgt sie in der angegebenen Datei. Die beiden Li sind die Schranken, wie größ eine Datei maximal werden kann, somit wird ein Volllaufen des Dateisystems ausgeschlossen.

log_on_success

Was wird protokolliert, falls der Server erfolgreich startet? Mögliche Einträge sind: Prozessnummer des Serv »PID«, Rechnername, von dem die Anforderung kam »HOST«, Benutzernummer, von dem die Anforderung kam »USERID«, der Exit-Status des Servers »EXIT« und die Dauer des Serverlaufs »DURATION«.

log_on_failure

Was wird protokolliert, falls der Server nicht gestartet werden konnte? Neben »HOST« und »USERID« (wie oben) sind möglich »ATTEMPT«, das die Tatsache des fehlgeschlagenen Starts notiert und »RECORD«, das weiterführende Informationen (sofern ermittelbar) vom entfernten Aufruf protokolliert.

rpc_version

Versionsnummer des RPC-Dienstes

env

Hier lassen sich zu einem Server zusätzliche Umgebungsvariablen angeben

passenv

port

Portnummer, an der der Server wartet, diese muss mit dem Eintrag in der Datei /etc/services (soweit vorhanden) übereinstimmen

redirect

Eine Anforderung wird an den angegebenen Rechner weitergeleitet. Dies ist nur bei TCP-Diensten möglich

bind

Bindet einen Dienst an ein spezielles Device. So kann z.B. bei einem Rechner mit zwei Netzwerkkarten siche gestellt werden, dass der Dienst nur über die eine Karte zugänglich ist

banner

Enthält einen Dateinamen, deren Inhalt auf dem zugreifenden Rechner angezeigt wird, falls ihm der Zugang verwehrt wird.

Bevor wir uns den Beispielen zuwenden, sind noch einige Anwendungen zu den vom **xinetd** akzeptierten Signalen notwendig. Bislang konnten Sie nahezu jeden Prozess mit dem Signal SIGHUP (1) zum erneuten Einlesen seiner Konfigurationsdateien bewegen. Im Falle des **xinetd** erreichen Sie damit allerdings nur, dass dieser sich mit einem Speicherabzug verabschiedet. Aus Sicherheitsgründen reagiert der **xinetd** auf einige Signale anders als gewohnt:

SIGUSR1 (10)

Einlesen der Konfigurationsdatei. Aktive Dienste bleiben aktiv.

SIGUSR2 (12)

Einlesen der Konfigurationsdatei. Aktive Dienste werden sofort beendet.

SIGQUIT (3)

Programmende, ohne laufende Dienste zu beenden

SIGTERM (15)

Programmende, laufende Dienste werden zuvor beendet

SIGHUP (1)

Anlegen eines Speicherauszugs und Programmende

SIGIO (29)

Interne Konsitenzprüfung des xinetd

Beispiel 1: Mit dem default-Eintrag spezifizieren wir einige Voreinstellungen, die für alle Dienste gelten, in denen sie nicht explizit überschrieben werden.

```
default
  {
   log_type      = FILE /var/log/xinetd.log
   log_on_success = HOST
   log_on_failure = HOST USERID
   instance      = 5
   disabled      = finger
  }
```

Beispiel 2: »telnet« soll nur von Rechnern des Netzwerkes 192.168.100 möglich sein, wobei die Rechner mit den Endnummern 56-192 ausgenommen werden sollen. Alle fehlgeschlagenen Kontaktversuche sollen über den Syslog-Mechanismus protokolliert werden, wobei die Herkunft den Sicherheitdiensten (auth) zugeordnet werden soll und die Meldung das Level »Warnung« erhält:

```
service telnet
  {
   socket_type = stream
   protocol    = tcp
   wait        = no
   user        = root
   server      = /usr/sbin/in.telnetd
   only_from   = 192.168.100.0
   no_access   = 192.158.100.56/192
   flags       = IDONLY
   log_on_failure += RECORD
   log_type    = SYSLOG auth warn
  }
```

Beispiel 3: »ftp« soll nur in den Nachtstunden erlaubt sein. Gleichzeitig werden 4 Zugriffe zugelassen.

```
service ftp
  {
   socket_type = stream
   wait        = no
   user        = root
   server      = /usr/sbin/wu.ftpd
   server_flags = -a
   log_on_success += DURATION
   access_times = 20:00-06:00
   instance    = 4
  }
```

Beispiel 4: Von den internen Diensten soll »time« sowohl über »udp« als auch über »tcp« bereitgestellt werden:

```
service time
  {
   id          = time_dgram
   socket_type = dgram
   wait        = no
   user        = root
  }

service time
  {
   id          = time_stream
   socket_type = stream
   wait        = no
   user        = root
  }
```

Beispiel 5: Zuletzt noch ein Beispiel zu einem RPC-Dienst:

```
service rstatd
  {
   type        = RPC
   socket_type = dgram
   wait        = yes
   user        = root
   server      = /usr/etc/rpc.rstatd
   rpc_version = 2-4
   env         = LD_LIBRARY_PATH=/etc/securelib
  }
```

Remote Procedure Call

Übersicht

Der Remote Procedure Call (**RPC**) ist ein Protokoll, das die Implementierung verteilter Anwendungen - also Netzwerkdienste - vereinfachen soll. Die dahinter steckende Idee ist, dass ein Programm eine Funktion eines Programms, das auf einem anderen Rechner läuft, nutzen kann, ohne sich um die zu Grunde liegenden Netzwerkdetails kümmern zu müssen. Der RPC arbeitet nach dem Client-Server-Modell.

Ein RPC-Aufruf arbeitet in den meisten Fällen **synchron**. Das lokale Programm, der Client, sendet eine Anforderung an das entfernte Programm, den Server, und unterbricht seine Arbeit bis zum Eintreffen der Antwort. In Verbindung mit **Threads** ist allerdings eine **asynchrone** Realisierung eines entfernten Funktionsaufrufs möglich.

Die Programmierung eines Programms, das RPC-Aufrufe verwendet, gestaltet sich recht einfach. In dem der Sprache C sehr ähnlichen Programmcode wird eine so genannte **Stub-Routine** verwendet, die als Platzhalter für den kompletten Code zur Realisierung des Netzwerkzugriffs dient. Später wird mit Hilfe des Programms **rpcgen** aus der Datei ein vollständiges C-Programm erzeugt.

RPC-Standards

Ursprünglich wurde das Protokoll von der Firma Xerox entwickelt. Heute existieren eine Reihe von Modellen und Implementierungen von Remote Procedure Calls, u.a.:

ONC RPC

Eine der ersten kommerziell verfügbaren Implementierungen wurde durch die Firma Sun verwirklicht. Das zunächst als SunRPC bezeichnete Modell liegt heute in zwei Implementierungen vor, ONC RPC (Open Network Computing; u.a. von Linux verwendet) und der hauptsächlich mit Solaris verbreitete TI RPC (transport independent). Während erstgenannter RPC einzig die Protokolle TCP und UDP unterstützt, kann der TI RPC auf einigen weiteren Transportprotokollen aufgesetzt werden.

Die Programmiersprache des ONC RPC ist die Remote Procedure Call Language »**RPCL**«. Des Weiteren werden drei Authentifzierungs-Schemata unterstützt: keine Authentifizierung (Voreinstellung), die Rechteprüfung über UID/GID und SecureRPC, eine Erweiterung, die auf DES-verschlüsselten Zeitstempeln basiert.

DCE RPC

Der RPC des Distributed Computing Environment ist weit weniger verbreitet als der ONC RPC und unter Linux ist keine Implementierung vorhanden. Der Unterschied zum ONC RPC liegt vor allem in der Aufrufsemantik.

DCE RPC - Anwendungen werden mit Hilfe der Interface Definition Language »**IDL**« programmiert. Die gebräuchlichste Authentifizierungsmethode ist Kerberos.

ISO RPC

Der Versuch der International Organization for Standardization die existierenden Modelle mit einem verbindlichen Standard unter einen Hut zu bringen, scheiterte kläglich. Bis heute existiert keine nennenswerte kommerzielle Implementierung dieses Modells.

Neben der Standardisierung der RPC-Interaktion und -Kommunikation definierte die ISO die Interface Definition Notation »**IDN**«, eine Sprache zur Erzeugung von RPC-Anwendungen.

Probleme mit RPCs

Da es sich nicht um einen lokalen Funktionsaufruf handelt, ergeben sich mit RPC's eine Reihe von Problemen, deren wichtigste kurz vorgestellt werden sollen:

Zunächst stellt sich die Frage: »Welcher Rechner bietet einen entsprechenden Dienst an?«

Unter Linux erfordern viele auf RPC basierende Dienste die konkrete Angabe des Servers. Andere Dienste verwenden Konfigurationsdateien, in denen die entsprechenden Server eingetragen sind. Eine weitere Möglichkeit ist die Verwendung eines Broadcast-Protokolls, das versucht, einen Server zu finden. Ein verbreitetes Protokoll ist z.B. das Ressource Location Protocol »RLP«.

Welches Transportprotokoll (TCP/ UDP/ andere) soll verwendet werden?

Hinter der Frage steht die Problematik der Sicherheit und Performance eines RPC-Aufrufs. Eine TCP-basierte Übertragung stellt die Unversehrheit der übertragenen Daten sicher, d.h. dass der Client sich nicht um die Sicherung des Datentransports zu kümmern hat. Da die TCP-Kommunikation synchron abläuft, arbeitet sie in vielen Fällen wesentlich langsamer als das paketorientierte UDP-Protokoll.

RPC-Dienste mit Anfragecharakter (z.B. NIS) werden daher meist über UDP abgewickelt. TCP wird bei intensivem Datentransfer wie NFS häufig eingesetzt. Einige Server unterstützen auch Multiprotokolle, wobei es dem Client überlassen ist, ob er eine Anfrage über TCP oder UDP stellt.

Was ist bei Ausfall des Dienstanbieters (Server)?

Bei einem lokalen Prozeduraufruf ist mit dem Rücksprung aus dieser die Abarbeitung beendet. Was aber, wenn der RPC-Aufruf nicht terminiert? Die Ursachen können vielgestaltig sein: der Server könnte ausgefallen oder überlastet sein, die Netzwerkverbindung könnte gestört sein...

Die meisten Realisierungen von Servern sehen keinerlei Statusinformationen auf Serverseite vor, d.h. die Fehlerbehandlung bleibt allein dem RPC-Client vorbehalten. Üblich sind Timeouts zur Erkennung von Problemen.

Semantik des Aufrufs (genau einmal, mindestens einmal)

Wie verhält sich ein Client, wenn der Server nicht rechtzeitig antwortet? Sendet er den Auftrag ein weiteres Mal ab, oder wartet er ggf. bis in alle Ewigkeit?

ONC RPC und DCE RPC bieten beide verschiedene Semantiken des Aufrufs. Welcher tatsächlich angewandt wird, hängt von der konkreten Realisierung ab. Werden z.B. serverseitig die Anforderungen der Clients verwaltet, so sollte ein Client einen Aufruf tatsächlich nur einmal initiieren dürfen. Merkt sich der Server den Zustand nicht, kann ein Client auch mehrfach eine Anfrage an diesen stellen, ohne dass dessen Arbeit beeinflusst werden würde. Beide RPC-Varianten ermöglichen auch Broadcast-RPCs, wo ein Client gleichzeitig Anfragen an mehrere Server richten kann und »No response«-RPC, wo der Client keine Antwort der Anfrage vom Server erwartet.

Der ISO-RPC lässt nur den einmaligen Aufruf einer RPC-Routine zu.

Wie werden Daten dargestellt (Wortbreite, big endian, little endian)?

Der Aufruf einer RPC-Funktion auf einer Intel-Maschine sollte im Idealfall auch von einem Server, der z.B. auf einer SPARC-Station läuft, behandelt werden können. Das Problem hierbei ist, dass beide Architekturen verschiedene Darstellungsformate verwenden. Nicht nur, dass neuere SPARCs mit einer 64 bit-Wortbreite und Intel-Rechner nach wie vor mit 32 bit arbeiten, sie ordnen intern die Bytes eines Wortes auch noch in unterschiedlichen Reihenfolgen an. Im Falle des RPC ist deshalb das Format der auszutauschenden Daten »big endian« durch das XDR-Protokoll (E**x**ternal **D**ata **R**epresentation) fest vorgegeben.

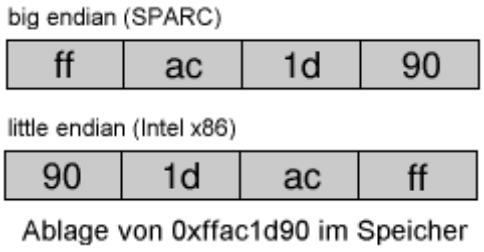


Abbildung 1: Byteanordnung auf SPARC und Intel

Wie gut ist die Performance?

Die Geschwindigkeit eines lokalen Funktionsaufrufes zu erreichen, ist ohnehin eine Wunschvorstellung. Dennoch sollte die Verzögerung durch den Netzwerktransport in akzeptablen Grenzen gehalten werden. Um trotz hoher Last die ständige Verfügbarkeit eines Servers zu gewährleisten, werden oft mehrere Serverprozesse parallel gestartet.

Wie steht es mit der Sicherheit ?

Zum einen sollte ein Server wissen, ob ein Client berechtigt ist, seine Dienste in Anspruch zu nehmen. Aber auch der Client möchte sicher gehen, dass er nicht von einem falschen Dienstanbieter Daten erhält und dass er seine Pakete an einen vertrauenswürdigen Rechner sendet. Ebenso kennen RPC-Implementierungen die Verschlüsselung von Daten während der Übertragung, um dem unerlaubten Abhören vorzubeugen.

RPC-Ablauf

Normalerweise erfolgt ein Aufruf einer RPC-Routine synchron, d.h. der Client sendet die Daten an einen Server und schläft bis zum Eintreffen der Antwort.

Der Server überwacht im einfachsten Fall den ihm zugedachten Port und beginnt, sobald er eine Anforderung eines Clients erkennt, mit der Bearbeitung des Auftrags. Dazu entpackt er die Daten des Pakets, ruft die behandelnde Routine auf, verpackt das Ergebnis und sendet die Antwort an den Client. Anschließend wartet der Server auf neue Aufträge. Abbildung 2 skizziert diesen einfachsten Fall.

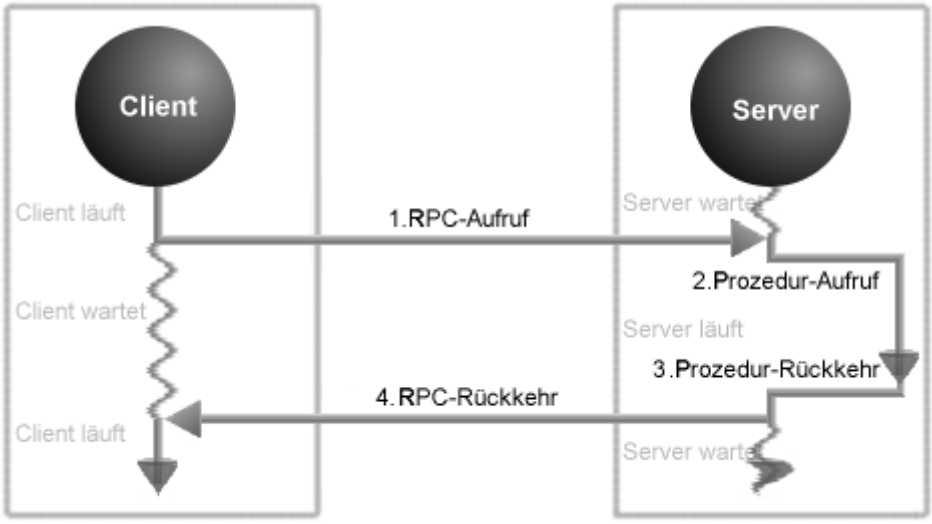


Abbildung 2: Ablauf eines RPC-Aufrufs

Wo liegt das Problem? Es existieren eine ganze Reihe von Standard-RPC-Diensten und von vielen Diensten bieten manche Server auch noch mehrere Versionen an. Wie also identifiziert ein Client den entsprechenden Dienst auf dem Server?

Ein RPC-Dienst wird durch genau drei Parameter eindeutig beschrieben:

1. Programmnummer
2. Versionsnummer
3. Transportprotokoll

Jeder RPC-Funktion einen eigenen, wohldefinierten Port zuzuweisen, war das Vorgehen in den allerersten Implementierungen. Aber schon bald stieß man auf das Problem der allmählich zur Neige gehenden freien Portnummern...

Die Lösung fand man durch Einführung eines neuen RPC-Dienstes, dessen Aufgabe die dynamische Zuweisung einer freien Portnummer zu einem lokalen Dienst ist, sobald eine Anforderung von einem Client eintrifft.

Bei diesem Dienst handelt es sich um den so genannten **Portmapper**.

Portmapper

An welchem Port der Portmapper auf Anforderungen lauscht, muss für jede Client-Anwendung bekannt sein. Deswegen überwacht der Portmapper stets **Port 111** sowohl über TCP als auch UDP.

Alle RPC-Dienste, die der eigene Rechner anbieten soll, müssen ihre Bereitschaft dem Portmapper signalisieren. Das impliziert, dass vor dem Start eines jeden RPC-Dienstes der Portmapper zu starten ist.

In jeder Linux-Distribution sollte das Programm **portmap** existieren, üblicherweise im Verzeichnis »/sbin«. Falls der Portmapper noch nicht aktiv ist, kann der Administrator diesen von Hand starten:

```
root@sonne> / sbin/ portmap
```

Die Arbeit des Portmappers lässt sich mit Hilfe des Kommandos **rpcinfo** überprüfen:

```
root@sonne> rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
```

An obiger Ausgabe ist zu erkennen, dass der Portmapper zurzeit der einzige aktive RPC-Dienst ist.

Sobald der Prozess des Portmappers läuft, überwacht er den Port 111 auf eintreffende Verbindungswünsche. Gleichzeitig führt er eine Tabelle mit allen notwendigen Informationen zu lokalen RPC-Diensten, die sich bei ihm registriert haben.

Trifft nun eine Anfrage von einem Client ein, schaut der Portmapper in seinen Tabellen nach, ob der erwünschte Dienst in der entsprechenden Version registriert ist. Wenn ja, antwortet er dem Client mit der korrekten Portnummer, an der der Server-Prozess aktiv ist. Der Client wendet sich des Weiteren direkt an den Server. Für weitere Anfragen wird dessen Portnummer lokal für eine gewisse Zeit zwischen gespeichert. Der Ablauf ist nochmals in Abbildung 3 skizziert:

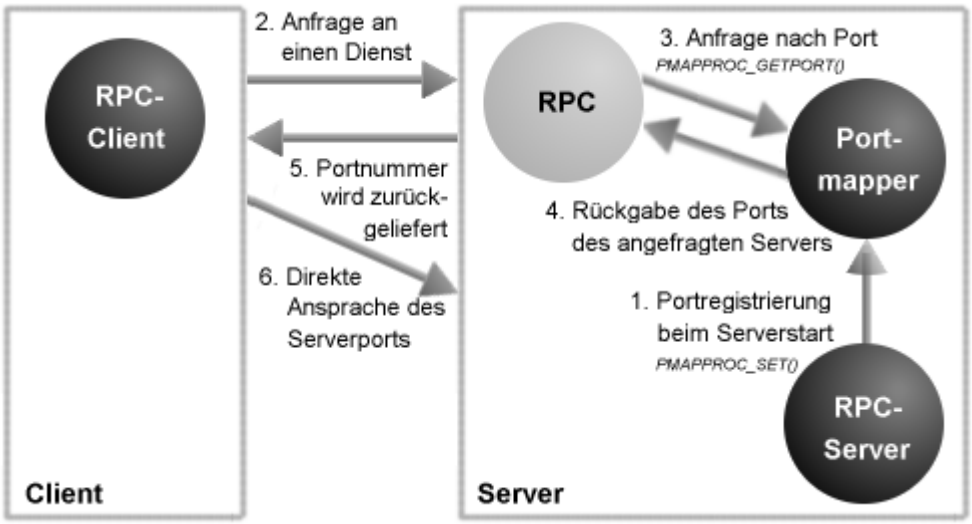


Abbildung 3: RPC-Client erfragt die Portnummer des Servers

Request For Comment

Übersicht

1969 verfasste der Student Steve Crocker einen Artikel "Host Software", in dem er zur öffentlichen Diskussion zu seinem Vorschlag aufrief. Er nannte die Dokumentation "Request for Comment" und schuf damit eine bis dahin nicht gekannte Plattform wissenschaftlichen Meinungsaustauschs.

Seinem Beitrag folgten Diksussionen verschiedenster Protokollverschläge für Netzwerke, die anfangs gemäß ihres Erscheinungsdatums nummeriert wurden. Erst 1983 schuf man im Internet Activities Board (IAB) die Stelle eines "RFC Editors", die die eingehenden Diskussionbeiträge erst nach einer eingehenden Begutachtung zur Veröffentlichung als RFC freigab. So beugte man der wachsenden Flut an "halbherzigen" Ausarbeitungen vor.

Die Bedeutung der RFC's demonstrieren die zahlreichen standardisierten Protokolle, die ihren Ursprung zumeist im Werdegang eines RFC's haben.

Die wichtigen RFCs

Die nachstehende Tabelle enthält die wichtigsten RFC's, die sich mit den im Buch behandelten Protokollen beschäftigen.

RFC-Nummer	Inhalt
826	Address Resolution Protocol (ARP)
862	Echo Protocol (ECHO)
864	Character Generator Protocol (CHARGEN)
867	daytime- Protocol
868	time-Protocol
951	Bootp - Internet bootstrap protocol (des Weiteren in 1532 und 1533)
1034	Domain Names - Concepts and Facilities
1035	Domain Names - Implementation and Specification
1094	Network File System (Version 2)
1831	Remote Procedure Call
1833	Portmapper (auch als RPCBind bezeichnet)

Netzwerk Clients

Überblick

Ziel des Kapitels

Inhalt des Kapitels

Client - Telnet & Co.

Übersicht

Die in diesem Abschnitt vorgestellten Client-Anwendungen arbeiten alle auf der Konsole. Ihre Gemeinsamkeit liegt im Versuch, die Ausführung von Kommandos auf entfernten Rechnern dem lokalen Verfahren anzugleichen. Der wesentliche Unterschied aus Sicht eines Anwenders ist die Sicherheit der einzelnen Mechanismen.

Telnet ermöglicht dabei »nur« eine Sitzung auf dem entfernten Rechner, während die r-Utilities und die Secure Shell auch Kommandos zum Kopieren zwischen Rechnern und zum entfernten Ausführen von Programmen beinhalten.

Etwas genauer werden die dahinter stehenden Prinzipien im Abschnitt Server - Telnet&Co beleuchtet.

Telnet

Telnet ermöglicht das Eröffnen einer Sitzung auf einem entfernten Rechner. Das zu Grunde liegende Protokoll **TELNET** wurde so entworfen, dass es auch in heterogenen Umgebungen zuverlässig arbeiten kann. Als Folge daraus ist die Arbeit über **telnet** halt doch nicht ganz so, als würde man lokal seine Eingaben tätigen.

In der Hauptsache ist es die Verfügbarkeit einiger Tastenkombinationen, die bei lokalem Aufruf eventuelle Sonderfunktionen implementieren, während einer Telnet-Sitzung allerdings als Steuerzeichen interpretiert werden und damit dort in der gewohnten Terminologie nicht zur Verfügung stehen.

Das Mapping zwischen den lokalen Einstellungen und denen auf dem Server wird daher auch als **Terminalemulation** bezeichnet und es existieren eine Reihe von Standards (bspw. vt100, vt220, xterm), die alle Clients und Server unterstützen sollten.

Zahlreiche Optionen des Telnet-Clients **telnet** beschäftigen sich daher auch mit den Einstellungen der Emulation und bei Initiieren einer neuen Telnet-Sitzung handeln Client und Server zunächst die Sitzungsparameter aus:

```
user@sonne> telnet
telnet> toggle options
Will show option processing.
telnet> open localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^ ]'.
SENT DO SUPPRESS GO Ahead
SENT WILL TERMINAL TYPE
SENT WILL NAWS
SENT WILL TSPEED
SENT WILL LFLOW
SENT WILL LINEMODE
SENT WILL NEW-ENVIRON
SENT DO STATUS
SENT WILL XDISPLOC
RCVD DO TERMINAL TYPE
RCVD DO TSPEED
RCVD DO XDISPLOC
RCVD DO NEW-ENVIRON
RCVD WILL SUPPRESS GO Ahead
RCVD DO NAWS
SENT IAC SB NAWS 0 80 (80) 0 22 (22)
RCVD DO LFLOW
RCVD DONT LINEMODE
RCVD WILL STATUS
RCVD IAC SB TERMINAL-SPEED SEND
SENT IAC SB TERMINAL-SPEED IS 38400,38400
```

```
RCVD IAC SB X-DISPLAY-LOCATION SEND
SENT IAC SB X-DISPLAY-LOCATION IS "sonne.galaxis.de:0"
RCVD IAC SB NEW-ENVIRON SEND
SENT IAC SB NEW-ENVIRON IS VAR "PRINTER" VALUE "lp" \
VAR "DISPLAY" VALUE "sonne.galaxis.de:0"
RCVD IAC SB TERMINAL-TYPE SEND
SENT IAC SB TERMINAL-TYPE IS "XTERM"
RCVD DO ECHO
SENT WONT ECHO
RCVD WILL ECHO
SENT DO ECHO
Welcome to Linux (i386) - Kernel 2.4.18 (pts/3).

sonne login:
```

Anhand der Ausgaben ist das Frage-Anwort-Spiel zwischen Client und Server leicht nachzuvollziehen. Der Client sendet »SENT« zum einen, bittend »WILL« (ich möchte...), zum anderen, fordernd »DO« (ich mache...) oder auch ablehnend »WONT« (ich möchte nicht...) seine Verhandlungspositionen zum Server, welcher antwortet und seinerseits Vorschläge zur Sitzungsgestaltung einbringt. Im Beispiel strebt der Client den »Linemode« an (SENT WILL LINEMODE), der Server lehnt den Wunsch ab (RCVD DONT LINEMODE)... Sind alle Parameter der Sitzung geklärt, sendet der Server die Login-Aufforderung.

Das Anmelden mit der Nutzerkennung »root« wird in den meisten Fällen scheitern, da der Terminal-Zugang nicht als sicher einzustufen ist. Die Terminals für das Root-Login müssen explizit in der Datei /etc/securetty benannt sein.

Während einer Sitzung kann die Terminalemulation des Servers beeinflusst werden, indem die dortige Shellvariable $TERM mit einem neuen Wert exportiert wird, auch kann der Kommandomodus von **telnet** über das so genannte Fluchsymbol [Ctrl][AltGr][9] erreicht werden. Nach Eingabe der meisten Kommandos gelangen Sie zur aktuellen Sitzung zurück:

```
user@sonne> telnet localhost -a
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Password:
Last login: Wed Jun 21 09:18:40 from localhost
user@sonne> export PS1="user@telnet> "
user@telnet> [Ctrl][AltGr][9]
telnet> status
Connected to localhost.
Operating in single character mode
Catching signals locally
Remote character echo
Local flow control
Escape character is '^]'.
<Enter>

user@telnet> logout
Connection closed by foreign host.
user@sonne>
```

Die wichtigsten Optionen beim Start

```
telnet [-a] [-d] [-x] [-l <Benutzername>] [Servername] [Portnummer]
```

Wichtige Optionen beim Start von **telnet** sind neben der Angabe des Zielrechners eine nachfolgende **Portnummer**, falls Telnet sich nicht an den Standardport 23 des Zielrechners verbinden soll. Diese Möglichkeit spielt vor allem zum Debuggen bestimmter Dienste eine Rolle, allerdings lassen sich nicht alle Dienste auf so einen interaktiven Plausch ein.

Das nachfolgende Beispiel zeigt eine andere Art Mails zu versenden, den direkten Dialog mit Sendmail (SMTP an

Port 25)...

```
user@sonne> telnet sonne smtp
Trying 192.168.10.101...
Connected to sonne.galaxis.de
Escape character is '^]'.
220 sonne.galaxis.de ESMTP Sendmail 8.12.2/8.12.2; Tue, 2 May 2002 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
< ENTER>
Date: Mon Apr 1 11:11:11 MEST 2000
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.
```

Mit der Option **-a** wird das automatische Anmelden beim Server versucht, d.h. der lokale oder der mit der Option **-l Name** angegebene Nutzername wird automatisch als Loginname gesetzt (indem die Variable $USER gesetzt und exportiert wird) und es erscheint nur noch die Passwortabfrage. Bei Eingabe eines falschen Passwortes erscheint ein neues Login-Prompt. Wie viele Fehlversuche zulässig sind, bevor der Server die Verbindung kappt, hängt von dessen Konfiguration ab.

Zum Debuggen einer Sitzung kann die Option **-d** angegeben werden, sie erfordert allerdings Root-Rechte.

Neuere Telnet-Versionen sollen auch verschlüsselte Sitzungen unterstützen, indem sie mit der Option **-x** initiiert werden, allerdings erntete ich bislang immer die Ausschrift «telnet: Warning: -x ignored, no ENCRYPT support.«.

Die wichtigsten Kommandos im interaktiven Modus

Um innerhalb einer Telnet-Sitzung Verbindung zu einem Server aufzunehmen, dient das Kommando **open**:

```
telnet> open sonne.galaxis.de 25
```

Die Kommandos **logout** und **close** besitzen dieselbe Bedeutung und beenden die aktuelle Verbindung. Ein **quit** beendet zusätzlich die Telnet-Sitzung.

Mit **set** bzw. **unset** lassen sich Variablen von Telnet setzen bzw. löschen. Variablen beeinflussen u.a. die Wirkungsweise von Kommandos oder sie steuern die Behandlung von Sonderzeichen. Geben Sie innerhalb von Telnet »set ?« ein und Sie erhalten eine Liste der unterstützten Variablen nebst kurzer Erläuterung.

```
telnet> set ?
echo            character to toggle local echoing on/off
escape          character to escape back to telnet command mode
rlogin          rlogin escape character
tracefile       file to write trace information to

              The following need 'localchars' to be toggled true
flushoutput     character to cause an Abort Output
interrupt       character to cause an Interrupt Process
quit            character to cause an Abort process
eof             character to cause an EOF
...
```

Das Kommando **status** wurde im letzten Beispiel schon benutzt und zeigt die wichtigsten Eigenschaften der aktuellen Sitzung an. Auch **toggle** war bereits in Erscheinung getreten. Es schaltet einfach die Flags des Arguments um (zwischen TRUE und FALSE). Zwei interessante Argumente zur Überwachung des Datenaustauschs zwischen Client und Server sind **options**, das die Protokollbearbeitung veranschaulicht, und **netdata**, womit der Netzwerkverkehr in hexadezimaler Form überwacht werden kann.

Die r-Utilities

Die r-Utilities sind eine Sammlung von Netzwerkkommandos, die einige wichtige Unix-Kommandos um die Netzwerkfähigkeit erweitern. Zu den Kommandos zählen: **rlogin**, **rsh**, **rcp**, **ruptime**, **rwho** und **rexec**. Trotz der mit den Kommandos verbundenen Sicherheitsrisiken ist ihr Einsatz in geschlossenen Unix-Netzwerken weit verbreitet.

Vorab sei bemerkt, dass die in Form der Manuals beiliegenden Dokumentationen zu den nachfolgenden Kommandos nicht mit den Möglichkeiten dieser übereinstimmen. Rufen Sie »< Kommando> --help« auf und vergleichen Sie die Liste der Argumente mit denen aus dem Manual.

Gleichsetzung von Benutzerkennungen

Diese Möglichkeit betrifft die Kommandos **rlogin**, **rsh** und **rcp** und erlaubt bei entsprechender Konfiguration das passwortfreie Anmelden auf dem Zielrechner. Die Authentifizierung eines Benutzers erfolgt dabei anhand von Tabellen, die Paare von Rechnernamen und Benutzerkennzeichen enthalten. Der Administrator des Servers kann hierzu eine global gültige Datei **/ etc/ hosts.equiv** anlegen, so dass ein in ihr enthaltener Benutzer, der vom benannten Rechner aus die Anmeldung unter dem selben Benutzernamen versucht, sofortigen Zugang erhält. Einziges Benutzerkennzeichen, dem niemals passwortfreier Zugang gewährt wird, ist **root**.

Sollte in der Datei »/etc/hosts.equiv« keine Übereinstimmung eines Rechner-Benutzer-Paares gefunden werden, so wird im Heimatverzeichnis des Benutzers nach einer Datei **.rhosts** gesucht. Der Inhalt der Datei ist allerdings nur relevant, wenn diese Datei **root** oder dem Benutzer selbst gehört und sie nur vom Eigentümer beschreibbar ist. Der Aufbau dieser Datei ist analog zum Aufbau der /etc/hosts.equiv:

```
user@sonne> cat ~/ .rhosts
localhost          user
erde.galaxis.de    tux
melmac.outside.all tux
melmac.outside.all alf
```

Der Benutzer »tux« erhält von den Rechnern »erde.galaxis.de« und »melmac.outside.all« aus passwortfreien Zugang unter der Kennung »user« auf dem Rechner »sonne.galaxis.de«. »alf« darf nur von »melmac.outside.all« aus ohne Angabe des Passwortes zugreifen und damit es auch lokal für den Benutzer selbst funktioniert, ist auch ein lokaler Eintrag enthalten.

rlogin

Remote Login gleicht funktionell sehr stark dem Kommando telnet. Es verbindet sich zu einem Server (rlogind, TCP-Port 513), der eine Login-Prozedur startet und die Verbindung über ein Pseudoterminal betreibt. Im

Unterschied zu **telnet** kennt **rlogin** kein Protokoll, um Übertragungsparameter mit dem Server auszuhandeln.

Verbinden Sie sich recht häufig mit einunddemselben Server, so lässt sich der Tippaufwand verringern, indem Sie einen Link unter dem Rechnernamen des Servers auf das Programm **rlogin** angelegen:

```
user@sonne> ln -s / usr/ bin/ rlogin localhost
user@sonne> ./ localhost
Last login: Sat Jul 8 16:42:56 from localhost
Have a lot of fun...
user@sonne> ~.
rlogin: closed connection.
```

Zum Beenden einer **rlogin**-Sitzung geben Sie entweder [Ctrl][D] oder das Escape-Zeichen (Voreinstellung ist die Tilde) gefolgt von einem Punkt ("~.") ein. Alternativ endet die Sitzung, sobald Sie die Shell auf dem Server beenden (exit).

Wichtigste Kommandozeilenoptionen sind **-l < Nutzername>** , das die Anmeldung auf dem Server unter dem angegebenem Namen versucht, und **-e < Escape-Zeichen<** , um ein anderes Zeichen (als die Tilde) als »Fluchtsymbol« zu vereinbaren.

rsh

Remote Shell ermöglicht das Ausführen von Kommandos auf einem anderen Rechner. Serverseitig wartet der Dämon **rshd** auf TCP-Port 514. Wird **rsh** ohne Argumente aufgerufen, so ruft das Kommando implizit **rlogin** auf, um eine Sitzung auf dem Server einzuleiten. **rsh** wird normalerweise mit dem Servernamen und dem zu startenden Kommando aufgerufen. Letzterem können beliebig viele Argumente folgen, die direkt dem Server übergeben werden. Vergessen Sie nicht, enthaltene Sonderzeichen vor der Interpretation durch die lokale Shell zu schützen.

```
tux@erde> rsh sonne w user
  5:09pm  up  9:33,  3 users,  load average: 0.16, 0.10, 0.02
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
user     tty2     -                 4:10pm 50:49   0.20s  0.13s  -bash
```

rsh bedingt den passwortfreien Zugang auf dem Server, sonst hagelt es nur ein »Permission denied«. Mit der Option **-l < Nutzername>** können Sie sich unter einem anderen Namen anmelden.

Einschränkungen: Ein paar Unterschiede zum lokalen Kommandostart gibt es dann doch... Zum einen geht der Rückgabewert des Kommandos verloren. Auch sollten Sie keine Editoren oder andere interaktiv arbeitende Kommandos entfernt starten, da Sie keine Möglichkeit haben, diese mit Eingaben zu füttern (der Start gelingt schon...).

rcp

Remote Copy ist das Netzwerkpedant zum lokalen **cp**. Nahezu alle Optionen (die im Netzwerk sinnvoll sind) des lokalen Vorbildes sind implementiert, u.a. auch **-r** zum rekursiven Kopieren ganzer Verzeichnisstrukturen. **rcp** vermag Dateien vom lokalen Rechner zu einem anderen zu übertragen, ebenso kann der umgekehrte Weg gegangen werden und es besteht sogar die Möglichkeit, den Kopiervorgang zwischen zwei entfernen Rechnern von der lokalen Konsole aus zu veranlassen.

Der Aufruf des Kommandos lautet:

```
rcp [[user1@]host1:]<Quelle(n)> [[user2@]host2:]<Ziel>
```

»user« ist nur anzugeben, wenn sich das Nutzerkennzeichen des lokalen Benutzers von denen auf den entfernen Rechnern unterscheidet. Ist der lokale Rechner Ort der Quellen oder des Zieles, so kann auch die Angabe dessen Namens entfallen. Quelle und Ziel können die üblichen Metazeichen enthalten, jedoch müssen diese vor der Interpretation durch die Shell geschützt werden. Wird bei der Angabe der Dateinamen auf die Pfade verzichtet, bezieht sich diese immer auf das Heimatverzeichnis.

rexec

Remote Execute ist eine weitere Möglichkeit, um Kommandos auf einem anderen Rechner auszuführen. Im Unterschied zu **rsh** wird hier mit einer Login-Prozedur begonnen und ein verschlüsseltes Passwort gesendet. Das Kommando selbst ist mit Vorsicht zu genießen, da das permanente Prozedere mit dem Anmeldevorgang oft verleitet, die Option **-p < Passwort>** zu verwenden. Dieses Passwort muss auf der Kommandozeile im Klartext angeben werden, d.h. ggf. für in der Nähe befindliche Personen lesbar!

Um die Abfrage des Loginnamens zu unterdrücken, kann dieser mit der Option **-l < Nutzername>** als Argument angegeben werden.

```
user@sonne> rexec -l tux -p SchAuT_weG erde.galaxis.de pwd
/home/tux
```

rwho und ruptime

Remote Who und *Remote Uptime* erweitern die gleichnamigen Kommandos auf die Netzwerkumgebung, d.h. mit **rwho** erhalten Sie Informationen über alle im lokalen Netzwerk angemeldeten Benutzer (analog zum lokalen **who**) und **ruptime** zeigt u.a. die Laufzeit aller Rechner des Netzes an (Zeit seit dem letzten Systemstart).

Beide Kommandos werten die Informationen der Dateien im Verzeichnis »/var/spool/rwho« aus.

Diese Datei wird in regelmäßigen Abständen vom **rwhod** aktualisiert, der die lokalen Informationen per Broadcast ins Netz entlässt und die Nachrichten anderer Dämonen empfängt. In großen Netzen kann dadurch die Netzlast enorm anwachsen. Deshalb und auch aus Sicherheitgründen werden diese Dienste meist deaktiviert.

Secure Shell

Solange wie es die Secure Shell schon gibt, sollte man meinen, angesichts der gravierenden Sicherheitslücken von Telnet und Remote Shell, sie hätte mit den Altlasten von Unix schon längst aufgeräumt. Doch leider ist Ihr Einsatz, trotz hoher Verfügbarkeit, noch nicht zur Selbstverständlichkeit erhoben worden.

Dabei sind verschlüsselte Übertragungen und passwortfreies Anmelden auf entfernten Rechnern doch genau das, was der Anwender sich wünscht. In diesem Abschnitt lernen Sie die Belange des ssh-Clients kennen.

Die Konfiguration des Clients

Die default-Einstellungen zum Secure Shell Client legt der Systemverwalter in der Datei **/ etc/ ssh_config** fest, dem Benutzer bleibt es überlassen, eigene Konfigurationen in einer Datei **~ / .ssh/ config** vorzunehmen. Existiert letztere Datei im Heimatverzeichnis des Benutzers, so wird das Kommando **ssh** diese beim Start einlesen, ansonsten entnimmt es die Informationen der globalen Datei.

```
user@sonne> cat / etc/ ssh_config
# Site-wide defaults for various options

#   Host *
#   ForwardAgent yes
#   ForwardX11 yes
#   RhostsAuthentication yes
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   FallBackToRsh yes
#   UseRsh no
#   BatchMode no
#   StrictHostKeyChecking no
#   IdentityFile ~/.ssh/identity
#   Port 22
#   Cipher idea
#   EscapeChar ~
```

In der Beispieldatei sind alle Zeilen auskommentiert. In den allermeisten Fällen sollte dies die praktischen Bedürfnisse auch abdecken. Was sich hier dennoch konfigurieren lässt, möchten wir Ihnen nicht vorenthalten (eine Beschränkung auf ausgewählte Einträge sei uns gestattet):

Host

Alle nachfolgenden Einstellungen, bis zur nächsten mit »Host« beginnenden Zeile betreffen einzig den/die angegebenen Rechner. Die Angabe kann die Wildcards »?« und »*« enthalten. Hiermit lassen sich also verschiedene Einstellungen für einzelnen Rechner und für Rechner eines Netzwerks treffen.

BatchMode

Die Abfrage nach dem Passwort oder der Passphrase (siehe später) wird unterbunden, somit lassen sich z.B Kommandos in Shellscripts ausführen. Mögliche Werte sind »yes« oder »no«.

Cipher

Hier wird der zu verwendende Verschlüsselungsalgorithmus angegeben. Allerdings sollten Sie aus idea, des 3des, blowfish, arcfour bzw. none nur den wählen, der sowohl von Ihrem Client als auch vom Server unters wird.

Compression

Schaltet die Komprimierung der zu übertragenden Daten ein bzw. aus. Mögliche Werte sind »yes« oder »nc

FallBackToRsh

Scheitert der Verbindungsaufbau zum Server (weil diese eventuell nicht aktiv ist), kann eine Verbindung üb rsh versucht werden. Den Wert auf »yes« zu setzen, beschwört den ganzen Ärger der unsicheren rsh-Verbindung herauf...

ForwardAgent

Die Authentifizierung kann über einen »Vermittler« vorgenommen werden. Um dieses automatisch zu veranlassen, kann ForwardAgent auf »yes« gesetzt werden. Dieser Vermittler wird meist das Programm **ssh agent** sein. Das Kommando sollte unmittelbar in der Login-Shell gestartet werden. Damit werden alle weite Prozesse automatisch Nachfahren des ssh-agent-Prozesses, letztlich auch der Aufruf eines ssh-Kommandos

ForwardX11

X11 Verbindungen können automatisch über einen sicheren Kanal umgeleitet und die Variable $DISPLAY gesetzt werden. Mögliche Werte sind »yes« oder »no«.

IdentityFile

Die Datei enthält den geheimen RSA-Schlüssel des Benutzers. In den meisten Fällen wird der Benutzer nur IdentityFile (~/.ssh/identity) verwenden, dann kann auf die Angabe verzichtet werden. Weicht der Dateinar davon ab, muss das Schlüsselwort gesetzt werden. Dabei muss der Dateiname mit der Tilde beginnen!

PasswordAuthentication

Hier wird festgelegt, ob eine Abfrage des Passwortes durchgeführt werden soll oder nicht. Steht der Wert a »no«, ist ein Anmelden auf einem entfernten Rechner nur möglich, wenn der eigene öffentliche Schlüssel d bekannt ist.

Port

Die Portnummer, an die sich **ssh** verbinden soll.

RhostsAuthentication

Hier wird festgelegt, dass bei der Authentifizierung die Dateien rhosts und /etc/hosts.equiv ausreichen. Dies

Rückfall in die rhost-Zeiten ist eine Sicherheitslcke und sollte demensprechend auf »no« bleiben.

RhostsRSAAuthentication

Hier können die Benutzung von rhosts und /etc/hosts.equiv erlaubt werden, allerdings nur wenn die RSA-H Authentifizierung erfolgreich war. Standardeinstellung ist »yes«.

RSAAuthentication

Hiermit kann die Authentifizierung mittels RSA-Verschlüsselung an- bzw. abgeschalten werden. Steht der W auf »yes«, sollte die Datei »~/.ssh/identity« oder ein Authentifizierungsagent existieren.

StrictHostKeyChecking

Ein »yes« verschärft die Sicherheit, indem das Anmelden nur auf Rechnern gestattet wird, die in den Datenbanken »/etc/known_hosts« bzw. »~/.ssh/known_hosts« enthalten sind. Steht hier »no« (Voreinstell werden neu besuchte Rechner automatisch der privaten Datei hinzugefügt.

TISAuthentication

Hier wird festgelegt, ob Authentifizierung ber TIS erlaubt ist. Diese Methode wird bei der Gauntlet Firewall bei dem freie Firewall Toolkit (FWTK) der Firma Trusted Information Systems (TIS) verwendet.

UseRsh

Steht der Wert auf »yes«, wird **ssh** sofort eine Verbindung über **rsh** aufbauen und alle Optionen mit Ausna des Host-Feldes ignorieren.

Die Verwendung des Clients

Es ist an der Zeit, eine **ssh**-Sitzung zu eröffnen. Um einen Eindruck vom Ablauf das Anmeldens zu erhalten, schalten wir die erweiterte Ausgabe ein (Option **-v**):

```
user@sonne> ssh localhost -v
SSH Version 1.2.27 [i686-unknown-linux], protocol version 1.5.
Standard version. Does not use RSAREF.
sonne: Reading configuration data /etc/ssh_config
sonne: ssh_connect: getuid 500 geteuid 0 anon 0
sonne: Allocated local port 1023.
sonne: Connecting to 127.0.0.1 port 22.
sonne: Connection established.
sonne: Remote protocol version 1.5, remote software version 1.2.27
sonne: Waiting for server public key.
sonne: Received server public key (768 bits) and host key (1024 bits).
sonne: Forcing accepting of host key for localhost.
sonne: Host '127.0.0.1' is known and matches the host key.
Creating random seed file ~/.ssh/random_seed. This may take a while.
sonne: Encryption type: idea
sonne: Sent encrypted session key.
sonne: Installing crc compensation attack detector.
sonne: Received encrypted confirmation.
sonne: Remote: Server does not permit empty password login.
sonne: No agent.
sonne: Doing password authentication.
user@127.0.0.1's password:
```

Angenommen, unser Passwort besitzt nur 3 Zeichen...:

```
user@sonne> ssh localhost
user@127.0.0.1's password:
Permission denied.
```

Offensichtlich verfügt der Server über eingebaute Regeln bezüglich der Passwort-Sicherheit. Ein Zugangsversuch mit »starkem« Passwort provoziert folgende Reaktion:

```
user@sonne> ssh localhost
user@127.0.0.1's password:
Last login: Fri Mar 31 21:45:35 2000 from localhost
Have a lot of fun...
No mail.
user@sonne>
```

Die folgende Sitzung läuft analog zur Arbeit am lokalen Terminal ab. Sämtlicher Datenverkehr geht verschlüsselt übers Netz.

Die Optionen zum Start von **ssh** sind vielfältig und die meisten werden Sie vermutlich niemals benutzen. Eine kleine Auswahl stellt die folgende Tabelle zusammen:

-l Nutzer

Auf dem Zielrechner meldet man sich unter dem Kennzeichen von »Nutzer« an, anstelle des lokalen Nutzerkennzeichens.

-p port

Es wird sich beim Zielrechner zum angegebenen Port verbunden anstatt zum Port 22. An diesem Port sollte allerdings auch ein ssh-Server lauschen. **ssh** kann nicht zum Debuggen anderer Dienste genutzt werden.

-f

ssh geht nach dem Verbindungsaufbau in den Hintergrund. Gleichzeitig wird die Standardeingabe von /dev lesen (Vergleiche Option »-n«).

-i Datei

Anstatt der Datei ~/.ssh/identity wird die angegebene als privater Schlüssel verwendet

-n

Die Standardeingabe wird von /dev/null lesen. Das ist notwendig, wenn die ssh ein X-Programm auf dem entfernten Rechner starten soll, da dieses die Eingaben von der Standardeingabe lesen soll und nicht die **ss** (Beispiel: »ssh erde -n xemacs &«).

Passwortfreies Login

Prinzipiell ist eine derartige Konfiguration auf **drei Wegen** möglich. Zum einen über die **Gleichsetzung von Nutzerkennungen** über Rechnergrenzen hinweg (analog dem Vorgehen in der »/etc/hosts.equiv«) und zum anderen durch die **Authentifizierung mittels der Hostschlüssel** und zum dritten durch eine **Kombination** aus beiden. Die Konfiguration der beiden erstgenannten Methoden sei nun vorgestellt:

Gleichsetzung von Nutzerkennungen

Diese Methode gilt als unsicher und wird deswegen in der Standardkonfiguration des Servers nicht unterstützt. Dennoch sei das Vorgehen auf Clientseite hier skizziert:

Root hat nun die Möglichkeit, in einer Datei »/etc/shosts.equiv« Paare aus Rechner- und optionalem Nutzernamen einzutragen. Einem Nutzer, der sich unter einem solchen Namen vom entsprechenden Rechner aus anmeldet, wird der Zugang ohne Angabe eines Passwortes gewährt. Er gilt als »vertrauenswürdig«. Der Aufbau der Datei »/etc/shosts.equiv« ist analog zur »/etc/hosts.equiv«.

Der gewöhnliche **Nutzer** kann ein analoges Verhalten durch Anlegen einer Datei ».shosts« in seinem

Heimatverzeichnis auf dem Zielrechner erreichen. Dazu trägt er in dieser Datei Paare von Rechner- und Nutzernamen ein. Von einem solchen Rechner aus ist das passwortfreie Anmelden unter der spezifizierten Nutzerkennung möglich.

Authentifizierung mittels Hostschlüssel

Das Vorgehen ist einfach:

1. Der Nutzer generiert ein Paar aus öffentlichem und privaten Hostschlüssel:

```
user@sonne> ssh-keygen
Initializing random number generator...
Generating p: .....++ (distance 502)
Generating q: .....++ (distance 364)
Computing the keys...
Testing the keys...
Key generation complete.
Enter file in which to save the key (/home/user/.ssh/identity):
Enter passphrase:
Enter the same passphrase again:
Your identification has been saved in /home/user/.ssh/identity.
Your public key is:
1024 33 144548000807386124839939532430378957162277213079174901455237068977\
139931644547879340831129643511346585433981981732906528611103615044612042 52\
860825515809240572983943183684618343601610394751130292219758202002951465 87\
337217929961550672876722156317073838072080319067844704449679307775461587 65\ 712311197196260549233
user@sonne
Your public key has been saved in /home/user/.ssh/identity.pub
```

Tipp: Drücken Sie bei der Frage nach den Passphrasen [ENTER]. Wenn Sie dort etwas eingeben, müssen Sie die Eingabe bei jedem Login über ssh wiederholen. Somit ermöglichen Sie ein von Ihrem eigentlichen Passwort unabhängiges Passwort.

2. Der Nutzer meldet sich auf dem Zielrechner an:

```
user@sonne> ssh erde
Last login: Sat Apr 1 11:38:19 2000
Have a lot of fun...
No mail.
```

3. Den öffentlichen Schlüssel ~/.ssh/identity.pub des lokalen Rechners (sonne) fügt der Nutzer an die Datei ~/.ssh/authorized_keys in seinem Heimatverzeichnis auf dem Zielrechner (erde) an.

   Zum Kopieren zwischen den Rechnern existieren zahlreiche Möglichkeiten:

   - Über FTP
   - Per Mail
   - Unter X: Kopieren mittels Maus
   - Mittels scp (Secure Copy):

```
user@erde> scp user@sonne:~/.ssh/identity.pub user@erde:~/temp.key
user@erde> cat temp.key >> .ssh/authorized_keys
```

Zukünftige Sitzungen sollten ohne Angabe eines Passwortes möglich sein.

Secure Copy

Während einer Secure Shell Sitzung würde der Aufruf des Kommandos cp nur das Kopieren von Dateien innerhalb des Dateisystems des Zielrechners ermöglichen. Um dennoch Daten zwischen Ziel- und Heimatrechner auszutauschen, ohne auf FTP oder rcp zurückgreifen zu müssen, steht das Kommando **scp** zur Verfügung.

scp verwendet zum Datentransfer die Secure Shell und bietet damit dieselben Authentifizierungsmechanismen und dieselbe Sicherheit. Der Aufruf besitzt folgendes Format:

Aufruf: scp [OPTIONEN] [[user@]host1:]filename1... [[user@]host2:]filename2

Die Angabe der Dateinamen unterscheidet sich nicht vom Vorgehen des »lokalen« cp, auch lassen sich wie gewohnt Verzeichnisse rekursiv mit der Option **-r** kopieren. Allerdings kommen nun noch die Namen der Nutzer und Rechner hinzu.

Der **Nutzer** (user) ist anzugeben, wenn sich das Nutzerkennzeichen auf dem Quell- bzw. Zielrechner vom lokalen Nutzerkennzeichen unterscheidet. Sonst kann die Angabe entfallen.

Der **Rechnername** (host) ist anzugeben, wenn es sich nicht um den lokalen Rechner handelt.

Im nachfolgenden Beispiel kopieren wir eine Datei vom Rechner »erde« unter dem dortigen Login »tux« auf den lokalen Rechner. Es sei vorausgesetzt, dass uns das passwortfreie Anmelden auf »erde« als »tux« möglich ist (falls dem nicht so ist, würden wir zur Eingabe eines Passwortes aufgefordert werden:

```
user@sonne> scp tux@erde:~/beispiel.txt .
beispiel.txt                  3 KB |   3.1 kB/s | ETA: 00:00:00 | 100%
```

Neben dem schon genannten **-r** zum rekursiven Kopieren sind noch die Optionen **-q** und **-C** nützlich, die die Statistikausgabe verhindern bzw. eine Komprimierung der zu übertragenen Daten vornehmen.

Network File System- Der Client

Übersicht

Das Network Filesystem ist die gebräuchlichste Methode unter Unix, um Verzeichnisse über Rechnergrenzen hinweg verfügbar zu machen. Einen kurzen Abriss zur Historie und Intention des NFS erhalten Sie im Abschnitt zum NFS-Server.

Voraussetzungen

Kernelunterstützung

Die Unterstützung des NFS-Dateisystems ist Aufgabe des Kernels. Vermutlich integriert der aktuelle Kernel bereits die erforderliche Eigenschaft; ein Blick in die Datei »/proc/filesystems« schafft Gewissheit:

```
user@sonne> cat / proc/ filesystems
nodev   sockfs
nodev   tmpfs
nodev   pipefs
nodev   proc
        ext2
nodev   devpts
        reiserfs
nodev   supermount
```

Im Beispiel fehlt offensichtlich ein entsprechender Eintrag für das NFS-Dateisystem. Bei aktuellen Kerneln, die heutigen Distributionen beiliegen, muss das noch nichts bedeuten... Eventuell ist die Unterstützung im Kernel nur als Modul kompiliert, und solange das Modul noch nicht geladen wurde, wird der Kernel das Dateisystem auch nicht kennen. Das Modul sollte nun von Hand geladen werden:

```
root@sonne> modprobe nfs
```

Eine Ausschrift der Art »*modprobe: Can't locate module nfs*« resultiert entweder aus einer falschen Konfiguration der Modulabhängigkeiten (unwahrscheinlich) oder aber aus einer fehlenden Unterstützung durch den Kernel. Die Generierung eines neuen Kernels wird notwendig sein...

Bei erfolgreichem »modprobe«-Aufruf erscheint folgende Zeile am Ende der Datei »/proc/filesystems«:

```
user@sonne> tail -2 / proc/ filesystems
nodev   supermount
nodev   nfs
```

Portmapper

Für die Grundfunktionalität eines NFS-Clients ist die Aktivierung des Portmappers nicht notwendig. Erst wenn Programme mit Dateisperren auf importierten NFS-Verzeichnissen arbeiten, werden auf Clientseite zwei RPC-Dienste (»rpc.lockd« und »rpc.statd«) und damit der Portmapper erforderlich.

Das Vorgehen zum Start des Portmappers und der beiden Dienste erfolgt analog zur Beschreibung im NFS-Server-Abschnitt. Beachten Sie auch die dortigen Hinweise zu ggf. erforderlichen Sicherheitseinstellungen in den Dateien »/etc/host.allow« und »/etc/hosts.deny«. Ebenfalls in der Abhandlung zum Server finden Sie eine Diskussion zur Arbeitsweise von Network-Lock-Manager (»rpc.lockd«) und Network-Status-Monitor (»rpc.statd«).

Bei aktuelleren RedHat- und SuSE-Distributionen finden Sie oft ein Skript /etc/init.d/nfs, über das Sie komfortabel den Client starten und beenden können (genau genommen startet das Skript auch nur die beiden RPC-Dienste und mountet alle in der Datei /etc/fstab erwähnten NFS-Dateisysteme).

Mounten eines NFS-Dateisystems

Welcher Server bietet was?

Für gewöhnlich wird der Administrator eines NFS-Servers die Verzeichnisse nur für konkrete Rechner exportieren und den Verantwortlichen dieser Clients alle notwendigen Informationen zum Zugriff zukommen lassen. Dennoch kann von einem Client leicht verifziert werden, ob ein Rechner als NFS-Server fungiert und welche Verzeichnisse er welchen Rechnern zur Verfügung stellt.

Das Kommando »showmount« dient zur Abfrage eines Servers. Werden, abgesehen vom Servernamen, keine weiteren Optionen angegeben, so werden die aktuell zum Server verbundenen Clients aufgelistet:

```
user@venus> showmount sonne.galaxis.de
All mount points on sonne.galaxis.de:
erde.galaxis.de
vernus.galaxis.de
```

Aus Sicht eines NFS-Clients ist die Option »-e« nützlich, die dem Server Auskunft über die exportierten Verzeichnisse entlockt:

```
user@venus> showmount -e sonne.galaxis.de
Export list for sonne.galaxis.de:
/home      *.galaxis.de
/usr/share *.galaxis.de
```

Eine umfassende Beschreibung von »showmount« finden Sie im Abschnitt zum NFS-Server.

Das Mount-Kommando

Auch zum Mounten von NFS-Verzeichnissen dient das Kommando **mount**. Es gelten dieselben Voraussetzungen wie zum Einhängen lokaler Dateisysteme: Die notwendigen **Rechte** müssen gegeben sein (auch auf Serverseite) und der **Mountpunkt** - also das Zielverzeichnis - **muss existieren**.

Der Aufruf von **mount** folgt nun diesem Schema:

```
mount -t nfs [ -o <Optionen> ] <Servername>:<Verzeichnis auf Server> <lokaler Mountpunkt>
```

Anstatt des Servernamens ist auch die Angabe einer IP-Adresse zulässig. Neben den allgemein gültigen *Optionen* für das Mountkommando steuern NFS-spezifische Optionen vor allem das Verhalten des Kommandos im Fehlerfall. Wir widmen uns ihnen im folgenden Abschnitt.

Als konkretes Beispiel soll vom NFS-Server »sonne.galaxis.de« das dort freigegebene Verzeichnis »/home« importiert werden. Als Mountpunkt wurde lokal ein Verzeichnis »/mnt/home« erzeugt. Der Aufruf sieht wie folgt aus:

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

Etwas Komfort

Da Administratoren - allen voran »Unix-ler« - bekanntlich recht bequeme Menschen sind und nach Automatisierung streben, gibt es auch hier einen Weg, um den Mount-Aufruf zu vereinfachen. Die kürzere und elegantere Schreibweise wäre ein Eintrag in die Datei /etc/fstab, die statische Informationen über Dateisysteme, ihre

Mountpunkte und Optionen enthält.

Eine typische fstab-Datei könnte wie folgt aussehen:

```
user@erde> cat / etc/ fstab
/dev/hda3       /             ext3      defaults 1 2
/dev/hda1       /boot         ext3      defaults 1 2
/dev/cdrom      /media/cdrom  auto      ro,noauto,user,exec 0 0
/dev/fd0        /media/floppy auto      noauto,user,sync 0 0
devpts          /dev/pts      devpts    defaults 0 0
proc            /proc         proc      defaults 0 0
usbdevfs        /proc/bus/usb usbdevfs  noauto 0 0
/dev/hda2       swap          swap      pri=42 0 0

# NFS-Einträge
sonne:/home     /home         nfs       bg,soft,intr,retry=5 0 0
sonne:/usr/share /usr/share   nfs       defaults 0 0
```

Zur Bedeutung der einzelnen Felder finden Sie Erläuterungen im Abschnitt Dateisysteme, /etc/fstab des Kapitels »Systemadministration«. Wichtig für unser Beispiel sind die beiden letzten Einträge. Anstelle der Gerätedatei bei lokalen Dateinamen tritt nun das zu importierende Verzeichnis in der Form »Server:/Pfad«. Als Typ des Dateisystems muss »nfs« in die dritte Spalte eingetragen werden. Die beiden letzten Spalten sollten bei importierten Verzeichnissen stets »0« sein, da sowohl ein Backup als auch eine Überprüfung des Dateisystems Aufgabe des Servers (der Rechner, auf dem diese Dateisysteme lokal liegen) ist.

Und der Zweck solcher Einträge?

Zum einen genügt nun ein einziger Aufruf, um alle in der Datei benannten NFS-Verzeichnisse gleichzeitig zu importieren:

```
root@erde> mount -a -t nfs
```

Zum anderen sinkt selbst bei Import einzelner Einträge der Tippaufwand, da als Argument für das Kommando »mount« entweder die Angabe der Server-Pfad-Kombination oder auch nur die Angabe des lokalen Mountpunkts genügt. Mit jedem der beiden folgenden Aufrufe könnte das Verzeichnis »/home« vom Server »sonne« gemountet werden:

```
# Angabe des Servers+ Pfads
root@erde> mount sonne:/ home
# Angabe des lokalen Mountpunkts
root@erde> mount / home
```

Mount-Optionen

Durch gezielten Einsatz einiger Optionen des **mount**-Kommandos lassen sich die Zugriffe auf Daten eines NFS-Verzeichnisses mitunter erheblich beschleunigen. Auch kann das Verhalten im Fehlerfall über Argumente gesteuert werden. Selbstverständlich lassen sich solche Angaben auch in der /etc/fstab dauerhaft nieder schreiben.

Die das NFS betreffenden Optionen sind:

rw, ro

Schreib- und Lesezugriff bzw. Nur-Lese-Zugriff. Beachten Sie, dass die Rechte höchstens weiter eingeschrä
werden können, d.h. ein vom Server nur-lesend-exportiertes Verzeichnis kann lokal nicht zum Schreiben
freigegeben werden, umgekehrt kann die Schreibberechtigung lokal verboten werden, selbst wenn der Serv
diese zuließe

fg

bg

Scheitert der Mountvorgang im ersten Versuch, wird er im Hintergrund (»background«) solange wiederholt, er erfolgreich war oder »rsize« erreicht wurde.

retrans= zahl

Anzahl der Wiederholungsversuche, um einen Mount durchzuführen. Der Default-Wert liegt bei 5

hard

Ein Programm wird während des Zugriffs auf ein NFS-Verzeichnis hängen bleiben, falls der Server zusammenbricht. Nach Wiederanlaufen des Servers fährt das Programm mit seiner Arbeit fort. ein hängend Programm, kann nur unterbrochen werden, wenn die Option »intr« angegeben wurde.

soft

Der Kernel wird, falls der Server eine bestimmte Zeit lang (»retrans*timeo«) nicht antwortet, einen Fehler generieren und die auf den Server wartenden Prozesse informieren. Die Zeitdauer zwischen den Versuchen kann mit »timeo=Sekunden« eingestellt werden.

intr, nointr

Möglichkeit des Abbruchs durch eine Tastenkombination (»interrupt«) bzw, das Verhindern derselben

remount

Aushängen eines Verzeichnisses, um es sofort wieder (beispielsweise mit neuen Optionen) einzuhängen

suid, nosuid

Möglichkeit zur Benutzung des SUID-Bits auf dem eingehängten Dateisystem

retry= zahl

Anzahl der erfolglosen Mount-Versuche (Voreinstellung ist 10000), bis endgültig abgebrochen wird

wsize= zahl

Setzt die Blockgröße beim Schreiben über NFS auf »Bytes« byte. Voreinstellung ist 1024, sollte aber auf 81 gesetzt werden.

rsize= zahl

Setzt die Blockgröße beim Lesen über NFS auf »Bytes« byte. Voreinstellung ist 1024, sollte aber auf 8192 gesetzt werden.

timeo= zahl

Zeitspanne für Wiederholversuche, angegeben in Zehntelsekunden

proto= protokoll

ab Version 3: Angabe des Protokolls (UDP oder TCP)

Network Information System - Der Client

Übersicht

Das Network Information System ist ein Verzeichnisdienst, der die Benutzung bestimmter Konfigurationsdateien von verschiedenen Rechnern aus ermöglicht. Insbesondere in Verbindung mit dem Network File System wird NIS verwendet, um Benutzern netzwerkweit dieselbe Umgebung zur Verfügung zu stellen. Eine ausführliche Darstellung zu den Möglichkeiten des NIS und einen historischen Abriss findet der interessierte Leser im Abschnitt über den NIS-Server.

Die Programme

Die notwendigen Programme zur Administration eines Clients sind **ypbind**, **yppoll** und **ypset**. Zur Arbeit über NIS stehen die Kommandos **ypcat**, **ypchfn**, **ypchsh**, **ypmatch**, **yppasswd** und **ypwhich** zur Verfügung und sollten nach Installation des entsprechenden Paketes (ein typischer Paketname lautet »ypclient...«) im Verzeichnis »/usr/sbin« (die 3 ersten Kommandos) bzw. »/usr/bin« vorhanden sein.

Da es sich beim Network Information Service um einen RPC-Dienst handelt, ist ein aktiver Portmapper erforderlich. Wie dieser zu starten ist, sollte aus dem entsprechenden Abschnitt zum Remote Procedure Call bereits bekannt sein.

Der Verantwortungsbereich

Jeder NIS-Client ist einer so genannten **NIS-Domain** zugeteilt. Und zu jeder NIS-Domain existiert mindestens ein Server, der die Datenbanken für seinen Verantwortungsbereich verwaltet und auf Anfragen der Clients antwortet. Innerhalb eines Netzwerkes können mehrere NIS-Domänen unabhängig voneinander existieren und ein einzelner Server kann mehrere Domänen verwalten.

Damit ein Client weiß, welcher NIS-Domain er zugehört, muss ihm diese vorab bekannt gegeben werden:

```
root@sonne> domainname
(none)
root@sonne> domainname nis.galaxis.de
root@sonne> domainname
nis.galaxis.de
```

Dieser Domainname hat nichts mit dem Internet-Domainnamen zu tun! Er könnte durchaus gleich lauten, ohne dass zwischen beiden eine Wechselwirkung bestehen würde. Der Systemverwalter hat jederzeit das Recht, den eigenen Rechner einer anderen NIS-Domain zuzuordnen.

Die Datei / etc/ nsswitch.conf

»Name Service Switch« bezeichnet einen Mechanismus, nachdem symbolische Rechnernamen in Adressen und umgekehrt umgewandelt werden. Linux übernahm das Vorgehen einer Lösung von SUN, wobei der Name der Datei nicht exakt den Kern trifft, denn /etc/nsswitch.conf konfiguriert weit mehr als die bloße Gewinnung von Rechnernamen und -adressen.

Einem NIS-Client stehen prinzipiell mehrere Wege offen, wie er an eine Information wie bspw. an die IP-Adresse zu

einem symbolischen Rechnernamen gelangen kann. Er könnte den NIS-Server konsultieren oder aber auch die lokale Datei /etc/hosts durchforsten. Und auch eine direkte Anfrage an einen DNS-Server ist denkbar. Für welchen Weg er sich letztlich entscheidet, ist in der Datei »/etc/nsswitch.conf« beschrieben:

```
user@sonne> cat / etc/ nnswitch.conf
...
hosts:     files dns
services:  files
protocols: files

passwd:    nis [NOTFOUND=return] files compat
shadow:    nis [NOTFOUND=return] files compat
group:     nis [NOTFOUND=return] files
...
```

An dieser Stelle verfolgen wir nur einen Auszug der für NIS relevanten Einträge aus obiger Beispieldatei. Eine vollständige Bechreibung der Syntax der Datei finden Sie unter Netzwerk-Grundlagen, Konfigurationsdateien, /etc/nsswitch.conf.

Der erste Eintrag einer Zeile ist der Name der betreffenden Datenbank. Er deckt sich mit den Namen der lokalen Konfigurationsdatei, womit sein Zweck klar sein sollte. Dem Doppelpunkt folgen die Mechanismen, die, in gegebener Reihenfolge, angewandt werden, um eine Information aus der betreffenden Datenbank zu gewinnen. Liefert ein Versuch kein Ergebnis, wird in der Voreinstellung das nächste Verfahren versucht.

So werden im Beispiel durch die Zeile

```
hosts:     files dns
```

Rechnernamen bzw. IP-Adressen zuerst in der lokalen Datei /etc/hosts gesucht (»files«) und nur wenn dort ein entsprechender Eintrag fehlt, wird der Domain Name Service (»dns«) bemüht. »dns« darf einzig in Zusammenhang mit »hosts« verwendet werden.

Informationen zu »services« und »protocols« werden laut obiger Konfiguration einzig durch die jeweiligen lokalen Dateien bereitgestellt. Auch dies ist typisch für Dateien statischer Natur (deren Inhalt sich i.d.R. niemals ändert).

Eine Befragung des NIS-Servers findet in obiger Konfiguration nur für Informationen aus »passwd«, »shadow« und »group« statt. Als Antwort auf eine Anfrage sind **SUCCESS** bei Erfolg, **NOTFOUND**, falls keine entsprechende Information gefunden wurde, **UNAVAIL**, falls der Dienst generell, und **TRYAGAIN**, falls der Dienst temporär nicht erreichbar ist (bspw. bei dessen Überlastung), möglich. Die Semantik für **SUCCESS** ist, dass die Befragung endet (»return«). Für alle weiteren Antworten ist das Fortfahren mit Befragung des nächsten konfigurierten Mechanismus die Voreinstellung (»continue«). In

```
passwd:    nis [NOTFOUND=return] files compat
shadow:    nis [NOTFOUND=return] files compat
group:     nis [NOTFOUND=return] files
```

modifizieren wir das Vorgehen dahingehend, dass eine Anfrage als gescheitert gilt, wenn der NIS-Server den gewünschten Eintrag nicht findet. »[NOTFOUND=return]« ist zu lesen als »Antwortet NIS mit »NOTFOUND«, dann kehre zurück (»return«)" (und schaue nicht mehr in den lokalen Dateien nach). Der Sinn? Ein Rechner könnte so eingerichtet werden, dass sich einige Benutzer selbst dann anmelden können, wenn der NIS-Server ausgefallen ist (Root sollte das stets können!). Bei aktivem Server sind für solche Benutzer jedoch die Informationen des Servers bindend.

Die letzte Option **compat** steuert nicht die Reihenfolge der Befragung, sondern aktiviert die Auswertung einer erweiterten Syntax in den Datenbanken »passwd«, »shadow« und »group«. Auf diese weiter gehenden Konfigurationsmöglichkeiten kommen wir weiter unten im Text zurück.

Die Datei / etc/ host.conf

Handelt es sich bei dem Client um ein älteres System, das noch auf der C-Bibliothek »libc5« (oder gar 4) basiert, spielt die Datei /etc/nsswitch.conf für NIS gar keine Rolle. Womöglich existiert sie nicht einmal.

Bei derartigen Clients ist einzig die Auflösung von Rechnernamen und IP-Adressen zu konfigurieren, wenn sie auch über NIS geschehen soll. Der Resolver der alten C-Bibliothek wird über die Datei /etc/host.conf gesteuert.

Der einzig für NIS relevante Eintrag betrifft die Ergänzung der mit »order« eingeleiteten Zeile um das Schlüsselwort **nis**:

```
user@sonne> cat /etc/host.conf
...
order hosts, nis, bind
...
```

Die Zeile im Beispiel legt fest, dass zunächst in der lokalen Datei »/etc/hosts« nachgeschaut wird (»hosts«). Schlägt die lokale Suche fehl, wird NIS befragt (»nis«) und bei negativen Bescheid von diesem wird letztlich der DNS konsultiert (»bind«; alternativ könnte hier auch »dns« stehen).

Start des Clients

Der NIS-Client wird durch Aufruf des Kommandos **ypbind** gestartet, wobei er den Namen des NIS-Servers in Erfahrung bringen muss, um ihn zu kontaktieren. Zwei Wege stehen einem Client hierfür zur Verfügung:

Serversuche per Broadcast

Den geringsten administrativen Aufwand erfordert die Suche eines für die NIS-Domain zuständigen Servers via Broadcast, d.h. der Client sendet eine Anfrage ins Netz mit der Bitte, ein verantwortlicher NIS-Server möge sich melden. Hierfür genügt die Angabe der entsprechenden Option beim Aufruf von **ypbind**:

```
root@sonne> /usr/sbin/ypbind -broadcast
```

Sind mehrere NIS-Server für eine Domain konfiguriert (ein Master, mehrere Slaves), so sollte jeder der Server auf die Anfrage reagieren. Der Client wird sich bei weiteren Anfragen allerdings stets an den Server wenden, vom dem die Antwort zuerst eintraf.

Die Broadcast-Methode sollten Sie nur in Netzwerken verwenden, die vollständig unter Ihrer Kontrolle sind, denn es ist ein Leichtes, einen fingierten NIS-Server aufzusetzen...

Serverspezifikation per Konfigurationsdatei

Das Kommando **ypbind** liest beim Start seine Konfigurationsdatei **/etc/yp.conf** ein. Sie wird verwendet, um das Vorgehen der Kontaktaufnahme zu einem NIS-Server einer Domain festzuschreiben.

Im einfachsten Fall wird eine Datei »/etc/yp.conf« Einträge folgender Art umfassen:

```
root@sonne> cat /etc/yp.conf
ypserver sonne.galaxis.de
ypserver erde.galaxis.de
ypserver 192.168.100.107
```

Die Beispielkonfiguration benennt drei NIS-Server für die lokale Domain (also die, die zuvor per »domainname« gesetzt wurde). **ypbind** setzt gleichzeitig eine Anfrage an alle drei Server ab und der, der zuerst antwortet, wird zum Server für den Client.

Die Datei »/etc/yp.conf« kann zwei weitere Arten von Einträgen enthalten:

```
domain nis.galaxis.de server sonne.galaxis.de
domain andere.nis.domain server anderer.nis.server
```

Für unterschiedliche NIS-Domains werden die Server angegeben. Es sind mehrere Angaben zu einer Domain zulässig. Durch eine solche Konfiguration kann ein Client seine Mitgliedschaft zu einer Domain ändern, zu einem Zeitpunkt kann er jedoch nur einer Domain angehören!

```
domain nis.galaxis.de broadcast
```

Ein NIS-Server für die Domain »nis.galaxis.de« wird per Broadcast ermittelt. Ein solcher Eintrag hat Vorrang vor einer Zeile, die einen konkreten Server für eine Domain benennt.

Die Arbeitsweise von ypbind

Bislang könnte der Eindruck entstanden sein, »ypbind« ist der NIS-Client. Dem ist nicht so, genau genommen existiert kein Clientprogramm sondern ein Rechner ist ein NIS-Client, wenn er Funktionen der Standard-C-Bibliothek aufruft, welche sich der NIS-Funktionalität bedienen. Solche Funktionen erfordern Informationen über die Bindung an einen NIS-Server, den sie letztlich kontaktieren, und jene Informationen liefert »ypbind«. Das Kommando ist somit die zentrale Instanz eines NIS-Clients.

Wenn »ypbind« startet, sucht es zunächst nach einem Server für die gesetzte NIS-Domain. Die Broadcast-Option der Kommandozeile genießt dabei Vorrang vor der Konfiguration der Datei »/etc/yp.conf«. Nach erfolgreicher Bindung an einen Server, sendet »ypbind« alle 20 Sekunden eine YPPROC-DOMAIN-Anforderung an den Server, um dessen Aktivität zu überprüfen. Antwortet der Server nicht oder meldet er, dass er für die Domain nicht länger zuständig ist, so sucht »ypbind« selbsttätig nach einem anderen Server (Broadcast oder gemäß Konfiguration in »/etc/yp.conf«). In 15-Minuten-Intervallen testet »ypbind«, ob ein anderer konfigurierter Server eventuell schneller antwortet als der aktuell gebundene. Falls ja, wechselt »ypbind« die Bindung zum neuen Server.

Die Optionen von ypbind

Neben »broadcast« versteht »ypbind» eine Reihe weiterer Kommandozeilenoptionen (Auswahl):

-c

»ypbind« prüft die Syntax der Datei »/etc/yp.conf« und endet anschließend.

-d | --debug

Start im Debug-Modus; nützlich bei der Fehlersuche.

-broadcast

Ein NIS-Server wird per Broadcast gesucht.

-no-ping

Die Überprüfung, ob der NIS-Server noch antwortet, wird abgeschaltet. Die Option ist bei Modemverbindun sinnvoll, um eine permanente Einwahl zu verhindern.

-f <Konfigurationsdatei>

»ypbind« verwendet die angegebene Konfigurationsdatei anstatt »/etc/yp.conf«.

-p <Port>

»ypbind« verwendet den angegebenen Port; im Zusammenhang mit Firewalls hilfreich.

-ypset

-ypsetme

Erlaubt das dynamische Ändern des zu verwendenden NIS-Servers nur durch den Administrator des lokalen Rechners (siehe »Konfiguration zur Laufzeit«).

Ein einfacher Test

Nach obigen Anpassungen und (Neu)Start von »ypbind« kann eine Überprüfung nicht schaden, ob die Verbindung zu einem NIS-Server steht. Bemühen Sie zunächst das Kommando **ypwhich**, um zu erfahren, zu welchem NIS-Server die Bindung besteht:

```
user@sonne> ypwhich
sonne.galaxis.de
```

Ob auch die Dateien korrekt dargeboten werden, erfahren Sie, indem Sie eine Liste der Nicknamen aller exportierter Datenbanken anfordern:

```
user@sonne> 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"
```

Konfiguration zur Laufzeit

ypbind nimmt beim Start Kontakt zu einem für die NIS-Domain zuständigen NIS-Server auf und hält die Bindung zu diesem aufrecht, bis entweder der Server ausfällt oder im Zuge der Reaktionszeittests ein anderer Server schneller antwortet und damit den Zuschlag erhält.

Das Kommando, um »ypbind« explizit einen neuen Server unterzuschieben, ist **ypset**. Mit

```
root@sonne> ypset melmac
```

wird »melmac« zum neuen NIS-Server für den Client »sonne«, allerdings nur, wenn »ypbind« tatsächlich eine Verbindung zu diesem herstellen konnte. Ansonsten begibt sich »ypbind« nach den schon bekannten Regeln auf die Suche nach einem gültigen Server.

Um den Server einer anderen als der lokal konfigurierten NIS-Domain zu setzen, ist diese Domain anzugeben:

```
root@sonne> ypset -domain andere.nis.domain venus
```

Schließlich kann auch der Rechnername angegeben werden, falls die Änderung auf einem anderen als dem lokalen Rechner vollzogen werden soll (dann muss das dortige »ypbind« aber mit der Option »-ypset« gestartet worden sein):

```
root@mars> ypset -domain nis.galaxis.de -h sonne erde
```

Änderungen an lokalen Dateien

Änderungen sind einzig an den Dateien der Benutzerverwaltung erforderlich.

Für Linuxsysteme, die auf früheren Versionen der Standard-C-Bibliothek basieren (libc4 und 5), ist die Datei /etc/passwd zu modifzieren, um Benutzerkonten vom NIS-Server verwenden zu können. Für die aktuellen Versionen der glibc2-Bibliothek kommen Anpassungen an den Dateien /etc/shadow und /etc/group hinzu.

Datenimport ohne lokale Zugangskontrolle

Um die Datensätze der oben genannten Dateien unverändert vom Server zu übernehmen, genügt das Einfügen einer Zeile mit einem Pluszeichen in obige Dateien:

```
# ab glibc-2:
root@sonne> vi /etc/passwd
...
user:x:501:100:Testuser:/home/user:/bin/bash
newuser:x:502:100:Neuer Benutzer:/home/newuser:/bin/bash
+
```

Beachten Sie, dass »normale« Zeilen, die solchen NIS-Einträgen folgen, keine Beachtung mehr finden, d.h. NIS-Datensätze sollten stets am Ende der Dateien angefügt werden.

Bei Verwendung der Standard-C-Bibliotheken der Versionen 4 und 5 müssen dem Pluszeichens 6 Doppelpunkte folgen:

```
# libc4 und libc5:
root@sonne> vi /etc/passwd
...
user:x:501:100:Testuser:/home/user:/bin/bash
newuser:x:502:100:Neuer Benutzer:/home/newuser:/bin/bash
+:::::
```

Der Grund für diese Syntax liegt in der steten Verwendung des Compat-Modus durch die alten C-Bibltiotheken (siehe nächster Abschnitt).

Datenimport mit lokaler Zugangskontrolle (Compat-Modus)

Die beschriebenen Mechanismen funktionieren mit der Standard-C-Bibliothek »glibc2« nur, wenn die korrespondierenden Einträge in der Datei nsswitch.conf das Schlüsselwort **compat** enthalten.

Benutzerinformationen sind sensible Informationen und die netzwerkweite Allgemeingültigkeit aller Parameter ist nicht in jedem Fall erwünscht oder sinnvoll. Z.B. könnte auf einem bestimmten Rechner nur eine konkrete Shell installiert sein. Sollte dennoch jeder auf dem NIS-Server eingetragene Benutzer sich dort anmelden dürfen, sollte sicher gestellt sein, dass auch diese Shell als Loginshell gesetzt ist - unabhängig von der Einstellungen in der NIS-Datenbank. Auch kollidierende Benutzer-/Gruppen-IDs lassen sich lokal korrigieren. Und im Sinne der Sicherheit enorm wichtig: Bestimmten Benutzern kann explizit der Zugang verwehrt werden.

Beginnen wir mit dem Sperren eines Zugangs. Dazu ist in die lokale Datei /etc/passwd der Benutzername mit einem vorangestellten Minuszeichen aufzunehmen:

```
root@sonne> vi /etc/passwd
...
# Benutzer »alf« darf nicht (Version 1)
-alf
+
```

Eine andere Variante, das Anmelden eines Benutzers zu unterbinden, ist die Vergabe eines ungültigen Passworts,

womit wir ein erstes Beispiel haben, wie ein vom Server gelieferter Wert lokal überschrieben werden kann:

```
root@sonne> vi /etc/passwd
...
# Benutzer »alf« darf nicht (Version 2)
+alf:!:::::
+
```

In obigen Beispielen ermöglichen wir durch die abschließende Zeile mit dem einzelnen Pluszeichen, das sämtliche Benutzer, die nicht zuvor explizit erwähnt wurden, unverändert vom NIS-Server übernommen werden. Wenn Sie auf letztere Zeile verzichten, erhalten demzufolge nur die zuvor aufgeführten Benutzer Zugang zum lokalen System.

Auf einem Mailserver muss zwar zu jedem Mailkonto dessen Benutzer eingetragen sein, dessen Anmeldung ist jedoch zumeist unerwünscht. Um nun allen auf dem NIS-Server eingetragenen Benutzern zwar das Mailkonto einzurichten, das Login zu verwehren, wird auf dem Mailserver (als NIS-Client) die Login-Shell auf »/bin/false« gesetzt:

```
root@sonne> vi /etc/passwd
...
# Benutzer erhalten eine ungültige Shell
+::::::/bin/false
```

Bislang galten Einstellungen für einen einzelnen Benutzer (+alf:::::) oder für alle Benutzer (+::::::). Flexibler ist das Schema durch Verwendung von Netzgruppen. Eine solche wird durch ein vorangestelltes »@« gekennzeichnet. Sie muss in der Datei netgroup konfiguriert sein:

```
root@sonne> grep NetAdmins /etc/netgroups
NetAdmins (sonne,user1,) (sonne,user2,) LocalAdmins
root@sonne> vi /etc/passwd
...
# Bezug auf eine Netzgruppe
+NetAdmins::666:100::/home/admingroup:/bin/ksh
```

Die Programme

Etliche Programme arbeiten im Verborgenen mit den NIS-Mechanismen zusammen, ohne dass der Benutzer dies merkt. So offenbart »login« niemandem, ob eine Benutzerauthentifizierung lokal oder über NIS erfolgt. Und auch die zahlreichen Netzwerkprogramme halten sich mit Hinweisen bedeckt, wie eine Rechneradresse gewonnen wurde.

Doch nicht alle Programme können von sich aus entscheiden, ob sie Aktionen via NIS oder über lokale Funktionen erledigen sollen. Wünscht ein Benutzer bspw. sein Passwort zu ändern, so wäre das lokal als auch auf dem NIS-Server denkbar. Bei ersterem beträfe die Änderung einzig die Anmeldung am Client-Rechner, bei letzterem gelte das Passwort fortan auf allen NIS-Clients.

Die »Arbeitsprogramme«

Genau genommen handelt es sich um ein einziges Programm, das entweder durch Optionen *oder* durch Aufruf unter anderen Namen die Funktionen verschiedener lokaler Programme übernimmt.

```
yppasswd [-f] [-l] [-p] [Benutzer]
ypchfn [Benutzer]
ypchsh [Benutzer]
```

yppasswd ist dabei das eigentliche Programm, das, wie der Name schon andeutet, zum Setzen des Passwortes in der NIS-Datenbank dient.

```
user@erde> yppasswd
Ändere NIS Account Informationen für user auf sonne.galaxis.de.
Geben Sie bitte Ihr altes Passwort ein:
Ändere NIS Passwort für user auf sonne.galaxis.de
Geben Sie bitte ein neues Passwort ein:
Bitte geben sie das neue Passwort erneut ein:

Das NIS Passwort wurde geändert auf »sonne.galaxis.de«.
```

Beachten Sie, dass das neue Passwort erst wirksam wird, nachdem auf dem NIS-Server die NIS-Maps aktualisiert wurden.

Mit der Option **-f** arbeitet »yppasswd» wie **ypchfn**, womit das so genannte GECOS-Feld modifiziert wird:

```
user@erde> ypchfn
Ändere NIS Account Informationen für user auf sonne.galaxis.de.
Geben Sie bitte Ihr Passwort ein:

Ändere vollen Namen für user auf sonne.galaxis.de.
Zum Akzeptieren, einfach Return drücken. Zum Löschen, tippen
Sie "none" ein.
Name [Testuser]: Beispielbenutzer
Büro [xxx]: none
Dienst. Telefon []: 0815 424242
Privat. Telefon []:
Fehler während dem Ändern der GECOS Informationen.
Die GECOS Informationen wurde nicht geändert auf »sonne.galaxis.de«.
```

Die Änderungen wurden abgewiesen? Dann hat Ihr Administrator - aus welchen Gründen auch immer - die Modifikation des Feldes unterbunden, indem er beim Start des zuständigen Daemons rpc.yppasswdd auf die Option »-e chfn« verzichtete.

Mit der Option **-l** arbeitet »yppasswd» wie **ypchsh**, womit die Login-Shell des Benutzers konfiguriert werden kann. Auch hierzu muss der Server-Administrator den Daemon rpc.yppasswdd mittels der Option »-e chsh« dazu ermächtigt haben.

```
user@erde> ypchsh
Ändere Login-Shell für user.
Password:
Enter the new value, or press return for the default
    Login Shell [/bin/tcsh]: /bin/bash
Shell changed.
```

Das Beispiel offenbart eine häufige Schwäche der Internationalisierung von Linux: Die unvollständige Übersetzung der Begleittexte. Aber das soll uns hier nicht weiter stören.
Wichtig beim Ändern der Shell ist, dass Sie diese unbedingt mit *vollständigem Pfad* angeben; sonst wird Ihnen das Anmelden mit dem nächsten Versuch nicht mehr möglich sein (weil die PATH-Variable der Shell noch nicht existiert).

Die Option »-p« von »yppasswd» ist eigentlich überflüssig, da die damit erzwungene Änderung der Passwortes ohnehin die Voreinstellung ist.

Der Administrator hat die Möglichkeit, durch Angabe des Benutzernamens als Kommadozeilenoption von »yppasswd», »ypchfn» bzw. »ypchsh», den jeweilgen Parameter eines jeden Benutzers zu ändern.

Die Diagnoseprogramme

Einige der hier vorgestellten Programme begegneten Ihnen bereits im Laufe des Abschnitts. Sie werden i.d.R. nur während des Einrichtens von Client und Server benötigt, um deren korrekte Arbeit zu überprüfen.

```
ypcat [ -kt ] [ -d Domain ] Mapname
ypcat -x
```

ypcat ermöglicht das Betrachten der Inhalte von Maps. Zwingend anzugeben ist der Name der Map, womit sie in der aktuellen NIS-Domain gesucht wird. Mit »-d Domain« kann die Map einer jeden anderen NIS-Domain betrachtet werden (sofern der NIS-Client die Rechte dazu besitzt).

Da zu einer NIS-Datenbank oft mehrere Maps mit verschiedenen Suchschlüsseln existieren, wird häufig für die gebrächlichste ein kurzer Aliasname gewälht (Bspw. wird die Passwortdatei nach Benutzerkennzeichen »passwd.byname« und nach Benutzer-IDs »passwd.byuid« sortiert gespeichert. Der Alias »passwd« verweist auf »passwd.byname«.). Welche Aliasse existieren, verrät »ypcat -x«.

Denkbar (aber unüblich) sind Konfigurationen, in denen ein Aliasname gleich einem konkreten Mapnamen lautet. Um eine solche Map dennoch betrachten zu können, ist die Option »-t« erforderlich, die die Aliasbindungen unterdrückt.

Die Option »-k« schreibt nochmals den Schlüsselwert explizit vor jeden Datensatz.

```
ypmatch [ -kt ] [ -d Domain ] Schlüssel ... Mapname
ypmatch -x
```

Im Unterschied zu »ypcat« beschränkt **ypmatch** die Auflistung auf die Datensätze, die mit den angegebenen Suchschlüsseln (auch mehrere) in der Map übereinstimmen. Die Optionen besitzen dieselbe Bedeutung wie unter »ypcat« beschrieben.

```
yppoll [ -h Rechner ] [ -d Domain ] Mapname
```

yppoll zeigt die Version und den Namen des NIS-Servers zu der angegebenen Map an. Mit »-h Rechner« kann ein konkreter Server benannt werden. Ohne diese Option wird der Server, den »ypwhich« liefert, befragt. Mit »-d Domain« werden die Informationen zu angegebenen anstatt zur aktuellen NIS-Domain geliefert. Als Mapname werden keine Aliase akzeptiert.

```
user@erde> / usr/ sbin/ yppoll passwd.byname
Domain "galaxis.de" wird unterstützt.
Der Zeitstempel der Map »passwd.byname« ist 1029506197. [Fri Aug 16 15:56:37 2002]
Der Master Server ist »sonne.galaxis.de«.
```

```
ypwhich [ -d Domain ] [ -Vn ] [ Rechnername ]
ypwhich [ -d Domain ] [ -t ] -m [ Mapname ]
ypwhich -x
```

ypwhich zeigt den zuständigen NIS-Server an. Die Optionen »-d Domain«, »-t« und »-x« besitzen dieselbe Bedeutung wie beim Kommando »ypcat« beschrieben.

Wird ein Mapname angegeben (Option »-m [Mapname]«), liefert »ypwhich« den Namen des Servers, der diese Map anbietet. Bei alleiniger Angabe von »-m« erhalten Sie eine Auflistung für alle existierenden Maps.

Ein Rechnername auf der Kommandozeile bewirkt, dass die Anfrage von diesem Rechner aus gestellt wird, d.h. die Frage lautet: »Welcher NIS-Server ist für diesen Rechner zuständig?«.

In der Voreinstellung wird nach dem NIS-Server gefahndet, der das NIS-Protokoll der Version 2 unterstützt. Mit »-V1« fragen sie nach dem zuständigen NIS-Server der Version 1 (unter Linux unterstützen die bekannten Implementierungen beide Versionen, sodass Sie dieselbe Antwort erhalten werden).

Administrative Programme

Zum Ausführen dieser Programme bedarf es der Rechte des Administrators.

Bereits ausführlich diskutiert wurden die Kommandos domainname zum Setzen des NIS-Domainnamens und ypbind, das die Bindung zum NIS-Server herstellt und überwacht. Fehlt noch die Beschreibung zu **ypset**, das »ypbind« anweist, sich an einen konkreten NIS-Server zu binden. Das klappt jedoch nur, wenn »ypbind« mit der Option »-ypset« (bzw. »-ypsetme«) gestartet wurde.

```
ypset [ -d Domain ] [ -h Rechnername ] NIS-Server
```

Bei alleiniger Angabe des NIS-Servers werden als Domainname die aktuelle NIS-Domain und als Rechnername der aufrufende Clientrechner angenommen. Bevor »ypbind« tatsächlich die aktuelle Bindung wechselt, wird überprüft, ob der neue NIS-Server erreichbar ist:

```
root@erde> ypset wega.galaxis.de
Auf »wega.galaxis.de« läuft kein ypserv.
```

Im Beispiel ist auf dem angegebenen Rechner kein NIS-Server aktiv. Die alte Bindung von »ypbind« bleibt erhalten.

Mit der Option »-d Domain« wird die Bindung zu einem Server für die angegebene NIS-Domain vorgenommen; mit »-h Rechnername« betrifft dies den ypbind-Prozess auf dem angegebenen Rechner. Jener muss zwingend mit der Option »-ypset« gestartet worden sein.

Domain Name Service - Der Client

Übersicht

Vermutlich haben Sie auf die Dienste des *Domain Name Service* bereits zugegriffen, ohne seine Existenz wahrzunehmen. Surfen Sie im Internet? Dann verwenden Sie sicher Angaben wie www.linuxfibel.de? Aber hinter dieser Angabe verbirgt sich nichts anderes als ein symbolischer Name für eine IP-Adresse. Und wer ermittelt die Adresse zum Namen? Der Domain Name Service.

Und elektronische Post ist ohne den DNS undenkbar. Denn nur ein DNS-Server weiß, wer der zuständige Mailserver für einen Empfänger ist.

Der Domain Name Service arbeitet transparent. Sie als Anwender werden von seiner Existenz, ist er erst einmal korrekt administriert, nichts bemerken, es sei denn, er verweigert einmal seinen Dienst... Einen Rechner zu einem DNS-Client zu ernennen, ist unter Linux ein einfaches Unterfangen, da die Standard-C-Bibliothek, ohne die kein Linuxsystem laufen könnte, alle notwendigen Routinen in Form des so genannten **Resolvers** bereits in sich birgt. Wenige Handgriffe genügen und der Resolver wird die Funktionalität des DNS gewährleisten.

Neben der Anpassung des Resolvers werden wir Ihnen die wichtigsten Werkzeuge rund um den Domain Name Service, die *Bind-Utils*, vorstellen. DNS arbeitet auch ohne sie, aber hin und wieder möchten Sie vielleicht doch konkrete Informationen zu einem gegebenen Rechnernamen oder einer IP-Adresse erfahren.

Der Vollständigkeit halber muss erwähnt werden, dass auf die Konfiguration eines DNS-Clients auch komplett verzichtet werden kann, falls der Rechner mindestens als so genannter Caching-Only-DNS-Server konfiguriert wird. Ein solcher recherchiert jeden Namen nur einmalig und hält die Informationen in einem Cache (»Zwischenspeicher«) vor. Wie lange die dortigen Daten ihre Gültigkeit behalten, ist konfigurierbar. Ein solches Vorgehen ist i.d.R. effizienter als ein DNS-Client, der zu jeder Namensauflösung einen externen Server kontaktiert. Der lokale Nameserver wird bspw. verwendet, wenn die Konfigurationsdatei des Resolvers fehlt oder keinen Nameserver-Eintrag enthält. Interessieren Sie sich für eine derartige Konfiguration, so finden Sie im Abschnitt zum DNS-Server die notwendigen Hinweise.

Der Resolver

Beim Resolver handelt es sich nicht um einen eigenen Prozess sondern um eine Bibliothek von Routinen, die von Prozessen der Netzwerkprogramme gerufen werden. Die relevante Konfigurationsdatei für den Resolver selbst ist /etc/resolv.conf.

Bezieht der Client die Daten seiner Netzwerkkonfiguration inklusive der Nameserver-Optionen via DHCP oder über das Point-To-Point-Protocol (dies ist bei Verbindungen über ein Modem der Fall), so wird die Datei »resolv.conf« i.d.R. vom entsprechenden Client-Prozess selbst erzeugt. Sie sind damit aus dem Schneider und müssen gar nichts konfigurieren (außer die Einwahl via PPP an sich bzw. den DCHP-Client).

Starten wir wiederum mit einer einfachen Beispielkonfiguration einer Datei »resolv.conf«:

```
# /etc/resolv.conf

search saxsys.de linuxfibel.de galaxis.de

nameserver 134.109.192.18
nameserver 192.168.85.1
```

search

Aus reiner Bequemlichkeit bevorzugt der Anwender oft die Angabe eines Rechnernamens ohne angehängte

Domainnamen (»erde« anstatt »erde.galaxis.de«). Ein solcher »Kurzname« ist im DNS-Namensraum kaum eindeutig, weshalb der Resolver versucht, diesen zu einem vollständigen Namen zu expandieren. Hierzu erweitert er jeden Rechnernamen, der *keinen Punkt enthält*, der Reihe nach um die unter **search** konfigurierten Suffixe. Kann ein resultierender Rechnername aufgelöst werden, endet die Suche erfolgreich der Beispielkonfiguration würde ein Rechnername »erde« zuerst zu »erde.saxsys.de«, dann zu »erde.linuxfibel.de« und zuletzt zu »erde.galaxis.de« erweitert werden. Erst letzterer Rechnername ergibt e gültigen, d.h. im DNS-Namensraum existenten, Namen. Beachten Sie, dass die **search**-Liste auf maximal 6 Domains mit zusammen 256 Zeichen beschränkt ist. Der **search**-Mechanismus arbeitet vor allem bei Zugrif entfernte DNS-Server extrem langsam. Eine gesetzte Umgebungsvariable **LOCALDOMAIN**, die eine kommaseparierte Liste von Domainnamen enthält, überschreibt die Angaben einer **search**-Liste. Vergleiche Sie auch die Anmerkungen zum unten beschriebenem **domain**-Eintrag.

nameserver

Für jeden zu befragenden Nameserver bedarf es eines eigenen **nameserver**-Eintrags in der Datei »/etc/resolv.conf«. Entsprechend der Reihenfolge der Einträge werden die Nameserver nacheinander kontaktiert, bis der erste erfolgreich eine Anfrage beantwortet. In den gängigen Implementierungen sind maximal drei Nameserver-Angaben zulässig. Fehlt ein **nameserver**-Eintrag, so wird die Anfrage an den lok Rechner gerichtet. Dieser muss zumindest als Caching-only-Nameserver konfiguriert sein (sonst schlägt jed Anfrage fehl).

Die beiden »gängigen« Einträge der Datei »/etc/resolv.conf« sind damit benannt; die weiteren Möglichkeiten beschreibt folgende Zusammenstellung:

domain

domain besitzt eine ähnliche Aufgabe wie **search**. Tatsächlich sollten nicht beide Einträge in einer »/etc/resolv.conf« Verwendung finden. Ist dem dennoch so, gilt der zuletzt gefundene Eintrag aus der Date Im Unterschied zu **search** enthält **domain** nur den Namen der lokalen Domain. Der Resolver wird jeden Rechnernamen, der keinen Punkt enthält, um diesen Domainnamen erweitern. Fehlen sowohl ein **domain**- auch ein **search**-Eintrag, so wird der durch *gethostbyname()* gelieferte Name verwendet.

sortlist

Verfügt ein Rechner über mehrere IP-Adressen, so kann eine Anfrage beim Resolver zu einer Liste von mehreren Adressen führen, deren interne Reihenfolge unspezifiziert ist. Um die Bevorzugung bestimmter Adressen zu erzwingen, können im **sortlist**-Eintrag die Adressen der Netzwerke angegeben werden, aus de Bereich Adressen zuvorderst in der Ergebnisliste zu platzieren sind.
Betrachten Sie als Beispiel ein umfangreiches lokales Netz, in dem mehrere Rechner über Schnittstellen zu anderen Netzen verfügen. Hier wäre es sicher effizient, wenn die Suche nach Adressen der lokalen Rechner »interne« Adresse liefern würde, um umfangreiches »Routen« zu »äußeren« Adressen zu unterbinden.

options

Derzeit kann das Verhalten des Resolvers über zwei Optionen beeinflusst werden.
debug aktiviert erweiterte Statusmeldungen des Resolvers, die eventuell zur Fehlersuche nützlich sein könn Mit **ndots:*n*** wird die Regel manipuliert, wie viele Punkte ein Rechnernamen enthalten muss, um das Anfüg der Standarddomain zu unterbinden. Die Voreinstellung lautet »ndots:1«, was letztlich dazu führt, dass die Standarddomain einzig angefügt wird, wenn der Rechnername keinen Punkt enthält oder - anders ausgedrü - enthält der Rechnername mindestens einen Punkt, so wird er nicht um die Standarddomain erweitert.

Und nochmals möchten wir unterstreichen, dass eine nicht existente oder eine leere Datei »resolv.conf« dazu führt, dass implizit als Nameserver der lokale Rechner angenommen wird und als Standarddomain die lokale Domain (die hoffentlich *gethostbyname()* liefert). Diese Konfiguration entspricht genau der client-seitigen Konfiguration eines Caching-Only-Nameservers.

Nslookup

Nslookup ist traditioneller Bestandteil der BIND-Software und dient (einzig) als Debugging-Werkzeug bei der Fehlersuche. Es erlaubt die direkte Abfrage eines Nameservers und Zugriff auf nahezu jede Information. Ab BIND

Version 9 ist Nslookup nur noch aus Kompatibilitätsgründen enthalten, da dessen Funktionalität komplett vom neueren dig übernommen wurde.

Kommandozeilenmodus

Im Kommandozeilenmodus wird **nslookup** mit dem aufzulösenden Rechnernamen aufgerufen:

```
user@sonne> nslookup www.linuxfibel.de

Note: nslookup is deprecated and may be removed from future releases. Consider using the `dig' or `host' programs instead.
Run nslookup with the `-sil[ent]' option to prevent this message from appearing.

Server:         10.0.0.2
Address:        10.0.0.2 # 53
www.linuxfibel.de    canonical name = www.fibel.de.
www.fibel.de         canonical name = meine.linux.fibel.de.

Name:       meine.linux.fibel.de.
Address:    194.180.239.13
```

Bei erfolgreicher Recherche antwortet **nslookup** mit Namen und Adresse des antwortenden Servers, der die Anfrage auflöste, und mit der gesuchten Adresse (oder mehrere Adressen, falls der Rechner über mehr als eine IP verfügt).

Zum »reverse lookup« rufen Sie das Kommando einfach mit der zu recherchierenden IP-Adresse auf.

Zur Unterdrückung der Warnung ob der »Veralterung« des Werkzeugs, können Sie die Option **-sil** bemühen.

Seine Stärken spielt **nslookup** allerdings erst im interaktiven Modus aus. Die dort verfügbaren Modifikatoren stehen zwar ebenso im Kommandozeilenmodus zur Verfügung (bspw. »nslookup **type= mx** www.google.de«), jedoch ist dieses Verfahren höchst ungeöhnlich, weshalb wir auf die Modifizierer erst im folgenden Abschnitt eingehen.

Interaktiver Modus

Den interaktiven Modus von **nslookup** erreichen Sie durch bloße Eingabe des Kommandos ohne jegliche Optionen. Beenden lässt sich das Programm durch Eingabe von **exit** oder mittels der Tastenkombination **[Ctrl][D]**.

Um das im Kommandozeilenmodus angeführte Beispiel interaktiv nachzuvollziehen, sind folgende Eingaben erforderlich:

```
user@sonne> nslookup
Note: nslookup is deprecated and may be removed from future releases. Consider using the `dig' or `host' programs instead.
Run nslookup with the `-sil[ent]' option to prevent this message from appearing.

Default Server:         10.0.0.2
Address:        10.0.0.2 # 53

> www.linuxfibel.de
Server:         10.0.0.2
Address:        10.0.0.2 # 53

www.linuxfibel.de    canonical name = www.fibel.de.
www.fibel.de         canonical name = meine.linux.fibel.de.

Name:       meine.linux.fibel.de.
Address:    194.180.239.13

> exit
```

Dig

```
user@sonne> dig www.linuxfibel.de

; <<>> DiG 9.1.3 <<>> www.linuxfibel.de
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 13063
;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.linuxfibel.de.            IN   A

;; ANSWER SECTION:
www.linuxfibel.de.      53411   IN   CNAME   www.fibel.de.
www.fibel.de.           53411   IN   CNAME   meine.linux.fibel.de.
meine.linux.fibel.de.   53411   IN   A       184.180.239.13

;; AUTHORITY SECTION:
linuxfibel.de.          53411   IN   NS      ns.dns-dsi.de.
linuxfibel.de.          53411   IN   NS      ns2.dsi.net.
linuxfibel.de.          53411   IN   NS      ns.dsi.net.

;; ADDITIONAL SECTION:
ns.dns-dsi.de.          85595   IN   A       213.83.53.194
ns2.dsi.net.            80799   IN   A       213.187.77.174
ns.dsi.net.             81978   IN   A       213.83.36.58

;; Query time: 3 msec
;; SERVER: 10.0.0.2#53 (10.0.0.2)
;; WHEN: Tue May 14 16:59:21 2002
;; MSG SIZE rcvd: 218
```

Dig (Domain Information Groper) ist das Nachfolgetool von nslookup. Wie im Beispiel zu sehen ist, bekommt der User einiges mehr an Informationen heraus, als beim alten Auflösevorgang. Hierbei wird auch angezeigt, welcher Autorität die Zone untersteht, sowie eine gleichzeitige zusätzliche Auflösung aller Nameserver.

Host

Ein weiteres Diagnose-Werkzeug ist host, das per Voreinstellung lediglich eine verkürzte Ausgabe der vorherigen Tools darstellt, sich jedoch mit diversen Schaltern und Parametern zu einem sehr mächtigen Werkzeug verwandeln kann.

```
user@sonne> host www.linuxfibel.de

www.linuxfibel.de. is an alias for www.fibel.de.
www.fibel.de. is an alias for meine.linux.fibel.de.
meine.linux.fibel.de. has address 194.180.239.13
```

Booten und Konfiguration im Netz

Übersicht

BOOTP

Sowie im Linuxumfeld bootpd und dhcpd als Bootp-Server geeignet sind, existieren auf Clientseite mehrere Programme, die Bootp-Anfragen stellen können. Bezogen auf ihre Verbreitung reduziert sich die Auswahl auf **bootpc**, das im selben Paket enthalten ist, wie der Daemon bootpd und auf **pump**, das die RedHat-basierten Distributionen zur Verfügung stellen. RedHat beschreitet den Weg, die Bootp-Anfragen durch den DHCP-Server abhandeln zu lassen, vielleicht resultiert hieraus die Verwendung eines eigenen Clientprogramms.

Testlauf mit bootptest

bootptest ist im Umfang des bootp-Pakets enthalten und zumindest bei der Suche nach Fehlern recht nützlich. Das Kommando sendet in Sekundenabständen BOOTPREQUESTS an einen angegebenen Server und gibt dessen Antwort - falls sie eintraf - aus. Das Programm setzt allerdings bereits eine eingerichtete Netzwerkverbindung voraus, zwar ist es möglich mit der Option **-h** eine Anforderung auf Basis der Hardwareadresse abzusenden, jedoch antwortet der Server nicht von Haus aus mit einem Broadcast, sondern mit einem an die IP des Clients gerichteten Paket. Ist die Schnittstelle des Clients allerdings noch nicht auf diese Adresse konfiguriert, wird der Kernel das Paket ablehnen.

bootptest hilft letztlich nur, um die Verfügbarkeit eines Bootp-Servers zu testen:

```
root@sonne> bootptest erde
bootptest: version 2.4.3
Sending to 192.168.100.100 (request) htype:0 hlen:0 xid:1457 C:192.168.100.99 vend-rfc1395
Recvd from 192.168.100.99 (reply) htype:0 hlen:0 xid:1457 C:192.168.100.99 Y:192.168.100.99 S:192.168.100.99
sname:"sonne" file:"/00:00:1C:D9:4C:C5" vend-rfc1395 SM:255.255.255.0 DNS:192.168.100.99 DNAM:"galaxis.de"
```

Der Client bootpc

Bevor wir uns der notwendigen Vorarbeit widmen, sollen die wichtigsten Optionen des Kommandos genannt werden.

```
bootpc [--dev device] [--debug] [--server addr] [--hwaddr addr] [--returniffail] [--timeout seconds] [--serverbcast] [--help]
[...]
```

--debug

Ausführliche Meldungen über die einzelnen Schritte

--dev Device

Name des Netzwerkdevices, über das die Kommunikation laufen soll. Wichtig ist die Angabe, wenn mehrere Geräte (Netzwerkkarten, Modems,...) vorhanden sind.

--help

Hilfe zum Programm

--hwaddr Hardwareadresse

Die Anfrage wird unter der angegebenen Hardwareadresse gestellt

--returniffail

Programmabbruch erfolgt bei einem Fehler, ansonsten wird ein Tastendruck erwartet

--server < Server-IP>

Adresse des zu befragenden Bootp-Servers (ohne die Angabe wird per Broadcast nach einem gesucht)

--serverbcast

Weist den Bootp-Server an, die Antwort als Broadcast zu senden. Zur Erstausstattung eines Clients mit eine Adresse ist die Angabe zwingend erforderlich!

--timeoutwait Sekunden

Wie lange soll maximal auf eine Antwort gewartet werden?

Der typische Zeitpunkt der Anwendung von **bootpc** ist im Verlauf des Bootvorgangs des Rechners. Häufigstens Ansinnen wird wohl der Bezug einer IP-Adresse von einem Server sein, was allerdings ein eingerichtetes Netzwerk bedingt. Der logische erste Schritt wird somit die Konfiguration der Netzwerkkarte sein. Hierfür benötigen wir jedoch eine IP-Adresse, die wir noch nicht kennen (können) - sonst müssten wir sie nicht erst anfordern...

Willkürlich eine IP-Adresse zu wählen kann scheitern, falls diese sich mit einer existierenden überschneidet. Sicherer ist daher die Wahl einer »ungütigen« IP-Adresse wie "0.0.0.0". Tatsächlich gibt sich **ifconfig** damit zufrieden:

```
root@sonne> ifconfig eth0 0.0.0.0 up
```

Das Symbol **eth0** im Beispiel bezieht sich auf eine Ethernet-Karte und kann verwendet werden, wenn die Karte entweder fest im Kernel eingebunden ist oder aber in der Datei /etc/modules.conf ein Alias unter dem Namen auf das tatsächliche Device existiert.

Wenn **bootpc** nun einen Broadcast aussendet, dann muss der Kernel wissen, wohin er dieses zu senden hat. Er benötigt in seiner IP-Routing-Tabelle einen Eintrag, der Pakete mit »unbekannten« Adressen an ein Device leitet. Für unser Beispiel heißt das, dass alle Pakete mit unbekanntem Empfänger an das Device **eth0** zu senden sind:

```
root@sonne> route add default eth0
```

Schließlich können die Parameter angefordert werden. Unbedingt erforderlich ist die Option **--serverbcast**, um dem antwortenden Bootp-Server mitzuteilen, er solle sein Paket doch als Broadcast senden und nicht an die dem Client zugedachte IP-Adresse richten. Nur ein Broadcast wird der lokale Kernel des Clients akzeptieren; auf eine IP-Adresse reagiert er erst, wenn es die »seine« ist:

```
root@sonne> bootpc --serverbcast
SERVER='192.168.100.100'
IPADDR='192.168.100.10'
BOOTFILE=''
NETMASK='255.255.255.0'
NETWORK='192.168.100.0'
brOADCAST='192.168.100.255'
DNSSRVS_1='192.168.100.1'
DNSSRVS_2='211.97.100.1'
DNSSRVS='192.168.100.1 211.97.100.1'
DOMAIN='galaxis.de'
SEARCH='galaxis.de'
```

Ersichtlich aus den Ausgaben sollte sein: **bootpc** allein richtet mir mein Netzwerk noch nicht ein. Tatsächlich liefert der Bootp-Client einzig die relaventen Informationen. Wie damit zu verfahren ist, liegt in Händen des Administrators. Vermutlich wird er den »Rest« von einem Shellskript erledigen lassen; wir demonstrieren hier pump is a daemon that manages network interfaces that are controlled by either the DHCP or BOOTP protocol.einzig die

Konfiguration der Netzwerkkarte und das Setzen der Routen anhand der neuen Parameter. Unser Skript beginnt wie folgt:

```
root@sonne> cat bootpc-script
# Nachfolgend stehen die Parameter in Variablen (SERVER, IPADDR,...) zur Verfügung
eval `bootpc --returniffail --timeoutwait 2 --serverbcast`

# Device 'eth0' mit »richtiger« IP-Adresse konfigurieren
ifconfig eth0 down
ifconfig eth0 $IPADDR netmask $NETMASK broadcast $brOADCAST up

# Default-Route setzen
route add default eth0
```

Zum Zwecke des Aufzeigens der möglichen Vorgehensweise sollte das Beispiel genügen. In realistischen Szenarien darf natürlich eine Fehlerbehandlung nicht fehlen. Das Einrichten zum Zugriff auf die gelieferten DNS-Server erfordert eine Modifikation der Datei /etc/resolv.conf. Hier sollte dringend Sorge getroffen werden, um den »ursprüglichen« Zustand während des Herunterfahrens des Systems zu rekonstruieren. Das Ganze läuft auf ein Init-Skript hinaus, das mittels der Parameter »start« und »stop« alternative Maßnahmen trifft.

Der Client pump

RedHat basierte Distributionen verwenden als Bootp- als auch als DHCP-Client häufig das Programm **pump**. »pump« arbeitet dabei als Daemon und administriert von Haus aus Netzwerk-Schnittstellen und einige Netzwerkdienste.

```
pump [-krRsd?] [-c file] [-h hostname] [-i interface] [-l hours] [--usage]
```

Die wichtigen Optionen sind:

-c <file>

Name einer Konfigurationsdatei, wenn es sich *nicht* um /etc/pump.conf handelt

-h <hostname>

Name/Adresse des zu befragenden Bootp- bzw. DHCP-Servers

-i <Interface>

Zu konfigurierenden Netzwerkgerät; Voreinstellung ist eth0

-k

Beendet den Daemon und deaktiviert alle Netzwerkgeräte

-l <hours>

Gültigkeitsdauer der vom Server empfangenen Daten in Stunden; die Option ist nur in Verbindung mit DHC wichtig

-R

Anforderung neuer Daten (auch ohne das die alten »verfallen« sind)

-s

Anzeige von Statusinformationen

-d

--no-gateway

Kein Setzen einer default-Route für das Netzwerkgerät

--help

Eine Kurzhilfe zum Kommando

Um mit **pump** das Netzwerk eines Clients via Bootp oder DHCP einzurichten, genügt auf Clientseite der Aufruf des Kommandos. Die nachfolgende Statusabfrage sollte berichten, welche Informationen vom Server bezogen wurden:

```
root@sonne> pump -s
Device eth0
        IP: 192.168.100.100
        Netmask: 255.255.255.0
        Broadcast: 192.168.100.255
        Network: 192.168.100.0
        Boot server 127.0.0.2
        Next server 192.168.100.10
        Domain: galaxis.de
        Nameservers: 192.168.100.1 211.97.100.1
```

Die Aufzählung der Optionen von **pump** deutete bereits die Möglichkeit an, das Verhalten des Daemons per Konfigurationsdatei zu beeinflussen. Der typische Name der Datei lautet **/ etc/ pump.conf**. Ihre Existenz genügt, um von **pump** beachtet zu werden. Die Direktiven in der Datei können sowohl einem speziellen Device zugeordnet werden oder als globale Direktive für alle zu konfigurierenden Netzwerkgeräte gelten:

```
# Globale Direktive(n)
domainsearch "galaxis.de outside.all"
retries 3

# Lokale Direktive(n) für eth0
device eth0 {
   nogateway
}
```

Jedes wiederholte Vorkommen einer Direktive überschreibt die bisherigen Werte. Als Direktiven sind möglich:

device < Devicename>

Bindet nachfolgende Direktiven an das bezeichnete Device; die Liste muss in geschweifte Klammern eingeschlossen sein

domainsearch < Suchpfad>

Anstatt den DNS-Suchpfad zur Auflösung unvollständiger Rechnernamen aus der Datei /etc/resolv.conf zu nehmen, wird der angegebene verwendet

nonisdomain

Der Nis-Domainname bleibt erhalten, selbst wenn der Server einen anderen mitteilt

nodns

Kein Überschreiben der Datei /etc/resolv.conf

nogateway

Ignorieren der Angaben zu default-Routen vom Server

retries < Anzahl>

Anzahl Wiederholungen, wenn eine Anfrage am Server scheiterte

timeout < Sekunden>

Eine unbeantwortete DHCP/Bootp-Anfrage gilt nach Ablauf der angegebenen Zeitspanne als fehlgeschlagen

script < Programm/ Skriptname>

DHCP

Booten übers Netz

WWW Clients

Übersicht

Galeon

Konquerer

Lynx

Mozilla

Netscape

W3M

WWWoffle

Mail Clients

Übersicht

Elm

Emacs

Kmail

Mail

Metamail

Netscape

Pine

Der Wachhund "xbiff"

FTP Clients

Übersicht

ftp

kftp

xftp

News Clients

Übersicht

Emacs

Knews

Netscape

Tin

Übersicht

Samba

Übersicht

Netzwerk Server

Überblick

Ziel des Kapitels

Inhalt des Kapitels

Server - Telnet & Co.

Übersicht

Vorbereitungen

Start der Serverdienste

Die Server für Telnet und die Remote Shell werden nahezu ausschließlich erst bei Bedarf durch den inetd gestartet. Überprüfen Sie vorab die Datei /etc/inetd.conf, ob die entsprechenden Einträge enthalten und freigeschaltet sind:

```
user@sonne> egrep 'telnet| rsh' /etc/inetd.conf
# If you want telnetd not to "keep-alives" (e.g. if it runs over a ISDN
# uplink), add "-n". See 'man telnetd' for more details.
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
# man-page of rlogind and rshd to see more configuration possibilities about
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -L
# shell stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -aL
# Try "telnet localhost systat" and "telnet localhost netstat" to see that
```

Entfernen Sie ggf. Kommentarzeichen vor den Einträgen oder ergänzen Sie obige Zeilen. Starten Sie nach Änderungen den inetd neu:

```
root@sonne> /etc/init.d/inetd restart
```

Enthält Ihre Distribution kein ähnlich lautendes Startskript, so sollten Sie den Daemon per Hand neu starten (killall inetd; /usr/sbin/inetd).

Verwenden Sie den moderneren xinetd (Voreinstellung bspw. bei RedHat), so sollte dessen Konfigurationsdatei /etc/xinetd.conf Einträge ähnlich der folgenden aufweisen:

```
user@sonne> less /etc/xinetd.conf
...
service telnet
  {
   socket_type = stream
   protocol    = tcp
   wait        = no
   user        = root
   server      = /usr/sbin/in.telnetd
   log_on_failure += RECORD
   log_type    = SYSLOG auth warn
  }

service shell
  {
   socket_type = stream
   protocol    = tcp
   wait        = no
   user        = root
   server      = /usr/sbin/in.rshd
   server_args = -aL
   log_on_failure += RECORD
   log_type    = SYSLOG auth warn
  }
...
```

Auch der xinetd ist neu zu starten, damit die Änderungen in der Konfiguration wirksam werden.

Der Ssh-Serverprozess wird i.d.R. als permanenter Dienst bereits während des Systemstarts aktiviert; eine Konfiguration zum Start durch den [x]inetd ist dennoch möglich.

Etwas Sicherheit

Zumindest Telnet und Rsh beinhalten einige bedenkliche Mechanismen, die Sicherheitsanforderungen in kritischem Umfeld nicht genügen. Aus diesem Grund wurden zusätzliche Barrieren eingeführt, die den Zugang zu den Diensten einschränken können.

Eine der Sicherheitsvorkehrungen wäre der konsequente Einsatz des moderneren Internet-Daemons xinetd anstatt des alteingesessenen inetd. Für jeden Dienst können Sie mittels der Einträge **only_from** bzw. **no_access** explizit festlegen, von welchen Rechnern/Netzwerken aus der Zugriff gestattet ist oder von wo aus er verwehrt wird.

In vielen Distributionen wird nach wie vor am Inetd als »Verwalter der Internet-Dienste« festgehalten. Dessen fehlendes Sicherheitskonzept übernimmt der TCP-Wrapper, der den Zugriff anhand der Einträge der Dateien /etc/hosts.allow und /etc/hosts.deny verifiziert. Der Zugang zu Telnet- und Remote-Shell-Server sollte - falls Sie auf die beiden Dienste nicht verzichten können - in offenen Netzwerken stets durch den TCP-Wrapper überwacht und nur für konkrete Rechner geöffnet werden. Die beiden Dateien /etc/hosts.allow und /etc/hosts.deny sollten um folgende Einträge ergänzt werden (im Beispiel gestatten wir den Zugriff von allen Rechnern des lokalen Netzwerkes aus:

```
root@sonne> vi /etc/hosts.deny
# Alles pauschal verbieten
ALL : ALL

root@sonne> vi /etc/hosts.allow
# Telnet und Rsh für Rechner des lokalen Netzwerkes öffnen; Rsh erfordert neben dem rsh-Daemon noch Zugang zum rlogin-Daemon
in.telnetd : LOCAL
in.rlogind : LOCAL
in.rshd    : LOCAL
```

In aktuellen Implementierungen des Ssh-Daemons ist die Auswertung der Dateien /etc/hosts.allow und /etc/hosts.deny zumeist fest verdrahtet; Sie können das leicht mit Hilfe von **strings** überprüfen:

```
user@sonne> strings /usr/sbin/sshd | egrep 'hosts.allow|hosts.deny'
hosts_allow_table
hosts_deny_table
/etc/hosts.allow
/etc/hosts.deny
```

Erhalten Sie obige Ausgaben, so muss der Zugriff auf den Daemon ebenso durch die Einträge in den beiden Dateien möglich sein. Bspw. genügt folgende Zeile, um den Rechnern des lokalen Netzwerks den Ssh-Zugang zu gewähren:

```
root@sonne> vi /etc/hosts.allow
# Zugang zur Secure Shell
sshd : LOCAL
```

Bei weiteren Zugangsproblemen auf einen der Dienste sollten Sie ggf. die Konfiguration der betreffenden Pluggable Authentificaton Modules prüfen.

Telnet

Allgemeines

Die Basis von Telnet (Telecommunication Network) ist so alt wie das Internet (genauer das ARPAnet) selbst, denn

die Motivation zur Verbindung von Rechnern entsprang dem Wunsch, vom »lokalen« Terminal Resourcen auf entfernten Rechnern in Anspruch nehmen zu können. So verfügten die 1969 verbundenen 4 Interface Message Processors (IMP) über eine rudimentäre Lösung der Terminalverbindung, die zumindest als Anfänge der Protokollentwicklung zu Telnet gelten dürften. 1970 markierte das Network Control Protocol (NCP) einen weiteren Meilenstein, indem es erstmals die Netzwerkschicht von den Anwendungsprozessen getrennt realisierte und eine Benutzerauthentifizierung einführte.

Die Erweiterung des ARPAnets um neue Rechner zeigte deutlich die Schwächen der bisherigen Realisierung auf: das System funktionierte nur zufriedenstellend, wenn die verbundenen Rechner den gleichen Zeichensatz verwendeten. Unter anderem diese Erfahrungen spiegelten sich im RFC-97 (1971) als erstem Vorschlag des Telnet-Protokolls wider. Weitere Protokolländerungen bzw. -erweiterungen spezifizierten u.a. die RFCs 318, 328, 340, 393 (1972) und 435 (1973). Der noch heutige gültige Standard wurde erst 1983 (RFC 854) festgeschrieben.

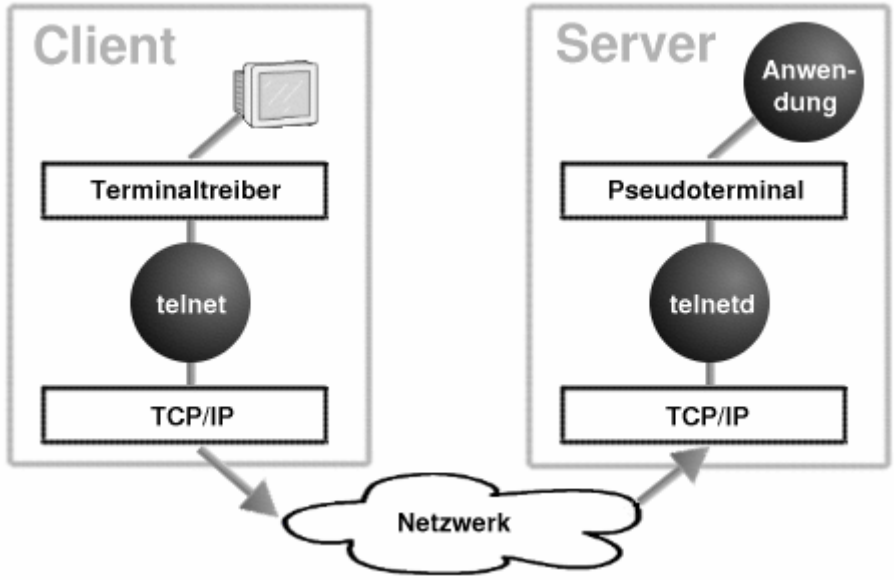


Abbildung 1: Interne Vorgänge während einer Telnet-Sitzung

Parameter zum Serverstart

Die r-Utilities

Secure Shell

Network File System - Server

Übersicht

Dieser Abschnitt basiert auf Recherchen von Bernd Reimann. Herzlichen Dank an ihn für die Zusammenarbeit!

Bevor wir uns dem Schwerpunkt der Administration eines NFS-Servers zuwenden, sollen kurze Betrachtungen zur Vergangenheit, Gegenwart und Zukunft der NFS-Entwicklung unter Linux etwas Hintergrundwissen vermitteln...

Historie

Bereits 1984, also lange bevor Linus Torvalds auch nur annähernd an Linux dachte und die Computerwelt noch von den Mainframes regiert wurde, widmete sich das Unternehmen *SUN Microsystems* der Aufgabe, ein System zu entwickeln, das den transparenten Zugriff auf die auf entfernten Rechnern liegenden Daten und Programme ermöglichen konnte.

Nahezu ein Jahr nach dieser Ankündigung (1985) stellte *SUN NFS Version 2 (NFS V2)* der Öffentlichkeit vor. Das Herausragende dieser Präsentation waren jedoch nicht die technischen Finessen, die das System mit sich bringen würde, sondern die Veröffentlichung der Spezifikation ansich. *SUN* durchbrach zum ersten Mal die Heiligkeit der Geheimhaltung solcher Interna und ebnete damit auch unwillkürlich den Siegeszug von NFS.

SUN stellte allen interessierten Firmen die Entwicklungsergebnisse als Lizenz zur Verfügung und ermöglichte somit einen de-facto-Standard in der UNIX-Welt, der sich bis heute durchgesetzt hat und immer noch eine der erfolgreichsten UNIX-Technologien darstellt. Nach und nach wurde NFS auf weitere Plattformen wie System V, BSD, AIX, HP-UX und später auch Linux portiert, wo NFS zur Standardausstattung einer jeden Distribution gehört. Selbst viele Nicht-Unix-Systeme, die es zu einer gewissen Verbreitung gebracht haben, warten mit einer NFS-kompatiblen Implementierung auf.

Protokollkollegen

1991 veröffentlichte *SUNSoft*, eine Tochtergesellschaft von *SUN*, erstmals eine komplette Sammlung zusammenwirkender Protokolle namens **ONC** (*Open Network Computing*), in der alle Bestandteile der Netzwerkkommunikation vom Zugriff auf die Netzwerkhardware bis hin zu Netzwerkanwendungen integriert wurden. Der NFS-Dienst ist Teil dieser ONC-Protokoll-Suite (Abbildung 1) und vertritt dort die Anwendungsebene, d.h. mit dem NFS-Dienst wird direkt gearbeitet.

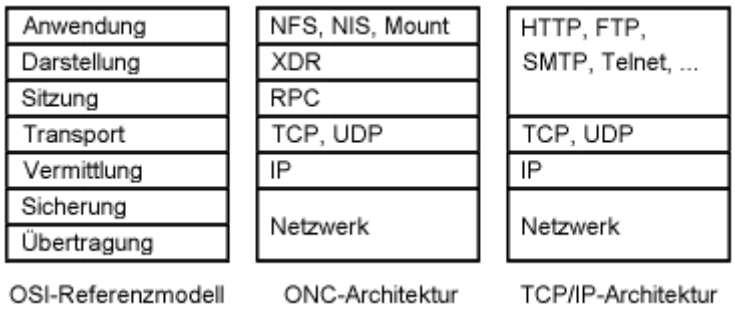


Abbildung 1: Vergleich verbreiteter Protokollstacks

Wie auch das OSI-Modell hat sich die ONC-Protokollsuite als Schema in der Praxis nicht durchgesetzt, auch wenn Protokolle wie XDR (External Data Representation) und der Remote Procedure Call (RPC) untrennbar mit dem NFS-

Dienst einhergehen. XDR dient der Konvertierung der zu übermittelnden Daten in ein standardisiertes Format. Dies ist notwendig, da das NFS-Protokoll nicht auf ein System und eine Hardware-Plattform beschränkt ist und unterschiedliche Systeme und Hardware mitunter unterschiedliche Darstellungen der Daten praktizieren.

Hinter dem »Entfernten-Prozedur-Aufruf« (RPC) steckt die Idee, dass Programme auf Funktionen zurückgreifen, die nicht zwingend auf dem gleichen Rechner erbracht werden, wobei sich der Funktionsaufruf aus Sicht des Programms nicht von einem »normalen« lokalen Funktionsaufruf unterscheidet. Der interessierte Leser findet im Abschnitt Remote Procedure Call des Kapitels »Netzwerk-Grundlagen« eine umfassende Diskussion der Mechanismen und Probleme des RPCs.

Auf der Anwendungsebene tummelt sich nicht nur der NFS-Dienst selbst, sondern auch noch (verwandte) Protokolle wie das Mount-Protokoll, der Netzwerk-Lock-Manager (NLM) und der Netzwerk-Status-Monitor (NSM). Das Mount-Protokoll ist für den »Einhängevorgang« des Dateisystems zuständig, ein konkretes Beispiel des doch recht komplexen Vorgangs folgt zu einem späteren Zeitpunkt. NLM und NSM ermöglichen den exklusiven Zugriff auf Dateien über Rechnergrenzen hinweg. Beide Dienste sind für den Betrieb von NFS nicht zwingend erforderlich, ob Sie sie benötigen, erfahren Sie im Abschnitt Interna.

Kernel-NFS vs. Nutzeradressraum-NFS

Wer mit der Materie wenig vertraut ist, wird womöglich zum ersten Mal mit beiden Begriffen konfrontiert sein. Eigentlich erübrigt sich eine Diskussion über Vor- und Nachteile, hat sich doch der kernel-basierte Serverdienst gegenüber der alteingesessenen Implementierung als im Nutzeradressraum tätiger Serverprozess eindeutig durchgesetzt. Und dennoch verlangt eine umfassende Betrachtung einen tieferen Blick hinter die Kulissen.

Der so genannte *User-Space NFS Server*, wie er vor wenigen Jahren noch jeder Linux-Distribution beilag, zeichnet sich vor allem durch eine einfach zu realisierende Implementierung aus. Allein die Tatsache, dass ein Client dieselbe Sicht auf ein gemountetes Dateisystem hat, wie auch der Server, ermöglicht ein simples Mapping von lokalen Gruppen- und Benutzeridentifikationen (gid, uid) auf die vom importierten Dateisystem verwendeten Nummern. Dem User-Space-NFS-Server stand hierzu ein Rpc-Dienst (rpc.ugidd) zur Seite, der anhand einer Tabelle die Umsetzung der Benutzeridentifikationen vornahm. Nicht zuletzt aus Sicherheitsgründen unterstützt der neue Kernel-NFS-Server nur eine stark eingeschränkte Variante dieses Verfahrens (vergleiche Squashing).

Wenn der kernel-basierte Server so aufwändig zu implementieren ist, warum hat man es nicht beim *User-Space NFS Server* belassen? Im Großen und Ganzen führen alle Überlegungen auf einen Grund zurück: *Im Kernel arbeitet der Server entschieden schneller!*. Der alte User-Space-NFS-Server kannte kein Threading (vereinfacht ausgedrückt, kann ein thread-basiertes Programm mehrere Aufgaben quasi gleichzeitig verrichten). Dieses Manko hätte eine Reimplementierung sicher noch behoben, aber auf Grund der strikten Schutzmechanismen im Betriebssystem bleiben dem Kernel-Server etliche Kopieroperationen erspart, da er bspw. die Berechtigung besitzt, Daten direkt in den Speicherbereich der Netzwerkkarte zu befördern. Ein Server im Nutzeradressraum müsste hierzu erst den Kernel bemühen, was allein schon einen Taskwechsel - und damit Zeit - bedingt. Weitere Unzulänglichkeiten sind die fehlende Unterstützung für NLM (*Network Lock Manager*) und NSM (*Network Status Monitor*) sowie die leidige Beschränkung der Dateigröße auf 2GB.

Die Vorteile des *Kernel-Space NFS Servers* liegen neben dem Geschwindigkeitsgewinn in dessen erweitertem Funktionsumfang. So beherrscht er mittels der NLM und NSM Protkolle das Setzen von Dateisperren auf exportierten Verzeichnissen. Und da die Überwachung der exports-Datei als Systemruf implementiert ist, müssen Server und Mountd nicht explizit über Änderungen unterrichtet werden. Die (theoretische) Beschränkung der Dateigröße liegt nun bei 2^{63} Bits.

Leider besitzt auch die Kernelrealisierung noch eine Schwäche. So werden Dateisysteme, die sich unterhalb eines exportierten Verzeichnisses befinden, nicht automatisch exportiert (bis auf einen Fall; siehe später). Für diese wird ein extra Eintrag in der exports-Datei und damit clientseitig ein weiterer Mountvorgang notwendig.

So komplex obige Ausführungen auch anmuten mögen, aus Sicht des Administrators bestehen bez. der Einrichtung von NFS keinerlei Unterschiede zwischen User- und Kernel-Space NFS.

Ausblick

Im kurzen historischen Abriss war von NFS Version 2 die Rede. Version 3 ist seit Jahren existent und an Version 4

wird gestrickt. Die aktuellen NFS-Server (Kernelversionen ab 2.2.19) unterstützen alle Forderungen der Version 2 und bereits wesentliche Teile von Version 3. Die letzte »Hürde« (NFS via TCP) zur vollständigen Umsetzung von Version-3-Funktionalität ist als experimenteller Patch (ab Kernel 2.4.19) bereits genommen, die stabile Version ist somit nur noch eine Frage der Zeit. Auch Entwicklungen zur Umsetzung von NFS der 4. Version sind angelaufen.

Schritte zum Serverstart

Ports, Ports, Ports

Da eine Kommunikation zwischen zwei Rechnern über sogenannte Sockets läuft, also die IP-Adresse und die zugehörige Portnummer eines Dienstes, muss natürlich der Client, der seine Anfrage an den Server richtet, auch diese Portnummer kennen. Auf Grund der Fülle existenter RPC-Dienste werden Portnummern für diese nicht mehr strikt statisch vergeben, sondern über einen zusätzlichen Dienst - den Portmapper oder auch RPC-Bind (SUN) genannt - verwaltet. Der Portmapper selbst lauscht an einem festgelegten Port (111 sowohl über UDP als auch TCP).

Dieser Portmapper ist in den meisten Distributionen standardmäßig installiert und wird oft durch ein Init-Skript in den Netzwerk-Runleveln beim Booten des Systems gestartet. Wird der Portmapper nur selten benötigt, spricht nichts gegen einen automatischen Start bei Bedarf durch den [x]inet-Daemonen.

Der Portmapper verwaltet eine Art kleine Datenbank, in der er verschiedene Informationen zu allen auf dem System aktiven RPC-Diensten hält. Ein RPC-Dienst muss sich deshalb beim Start beim Portmapper registrieren und ihm u.a. die von ihm verwendete Portnummer mitteilen. Bei Anfragen eines RPC-Clients an den Portmapper nach einem konkreten RPC-Dienst (*PMAPPROC_GETPORT()*), durchsucht dieser seine Datenbasis und übergibt, falls der Dienst verfügbar ist, dessen Portnummer. Alle Clientzugriffe erfolgen später direkt an diesen Port (also unter Umgehung des Portmappers; Abbildung 2).

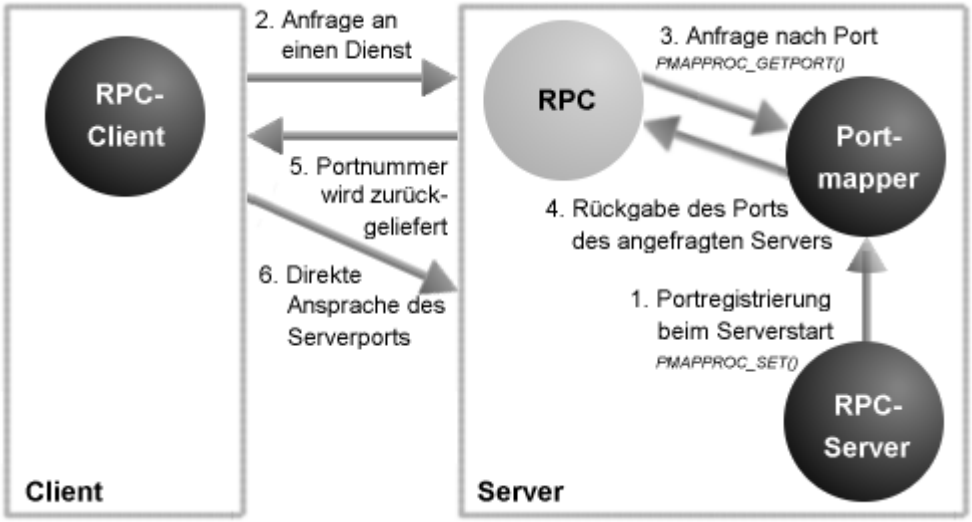


Abbildung 2: Kontaktaufnahme des Clients zum Server

RPC-Dienste und Sicherheit

Der Zugang zu fast jedem RPC-Dienst unter Linux wird in aktuellen Programmversionen durch die Konfigurationen der Dateien /etc/hosts.allow und /etc/hosts.deny geregelt. Sie können leicht feststellen, ob die Dateien für ein Programm relevant sind:

```
# Wertet 'rpc.mountd' die Dateien hosts.allow und hosts.deny aus?
user@sonne> strings /usr/sbin/rpc.mountd | grep hosts
hosts_access_verbose
hosts_allow_table
hosts_deny_table
/etc/hosts.allow
```

```
/etc/hosts.deny
@(#) hosts_ctl.c 1.4 94/12/28 17:42:27
@(#) hosts_access.c 1.21 97/02/12 02:13:22
```

Insofern ein Dienst nicht in einer der Dateien »hosts.allow« bzw. »hosts.deny« aufgeführt ist, ist der Zugang zu diesem gewährt. In der Datei »hosts.allow« kann der Zugang explizit erlaubt und in »hosts.deny« verboten werden, wobei bei widersprüchlichen Einträgen in beiden Dateien die Angaben aus »hosts.allow« gelten.

In sicherheitskritischem Umfeld ist es nicht ungewöhnlich, zunächst pauschal sämtliche Dienste in »hosts.deny« zu sperren, um die wirklich benötigen in »hosts.allow« explizit für auserwählte Rechner zuzulassen. Bez. der auf einem NFS-Server notwendigen Dienste, wären folgende Einträge in die Dateien erforderlich, um den Zugriff für Rechner des lokalen Netzwerks (192.168.100.0) zuzulassen:

```
root@sonne> vi / etc/ hosts.deny
# Alles verbieten
ALL : ALL

root@sonne> vi / etc/ hosts.allow
# Alle für NFS notwendigen Dienste für die lokale Domain freigeben:
portmap    : 192.168.100.0/255.255.255.0
mountd     : 192.168.100.0/255.255.255.0
lockd      : 192.168.100.0/255.255.255.0
statd      : 192.168.100.0/255.255.255.0
rquotad    : 192.168.100.0/255.255.255.0
```

Die Namen der einzelnen Dienste könnten auf Ihrem System von denen im Beispiel abweichen. Im Zweifelsfall sollten Sie nach der Registrierung der RPC-Dienste beim Portmapper die Namen der Ausgabe von »/usr/sbin/rpcinfo -p« entnehmen (mehr Informationen dazu im weiteren Text). »lockd« und »statd« sind einzig erforderlich, wenn Programme auf dem Client mit Dateisperren arbeiten (vergleiche Anmerkungen in der Einleitung); der RPC-Dienst »rquotad« ist notwendig, wenn auf den exportierten Verzeichnissen Quotas Verwendung finden.

Registrierung

Vergewissern Sie sich zunächst, dass der Portmapper auf Ihrem System aktiv ist. Dies kann entweder über die Statusabfrage des Runlevelskripts (falls ein solches existiert) erfolgen oder durch Suche in der Ausgabe des Kommandos ps:

```
# Abfrage über Runlevelskript (dessen Name kann auf Ihrem System abweichend lauten!)
root@sonne> / etc/ init.d/ portmap status
Checking for RPC portmap daemon:                    running

# Abfrage mittels ps
user@sonne> ps ax |  grep portmap
  399 ?        S      0:00 /sbin/portmap
 1578 pts/3    S      0:00 grep portmap
```

Ist der Portmapper inaktiv, dann starten Sie ihn entweder über das Runlevelskript oder per Hand:

```
# Start über Runlevelskript (dessen Name kann auf Ihrem System abweichend lauten!)
root@sonne> / etc/ init.d/ portmap start
Starting RPC portmap daemon                         done

# Start per Hand
root@sonne> / sbin/ portmap
```

In den aktuellen Distributionen erledigt den Start des NFS-Servers und aller weiteren notwendigen Dienste ein Skript, bei SuSE ist dies bspw. »/etc/init.d/nfsserver«:

```
# NFS-Serverstart bei SuSE
root@sonne> / etc/ init.d/ nfsserver start
Starting kernel based NFS server                    done
```

Existiert kein solches Skipt, müssen Sie die Dienste per Hand aktivieren:

```
root@sonne> /sbin/rpc.statd
root@sonne> /sbin/rpc.lockd
root@sonne> /usr/sbin/rpc.mountd
root@sonne> /usr/sbin/rpc.nfsd
root@sonne> /usr/sbin/rpc.rquotad
```

Die Dienste »/sbin/rpc.statd« und »/sbin/rpc.lockd« bzw. »/sbin/rpc.rquotad« sind nicht zwingend notwendig (siehe Anmerkungen in der Einleitung und unter Interna).

Vom erfolgreichen Registrieren der RPC-Dienste beim Portmapper können Sie sich durch einen Aufruf von »rpcinfo« überzeugen. In Verbindung mit der Option »-p« erfahren Sie, welche Dienste mit welchen internen Daten aktuell verwaltet werden:

```
user@sonne> rpcinfo -p
   Program Vers Proto   Port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100021    1   udp   1024  nlockmgr
    100021    3   udp   1024  nlockmgr
    100024    1   udp   1025  status
    100024    1   tcp   1024  status
    100011    1   udp    969  rquotad
    100011    2   udp    969  rquotad
    100005    1   udp   1033  mountd
    100005    1   tcp   1025  mountd
    100005    2   udp   1033  mountd
    100005    2   tcp   1025  mountd
    100003    2   udp   2049  nfs
```

Die Ausgaben könnten auf Ihrem System durchaus von obigem Beispiel abweichen, so sind abweichende Portnummern oder ähnliche Namen für die Dienste ebensowenig Besorgnis erregend, wie mehrfaches Auftauchen eines RPC-Dienstes in der Tabelle, da in vielen Konfigurationen aus Effizienzgründen gleich mehrere Instanzen eines Dienstes aktiviert werden.

Verzeichnisexport

Exportfs

Bevor ein Client ein Verzeichnis vom Server importieren kann, muss dieser dieses explizit mit den erforderlichen Rechten exportieren. Der Export eines Verzeichnisses erfolgt mit dem Kommando **exportfs** (das Kommando existiert nur in Zusammenhang mit dem Kernel-NFS):

```
/usr/sbin/exportfs [-avi] [-o Mountoptionen,..] [Client:/Pfad..]
/usr/sbin/exportfs [-av] -u [Client:/Pfad..]
/usr/sbin/exportfs -r [-v]
```

Der Export von Verzeichnissen mittels »exportfs« ist zumeist nur in der Testphase des NFS-Servers oder bei einmaliger Freigabe eines Verzeichnisses erforderlich. »exportfs« wird hauptsächlich genutzt, um nach Änderungen in der Datei »/etc/exports« den Server (genauer: rpc.nfsd und rpc.mountd) zu instruieren, seine Konfiguration neu einzulesen. Bei den Mountoptionen des Kommandos handelt es sich um eben diese, die anschließend in Zusammenhang mit der Datei »/etc/exports« vorgestellt werden; an dieser Stelle verzichten wir auf ihre Diskussion und veranlassen in einem ersten Beispiel den NFS-Server, das Verzeichnis »/home/user« für den Rechner »venus« freizugeben:

```
root@sonne> exportfs venus.galaxis.de:/home/user
```

Prinzipiell können Sie für die Rechnernamen dieselbe Syntax wie in »/etc/exports« verwenden; vergessen Sie

jedoch nicht, enthaltene Sonderzeichen vor Interpretation durch die Shell zu schützen. Wird - wie im Beispiel - auf Mountoptionen verzichtet, gelten async, ro, root_squash, secure (Erklärung im Anschluss).

Um das Verzeichnis vom Export wieder auszuschließen, ist die Option »-u« dienlich:

```
root@sonne> exportfs -u venus.galaxis.de:/home/user
```

Die Option »-a« exportiert alle (auf der Kommandozeile **und** in /etc/exports angegebenen) Verzeichnisse; in Zusammenhang mit »-u« werden sämtliche exportierten Verzeichnisse vom Export ausgeschlossen. Um in solchen Fällen die Auswertung der Datei »/etc/exports« zu verhindern, dient »-i«. Schließlich synchronisiert die Option »-r« die aktuelle Datenbasis des NFS-Servers (/var/lib/nfs/xtab) mit den Einträgen der Datei »/etc/exports«.

Nach Änderungen in der Datei »/etc/exports« führt folgender exportfs-Aufruf zu einer Aktualisierung beim Server:

```
root@sonne> exportfs -ra
```

Anmerkung: Beim älteren User-Space-NFS-Server gelingt der Export von Verzeichnissen einzig über Einträge in der Datei »/etc/exports«. Anschließend muss dem Serverprozess das Signal SIGHUP gesendet werden (kill -HUP <Prozess-ID des Servers>).

Die Datei /etc/exports

Anstatt Dateisysteme einzeln via »exportfs« zu exportieren, wird die Konfiguration mittels der Datei »/etc/exports« bevorzugt (...und in Verbindung mit dem User-Space-NFS-Server ist dies die einzige Möglichkeit der Konfiguration).

Der Aufbau eines Eintrags in die Datei ist einfach:

```
Verzeichnis    Client[(Mountoption,...)] [Client[(Mountoption,...)] ]
```

Verzeichnis bezeichnet das zu exportierende Verzeichnis. Es folgt eine Liste der Client-Rechner, die zum Importieren dieses Verzeichnisses befugt sind. Optional kann der Zugriff auf die Daten für einen Client gegenüber der Voreinstellung (async, ro, root_squash, secure) weiter eingeschränkt oder auch gelockert werden.

Die Syntax für *Client* ist etwas weiter gefasst. Hier sind nicht nur die Angabe starrer Rechnernamen gestattet, sondern ebenso die Bezeichnung von Netzgruppen (die in der Datei /etc/netgroup definiert sein müssen) oder die Verwendung von Wildcards zur Definition von Mustern. Damit ergeben sich folgende Varianten zur Angabe:

erde.galaxis.de

Angabe eines konkreten Rechners (oder auch IP-Adresse)

@Gateways

Bezug auf eine Netzgruppe, die der Datei /etc/netgroup definiert sein muss

***.galaxis.de**

Angabe von Mustern; hier: alle Rechner aus der Domäne »galaxis.de«

192.168.100.0/255.255.255.0

192.168.100.0/22

Angabe von IP-Netwerken inklusiver Subnetzmaske, welche für den Fall, dass allein führende Bits zur Maskenbildung herangezogen werden, auch als Anzahl der Bits (22 im Beispiel) spezifiziert werden kann

Jeder Angabe eines Clients kann eine Liste vom Optionen folgen, die den Zugriff auf die exportierten Daten steuern. Innerhalb der kommaseparierten Liste sind keine Leerzeichen statthaft! Mögliche Werte sind:

secure, insecure

Client-Anfragen werden nur von vertrauenswürdigen Ports (Portnummern unterhalb 1024) akzeptiert (»secu Voreinstellung); mit »insecure« werden auf Anfragen an höhere Ports akzeptiert

ro, rw

Das Verzeichnis wird schreibgeschützt (»read only«, Voreinstellung) bzw. mit vollen Lese- und Schreibrecht für den Client (»read/write«) exportiert

sync, async

Der Server darf den Vollzug eines Schreibvorgang dem Client erst melden, wenn die Daten tatsächlich auf d Platte geschrieben wurden (Ausschalten des Plattencaches). Die Voreinstellung ist **async**.

wdelay, no_wdelay

Die Option wird nur in Zusammenhang mit »sync« beachtet und erlaubt dem Server die Bestätigung eines Schreibvorgangs zu verzögern, falls mehrere Schreibvorgänge von einem Client zur gleichen Zeit im Gange Anstatt jeden zu bestätigen, sendet der Server nur eine einzige Antwort nach Vollzug aller Schreiboperation (betrifft »wdelay«, Voreinstellung).

hide, nohide

Exportiert der Server ein Verzeichnis, in dem wiederum ein anderes Dateisystem gemeountet ist, so wird di nicht an einen Client exportiert (»hide«, Voreinstellung); die »nohide«-Option (also den impliziten Export) funktioniert jedoch nur, wenn es sich bei der Clientangabe um einen Rechnernamen (keine Wildcards, IP-Netzwerke und Netzgruppen!) handelt.

subtree_check, no_subtree_check

Werden nur Teile eines Dateisystems vom Server exportiert, so muss der Server prüfen, dass Zugriffe nur a Dateien erfolgen, die innerhalb dieses Teilbaums liegen (»subtree_check«, Voreinstellung). Dies erhöht zwa Sicherheit allerdings auf Kosten der Geschwindigkeit, sodass die Prüfung mit »no_subtree_check« abgescha werden kann.

root_squash, no_root_squash

Root erhält die UserID des Pseudobenutzers »nobody«, womit der Root-Benutzer des Client-Rechners keine Root-Rechte auf dem vom Server importierten Verzeichnis erhält (Voreinstellung); mit »no_root_squash« bleiben die Root-Rechte auf Clientseite auf dem Verzeichnis erhalten.

all_squash, no_all_squash

Alle Zugreifenden erhalten die Nobody-UID; Voreinstellung ist »no_all_squash«, womit die Nutzerkennunge erhalten bleiben

anongid= gid

Squashing der Gruppe; die Gruppen-ID wird auf »gid« gesetzt. Bei dieser Option kann Root entscheiden, m welcher Server-GID die Client-Benutzer arbeiten sollen, sobald sie Zugriff auf den Server haben

anonuid= uid

Squashing des Benutzers. Die zugreifenden Benutzer bekommen die UID »uid« verpasst

Die Optionen zum Verändern der Nutzerkennungen werden als »Squashing« bezeichnet und werden im folgenden Abschnitt detailliert diskutiert.

Die Datei /etc/exports - Ein Beispiel

Als reales Szenario nehmen wir an, dass unser NFS-Server (»sonne.galaxis.de«) als Datei- und als NIS-Server konfiguriert wurde. Die zentrale Aufgabe eines NIS-Servers ist immer noch die Bereitstellung einer zentralen Benutzerverwaltung, in Kombination mit einen NFS-Server werden meist auch die Heimatverzeichnisse der Benutzer zentral gehalten, sodass ein Benutzer sich an einen beliebigen Rechner eines Pools anmelden kann und überall seine gewohnte Umgebung vorfinden wird.

Des Weiteren gehen wir davon aus, dass die Clients nur über eine beschränkte Plattenkapazität verfügen, sodass sie etliche Daten, die nicht der Grundfunktionalität dienen, ebenso via NFS importieren (wir gehen jetzt nicht soweit, festplattenlose Clients in die Betrachtung einzubeziehen, die sogar ihr Root-Dateisystem und den Swap via NFS beziehen).

In der nachfolgenden Beispieldatei finden Sie weitere Anmerkungen in Form von Kommentaren, warum welche Option für welches Verzeichnis gesetzt wurde; die Überlegungen sollten auf etliche Rahmenbedingungen übertragbar sein.

```
root@sonne> vi /etc/exports
# Wir nehmen an, dass /home auf einer eigenen Partition liegt, sodass sich die Prüfung, ob eine Datei innerhalb des exportierten Verzeichnisses liegt, erübrigt. Dass Benutzer innerhalb ihrer Heimatverzeichnisse auch Schreibrechte erhalten sollten, ist verständlich. Da unsere Netzwerk als sichere Zone anzusehen ist, verzichten wir auf die Einschränkung der zulässigen Portnummern:
/home          *.galaxis.de(rw,no_subtree_check,insecure)
# Wie der Name schon sagt, sind Daten unter /usr/share geeignet, um an zentraler Stelle für verschiedene Clients bereit gestellt zu werden. Kein Client muss diese Daten verändern können, deshalb setzen wir gleich die Benutzerkennungen auf den sehr restriktiven Zugang »nobody«
/usr/share     *.galaxis.de(ro,all_squash)
```

Abschließend sei darauf hingeweisen, dass auf Clientseite beim Import eines Verzeichnisses die Rechte weiter eingeschränkt werden können. Die Informationen dazu finden Sie im Abschnitt zum NFS-Client.

Squashing

Gerade wurde bei den Optionen und Parametern der Begriff »Squashing« eingeführt, doch was bedeutet er?

Lassen Sie es mich an einem Beispiel erklären: Auf dem System »sonne« (NFS-Server) existieren in einem exportierten Verzeichnis folgende Dateien:

```
user@sonne> ls -l
-rw-r-----   1  root     root    166 Apr 27 10:45 foo.bar
-rw-------   1  tux      users    16 Apr 27 10:45 testdaten.txt
```

Des weiteren existieren Benutzer mit folgender UID:

root 0
tux 501

Auf System »venus« (unser NFS-Client) existieren Benutzer mit folgenden UIDs:

root 0
alf 501
tux 502

Was passiert nun, wenn wir auf dem Client eine Serverfreigabe mounten?

Dazu muss man wissen, dass Unix die Zugriffsrechte nicht aufgrund der Benutzernamen verwaltet, sondern einzig anhand der zugrundeliegenden UIDs. In unserem Falle hat der Benutzer »alf« (auf dem Client) plötzlich Zugriff auf die Datei »testdaten.txt« von Benutzer «tux«, da beide dieselbe UID 501 besitzen. Gleiches gilt natürlich ebenso für den Benutzer »root«.

Um dies zu verhindern, gibt es das »Squashing«. Hierbei werden die UIDs und GIDs der auf die gemounteten Verzeichnisse zugreifenden Benutzer auf eine neutrale UID (der Pseudouser *nobody* mit UID -2) und GID (*nogroup* mit GID -2) gesetzt, wenn dieses mit der Option »all_squash« exportiert wurde. Die UID 0 (GID 0; also Root) wird in der Voreinstellung stets nach »nobody/nogroup« gemappt; erst die Option »no_root_squash» verhindert die

Umsetzung.

Der alte User-Space-NFS-Server erlaubte das »Squashen« jeder UID auf jede beliege andere und ebenso konnte jede Gruppenkennung (GID) auf jede andere gemappt werden. Der neuer Kernel-NFS-Server arbeitet wesentlich restriktiver und gestattet einzig, anstatt dem Pseudobenutzer »nobody« bzw. der Pseudogruppe »nogroup« mittels »anonuid=<UID>« bzw. »anonguid=<GID>« andere Kennungen zu vergeben. Diese gelten dann jedoch für beliebige Gruppen oder Benutzer.

Bez. unserer oben angegebenen Benutzerstrukturen würde bei Export des Verzeichnisses mittels der Option »all_squash« der Client-Benutzer »alf« nur noch ein »Nobody« sein und hätte somit nur Lesezugriff auf die Datei »testdaten.txt«

Etwas Statistik

Nicht zuletzt beim Aufspüren von Fehlkonfigurationen sind zwei Kommandos zur Diagnose des NFS-Servers nützlich. Zum einen erlaubt **showmount** einen Einblick sowohl auf die vom Server bereitgestellten Dateisysteme als auch auf die momentan aktiven Clients. Zum zweiten erlaubt **nfsstat** die Auswertung zahlreicher statistische Daten des Kernel-NFS-Servers.

showmount

/usr/sbin/showmount [-adehv] [NFS-Server]

Ohne Argumente aufgerufen, listet das Kommando die Namen der NFS-Clients auf, die aktuell Verzeichnisse vom Server importiert haben; bei Verwendung der Option »-a« werden zusätzlich die Verzeichnisnamen aufgeführt:

```
user@sonne> /usr/sbin/showmount
All mount points on sonne:
venus.galaxis.de:/home/
venus.galaxis.de:/usr/share/
erde.galaxis.de:/home/
```

Um einzig die aktuell von Clients importierten Verzeichnisnamen anzuzeigen, nicht aber die Namen der Clientrechner selbst, dient die Option »-d«. »-h« schreibt eine Kurzhilfe aus; »-v« die Versionsnummer des Programms.

Vor allem für Clientrechner interessant ist die Option »-e«, mit der sie, wenn Sie auf der Kommandozeile noch den Namen eines NFS-Servers angeben, testen können, welche Verzeichnisse ein Server für welchen Client exportiert:

```
user@erde> /usr/sbin/showmount -e sonne.galaxis.de
Export list for sonne.galaxis.de:
/home      *.galaxis.de
/usr/share *.galaxis.de
```

nfsstat

/usr/sbin/nfsstat [-anrcs] [-o <Kategorie>

Der Zugang zu den Statistiken im Kernel trägt eher informativen Charakter und verdeutlichen, welche Anforderungen die Clients tatsächlich stellen. Die Unterscheidung nach NFS Version 2 und 3 zeigt, in welchem Verhältnis die Clients nach dem alten bzw. nach dem neuen Protokoll arbeiten; im nachfolgend dargestellten Beispiel habe ich der besseren Übersicht wegen die Tabelle für die Versionen 2 entfernt.

Die Optionen des Kommandos helfen, die Ausgaben auf einen konkreten Typ zu beschränken. So begnügt sich »nfsstat -s« auf die Ausschrift der Statsitik der Serverseite, während »nfsstat -c« nur die Client-Daten offenbahrt.

»nfsstat -r« beschränkt sich auf Angabe der RPC-relevanten Daten; »nfsstat -n« tut dasselbe für die NFS-spezifischen Informationen.

```
root@sonne> nfsstat
Server rpc stats:
calls      badcalls   badauth    badclnt    xdrcall
20860      0          0          0          0
Server nfs v3:
null       getattr    setattr    lookup     access     readlink
0       0% 264     1% 0       0% 1148    5% 14726   70% 32      0%
read       write      create     mkdir      symlink    mknod
4485    21% 0       0% 0       0% 0       0% 0       0% 0       0%
remove     rmdir      rename     link       readdir    readdirplus
0       0% 0       0% 0       0% 0       0% 188     0% 14      0%
fsstat     fsinfo     pathconf   commit
1       0% 2       0% 0       0% 0       0%

Client rpc stats:
calls      retrans    authrefrsh
775        0          0
Client nfs v3:
null       getattr    setattr    lookup     access     readlink
0       0% 263     33% 0      0% 0       0% 494     63% 2      0%
read       write      create     mkdir      symlink    mknod
1       0% 0       0% 0       0% 0       0% 0       0% 0       0%
remove     rmdir      rename     link       readdir    readdirplus
0       0% 0       0% 0       0% 0       0% 0       0% 14      1%
fsstat     fsinfo     pathconf   commit
0       0% 1       0% 0       0% 0       0%
```

Unter *Kategorie* können nochmals die Statistiken speziell für RPC (»nfsstat -o rpc«) und NFS (»nfsstat -o nfs«) abgerufen werden. Zusätzlich sind Informationen über Auslastung des Transportprotokolls (»nfsstat -o net«), des Dateihandle-Caches (»nfsstat -o fh«) und des Antwortcaches (»nfsstat -o rc«) abrufbar.

```
root@sonne> nfsstat -o net
Server packet stats:
packets    udp        tcp        tcpconn
0          20860      0          0
Client packet stats:
packets    udp        tcp        tcpconn
0          0          0          0
```

Der Mountvorgang intern

Die Abläufe eines vom NFS-Clients initiierten Dateiimports sind recht komplex. Die Verwendung des hierzu notwendigen Kommando *mount* finden Sie im Abschnitt zum NFS-Client. Hier schauen wir hinter die Kulissen, also auf das Zusammenspiel der einzelnen Komponenten bis letztlich das Dateihandle, eine vom Server vergebene eindeutige Kennziffer für das Verzeichnis, zum Client gelangt.

Ausgangspunkt sei folgender erstmaliger Mount-Aufruf:

```
root@venus> mount -t nfs sonne:/ home / home
```

Anmerkung: Bei entsprechender Konfiguration kann ein Mountvorgang auch einem »normalen« Benutzer gestattet werden oder gar per Automounter automatisch bei Bedarf erfolgen.

Zum Zeitpunkt der ersten Kontaktaufnahme zum NFS-Server fehlt dem Client die Kenntnis der korrekten Portnummer, an dem dieser wartet. Deshalb geht die erste Anfrage an den Portmapper des Servers, an welchem Port der Mount-Daemon wartet. Mittels einer weiteren Anfrage beim Portmapper sollte der Client auch noch die Portnummer des NFS-Daemons in Erfahrung bringen (falls notwendig, erfragt der Client auch noch die Portnummern der RPC-Dienste NSM und NLM). Die Prinzipien des Verfahrens sollten Ihnen aus dem Abschnitt Schritte zum Serverstart bekannt sein. Einmal erfragte Portnummern werden solange lokal vorgehalten, bis ein

Zugriff auf einen solchen Port einen Fehler meldet (oder der Client neu bootet;-).

Mit der Erkenntnis, an welchem Port der Mount-Daemon des Servers wartet, wendet sich der Client nun direkt an diesen. Der Mount-Daemon durchsucht nun die Datei »/var/lib/nfs/etab«, ob das erwünschte Verzeichnis zum Export freigegen ist und ob der Client zum Import berechtigt ist.

```
user@venus> cat / var/ lib/ nfs/ etab
/usr/share *.galaxis.de(ro, async, wdelay, hide, secure, root_squash, all_squash,subtree_check, secure_locks,
mapping=identity, anonuid=-2, anongid=-2)
/home/  *galaxis.de(rw, async, wdelay, hide, secure, root_squash, no_all_squash, no_subtree_check, secure_locks,
mapping=identity, anonuid=-2, anongid=-2)
```

Verlaufen die Überprüfungen erfolgreich, so wendet sich der Mount-Daemon an den NFS-Daemon (Funktion *GETFH ()*), der ein eindeutiges Dateihandle, das das exportierte Verzeichnis referentiert, erzeugt. Dieses Handle ist ein Datenobjekt und enthält eine eindeutige Zufallszahl, die die Identifikation der Datei (ein Verzeichnis ist unter Unix letztlich auch nur eine Datei) gewährleisten soll und nur einmalig vergeben wird. Dieses Dateihandle übergibt der NFS-Daemon dem Mount-Daemon, der es zum Client sendet (Abbildung 3).

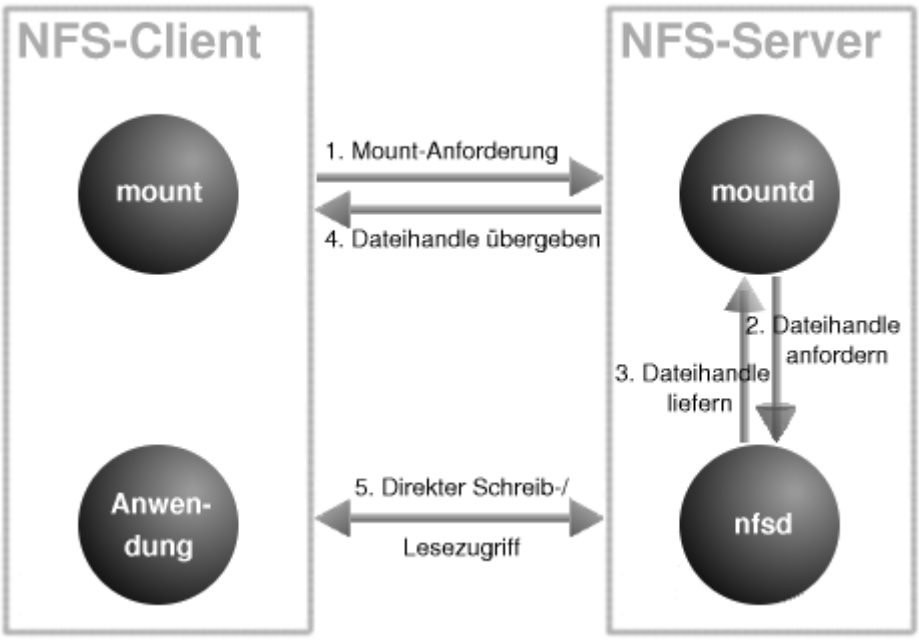


Abbildung 3: Ablauf eines NFS-Mounts

Alle weiteren Zugriffe auf Dateien des importierten Verzeichnisses erfolgen durch direkten Kontakt zwischen NFS-Client und dem NFS-Daemon-Prozess. Der Client gibt jeweils das Dateihandle an, womit der Server das Verzeichnis kennt, von dem aus er nach angeforderten Dateien suchen muss. Was in der Abbildung nicht dargestellt wurde, ist die eventuell notwendige Interaktion zwischen Lock-Manager und Status-Monitor, falls beim Zugriff auf Dateien mit Sperren gearbeitet werden soll.

Interna

Rein informellen Charakter tragen die weiteren Ausführungen, die Apsekte ansprechen, die für die Administration von NFS selbst unerheblich sind.

Cachefs

Oftmals ist es wünschenswert, wenn es eine Art Zwischenspeicher für die Netzzugriffe gibt, um die Netzbelastung so gering wie möglich zu halten. Hierfür gibt es von SUN (mal wieder!) ein Dateisystem namens Cachefs, das genau dieses »Caching« übernimmt. Leider existiert noch keine Implementierung dieses Dateisystems für Linux, jedoch wird es unter anderen Unixen häufig verwendet. Bei Cachefs wird ein Puffer-Verzeichnis angelegt, das nach und nach mit den vom Server angeforderten Daten gefüllt wird. Bei jedem Datenzugriff wird nun zunächst geprüft,

ob diese eventuell bereits im lokalen Puffer enthalten sind. Falls ja, muss die Datei nicht erst erneut übers Netz vom Server geladen werden, was einen mitunter erheblichen Geschwindigkeitsvorteil mit sich bringt, da lokale Bus- und Dateisysteme immer schneller sind als Netzleitungen (oder haben sie ein Gigabit-Ethernet zu Hause?).

Quota

Quotas dienen der Beschränkung des einem Benutzer zur Verfügung stehendem Plattenspeicherplatzes. Die Quotas selbst werden nur auf dem Server eingerichtet und ihre Einhaltung überwacht. Um Benutzern auf den Clients eine Abfrage ihres Quotastatus zu ermöglichen, ist auf dem NFS-Server der RPC-Dienst **rpc.rquotad** zu starten.

Network State Monitor (NSM) und Network Lock Manager (NLM)

Um gleichzeitige Zugriffe auf einunddieselben Dateien koordinieren zu können, behilft sich NFS eines Network-Lock-Managers (rpc.lockd) und eines Network-State-Monitors (rpc.statd). Das allein garantiert noch keine Dateisperren, denn diese müssen die auf NFS-importierte Dateien zugreifenden Programme auf Clientseite implementieren. Und dies tun bis heute die wenigsten Programme unter Linux. Daher ist es in den meisten Linuxnetzwerken gar nicht erforderlich, rpc.statd und rpc.lockd zu aktivieren.

Prinzipiell fordert ein entsprechend realisiertes Programm vom Kernel eine Dateisperre an. Der Kernel erkennt, dass die Datei auf einem via NFS gemountetem Dateisystem liegt und leitet die Anforderung an den lokalenLock-Manager weiter. Dieser sendet die Anforderung an den Lock-Daemon auf dem Server, welcher den Vorgang akzeptiert oder ablehnt. Das Ergebnis schickt er zum Lock-Daemon des Clients zurück, der es dem Kernel reicht.

Angenommen, während einer aktiven Dateisperre startet entweder der Client oder der Server neu, was passiert mit den Sperren? Jetzt kommt der Status-Monitor ins Spiel. Wenn auf Serverseite der Lock-Manager für einen Client eine Dateisperre anlegt, so unterrichtet er den Status-Monitor, welcher den Namen des Client-Rechners unter /var/lib/nfs/sm ablegt. Auf Clientseite erfährt der Status-Monitor die akzeptierte Sperre ebenso vom lokalen Lock-Manager, nur speichert er den Namen des Servers, von dem die Sperre gesetzt wurde.

Bootet nun ein Client neu, prüft der lokale rpc.statd sofort, ob irgendein Server zu unterrichten ist. Falls ja, kontaktiert er dessen Status-Monitor, der den Lock-Manager auf Serverseite unterrichtet, welcher wiederum alle Dateisperren des Clients aufhebt.

Fällt allerdings der Server aus, so unterrichtet nach einem Neustart dessen Status-Monitor die Kollegen auf allen Clients, die aktuelle Dateisperren besaßen. Die Lock-Manager auf den Clients versuchen nun zunächst die Dateisperren vom Server erneut anzufordern. Der Lock-Manager auf dem Server wird die durch den Neustart ungültigen Sperren noch »eine Zeit lang« reservieren und einzig dem »alten« Besitzer (also Client) gestatten, die Sperre unverzüglich anzufordern. So wird dem Client eine Chance erteilt, die verlorenen Sperren zu erneuern.

Unterschiede NFSV2 und NFSV3

Obwohl NFS V2 schon mehrere Jahre nach seiner Fertigstellung einen neuen großen Bruder erhielt (NFS V3), ist es trotzdem immer noch ein Standard und wird bei Linux weitestgehend eingesetzt. Langsam aber sicher findet der Umstieg auf NFS V3 statt, der aus Kompatibiltätsgründen fast alle alten Prozeduren übernimmt und neue Eigenschaften einbringt. So kann bei Version 3 nun zwischen den beiden Protokollen UDP und TCP ausgewählt werden, was bei Einsatz von TCP eine Entlastung des Dienstes bedeutet, da TCP nun die Fehlersicherung übernimmt und somit eine sichere Leitung gewährleistet - jedoch zum Preis einer erhöhten Netzbelastung (an der TCP-Unterstützung durch den Kernel-NFS-Daemon wird derzeit gearbeitet; vergleiche »Ausblick« in der Einleitung).

Die Größenlimits werden den heutigen Verhältnissen angemessener gestaltet bzw. ganz aufgehoben. Es wird ein asynchroner, zusätzlicher Schreib- und Lesemodus eingeführt, der die Geschwindigkeit deutlich erhöhen wird.

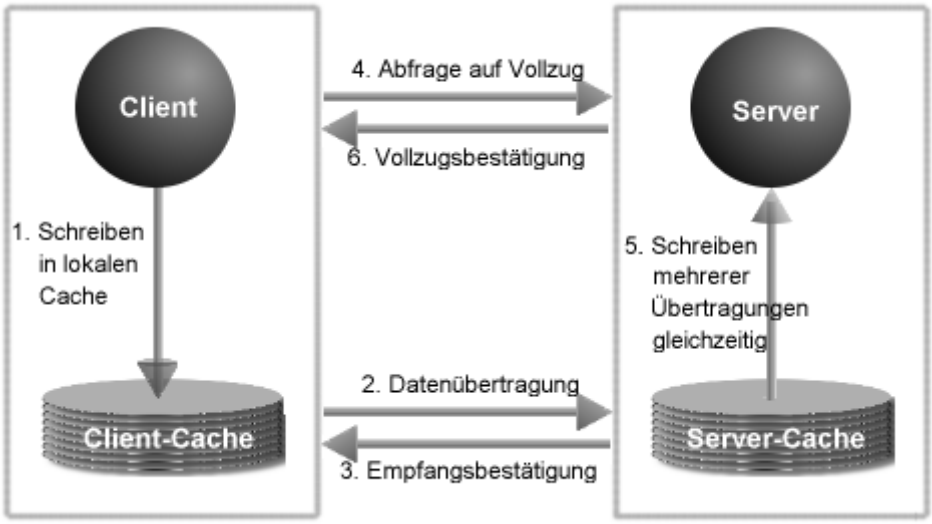


Abbildung 7: Asynchrones Schreiben in NFS-3

Dabei wird nicht mehr wie beim synchronen Zugriff ein Arbeitsvorgang komplett abgeschlossen, sondern die gecachten Daten werden beim Server zwischengespeichert und »bei geeigneter Gelegenheit« auch gesammelt festgeschrieben. Erst auf die endgültige Bestätigung hin verwirft der Client seine Daten im Cache.

Auch bei den Lesezugriffen wurden neue Prozeduren eingeführt. So wurden beim Objektaufruf (bspw. *ls -al*) zuerst alle Daten übertragen (*READDIR()*) und danach jeweils die Attribute (*GETATTR()*) der Objekte. Dies wird in Version 3 durch eine vollständig neue Prozedur (*READDIRPLUS()*) ersetzt, die alles in einem Aufwasch erledigt und die erhoffte Steigerung der Geschwindigkeit nach sich zieht.

NFS nach Version 3 arbeitet anstatt mit 32 Bit mit 64 Bit langen Dateizeigern, sodass auch hier die insbesondere für Datenbankanwendungen schmerzlichen Schranken der Datei- und Dateisystemgrößenbeschränkung fallen.

Mittlerweile wird eine vierte Version (NFSV4) erstellt. Die Tests sind weitestgehend abgeschlossen und nun wird an einer Verabschiedung, die voraussichtlich 2002 erfolgen wird, gearbeitet.

Network Information System

Übersicht

Als Benutzer erwarten Sie in einem lokalen Netzwerk, dass Sie, unabhängig vom konkreten Rechner, an dem Sie sich gerade anmelden, stets die gleiche Umgebung vorfinden. Der naive Ansatz, solchen »Komfort« zu realisieren, wäre aus Benutzersicht, Daten, die Sie an einem Rechner ändern, bei Beendigung der Sitzung auf allen anderen Rechnern zu replizieren. Natürlich wäre eine solche Lösung zum einen ziemliche Vergeudung von Speicherplatz (da die Daten mehrfach vorhanden wären) als auch, und der Grund wiegt wohl schwerer, nur mit hohem Aufwand zu betreiben. Letzterer ließe sich sicher mittels eines Skripts automatisieren und damit reduzieren... aber die meisten Anwender wären mit dem Verfassen eines solchen überfordert.

Selbst auf den Schultern des Administrators würde unnötig Mehrarbeit lasten, denn unter seine Verantwortung fiele es, jedem Benutzer Zugang zu jedem Rechner einzuräumen. Der nächste Auftrag für Sisyphus stünde an, sollten die Daten zu einem Benutzer geändert oder dessen Zugang gelöscht werden. Ob die Datenkonstistenz in umfangreichen Netzwerken noch gewährleistet wäre? Wohl kaum!

Der Ausweg ist die zentrale Lagerung wichtiger administrativer Daten und ein Dienst, der diese netzwerkweit zur Verfügung stellt. Eben der Network Information Service (NIS).

Gelbe Seiten

»Gelbe Seiten« ist der Inbegriff für eine Sammlung nützlicher Informationen. Wohl deshalb nannte *SUN Microsystems* seine erste Version eines solchen Dienstes »Yellow Pages«. Doch kollidierte der Name mit einem eingetragenen Warenzeichen der *British Telecom*, sodass SUN nach kurzem rechtlichen Gefecht auf den Begriff *Network Information Service* auswich. An den ürsprünglichen Namen erinnern noch die zahlreichen Programme rund um NIS, die, von wenigen Ausnahmen einmal abgesehen, allesamt mit dem Präfix »yp« beginnen.

SUN legte die Spezifikation zu NIS offen und reichte gleich noch eine freie Referenzimplementierung nach, die von BSD aufgegriffen und als Basis einer eigenen Realisierung von NIS diente. Die NIS-Client-Funktionalität in der Bibliothek »GNU-libc Version 5« basiert komplett auf diesem BSD-Code.

Der in den aktuellen »Glibc«-Versionen (»glibc-2«) enthaltene NIS-Code ist eine komplette Neuimplementierung der alten Routinen, ergänzt um neue Funktionen zur Zusammenarbeit mit SUN's NIS-Nachfolger *NIS+* und einem erweiterten Konfigurationsschema. Im Wesentlichen werden wir nachfolgend auf die Belange der Konfiguration dieser NIS-Version eingehen und auf Abweichungen des alten Schemas nur kurz verweisen.

NIS vs. NIS+

SUN's NIS+ hat - abgesehen vom Wortstamm - nicht viel mit dem traditionellen NIS gemeinsam. Anstatt einer einzelnen Domain setzt NIS+ auf einen hierarchischen Verwaltungsbereich ähnlich der Strukturierung des Domain Name Service. Ein Knoten der Baumstruktur definiert eines von 6 möglichen NIS+-Objekten, wobei die Wurzel stets ein Objekt vom Typ »directory« ist. Innerhalb des Baums existieren die speziellen »directory«-Objekte »org_dir« und »groups_dir«. Ersteres enthält die zu verwaltenden Daten in Form von Tabellen und letzteres umfasst Einträge, die den Zugriff auf die Daten kontrollieren. Ein Paar von Objekten »org_dir« und »groups_dir« bildet zusammen mit dem übergeordneten »directory«-Objekt eine NIS+-Domain. Um das Schema flexibel zu halten, sind innerhalb der Tabellen Verweise auf Objekte aus anderen NIS+-Domains zulässig.

Die aus Anwendersicht wichtigsten Neuerungen von NIS+ sind die Unterstützung von Datenverschlüsselung und Authentifizierung via SecureRPC.

Bislang existiert keine Implementierung eines NIS+-Servers für Linux, sodass wir es bei dieser knappen Einführung zu NIS+ belassen.

Die Datenbasis

Da die Suche nach Datensätzen in reinen ASCII-Dateien recht ineffizient ist, hält der NIS-Server die Daten in so genannten **Maps** vor. Die Daten der Maps sind im DBM-Format (Database Management) organisiert, sodass der Zugriff mittels Hashing-Funktionen enorm beschleunigt erfolgt. Die Maps müssen hierzu zunächst aus den »Original-ASCII-Dateien«, wie bspw. der /etc/passwd oder der /etc/group generiert werden. Diese Originale werden auch als *Master-Dateien* bezeichnet und oft existieren mehrere Maps für eine Master-Datei - für jeden Suchschlüssel eine. Im Falle der Passwortdatei sind die beiden Maps »passwd.byname« und »passwd.byuid« üblich, die die schnelle Suche per Benutzername bzw. per UserID ermöglichen.

Für bestimmte Maps werden gern Spitznamen vergeben, um den Schreibaufwand beim Zugriff auf diese möglichst gering zu halten. Die einzigen Programme, die diese Spitznamen verstehen, sind **ypcat** und **ypwhich** (Vergleiche Diagnose). Mit ersterem Programm können alle bekannten Spitznamen in Erfahrung gebracht werden:

```
user@sonne> ypcat -x
Benutze "ethers"        für Map "ethers.byname"
Benutze "aliases"       für Map "mail.aliases"
Benutze "services"      für Map "services.byname"
Benutze "protocols"     für Map "protocols.bynumber"
Benutze "hosts"         für Map "hosts.byname"
Benutze "networks"      für Map "networks.byaddr"
Benutze "group"         für Map "group.byname"
Benutze "passwd"        für Map "passwd.byname"
```

Welche Informationen via NIS?

Die Motivation zum Einsatz von NIS entspringt wohl in nahezu allen Fällen dem Wunsch, auf allen Rechnern eines lokalen Netzwerks den Benutzern eine identische Umgebung anzubieten. Demnach sind die Konfigurationsdateien der Benutzerverwaltung, insbesondere die Dateien »/etc/passwd« und »/etc/group«, heiße Anwärter auf die Verwaltung durch einen NIS-Server. Normalerweise geht die zentrale Verwaltung der Benutzer einher mit einer zentralen Bereitstellung der Home-Verzeichnisse über NFS.

Prinzipiell kann jede ASCII-Datei via NIS den Clients zur Verfügung gestellt werden. Wirklich sinnvoll erscheint eine solche Konfiguration allerdings nur bei Dateien, die für alle Clients identisch aufgebaut und deren Inhalte dynamischer Natur sind. Denn so erspart sich der Administrator den Aufwand, nach Änderungen an einer Datei diese auch auf allen Clients nachziehen zu müssen.

Der Aufwand zur Erzeugung einer Map ist vergleichsweise enorm - solange kein geeignetes Skript sich der Sache annimmt. Den meisten Distributionen liegt daher ein Makefile bei, das die Map-Erzeugung für gängige Dateien automatisiert. Was eine »gängige Datei« ist, ist hierbei Ermessensfrage des Distributors; Maps für /etc/networks, /etc/hosts, /etc/protocols, /etc/services, /etc/ethers,... gehören (fast) immer zum Standardrepertoire.

Abschließend sei darauf hingewiesen, dass es eine Konfigurationsfrage des NIS-Clients ist, welche der zur Verfügung stehenden Maps er vom Server bezieht und welche Dateien er lokal betrachtet (Vergleiche /etc/nsswitch.conf beim NIS-Client).

Manuelles Erzeugen der Maps

Das Kommando zum Erzeugen einer Map aus einer Master-Datei ist **makedbm**.

/usr/lib/yp/makedbm [Optionen] Master-Datei Map-Name

Die wichtigsten Optionen sind »-c«, womit **makedbm** nach Erzeugung der Map den NIS-Server (das Programm **ypserv**) anweist, seinen Cache zu leeren (der Server bekommt somit die Änderungen mit) und »-r«, womit Kommentarzeilen (Zeilen, die mit »#« beginnen) automatisch entfernt werden.

Ältere Versionen der Bibliothek »glibc« (Versionen < 2.2) unterstützten einzig NIS-Schlüssel und -Daten mit einer

maximalen Länge von 1024 Bytes (genau genommen entstammt diese Beschränkung dem NIS-Protokoll). Um die Zusammenarbeit aktueller Server auch mit Clients zu gewährleisten, welche derartige Bibliotheken verwenden, überprüft »makedbm« in der Voreinstellung die Schlüssel- und Datensatzlänge und verweigert die Map-Generierung bei Überschreitung der Limits. Erst die Option »--no-limit-check« hilft in einem solchen Fall weiter (passen Sie ggf. den makedbm-Aufruf im weiter unten beschriebenen Makefile an!). In heterogenen Netzwerkumgebungen ist diese Option mit Vorsicht zu genießen, da etliche NIS-Clients kommerzieller Unix-Systeme mit langen Datensätzen und Schlüsseln nicht zusammen arbeiten.

»Makedbm« arbeitet zeilenweise, indem alle Zeichen bis zum ersten Leerzeichen oder Tabulator als Schlüssel und der Rest der Zeile als Daten interpretiert werden. Spätestens jetzt sollte Ihnen bewusst werden, dass die Map-Generierung doch nicht in jedem Fall trivial ist, denn nicht immer sollte die erste Spalte als Schlüssel dienen und - noch schlimmer - nicht jede Master-Datei trennt die Spalten per Leerzeichen oder Tabulator. Bestes Beispiel ist die Datei »/etc/passwd«, die den Doppelpunkt als Separator nutzt. Und die Lösung des »Problems«?

»Makedbm« erhält die Daten in »geeignet aufbereitetem« Format, i.d.R. generiert ein Awk-Skript die korrekten Datensätze. Ein Aufruf zum Erzeugen der Map »passwd.byname« könnte wie folgt aussehen:

```
user@sonne> awk -F: '{ if ( $1 != "" && $3 >= 500) print $1"\t"$0}' /etc/passwd | /usr/lib/yp/makedbm -m
sonne.galaxis.de - passwd.byname
```

Obigen Aufruf finden Sie in etwas abgewandelter Syntax in der Datei »/var/yp/Makefile«. Knapp »umrissen« finden folgende Schritte statt: Awk verwirft zunächst sämtliche Leerzeilen und Datensätze, deren Benutzerkennzeichen kleiner als 500 sind (i.d.R. werden »reale« Nutzeridentifikationen ab 500 vergeben). In allen übrigen Zeilen extrahiert Awk die erste Spalte (Benutzerkennzeichen), die als Schlüssel in der Map dienen soll, und schreibt deren Inhalt sowie nochmals die gesamte Eingabezeile, voneinander durch einen Tabulator getrennt, auf die Standardausgabe. Makedbm liest von der Standardeingabe (symbolisiert durch das Minus vor dem Map-Namen) und schreibt das Ergebnis in die Map »passwd.byname«.

Den Inhalt einer Map vermag ein Hex-Editor darzustellen (das folgende Beispiel dient einzig der Demonstration; wie viele Bytes tatsächlich vor den relevanten Informationen stehen, differiert von Map zu Map, sodass der Zahlenwert der Option »-j« vermutlich nicht korrekt sein wird):

```
user@sonne> od -k 12288 -c passwd.map
0030000  Y  P  _  M  A  S  T  E  R  _  N  A  M  E  s  o
0030020  n  n  e  .  g  a  l  a  x  i  s  .  d  e  Y  P
0030040  _  L  A  S  T  _  M  O  D  I  F  I  E  D  1  0
0030060  2  3  3  8  1  6  6  1  n  o  b  o  d  y  n  o
0030100  b  o  d  y  :  x  :  6  5  5  3  4  :  6  5  5
0030120  3  3  :  n  o  b  o  d  y  :  /  v  a  r  /  l
0030140  i  b  /  n  o  b  o  d  y  :  /  b  i  n  /  b
0030160  a  s  h  u  s  e  r  u  s  e  r  :  x  5  0  x
0030200  :  5  0  0  :  1  0  0  :  B  e  i  s  p  i  e
0030220  l  u  s  e  r  :  /  h  o  m  e  /  u  s  e  r
0030240  :  /  b  i  n  /  b  a  s  h
```

Automatisches Erzeugen der Maps

Das dem NIS-Paket beiliegende Makefile enthält Steueranweisungen, um die Maps zu den gängigsten Dateien automatisch zu generieren. Ein Makefile enthält mehrere Einsprungspunkte, die so genannten Ziele (Targets), wobei das erste Ziel der Datei die voreingestellte Einsprungsmarke ist. Dieses default-Ziel testet die Umgebung und verzweigt seinerseits zum mit »all« bezeichneten Ziel. Der einfachste Weg, alle notwendigen Maps automatisch zu erzeugen, ist, dieses »all«-Ziel entsprechend anzupassen. Suchen Sie hierzu die mit **all:** beginnende Zeile in »/var/yp/Makefile«:

```
root@sonne> vi /var/yp/Makefile
...
#all: passwd group hosts rpc services netid protocols netgrp mail \
#     shadow publickey networks ethers bootparams printcap \
#     amd.home auto.master auto.home auto.local passwd.adjunct \
#     timezone locale netmasks
all: passwd group rpc services netid
```

```
...
```

Im vorliegenden Makefile sind alle denkbaren Ziele als Kommentare (die mit »# « beginnenden Zeilen) beschrieben. Aktuell hängt »all« von den Zielen »passwd«, »group«, »rpc«, »services« und »netid«, ab, d.h. mit Bearbeitung von »all« werden auch die anderen Ziele angesprungen. Genau hinter jenen Zielen verbergen sich die komplexen Anweisungen zum Generieren der entsprechenden Maps. Fügen Sie also die Namen der von Ihnen gewünschten Ziele hinter »all« ein oder entfernen Sie sie, wenn sie nicht durch NIS verwaltet werden sollen.

Die Datei /etc/shadow darf nicht bei Systemen via NIS verwaltet werden, wenn diese noch auf die alte Bibliothek »libc5« aufsetzen, da Shadow-Passwörter von dieser nicht unterstützt werden.

Bei Problemen mit der Datensatzlänge (die Generierung der Maps bricht mit entsprechender Meldung ab) sollten Sie den Aufruf von makedbm im Makefile anpassen:

```
root@sonne> vi /var/yp/Makefile
...
# Orignaler Aufruf:
# DBLOAD = $(YPBINDIR)/makedbm -c -m `$(YPBINDIR)/yphelper --hostname`
# Geänderter Aufruf:
DBLOAD = $(YPBINDIR)/makedbm --no-limit-check -c -m `$(YPBINDIR)/yphelper --hostname`
...
```

Der Aufruf kann sich bei anderen NIS-Versionen bzw. Distributionen (hier SuSE 8.0) vom Beispiel unterscheiden, wichtig ist, dass Sie die Option »--no-limit-check« ergänzen, den Rest aber unverändert belassen.

Als letzte Voraussetzung zur Genierung der Maps muss der NIS-Domainname gesetzt werden. In der folgenden Beschreibung zum Master-Server finden Sie eine weitergehende Erläuterung. Für unser Beispiel verwenden wir als NIS-Domainnamen »galaxis.de«:

```
# NIS-Domainname setzen, falls er noch nicht gesetzt ist:
root@sonne> [ -z `domainname` ] && domainname galaxis.de
```

Der schnellste Weg, die Maps zu erzeugen, führt über den Aufruf von make:

```
root@sonne> make -C /var/yp
make: Wechsel in das Verzeichnis Verzeichnis »/var/yp«
gmake[1]: Wechsel in das Verzeichnis Verzeichnis »/var/yp/galaxis.de«
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
gmake[1]: Verlassen des Verzeichnisses Verzeichnis »/var/yp/galaxis.de«
make: Verlassen des Verzeichnisses Verzeichnis »/var/yp«
```

Obiger Aufruf arbeitet ohne Interaktion mit dem Benutzer und eignet sich daher für die Automatisierung (siehe nachfolgenden Punkt »Aktualisierung der Maps«). Für den Fall, dass neben dem Master-Server noch weitere NIS-Server in der Domain arbeiten sollen, ist das Skript **ypinit** für die Map-Generierung zu bevorzugen, da es zur Eingabe der Slave-Server auffordert und diese Liste in der Datei »/var/yp/ypservers« hinterlegt:

```
root@sonne> /usr/lib/ypinit -m

At this point, we have to construct a list of the hosts which will run NIS servers. sonne.saxsys.de is in the list of NIS server
hosts. Please continue to add the names for the other hosts, one per line. When you are done with the list, type a <control
D>.
        next host to add: sonne.saxsys.de
        next host to add: [Ctrl][D]
The current list of NIS servers looks like this:
```

```
sonne.saxsys.de

Is this correct?  [y/n: y]  y
We need a few minutes to build the databases...
Building /var/yp/galaxis.de/ypservers...
Running /var/yp/Makefile...
gmake[1]: Wechsel in das Verzeichnis Verzeichnis »/var/yp/galaxis.de«
Updating passwd.byname...
...
```

Die Option **-m** ist auf dem Masterserver erforderlich. Im Unterschied zum direkten Aufruf von »make« erzeugt »ypinit« alle Maps generell neu, selbst wenn sich an den Masterdateien nichts geändert haben sollte. Letztlich ist »ypinit« wohl nur zum erstmaligen Erzeugen der Maps hilfreich oder im Falle sich ändernder Slave-Server. Aber selbst dann bleibt es Ihnen überlassen, ob Sie »make« bevorzugen und die Datei »/var/yp/ypservers« von Hand anpassen (jeden Server auf eine neue Zeile!).

Falls der NIS-Server nicht läuft, erhalten Sie bei der Map-Generierung - ob via »ypinit« oder »make« ist unerheblich - Fehlermeldungen der Art: »failed to send 'clear' to local ypserv: RPC: Program not registered«. Die Erzeugung der Maps wird dadurch nicht beeinträchtigt.

Im Verzeichnis »/var/yp« finden Sie anschließend das Unterverzeichnis »galaxis.de« (der Name entspricht exakt dem NIS-Domainnamen), das die erzeugten NIS-Maps enthält.

Aktualisierung der Maps

Änderungen an den vom NIS-Server bereitgestellten Konfigurationsdateien werden i.d.R. nicht direkt an den Maps vorgenommen, sondern mit eventuell vorhandenen Werkzeugen an den Master-Dateien in »/etc«. Der Administrator sollte jede Modifikation schnellstmöglich in die NIS-Maps übernehmen, nur... ob er wirklich stets dran denkt?

Der erfahrene Admin sorgt vor und automatisiert die Anpassung mittels eines Cron-Jobs. Im Terminkalender »/etc/crontab« steht dann bspw. folgender Eintrag:

```
root@sonne> vi /etc/crontab
...
*/10 * * * *   root   make -s -C /var/yp
...
```

Mit jenem Eintrag würde make alle 10 Minuten die Maps aktualisieren, insofern sich an den zugehörigen Masterdateien seit der letzten Generierung etwas geändert haben sollte. Im schlimmsten Fall könnte sich ein Benutzer, der soeben sein Passwort geändert hat, erst nach 10 Minuten mit dem neuen Passwort am System anmelden.

Master-Server

Verantwortungsbereich

Jeder NIS-Server ist für genau einen Verantwortungsbereich zuständig. Die Zugehörigkeit zu diesem Bereich ist durch die Verwendung einunddesselben (NIS-)**Domainnamens** gegeben. Dieser Domainname hat *nichts* mit der gleichnamigen Bezeichnung einer DNS-Domain zu tun, auch wenn er im Falle von NIS+ gleich lauten sollte (nicht muss!). Im Unterschied zum DNS wird beim NIS-Domainnamen die Groß- und Kleinschreibung unterschieden.

Der NIS-Domainname muss vor dem Start des NIS-Servers gesetzt sein, hierzu bedient sich der Administrator des Kommandos **domainname**:

```
root@sonne> domainname
```

```
root@sonne> domainname galaxis.de
root@sonne> domainname
galaxis.de
```

I.d.R. wird der Domainname unmittelbar in einem der Skripte während des Bootens des Systems gesetzt (bei SuSE bwsp. in »/etc/init.d/boot.localnet«). Um den Namen permanent zu speichern, wird dieser in der Datei »/etc/defaultdomain« abgelegt. Das Kommando »domainname« kann nun mittels der Option »-F *Datei*« zur Entnahme des Domainnames aus einer Datei angewiesen werden.

/var/yp/securenets

Da die Zugehörigkeit zu einer NIS-Domain allein durch den gemeinsamen Domainnamen gegeben ist, kann ein jeder Rechner schon durch Verwendung des gleichen Namens Mitglied der Domain werden. Als Mitglied verfügt er über Zugriff auf sämtliche vom NIS-Server verteilte Maps. In offenen Netzwerken birgt diese einfache Verfahrensweise ein hohes Sicherheitsrisiko in sich.

Ein gangbarer Weg der Zugriffskontrolle, der durchaus auch in der Praxis angewandt wird, ist der Start des NIS-Servers bei Bedarf durch den Inetd, wobei ein zwischengeschalteter TCP-Wrapper die Verifizierung des Clients übernimmt. Allerdings wird der NIS-Server meist als permanter Server-Dienst betrieben, sodass anstatt des Wrappers ein anderes Verfahren tritt. Die neueren Implementierungen des Server-Prozesses verwenden daher die Datei »/var/yp/securenets«, die die Zugangsregeln enthält.

Die Datei »/var/yp/securenets« besteht aus Zeilen mit jeweils einem Paar aus Netzmaske und Netzwerkadresse. Die Adresse eines zugreifenden Clients wird bitweise UND mit der Netzmaske verknüpft. Stimmt das Ergebnis mit der Netzwerkadresse überein, ist der Zugriff gestattet. Bei Abweichung ignoriert der NIS-Server die Anforderung und protkolliert den fehlgeschlagenen Kontakt.

Anstatt der Netzmaske 255.255.255.255 darf das Schlüsselwort »host« verwendet werden. Die nachfolgende Datei enthält eine kommentierte Beispielkonfiguration:

```
root@sonne> vi / var/ yp/ securenets
# Zugang für »localhost« explizit erlauben:
host            127.0.0.1

# Zugang für Rechner aus dem Netz 192.168.100.0 erlauben:
255.255.255.0       192.168.100.0

# Zugang jedem Rechner erlauben (Vorsicht!):
#0.0.0.0          0.0.0.0
```

Eine fehlende Datei »/var/yp/securenets« gestattet den Zugang für jeden Rechner.

Nach Änderungen an der Datei kann der laufende NIS-Server durch das Signal SIGHUP zum Einlesen der Konfiguration bewogen werden:

```
root@sonne> killall -HUP ypserv
```

/etc/ypserv.conf

Die Datei »/etc/ypserv.conf« enthält vier Optionen für den NIS-Server »ypserv« und weitere Regeln zur Steuerung des Zugriffs von Clients auf die verschiedenen Maps. Letztere Regeln betreffen ebenso den Map-Transfer-Server »rpc.ypxfrd« der bei Betrieb von NIS-Slave-Servern eine Rolle spielt und später im Abschnitt betrachtet wird.

Jede *Option* steht auf einer extra Zeile und besteht aus dem Schlüsselwort, gefolgt von einem Doppelpunkt und dem Wert der Option. Nicht explizit erwähnte Optionen werden mit ihrem Default-Wert interpretiert, der in nachfolgender Aufzählung in Klammern unterstrichen angegeben ist:

dns

files

[30], Anzahl der Map-Dateihandle, die der Server maximal im Cache zwischen speichern soll.

trusted_master

Diese Option ist für NIS-Slave-Server relevant und enthält den vollständigen Namen des Master-Servers. De Slave-Server akzeptiert neue Maps nur vom angegebenen Master-Server. Fehlt diese Option, wird der Slave keine neuen Maps akzeptieren.

xfr_check_port

[yes, no], der NIS-Server wartet an Ports mit Nummern < 1024.

Die Optionen »tryresolve« und »sunos_kludge« sind veraltet und werden nicht mehr unterstützt; »trusted_master« wurde erst mit ypserv-2.2 eingeführt.

Die Datei »/var/yp/securenets« kontrolliert den Zugriff zum NIS-Server, wobei dieser entweder gewährt oder verhindert werden konnte. Mit den Regeln aus der Datei »/etc/ypserv.conf« kann der Zugriff detaillierter auf konkrete Maps beschränkt werden (natürlich muss ein betreffender Client durch »/var/yp/securenets« überhaupt den NIS-Server kontaktieren dürfen).

Eine Zeile für eine *Zugangsregel* besteht aus 4 durch Doppelpunkte getrennten Spalten mit folgender Bedeutung (bei älteren NIS-Implementierungen existierte noch eine optionale fünfte Spalte):

IP-Adresse

Die IP-Adresse, für die die Regel gilt. Netzwerkadressen und Wildcards sind ebenso zulässig.

NIS-Domain

Name der NIS-Domain, für die diese Regel gilt. Ein Stern »*« steht für »alle Domains«.

Map-Name

Name der Map, für die die Regel gilt. Ein Stern »*« steht für »alle Maps«.

Sicherheit

Die 3 möglichen Werte sind »none«, »port« und »deny«.

»none« erlaubt den Zugriff stets, während »deny« diesen konsequent ablehnt. Mit »port« werden Zugriffe gestattet, wenn sie an einem Port mit einer Nummer < 1024 erfolgen.

Anmerkung: Die Datei »ypserv.conf« in älteren NIS-Implementierungen kannte keine Spalte »NIS-Domain« Dafür existierten zwei Spalten, die es ermöglichten, ein bestimmtes Feld einer Map zu verbergen, indem dessen Inhalt bei der Auslieferung durch ein *x* ersetzt wurde. Anwendung fand die Methode zum Ausblenden des Passwortfeldes entsprechender Maps (passwd.*, groups.*). Die aktuelle Implementierung maskiert Passwörter von Haus aus, sobald eine Regel für einen Client zutrifft. Ohne Regel ist der Zugriff stets erlaubt, womit keine Maskierung stattfindet!

Die nachfolgende Datei enthält eine kommentierte Beispielkonfiguration:

```
root@sonne> vi / etc/ ypserv.conf
# Allgemeine Optionen:
dns          : no
files        : 30
```

```
trusted_master : server.galaxis.de
xfr_check_port : yes

# Zugangsregeln:
# Keine Einschränkung für Rechner des lokalen Netzwerks
192.168.100.0/255.255.255.0 : galaxis.de : *             : none

# Kein Zugriff auf passwd.* aus folgendem Netzwerk
192.168.12.0/255.255.254.0  : galaxis.de : passwd.byuid  : deny
192.168.12.0/255.255.254.0  : galaxis.de : passwd.byname : deny

# Kein Zugriff für Rechner aus anderen Netzwerken
*                           : *          : *             : deny
```

Achten Sie auf die Reihenfolge der Regeln, denn die erste zutreffende gilt!

Portmapper

Die Server-Dienste rund um das Network Information System basieren allesamt auf den Remote Procedure Call (RPC). Jeder RPC-Server muss sich zum Zeitpunkt seines Starts beim so genannten Portmapper registrieren, d.h. der Portmapper muss vor jedem zu startenden RPC-Dienst aktiviert werden.

Wir verzichten hier auf eine nochmalige Diskussion des RPC-Mechanismen. Ausführliche Einblicke gewährt ein eigener Abschnitt zum Remote Procedure Call. In der Diskussion zum NFS-Server finden Sie weitere Hinweise.

Ob der Portmapper auf Ihrem System aktiv ist, können Sie u.a. durch Aufruf von »rpcinfo - p« in Erfahrung bringen. Läuft er nicht, ähnelt die Ausgabe der folgenden:

```
root@sonne> rpcinfo -p
rpcinfo: Kann den Portmapper nicht erreichen: RPC: Fehler des entfernten Systems - Verbindungsaufbau abgelehnt
```

Eventuell existiert in Ihrem System ein Runlevel-Skript zum Start des Portmappers. Bei SuSE und RedHat ist dies bspw. »/etc/init.d/portmap«. Rufen Sie als Root das Skript mit dem Argument »start«:

```
root@sonne> /etc/init.d/portmap start
Starting RPC portmap daemon                                    done
```

Fehlt ein solches Skript, dann suchen Sie in Ihrem System nach einem Kommando mit dem Namen »portmap« (manchmal auch »portmapper«) und starten dieses.

Start der Programme

Der NIS-Server wird mittels des Kommandos **ypserv** gestartet. Auch zum Start dieses Kommandos bringen viele Distributionen ein entsprechendes Runlevelskript (oft »/etc/init.d/ypserv«) mit, das mit der Option »start« auszuführen ist. Nach geglücktem Start sollte »rpcinfo - p« in etwa folgende Ausgabe produzieren:

```
# Start »von Hand«
root@sonne> ypserv
root@sonne> rpcinfo -p
   Program Vers Proto   Port
   100000    2   tcp    111  portmapper
   100000    2   udp    111  portmapper
   100004    2   udp    986  ypserv
   100004    1   udp    986  ypserv
   100004    2   tcp    989  ypserv
   100004    1   tcp    989  ypserv
```

Warum taucht der Server gleich vierfach in der Ausgabe auf? Tatsächlich ist nur ein einziges Serverprogramm aktiv, das sich mehrfach beim Portmapper registriert. Der Portmapper benötigt genaue Informationen, über welche Fähigkeiten ein Serverdienst verfügt. Der NIS-Server vermag Anforderungen sowohl über TCP als auch über UDP

zu behandeln und unterstützt hierbei jeweils die Protokollversionen 1 und 2. Die Version 1 wird in aktuellen Konfigurationen oft fehlen, sie ist ohnehin nur bei vorhandenen NIS-Clients notwendig, die auf SunOS-4-Systemen laufen.

Auch zum Start des NIS-Servers existieren in den Distributionen entsprechende Runlevel-Skripte. Falls vorhanden, sollten Sie diese auch nutzen, da intern eine rudimentäre Prüfung der korrekten Konfiguration stattfindet. Bei SuSE und RedHat finden Sie das Skript »/etc/init.d/ypserv«, das mit der Option »start« zu rufen ist.

```
# Start mittels Runlevel-Skript (hier SuSE und RedHat)
root@sonne> /etc/init.d/ypserv
Starting ypserv                                                    done
```

Nun haben Sie einen NIS-Server laufen, Ihre Benutzerverwaltung ist zentralisiert... doch was hat ein Benutzer zu tun, um bspw. sein Passwort zu ändern? Das Kommando passwd manipuliert nur lokale Dateien, der Zugriff auf die Server-Datenbanken ist ihm verwehrt. Auf Clientseite wird sich ein Benutzer daher an neue Kommandos gewöhnen müssen (im Falle des Passworts »yppasswd«), auf Serverseite steht dem NIS-Server ein weiterer Daemon zur Seite, der sich um Belange der Benutzerdaten kümmert. Der Server **rpc.yppasswdd** nimmt diesbezügliche Anforderungen von NIS-Clients entgegen, prüft die Berechtigungen und manipuliert die Felder der Passwortdateien. Abschließend führt »rpc.yppasswdd« das Skript »pwupdate« aus, das die Maps »passwd.*« und »shadow.byname« neu generiert.

In der Voreinstellung gestattet »rpc.yppasswdd« einzig das Setzen der Passwörter. Mittels der Optionen »-e chsh« und »-e chfn« kann den Benutzern der Zugriff auf das Shell- bzw. das Info-Feld ermöglicht werden. Auch die Verwendung alternativer Dateien kann »rpc.yppasswdd« mittels »-p <Passwortdatei>« bzw. »-s <Shadowdatei>« naheglegt werden.

Der Start des Server »rpc.yppasswdd« erfolgt entweder über das gleichnamige Kommando oder über ein Runlevelskript (insofern vorhanden):

```
# Start »von Hand«
root@sonne> /usr/sbin/rpc.yppasswdd

# Start mittels Runlevel-Skript (hier SuSE und RedHat)
root@sonne> /etc/init.d/yppasswdd
Starting yppasswdd                                                 done
```

Noch ein dritter Dienst könnte serverseitig von Interesse werden. Nämlich dann, wenn neben dem Master-NIS-Server auch noch ein oder mehrere Slave-NIS-Server die Clients einer Domaine bedienen und im Zuge sich häufig änderter und umfangreicher Maps die Geschwindigkeit der Aktualisierung eine zu wünschen übrig lässt.

Der Master-Server wird die Slave-Server über geänderte Maps unterrichten. Das Standardverfahren sieht nun vor, dass die Slaves das Kommando »/usr/lib/yp/ypxfr« starten, welches die Maps vom Master-Server herunter lädt. Die Kopien landen zunächst in einem temporären Verzeichnis und erst nach erfolgreichem Abschluss aller Transfers wird der Slave hieraus seine neue Datenbasis generieren.

Wird serverseitig der Dienst **rpc.ypxfrd** aktiviert, kümmert sich das zu Grunde liegende RPC-basierte Dateitransferprotokoll um eine gesicherte Übertragung, sodass ein Client die Maps vom Server direkt übernimmt, anstatt daraus erst eine neue Datenbasis zu erzeugen.

Obwohl »rpc.ypxfrd« durchaus per (x)inetd gestartet werden könnte, empfehlen die Entwickler wegen dessen gemächlichen Startzeiten den Start im Zusammenhang mit »ypserv«.

```
root@sonne> /usr/sbin/rpc.ypxfrd
```

Bei SuSE existiert gar ein Runlevelskript namens »/etc/init.d/ypxfrd«.

Diagnose

Eine Überprüfung der Funktionalität des Servers erfolgt mittels der Client-Werkzeuge zum Zugriff auf die Informationen der Maps. Den einzelnen Kommandos widmen wir uns konkret in der Diskussion zum NIS-Client; hier interessiert und einzig, ob der Server antwortet.

Dazu starten wir auf dem NIS-Server gleichzeitig einen NIS-Client. Ein simpler Aufruf von **ypbind** genügt:

```
root@sonne> /usr/sbin/ypbind -broadcast
```

Die Option »-broadcast« erzwingt eine Suche nach einem NIS-Server für die Domain. Auf sie kann verzichtet werden, wenn der Server in der Datei /etc/yp.conf eingetragen ist. Diese »saubere« Lösung ist aber Bestandteil der Client-Konfiguration und sollte uns an dieser Stelle nicht näher interessieren.

Der einfachste und Aussage kräftigste Test ist die Anfrage, welcher Rechner denn nun der Master-Server für eine Domain ist. Ein **ypwhich**, gerufen ohne Argumente, sollte den Server benennen:

```
user@sonne> ypwhich
sonne.galaxis.de
```

Wenn hier schon etwas schief läuft, dann ist mit aller Wahrscheinlichkeit kein Server aktiv. Überprüfen Sie ggf. die am Portmapper angemeldeten Dienste.

Als Weiteres sollten Sie den Zugriff auf eine komplette Map verifizieren, bspw. indem Sie **ypcat** mit dem Namen der gewünschten Map konsultieren:

```
user@sonne> ypcat passwd
nobody:*:65534:65533:nobody:/var/lib/nobody:/bin/bash
user:esL06uX_l3kjg:500:100:Beispieluser:/home/user:/bin/bash
tux:aZxx0Xpl3HgH:501:100:Beispieltux:/home/tux:/bin/bash
```

Solange die geforderte Map existiert, wird ihr Inhalt aufgelistet werden. Mit »ypcat -x« bringen Sie die Namen aller vom Server bedienten Maps in Erfahrung.

Schon konkreter, und in Hinsicht auf die spätere transparenten Zugriff auf Informationen der Maps von Bedeutung, ist eine gezielte Anfrage nach einem einzelnen Datensatz einer Map. So sollte folgender Aufruf die Information zum Benutzer »user« aus der Passwort-Datei extrahieren:

```
user@sonne> ypmatch user passwd
user:esL06uX_l3kjg:500:100:Beispieluser:/home/user:/bin/bash
```

Wenn soweit alles funktioniert, sollten Sie die einzelnen Schritte auf entfernten Clients wiederholen. Die Feuerprobe hat Ihr Server allerdings erst bestanden, wenn auch das Login via NIS von den Client-Rechnern aus funktioniert. Gelingen die obigen Schritte, während das Login missglückt, sollten Sie das Kommando **getent** bemühen, um den tasächlich vom NIS-Server gelieferten Daten auf die Schliche zu kommen.

```
tux@erde> getent passwd user
user:esL06uX_l3kjg:500:100:Beispieluser:/home/user:/bin/bash
```

Ggf. erkennen Sie hier Felder, die vom Server maskiert wurden. Allerdings sollte dies nur bei älteren Implementierungen noch der Fall sein.

Slave-Server

In Netzwerken mit nur wenigen angeschlossenen Rechnern wird ein einzelner Rechner als NIS-Server vollauf genügen. Zuverlässige Hardware vorausgesetzt, ist die Ausfallwahrscheinlichkeit des Servers gering, zumindest wenn er nicht zusätzlich als Arbeitsplatzrechner genutzt wird. In großen Netzwerken erweisen sich auch weniger die (Linux-)Server als Quelle von Ausfällen, sondern vielmehr die Netzwerkhardware wie Router, Switches oder einfach nur die Verkabelung. Ein stabiler Server nützt dann herzlich wenig, wenn der Client wegen unterbrochener

Leitungen keinen Kontakt zu diesem findet. Hier bieten sich Slave-Server in den unabhängigen Netzsegmenten an, um zum einen den Master-Server zu entlasten und zum anderen bei Ausfall dessen die wesentlichen Aufgaben weiter anbieten zu können. »Wesentlichen« Aufgaben meint damit die Bereitstellung der Informationen der Maps. Änderungen an diesen, wie bspw. das Manipulieren von Passwörtern mittels »yppasswd«, kann jedoch einzig vom Master selbst vorgenommen werden und ist bei Ausfall dessen nicht möglich.

Um einen Rechner in den Stand eines NIS-Slave-Server zu erheben, muss dieser ebenso das Kommando **ypserv** ausführen. Das impliziert, dass sowohl der NIS-Domainname gesetzt als auch der Portmapper aktiv sein muss.

```
root@erde> domainname galaxis.de
root@erde> rpcinfo -p 2>/dev/null || portmap
root@erde> rpcinfo -t localhost ypserv 2>/dev/null || ypserv
```

Woher erfährt der Slave nun den Namen des Master-Servers und die Namen der von ihm verwalteten Maps? Indem der Slave-Server sich der Funktionen eines NIS-Clients bedient. D.h. das Kommando **ypbind** ist zu starten:

```
root@erde> rpcinfo -t localhost ypbind 2>/dev/null || ypbind -broadcast
```

Dieselben Bibliotheksfunktionen, denen sich bspw. das Kommando **ybwhich** bedient, nutzt der Slave, um die Informationen vom Master anzufordern.

Als abschließenden Schritt muss auf dem Slave die Datenbasis angelegt werden:

```
root@erde> usr/lib/yp/ypinit -s `ypwhich`
```

Anstatt »ypwhich« können Sie auch direkt den Namen des Master-Servers oder eines anderen Slave-Servers mit aktueller Datenbasis angeben.

Ggf. sollten Sie auf dem Master-Server noch den Dienst **rpc.ypxfrd** starten. Näheres dazu finden Sie im Abschnitt zum Master-Server (»Start der Programme«).

Um die Funktionalität zu verifizieren, könnten Sie bspw. den Master-Server kurzfristig außer Dienst setzen und NIS-Anfragen absetzen, die nun von einem Slave-Server beantwortet werden sollten.

Domain Name Service

Autor: Bernd Reimann

Übersicht

Anmerkung: In der Literatur finden Sie die Bezeichnungen *Domain Name System* und *Domain Name Service*. Im Grunde meinen beide einunddasselbe. *Domain Name System* bezeichnet genau genommen die *Methode* der Umsetzung von IP-Adressen in symbolische Namen. Die technische Umsetzung hingegegen, also alle Routinen zusammen genommen, die das System implementierten, formen einen Dienst - den *Domain Name Service*. Der Praktiker spricht also vom *Service* und der Theoretiker vom *System*.

Vorläufer des DNS

Zu Beginn der sechziger Jahre, als die USA und die damalige UdSSR sich mitten im Kalten Krieg befanden und das Internet (bzw. *ARPAnet*) als solches noch gar nicht existierte, gab das amerikanische Verteidigungsministerium (*Department of Defense, DoD*) den Auftrag, ein dezentrales, ausfallsicheres und technisch fortschrittliches Netzwerk zu entwickeln und zu implementieren, um die vermeintliche Bedrohung durch die UdSSR möglichst gering zu halten.

Einige der renommiertesten Universitäten, Organisationen und Unternehmen arbeiteten an der Umsetzung und 1969 war es schließlich soweit. Vier Universitäts-Rechner in Kalifornien (*University of California*, Los Angeles, *University of California Santa Barbara*, *Stanford Research Institute*) und Utah (*University of Utah*) bildeten die Grundlage des neuen ARPAnets und konnten miteinander kommunizieren. In diesem Stadium der Entwicklung wurden die Informationen der verschiedenen Systeme noch in einer einzigen Datei, der »HOSTS.TXT«, verwaltet, von der alle angeschlossenen Systeme über identische Kopien verfügten. Diese Datei ist quasi ein Vorläufer der heutigen /etc/hosts-Datei.

Von der Idee und des dahinter steckenden Aufwandes war dies eine außerordentlich clevere und einfach zu handhabende Administrationsangelegenheit. Doch mit der Umstellung des Netzwerkprotokolle auf TCP/IP Mitte der siebziger Jahre stieg die Anzahl der zu verwaltenden Systeme immens an, womit die Aktualität der Datei »HOSTS.TXT« kaum mehr gewährleistet werden konnte und deren Umfang unpraktikable Ausmaße annahm. Der ursprüngliche Ansatz, eine dezentrale Verwaltung zu erschaffen, wurde aufgrund des benötigten Überblicks bei der Vergabe neuer Rechnernamen und -adressen »ad absurdum« geführt.

So war es unumgänglich, dass in den achtziger Jahren - die Anzahl der verbundenen Rechner betrug mittlerweile über 1000 - ein neues System, das Domain Name System (DNS), entwickelt und eingeführt wurde.

Ziele und Funktionen des DNS

Wie bereits erwähnt, übernimmt DNS die Aufgabe der Umwandlung von Internetnamen in numerische Internetadressen und umgekehrt. Diese Auflösung basiert auf der Verwendung des TCP/IP-Protokoll-Stacks. Hierbei wird jedem öffentlich zugänglichen System (Host, Router, Gateway, ...) eine eindeutige 32-Bit große binäre Zahl zugeordnet (IP-Adresse), über die es angesprochen werden kann. Zur Vereinfachung des Umgangs mit IP-Adressen werden diese in vier Oktetts unterteilt und in Form von durch Punkte getrennten Dezimalzahlen dargestellt (»11111111 11111111 11111111 11111111« ➔»255.255.255.255«). Ein Oktett besitzt somit einen Werteumfang von 0..255 ($2^8 = 256$). Seltener anzutreffen ist die hexadezimale Notation (»FF.FF.FF.FF«), wobei diese vorrangig beim Nachfolger des jetzigen IP-Adressformats angewandt wird (IPv6).

Da das menschliche Gehirn bekanntlich besser mit Namen als mit Zahlen umgehen kann, wird jeder dieser Adressen ein eindeutiger Namen zugeordnet. So lässt sich grundsätzlich mit der IP-Adresse »216.239.39.101« als auch mit dem Namen »www.google.de« die gleiche Web-Seite auf dem gleichen System erreichen. Da die Vermittlung im Internet aber einzig auf IP-Adressen basiert, ist bei Verwendung von symbolischen Rechnernamen vorab eine Adressauflösung durchzuführen.

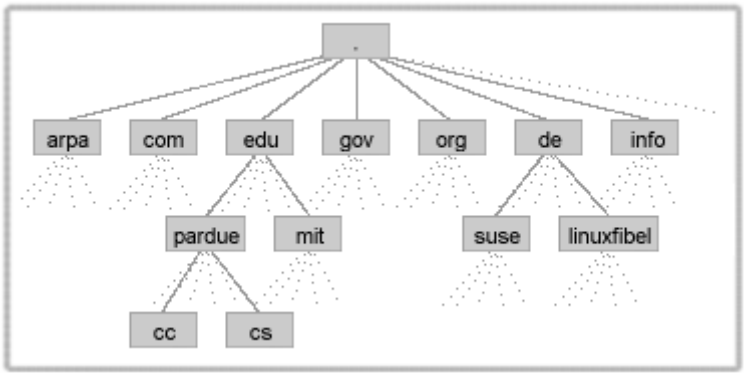


Abbildung 1: DNS-Namensraum

Um diese Auflösung zu verstehen, sollten Sie den Aufbau des DNS-Namensraumes kennen (Abbildung 1). Ausgehend von einer Wurzel (root), die die Bezeichnung ».« trägt, sind die Domains in einer baumartigen Struktur organisiert. Die der Wurzel folgende Ebene umfasst die so genannten *Top Level Domains (TLD's)*. Eine Ebene tiefer folgen die Subdomains, denen entweder unmittelbar die Rechnernamen folgen oder aber eine weitere Ebene lokaler Domains, unterhalb derer dann die Rechnernamen liegen.

Top-Level-Domains

Die Top-Level-Domains (TLD) gliedern sich in zwei Kategorien ein. Zur Kategorie der so genannten *generischen TLD's* zählen »org« (Non-Profit Organisationen), »mil« (militärische Einrichtungen), »com« (Kommerzielle Unternehmen), »edu« (Bildungseinrichtungen), »net« (Netzorganisationen) und »gov« (Regierungsbehörden). Im Jahre 2000 erfuhren die generischen TLD's eine Erweiterung um »biz«, »info«, »coop«, »aero«, »name«, »pro« und »museum«.
Die zweite Kategorie beinhaltet die *länderspezifischen TLD's*, wie bspw. »de« (Deutschland), »uk« (Grossbritannien), »aw« (Aruba), »cc« (Cocos Islands), »jp« (Japan) oder »ch« (Schweiz). Eine vollständige Liste aller TLD's wird von der *Internet Assigned Numbers Authority* (IANA), der Dachorganisation des Internets, verwaltet.

Die Abarbeitung eines Internetnamens, bspw. »www.linuxfibel.de«, erfolgt hierarchisch von rechts nach links, d.h. (vergleiche Abbildung 1) beginnend in der Wurzel führt der Zweig »de« (TLD) zum Zweig »linuxfibel« (registrierte Domain), der wiederum im Punkt »www« (Name eines Rechners der Domain »linuxfibel.de«) endet. Die Suche nach einem entsprechenden System erfolgt also stets baumabwärts beginnend im Root-Verzeichnis.

Korrekterweise müsste eine Domain immer mit einem abschliessenden Punkt, der das Root-Verzeichnis repräsentiert, geschrieben werden. Allerdings kann dies vernachlässigt werden, da inzwischen alle DNS-Werkzeuge hinreichend intelligent und tolerant sind, um trotzdem ein positives Ergebnis zurück zu melden. Bei einer Bind-Konfiguration wäre jedoch ein fehlender Punkt am Ende fatal, wie Sie später noch erfahren werden.

Forward- und Reverse-Lookup

Bei einer Namensauflösung wird i.d.R. das Äquivalent zu einem Internetnamen in Form einer Internetadresse, also eine Umsetzung von Namen nach Adresse, gesucht.

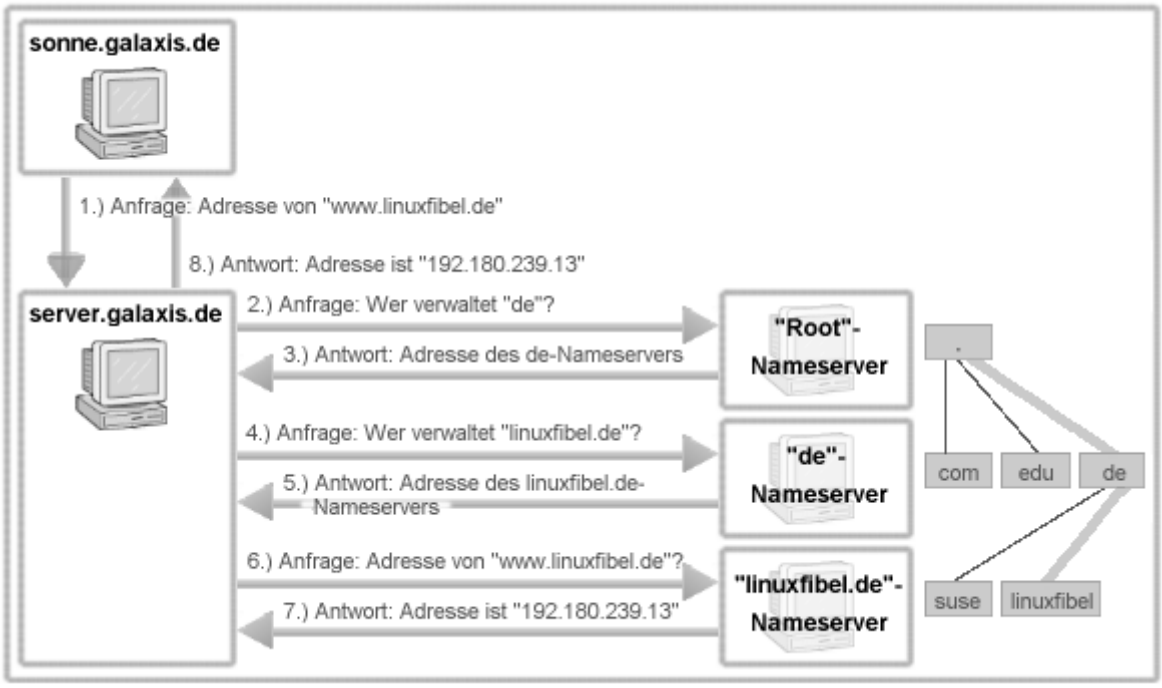


Abbildung 2: Auflösung der IP-Adresse eines Rechnernamens

Bedarf eine Anwendung der IP-Adresse zu einem Rechnernamen (»www.linuxfibel.de« vergleichen Sie das Beispiel in Abbildung 2), so übernimmt der so genannte *Resolver*, eine Sammlung von Funktionen aus der C-Bibliothek, die weitere Recherche. In typischen Konfigurationen wird der Resolver eines Clients (»sonne.galaxis.de«) zunächst die lokale Datei /etc/hosts nach entsprechenden Einträgen durchforsten. Wird er dort nicht fündig, reicht er die Anfrage an den zuständigen Nameserver weiter (in Abbildung 2 als »server.galaxis.de« bezeichnet). Jeder Nameserver verfügt über einen Cache, indem er die Daten der zuletzt recherchierten Anfragen eine Zeit lang zwischenspeichert. Erst wenn der lokale Nameserver die Adresse nicht in seinem Cache hat, beginnt der mit der Auflösung des Namens von $raquo;hinter her«. D.h. er fordert einen der Rootserver an (eine Liste solcher hält der Nameserver in einer Konfigurationsdatei), ihm die Adresse des für die Domain »de« zuständigen Nameservers mitzuteilen. An die gelieferte Adresse sendet er die folgende Anfrage nach der Adresses des für »linuxfibel.de« zuständigen Nameservers. Bei letzterem Server erhält er schließlich die gewünschte Informationen, die er sowohl in seinen Cache einträgt, als auch zum anfragenden Clientrechner weiterleitet.

Zur Auflösung in die Gegenrichtung (Adresse in Namen), das sogenannte *Reverse Lookup*, wurde eine neue Domain »in-addr.arpa« (***A**ddress and **R**outing **P**arameter **A**rea domain*) eingeführt. Unterhalb dieser speziellen Domain existieren 256 Subdomains (0..255). Insgesamt wiederholt sich diese Unterteilung viermal, bis die 32-Bit-Adresse vollständig dargestellt ist (Abbildung 3).

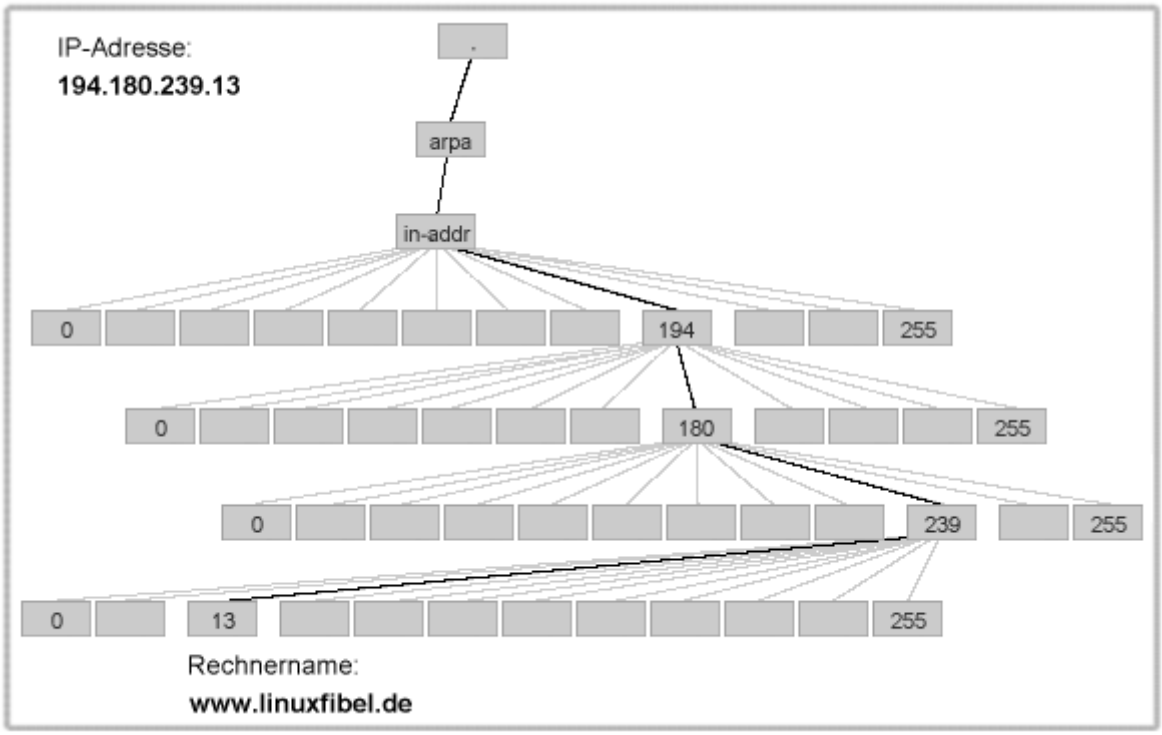


Abbildung 3: Reverse-Lookup mit Hilfe der Pseudodomain »in-addr.arpa«

Für das kommende IP-Adressformat IPv6 wurde die Domain »ip6.arpa« eingeführt, die nach einem ähnlichen Prinzip funktioniert, wie »in-addr.arpa« für IPv4 (offizielle Bezeichnung des IP für 32 Bit Adressen).

Anmerkung: Eine weitere Pseudodomain »e164.arpa« dient der Recherche von *whois*-Anfragen.

Unterschiede zwischen Domains und Zonen

Eine sehr wichtige Unterteilung in diesem Schema ist, um Verwirrungen auszuschließen, der Unterschied zwischen Domain und Zone. Leider ist dieser Unterschied nicht allzu groß und daher auch nicht allzu offensichtlich, aber immerhin vorhanden. Eine Domain beinhaltet alle unter ihm liegenden Domains, Zonen und Systeme (Hosts), d.h. alle baumabwärts liegenden Verwaltungseinheiten gehören zu dieser Domain. Da dies aber keinerlei Sinn macht alles auf dieser Ebene zu verwalten, wurden sogenannte Zonen eingerichtet. Diese Zonen konnten nun an die verschiedenen Einrichtungen und Unternehmen abgegeben werden, die die Verwaltung übernehmen und neue Knoten zeitnaher und schneller einbinden konnten. Jede Zone ist in sich abgeschlossen und beinhaltet nur die Knoten, die direkt zu ihr gehören. Sollten weitere Unterteilungen benötigt werden, so werden neue Zonen errichtet, die jedoch völlig unabhängig voneinander sind.

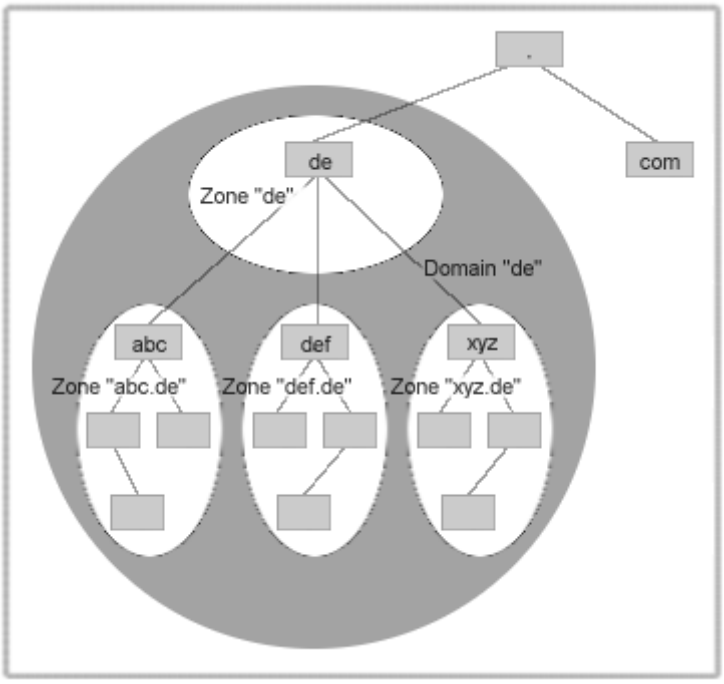


Abbildung 4: Zone und Domain

An dieser Stelle werden Namensserver (Nameserver) eingesetzt um die Verwaltung der Zonen zu übernehmen. Hierbei kann jedem Nameserver eine oder auch mehrere Zonen delegiert werden, für die er zuständig ist. Man spricht auch von "Autorität" einer Zone.

Generell besitzt jede Zone einen primären Nameserver ("primary Master", "Primary") und evtl. mehrere sekundäre Nameserver ("secondary Master", "Secondary"), die die Zonendaten untereinander austauschen, um auf dem neuesten Stand zu bleiben.

Eine Domain ist also ein logischer Oberbegriff, der zwar im täglichen Sprachgebrauch sehr häufig verwendet wird, im DNS wird jedoch lediglich mit Zonen gearbeitet. Oftmals findet auch keinerlei Unterscheidung statt, so dass die beiden Begriffe äquivalent behandelt werden.

Die Bind-Software

Die Entstehungsgeschichte der Bind-Software

Die Entwicklung des *Berkeley Internet Name Domain Pakets*, kurz *Bind*, startete Anfang der achtziger Jahre an der *University of Berkeley* als Diplomarbeitsprojekt einiger Studenten und stand unter Aufsicht namhafter Organisationen wie *Defense Advanced Research Projects Administration* (DARPA) und der *Computer Systems Research Group* (CSRG). Letztere Organisation zeichnete für die Versionen bis einschließlich 4.8.3 verantwortlich.

Zwischen 1985 und 1987 widmeten sich vorrangig Angestellte von *Digital Equipment Corporation* (DEC) der weiteren Entwicklung von Bind. Unter Federführung von DEC erschienen die Versionen 4.9 und 4.9.1. Die Version 4.9.2 sponserte *Vixie Enterprices*; *Paul Vixie* wurde zum Chefarchitekten von Bind.

Ab 4.9.3 ging die Verantwortung an das 1993 gegründete *Internet Software Consortium* (ISC) über, das von nun an für die Entwicklung und den Vertrieb bis einschließlich Version 8.2 zuständig war. Diese Organisation bezog ihre Mittel hauptsächlich von Sponsoren, die allerdings über die Jahre hinweg immer weniger freigiebig wurden, so dass mit Version 9 die Verantwortung an das Unternehmen *Nominum Inc.* abgegeben wurde, das seither die Weiterentwicklung vorantreibt.

Mit dem Start seiner Entwicklung wurde Bind zum Quasi-Standard und ist nahezu mit DNS selbst gleichzusetzen. Dieser Marktführerschaft trägt sogar *Microsoft* Rechnung, indem sie eine NT-Version von Bind herausbrachten, die allerdings in der internationalen IT-Infrastruktur keine bedeutende Rolle spielt.

Überblick über die einzelnen Versionen und ihre Merkmale

Die Wechsel der Verantwortlichkeiten für die Entwicklung von Bind deckt sich stets mit entscheidenden Versionssprüngen. Jede dieser drei Hauptversionen beinhaltete grundlegende Änderungen und Erweiterungen.

Die 4er Versionen brachten den Durchbruch von Bind und setzten so den Startpunkt als de-facto-Standard. Fortan fehlte Bind in kaum einem der Unix-Systeme. Das Kernstück in Bind-4 bildete eine Datei namens »/etc/named.boot«, in der sich die Informationen über die Lage aller weiteren Konfigurationsdateien, die zum Betrieb notwendig waren, befanden. Dem Konzept der zentralen Datei ist Bind bis heute treu geblieben.

Funktionen wie *Caching*, d.h. das Vorhalten von Adressinformationen bereits aufgelöster Namen bzw. Adressen, *Forwarding*, dem Weiterleiten an bspw. dedizierte Nameserver, oder *Query Logging*, die Analyse getätigter Auflösevorgänge, wurden implementiert. Bind-4 brachte allerdings ebenso ein erhöhtes Sicherheitsrisiko mit sich, da etliche entscheidende Programmteile mit teils extremen Fehlern (Bugs) behaftet waren. Trotzt zahlreicher Nachbesserungen am Quellcode konnte Bind-4 nie seinen Ruf als potentielle Sicherheitslücke abwerfen und verschwand mit Erscheinen der Version 8 binnen kurzer Zeit von der Servern.

In der Version Bind-8 wurden weite Teile des Quellcodes neu- bzw. komplett umgeschrieben. Weiterhin kamen neue Funktionen hinzu, die heute nicht mehr wegzudenken sind, wie dynamische Updates, Benachrichtigungen der Slaveserver bei Änderungen, ACLs zur Zugriffskontrolle auf den Server, verbesserte Zonentransfers und Updates sowie eine verbesserte Performance. Das Sicherheitsrisiko sank durch die Neuprogrammierung enorm.

Die aktuellste Version ist Bind-9. Auch hier wurden Sicherheit und Stabilität zum obersten Gebot. Signaturen ermöglichen nun gesicherte Zonenzugriffe (*DNS Security*, DNSSEC); *Transaction Signatures* (TSIG) ermöglichen signierte DNS-Anfragen. Das IPv6-Protokoll wird in Bind-9 vollständig unterstätzt, indem es neue Ressource-Einträge implementiert (»A6«, »AAAA«, »DNAME«) und beide Schreibformate der 128-Bit-Adressen akzeptiert (»Nibble« [veraltet] und »Bitstring«). Zur effizienteren Arbeit können Zonen mit konkreten Views angelegt werden, womit Definitionen getrennter Zonen für Anfragen aus dem internen und dem externen Netz möglich sind. Derartige Zonen sind komplett voneinander abgeschottet und parallel auf einunddemselben System lauffähig. Als wesentliche Neuerung unterstützt Bind-9 nun Multiprozessorsysteme. Wie auch seine Vorgänger ist Bind-9 für alle verbreiteteten Plattformen verfügbar.

Die wichtigsten DNS/Bind-Programme (Bind-Utils) kurz beschrieben

Zu diesem Zeitpunkt sollen die verschiedenen Programme und Daemons nur kurz benannt und ihr Zweck erläutert werden. Weitere Informationen zu den administrativen Vertretern der Programme erhalten Sie im weiteren Text. Die Hilfsprogramme zum Zugriff auf Informationen des DNS werden im Zusammenhang mit dem DNS-Client diskutiert.

Dig

Dig (*Domain Information Groper*) ermöglicht die interaktive Befragung von DNS-Servern. Es vollzieht DNS-Recherchen und veranschaulicht die Antworten aller befragten Server. Dig ist somit vorallem zur Diagnose der DNS-Funktionalität nützlich.

Dnssec-keygen, dnssec-makekeyset, dnssec-signkey, dnssec-signzone

Bei diesen Tools handelt es sich um die DNS Security Extensions, die dafür gedacht sind ein gesicherte Kommunikation zwischen den einzelnen Nameserver aufzubauen. Vor Bind9 war nur eine rudimentäre Unterstützung dieser Tools vorhanden. Das Prinzip, dass hier zugrunde liegt ist das Public-Key-Verfahren, bei dem ein Schlüsselpaar erstellt wird und der öffentliche Schlüssel bei einer Zertifizierungsstelle hinterlegt wird.
Da dieses Verfahren noch relativ neu ist und auch umwälzende Veränderungen weltweit haben wird, ist es kaum vorstellbar, dass es sich in nächster Zeit schnell durchsetzen kann.
dnssec-keygen: Mit diesem Befehl wird das Schlüsselpaar erstellt. Sie können wie bei PGP hier auch den Verschlüsselungs-Algorithmus und die Schlüssellänge mitgeben.
dnssec-makekeyset: Hiermit wird eine Datei erstellt, bei der Ihre Zone eingebunden ist und an den Betreiber Ihrer Parent-Zone geschickt wird, der diese signieren und akzeptieren kann. Sie sehen hier beginnen bereits die grössten Schwierigkeiten, da es sich hierbei meist um .de-Zonen handelt, auf die dann eine grosse Flut hereinbricht.
dnssec-signkey: Damit signiert der Parent-Zone-Betreiber Ihre Schlüssel mit seinem eigenen privaten Schüssel

dnssec-signzone: Letztendlich müssen Sie nun noch Ihre Zone mit dem erhaltenen Schlüssel signieren um Steueranweisungen sicher zu erhalten

Host

Host ist ein weiteres DNS-Abfragewerkzeug. In der Voreinstellung begnügt es sich mit der Ausgabe eines Adresse zu einem gegebenen Rechnernamen bzw. umgekehrt. Verschiedene Optionen ermöglichen ähnlich komplexe Recherchen wie »dig«.

Lwresd

Der Lightweight Resolver Daemon (lwresd) ist eine abgespeckte caching-Only Version des normalen Bind-Nameservers, der auf einem eigenen Protokoll (lrp - Lightweight Resolver Protokoll) basierend arbeitet. Normalerweise hört er lediglich auf Anfragen die über das Loopback-Interface (127.0.0.1) kommen.

Named-checkconf

Mit **named-checkconf** wurde ein Tool entwickelt, dass die Syntax der Konfigurationsdatei überprüft und gegebenenfalls interveniert. Sollte sich ein Fehler eingeschlossen haben so erscheint eine Fehlermeldung und die Zeilenangabe. Bei einer richtigen Konfigurationsdatei erhält man einen Return-Code von 0 und keine weitere Meldung. Standardmässig wird die /etc/named.conf als Konfigurationsdatei eingelesen. Sollten Sie diese also irgendwo anders abgelegt haben, so müssen Sie diese explizit angeben, damit die Überprüfung stattfinden kann.

named-checkzone

Ein weiteres Tool ist **named-checkzone**. Mit diesem Tool können Sie Ihre Zonen auf Fehler testen.
Bsp: *named-checkzone zoneA.de /var/named/master/db.zoneA*
Bei Beendigung wird ebenfalls ein Return-Code ausgegeben und evtl. eine kleine Zusammenfassung, ob die Überprüfung etwas ergeben hat.
Bsp: *zone zoneA.de/IN: loaded serial 2003041108*
ok

Nslookup

Nslookup liegt Bind-9 nur noch aus Kompatibilitätsgründen bei. Es ist quasi die »alte« Version von »dig« mit ähnlichem Funktionsumfang aber auch anderem Bedienkonzept (es beinhaltet einen interaktiven Modus). Nslookup beherrscht nicht den Umgang mit Adressen nach IPv6.

Rndc

Das RNDC ist ein Nameserver Kontrolltool, mit dem Anweisungen an den Nameserver geschickt werden können um ihn zu steuern. Dieses Tool ist neu seit Bind9 und löst damit den ndc der Vorgängerversionen ab. Prinzipiell wird in der Konfigurationsdatei des Tools eine Schlüsseldefintion vorgegeben, die den Algorithmus und ein Passwort enthält, mit diesem dann Clientsysteme Änderungen und Steuerungen am Server vornehmen dürfen.

Sicherheit von Bind

Der Domain Name Service ist eine der Schlüsselkomponenten des Internets und folglich ein vorrangiges Ziel potentieller Angreifer. Bei der Komplexität der Software sind Fehler kaum auszuschließen und in jeder bisherigen BIND-Version wurden mehr oder weniger gravierende Schwachstellen aufgedeckt. Gerade als Administrator eines öffentlichen DNS-Servers sollten Sie deshalb gewissenhaft die in Mailinglisten diskutierten Sicherheitsthemen verfolgen und ggf. ihre Serversoftware auf den aktuellsten Stand halten.

Inwiefern die aktuelle Bind-Auflage Fehler beinhaltet, vermag niemand zu erraten. Aber die problematischsten Schwachstellen früherer Versionen sollen im nachfolgenden Text Erwähnung finden.

Erlangen von Root-Rechten

Das höchste der erreichbaren Ziele für jeden Angreifer ist es, die UID 0 zu erhalten, d.h. Herr über den root-Account zu werden. Wurde der root-Account erst einmal übernommen, kann das System zu allem missbraucht werden, ohne die Gegenmaßnahmen fürchten zu müssen. Dieses Erlangen wird oftmals durch Fehler in der Software (Bugs) unterstützt, dass durch sogenannte Exploits ausgenutzt wird. Allerdings sind bei Bind9 bis dato noch keine Exploits bekannt, die fehlerhafte Codeteile ausnutzen können, was auch sehr für den Einsatz in unserer DMZ spricht. Bind4 und Bind8 dagegen waren, vor allem in der Frühphase ihrer Entwicklung, stark von Bugs betroffen und hatten immense Sicherheitsprobleme. Einer dieser Bugs soll hier einmal kurz exemplarisch dargestellt werden.

NXT-Bug: (Einstufung: critical)

Der NXT-Eintrag in den Konfigurationsfiles dient dazu, dass negative Antworten über signierte Zonen die nicht vorhanden sind, trotzdem abgeschickt werden. Er gibt an, welches der nächst erreichbare Zonen-Name ist.

Ein Angreifer, der diesen Bug ausnutzt, muss die DNS-Datenbank verändern und seinen Nameserver als Zonen-Autorität einsetzen. Danach wird eine Anfrage auf die Zone des Angreifers gestartet, der nun die Auflösung von sich selbst erhält. Wenn nun Bind versucht, die Abfrage abzuschließen, indem er eine Verbindung zu dieser Zone zu etablieren, gerät es in einen kritisch instabilen Zustand. Nun tritt der Exploit in Kraft und führt die folgenden Kommandos aus cd /; uname -a; pwd; id womit ein Buffer Overflow erreicht wurde, Bind abstürzt und der Angreifer eine Shell erhält, inklusive sämtlicher Privilegien, mit denen Bind gestartet wurde (meist root).

Eine weiterer Bereich, in dem Schwachstellen auftreten können, ist der administrative Bereich. Hier werden oftmals Konfigurationsfehler, wie zu geringe Zugangsbeschränkungen (Authentifizierung) oder falsche Zoneninformationen, in den Konfigurationsdateien gemacht meist lediglich zum debuggen, oftmals aber auch aus Unkenntnis oder Faulheit. Diese Schwächen können leicht zum Verlust der Kontrolle oder der Daten des Systems führen, werden jedoch durch die neuen Features wie ACL oder DNSSEC, die später noch ausführlich beschrieben werden, entschärft.

Stillegen eines Nameservers

Ein weitere verbreitete Methode einen Nameserver zu attackieren, ist die Möglichkeit ihn durch sogenannte Denial-of-Service-Attacken (DoS) einfach stillzulegen damit er seine Zoneninformation, die er vorhält, nicht mehr weiterverbreiten kann. Als Folge davon kann es zu einem sehr großen Chaos und Durcheinander kommen, da doch viele Anwendungen lediglich mit einer Namensauflösung funktionieren bzw. diese zwingend benötigen.

Insgesamt wird dem DNS-DoS weit weniger Aufmerksamkeit geschenkt, als dem bekannteren IP-Spoofing, mit dem Web-Server attackiert werden, was zur Folge hat, dass die Administratoren dieses Thema nur unzureichend ernst nehmen.

Umleiten einer Domain

Ein noch effektiveres Ergebnis wird durch die Umleitung von Domains erzielt. Hierbei legt der Angreifer in erster Linie kein destruktives Verhalten an den Tag, sondern will die Fähigkeit des bestehenden Nameservers und der Domain für sich ausnutzen. Die Verluste, die dabei entstehen lassen sich teilweise nur sehr schlecht einschätzen, da meist ein Vertrauensbruch zwischen Kunde und Anbieter entsteht. Denn wer wird seine Daten oder Geld weiterhin an einen Anbieter weitergeben, der sich nicht mal selbst schützen kann. Entgangene Gewinne in der Zeit der Umleitung und danach sind ebenfalls als ein großer Verlust zu betrachten, der vor allem Anbieter betrifft, die ihr Geld mit Informationen oder Angeboten aus dem Internet verdienen, wie amazon.de, ebay oder auch booxtra.

Mit Cache Poisoning versucht der Angreifer den bestehenden Puffer-Cache infolge von Zusatzinformationen dahingehend umzuleiten, dass ein Besucher nicht mehr die richtige IP-Adresse zurückgeliefert bekommt, sondern die neue (falsche) Adresse. Diese neuen Informationen erhalten außerdem einen sehr langen Lebenszeit-Eintrag (Time To Live, TTL), da Bind jeden Eintrag erst nach Ablauf der TTL aus seinem Cache verbannt und somit Falschinformationen lang in Umlauf sind.

Beim Response Spoofing (Query-ID-Prediction) erfolgt die Manipulation auf der Paketebene. ähnlich eines TCP/IP-

Handshackes wird in den Paketen eine 16-Bit-ID mitgesendet, die oftmals sequentiell erhöht wird. Hierbei wird versucht bei einer Kommunikation diese ID vorherzusagen und vor der eigentlichen Antwort eine gefälschte Nachricht zu senden. Die eigentliche Antwort wird bei Eintreffen sofort ignoriert und verworfen, da die ID nicht mehr korrekt, bzw. nicht mehr zuordenbar ist.

Bei der Man-in-the-Middle-Attacke setzt der Angreifer ebenfalls eine Falschinformation bei einem Nameserver ein und bekommt nun alle Anfragen zu dieser bestimmten Adresse weitergeleitet. Da dies generell jedem Anwender auffallen würde, wenn er an einem anderen Punkt herauskommt als erwartet, leitet der Angreifer die Anfrage an das ursprüngliche Ziel weiter und arbeitet nun als Vermittler. Warum aber der Aufwand? Gesetzt der Fall, dass das Ziel eine Bank oder ein Anbieter ist, das heikle Daten erwartet, bspw. PINs, TANs oder Kreditkartennummern, würde der Angreifer alle diese Daten zu Gesicht bekommen, abspeichern und weiterleiten. Der Anwender bekommt von dieser Zwischenspeicherung überhaupt nichts mit, denn er bekommt nur sein erwartetes Login oder seine gültige Verifikation zu sehen. Alles was dazwischen geschehen ist, bleibt ihm weitestgehend verborgen.

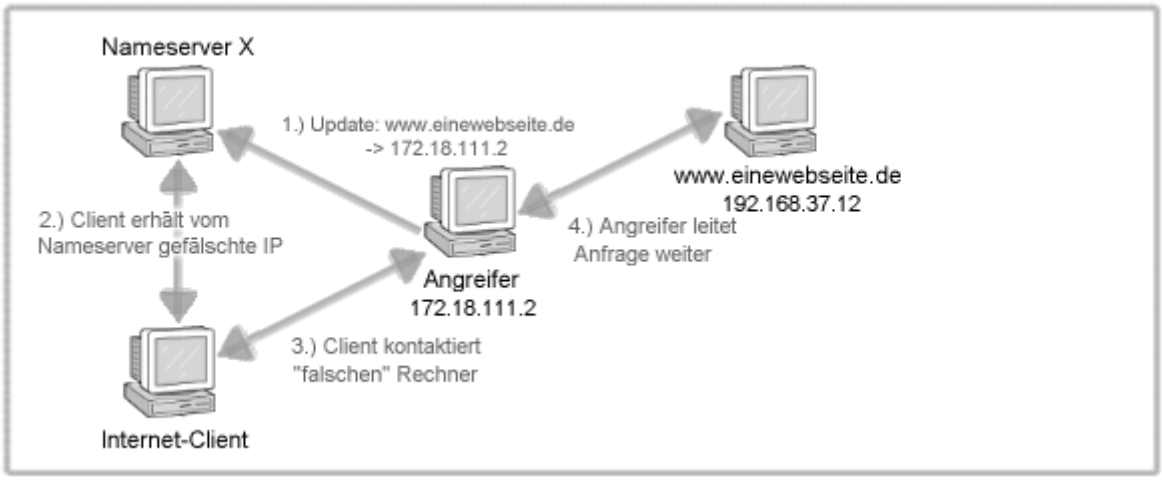


Abbildung 5: Man-in-the-Middle-Attacke

Eine strikte Trennung zwischen den einzelnen Angriffszielen ist nicht möglich, da es oft ein Mix aus allen möglichen Varianten ist. Generell lässt sich sagen, dass zwar Bind9 viele Sicherheitsrisiken durch seine Features ausgeräumt hat, aber bei rein destruktiven Angriffen auf Hardware (siehe DoS) dennoch keine Chancen hat.

Standard-Konfigurationen von Bind

Die vorliegenden Beispiele basieren auf der derzeit (September 2002) aktuellesten Bind-Version 9.2.1. Des Weiteren erfordert die Unterstützung für DNSSEC die Installation von OpenSSL > 0.9.5. Alle Pfadangaben beziehen sich somit auf eine Standardinstallation, d.h. die Datei »named.conf« liegt in »/etc« und die Programme rund um Bind in den Verzeichnissen »/usr/local/bin« bzw. »/usr/local/sbin«.

Die Datei »/etc/named.conf«

Zentraler Anaufpunkt einer jeden Bind-Konfiguration ist die Datei »/etc/named.conf«. Sie enthält die frei wählbaren Namen der Dateien mit den eigentlichen Datenbankeinträgen. Des Weiteren umfasst sie u.a. Steuerdaten, wie Protokollierung, Zugangsbeschränkungen, Schlüssel- und Zonendefinitionen oder Startoptionen für den Bind-Daemon »named«.

Vorerst begnügen wir uns mit einer relativ einfach aufgebauten Datei »named.conf«, anhand derer die Schritte zu einem funktionierenden Nameserver demonstriert werden. Im Verlauf des Abschnitts werden Sie mit etlichen weitergehenden Möglichkeiten der Konfiguration in dieser Datei konfrontiert werden.

```
# /etc/named.conf
# Globale Optionen
options {
   directory "/etc/named";
};
```

```
# Die Root-Zone
zone "." in {
   type hint;
   file "root.hint";
};

# Die lokale Zone
zone "localhost" in {
   type master;
   file "localhost.zone";
};

zone "0.0.127.in-addr.arpa" in {
   type master;
   file "127.0.0.zone";
};

# Die zu verwaltenden Zonen
zone "zoneA.de" in {
   type master;
   file "master/db.zoneA";
};

zone "0.1.10.in-addr.arpa" in {
   type master;
   file "rev/master/db.10.1.0";
};

zone "zoneB.de" in {
   type slave;
   file "slave/db.zoneB";
   masters { 10.2.0.1; };
};

zone "0.2.10.in-addr.arpa" in {
   type slave;
   file "rev/slave/db.10.2.0";
   masters { 10.2.0.1; };
};
```

In dieser kleinen Beispielkonfiguration sieht man, dass der Nameserver, der diese Datei verarbeitet, für die Zone zoneA.de als Master fungiert, d.h. er der primäre Ansprechpartner anderen zur Verfügung steht und das alle änderungen von ihm ausgehen. Der Parameter file beschreibt den Pfad, in dem die Zoneninformationen in verschiedene listenartigen ASCII-Dateien abgelegt werden. Dies ist nicht notwendigerweise zwingend, jedoch bei zunehmender Anzahl zu verwaltender Zonen, würden sonst alle Listen im definierten Root-Verzeichnis eingestellt. Es dient lediglich zur übersichtlichen Haltung der Daten. Bspw. kann der Administrator bei einem manuellen Zonentransfer der Einfachheit halber alle Listen im Verzeichnis rev/slave und slave löschen, ohne vorher überprüfen zu müssen, welche Zonen sich dahinter verbergen.

Zusätzlich zu diesen Daten fehlt noch der Eintrag für den Localhost (127.0.0.1) und die Root-Dateien. Der Localhost ist normalerweise immer sein eigener Master-Nameserver kann aber auch als Slave von anderen fungieren, da diese Informationen stets die gleichen sein müssten. Sicherer ist es, wenn eine Master-Zonendatei lokal vorgehalten wird, da bei einem Ausfall der Localhost nicht mehr angesprochen werden kann.

Die Root-Datei beinhaltet die Adressen aller Root-Server. Dies ist notwendig, wenn der Nameserver die Anfrage nicht auflösen kann und, wie in Abbildung 2 zu sehen ist, ganz oben in der Hierarchie beginnen muss, nach dem Ergebnis zu suchen. Diese Datei ist allerdings bei einem Forwarding nicht erforderlich, da alle unbekannten Abfragen immer an den Forwarder weitergeleitet werden. Eine aktuelle Version dieser Datei kann beim Internic bezogen werden.

Die Root-Zone

Der zentrale Punkt des DNS-Namensraums ist die Wurzel. Vermag ein Nameserver eine Anfrage nicht selbst aufzulösen, beginnt er mit der Suche nach dem Ergebnis ganz oben in der Hierarchie, also bei einem Nameserver, der die Root-Zone verwaltet (Vergleichen Sie auch Abbildung 2). Dessen Adresse muss jeder Nameserver zwingend

kennen. Deshalb liegt jedem Bind-Paket eine Datei bei, die die Liste der Root-Server enthält. Die Adressen der Root-Server werden i.d.R. niemals geändert, sodass Sie von der Aktualität der Liste ausgehen können.

Die lokale Zone

Zonendefinitionen der zu verwaltenden Zonen

In den Zonendefinitionen selbst werden die einzelnen Systeme auf Adressen abgebildet.

```
# /etc/named/master/db.zoneA
$TTL 24h

zoneA.de. IN SOA server1.zoneA.de. admin.server1.zoneA.de. (
          2002060301  ; Seriennummer [hier:JJJJMMTT0-99]
          24h         ; Refresh
          1h          ; Retry
          1w          ; Expire
          1h)         ; negativer TTL-Cache

zoneA.de.              IN  NS   server1.zoneA.de.
zoneA.de.              IN  NS   server2.zoneA.de.

server1.zoneA.de.       IN  A    10.1.0.1
server2.zoneA.de.       IN  A    10.1.0.2
system1.zoneA.de.       IN  A    10.1.0.50
system2.zoneA.de.       IN  A    10.1.0.51
server3.zoneA.de.       IN  A    10.1.0.3
server3.zoneA.de.       IN  MX   10 server3.zoneA.de.

ns1.zoneA.de.           IN  CNAME server1.zoneA.de.
ns2.zoneA.de.           IN  CNAME server2.zoneA.de.
www.zoneA.de.           IN  CNAME server2.zoneA.de.
mailbox.zoneA.de.       IN  CNAME server3.zoneA.de.
```

Neu seit Bind8.2 ist der Eintrag TTL. In früheren Versionen wurde diese Option an anderer Stelle aufgeführt, aber seit der Veröffentlichung von RFC 2308 musste die TTL-Anweisung an einem anderen Ort angegeben werden und hierfür wurde die erste Zeile gewählt. Dieser Wert gibt an, wie lange ein abfragender Nameserver die Daten in seinem Cache halten darf, bevor er die Daten aus dem Cache wieder entfernt.

Der zweite wichtige Eintrag ist der SOA (Start of Authority)-Eintrag. In dieser Zeile stehen die Zone, die Klasse, der Ressource Record selbst, die Autorität der Zone und den Ansprechpartner (Mailadresse auf der Autorität) bei Problemen.

Zusätzlich zu diesen Angaben beinhaltet dieser SOA-Eintrag noch einige Steuerdaten, wie Seriennummer, Aktualisierung oder Zeitpunkt der Ungültigkeit, wobei der wichtigste die Seriennummer darstellt, wie im nächsten Kapitel noch beschrieben wird.

Nachfolgend wird die Abbildung der Namen und Adressen vorgenommen. Die NS-Records geben den Namen aller Nameserver an, die mit der Zone etwas zu tun haben, die A-Records sind eine einfache Abbildung von Namen auf IP-Adressen, Mailserver-Einträge (MX) und CNAME-Records (canonical names) sind Aliase auf bereits beschriebene Namen, womit es möglich ist, ein System mit mehreren Namen anzusprechen.

Die andere wichtige Datei für eine Zone ist die in-addr.arpa -Datei, mit der das Reverse-Lookup ermöglicht wird. Hier auch ein kurzer Auszug mit den wichtigsten Informationen und Einträgen.

```
# /etc/named/rev/master/db.10.1.0
$TTL 24h

0.1.10.in-addr.arpa.IN SOA server1.zone.de.
admin.server1.zoneA.de. (
          2002060301  ; Seriennummer [hier:JJJJMMTT0-99]
          24h         ; Refresh
```

```
        1h      ; Retry
        1w      ; Expire
        1h)     ; negativer TTL-Cache

0.1.10.in-addr.arpa.    IN  NS   server1.zoneA.de.
0.1.10.in-addr.arpa.    IN  NS   server2.zoneA.de.

1.0.1.10.in-addr.arpa.  IN  PTR  server1.zoneA.de.
2.0.1.10.in-addr.arpa.  IN  PTR  server2.zoneA.de.
3.0.1.10.in-addr.arpa.  IN  PTR  system1.zoneA.de.
    .
    .
    .
```

Die ersten beiden Zeilen haben dieselbe Bedeutung wie in der db.zoneA -Datei und dürfen auch hier nicht fehlen. Da diese Datei für das Reverse-Lookup zuständig ist werden hier mit Pointer-Records (PTR) von Adressen auf Namen verwiesen. Dabei wird die IP-Adresse in umgekehrter Form inklusive der Domain in-addr und Arpa angeben (Vgl. Abbildung 4 und 5), da die Auflösung, gleich der Auflösung von Namen, von rechts nach links erfolgt.

Wichtig bei beiden Dateien ist der Punkt hinter den Adress- und Namensangaben . Dieser Punkt symbolisiert die Wurzel (Root), von der alles ausgeht. Wird dieser Punkt nicht angegeben, so hat dies fatale Folgen, denn Bind interpretiert alle Angaben ohne Punkt als relativ und hängt den Zonennamen an. Als Beispiel hier ein Systemeintrag:

```
server1.zoneA.de.       IN  A    10.1.0.1
```

Dieser Name wird durch den fehlenden Punkt am Ende als server1.zoneA.de.zoneA.de. gesetzt und der Nameserver reagiert auch nur auf diesen Namen, d.h. er kennt das System server1 überhaupt nicht. Alternativ dazu gibt es die Möglichkeit den Punkt wegzulassen.

```
server1             IN  A    10.1.0.1
```

In diesem Fall wird richtigerweise an den Namen server1, die Zone .zoneA.de. angehängt. Sind diese beiden Zonen-Dateien auf dem Server vorhanden, so kann das System bereits alle Anfragen zu der verwalteten Zone beantworten.

Um nicht in jede Konfigurations-Datei immer dieselben gleichbleibenden Daten, wie MX-Einträge, SOA-Einträge oder Steuerdaten, eintragen zu müssen, werden diese oft in Dateien außerhalb abgelegt und per INCLUDE-Anweisung in diese Konfigurationsdateien eingefügt. Der Vorteil dieser Vorgehensweise ist vor allem bei änderungen an diesen Daten zu spüren. Sollte sich der SOA-Eintrag ändern oder ein zusätzlicher MX-Record dazukommen muss der Neueintrag lediglich in einer, anstelle von 300 Dateien vorgenommen werden.

Master- und Secondaryzonen

Im vorhergehenden Kapitel ging es rein um die Master-Dateien auf einem Server. Ein System, das als Master für eine Zone fungiert muss die beiden gerade genannten Dateien vorhalten und den Slave-Servern (falls vorhanden) zur Verfügung stellen. Der Slave-Server erkennt durch die Einträge in der /etc/named.conf für welche Zonen er Daten und von welchem Server er diese abrufen muss, d.h. auf ihm werden diese beiden Dateien keinesfalls erstellt. Schon allein aus Redundanz- und Fehlergründen werden die Dateien einmal erstellt (Master) und bei Bedarf transferiert (Slave), so minimiert sich die Pflege erheblich. Dies bedeutet auch, dass beide Zoneninformationsdateien, sowohl auf dem Master- als auch auf dem Slave-Nameserver, vollkommen identisch sind und beide dieselben Records beinhalten [Vgl. G].

Der Slave-Nameserver wird auch nicht bei jeden Aufruf einen Transfer initiieren, sondern lediglich, wenn die TTL ausläuft, die Zonendaten aus dem Cache entfernt oder sich die Seriennummer erhöht. Nun wird auch der Sinn der Seriennummer deutlich. Der Administrator muss bei jeder änderung die Seriennummer erhöhen. überprüft nun der Slave-Nameserver die Zonendateien und stellt fest, dass sich die Seriennummer erhöht hat, initiiert er sofort einen Transfer und bringt seinen Informationsbestand auf den neuesten Stand. Schneller geht es durch die Option notify. Hierbei versendet der Master-DNS eine Benachrichtigung an alle Slave-Nameserver, damit ein Update durchgeführt werden kann.

Nameserver-Start

Der Nameserver-Dienst verfügt über ein paar Startoptionen, die Ihnen die Möglichkeit geben, Ihre Umgebung individuell einzustellen. Hier die wichtigsten:
-c config-file: Alternativer Pfad für die named.conf-Datei, in der die gesamte Konfiguration abgelegt ist.
-d debug-level: Je höher das Debugging-Grad ist, desto mehr Meldungen sind zu sehen
-f : Dies weist den Dienst an, im Vordergrund zu arbeiten (also das Gegenteil zu den Daemons, die lediglich bei Anfragen aktiv werden)
-g : Standardmässig arbeitet der Serverdienst im Hintergrund und wird nur aktiv, wenn eine Anfrage auf seinem Port eintrifft
-p Port: Einstellmöglichkeit eines anderen Port. Der Standard ist Port 53
-t directory: Wenn Sie ein chroot-Umgebung implementiert haben, können Sie diese hier mitgeben
-u user: Dies ist der User, unter dem der Nameserver seinen noramlen Dienst erledigt. Normalerweise ist dies der User named, der bei der Installation angelegt wird.
-v: Gibt die Version mit aus

Erweiterte Konfigurationen

Caching und Forwarding

Jedem Nameserver stet ein Cache zur Verfügung, in dem er alle bereits getätigten Abfragen hinterlegt. Dies dient in erster Linie zum schnelleren Bearbeitung von Anfragen, denn wenn das Ergebnis bereits im Cache vorhanden ist, wird sofort geantwortet und andere Nameserver werden außen vor gelassen. Dies geschieht so lange, bis der Cache entweder gelöscht oder veraltet ist (TTL). Erst dann wird wieder der Weg über andere (evtl. Root-Server) gewählt. Eine besondere Form dieses Caches ist ein Cache-Only-DNS. Dieser Cache-Only-DNS stellt seinerseits keine Anfragen, sondern leitet erhaltene Anfragen lediglich weiter und speichert die erhaltenen Ergebnisse. In dieser Konfiguration beschränkt sich die /etc/named.conf nur auf das nötigste an Optionen. Die wichtigste Option stellt das Forwarding dar.

```
# /etc/named.conf

options {
   directoy /etc/named ;
   forwarders { 122.32.141.4; 193.132.41.41; };
   forward only;
};
```

Beim Forwarding werden die Anfragen exakt an die in den Optionen angegebenen Nameserver weitergegeben. Diese Forwarder sind meist gut dimensionierte Nameserver größerer Firmen bzw. Provider, die über eine hohe Bandbreite auf die Root-Server zugreifen können. Mit dem Forwarding wird auch die eigene Vorhaltung der Root-Server-Datei hinfällig, da alles über diese Forwarder läuft. Eine weitere verschärfte Form eines Forward-DNS ist die Option forward only. Bei dieser Betriebsart leitet der Nameserver ebenfalls alle Anfragen weiter, jedoch verlässt er sich 100%-ig auf die Forward-DNS. Sollte der Forward-DNS kein Ergebnis oder eine Timeout zurückliefern, kontaktiert unser Nameserver keinen weiteren externen Nameserver, sondern vertraut dem Ergebnis des Forward-DNS. Allerdings bleiben die internen Zonen von diesem Forwarding unbetroffen, d.h. wenn Auflösungsanfragen über eine Zone deren Autorität (Master oder Slave) er ist, ankommen, beantwortet er diese natürlich.

Weiterführende Sicherheitsfunktionen und Schutzmechanismen

Diese grundlegenden Konfigurationen ermöglichen zwar einen Betrieb des Nameservers, jedoch gibt es unter Bind9 noch einige weitere fortgeschrittene Funktionen, die ein sichereres und differenzierteres Arbeiten und Administrieren ermöglichen. Die Entwicklung dieser Funktionen zielt in erster Linie in Richtung Sicherheit mit kryptografischen Elementen oder Zugangsbeschränkungen ab und genau diese Features und Funktionen werden in unserem Konzept benötigt um etwas hardening [Vgl. Anhang A Hardening] durchzuführen.

chroot Die abgesicherte Umgebung

Beim chroot (change root) handelt es sich um eine geschützte Arbeitsumgebung für Bind. Es wird, ähnlich wie bei

FTP- oder HTTP-Servern, eine passende Dateistruktur in einem geschützten Bereich aufgebaut, eine Art Gefängnis , bei dem /chroot als Root-Verzeichnis fungiert. Als weitere Sicherheitsmassnahme wird in diesem Zusammenhang auch noch der Besitzer des Daemons, meist root, auf einen ungefährlicheren User gesetzt. Dies geschieht aus dem Grunde, falls ein Angreifer jemals die Kontrolle über den Nameserver, bspw. wie beschrieben durch Buffer Overflows, erlangt, so besitzt er sofort die Rechte, des Users, der den Daemon gestartet hat. Wenn dieser Fall eintritt, findet sich der Angreifer als unterprivilegierter User in einem eingeschränkten Bereich wieder, was doch sehr schnell das Interesse sinken lässt.

Oftmals wird dieser geschützte Bereich auch noch auf ein zusätzliches Filesystem ausgelagert, um bei einer eventuellen DoS-Attacke das eigentliche System nicht der Gefahr auszusetzen, einen Reboot machen zu müssen. Hier kann beim Erkennen, sofort das Filesystem abgehängt werden und das Betriebssystem bleibt weitestgehend unbeeindruckt davon. Da /chroot nun als Wurzelverzeichnis dient, sind andere Verzeichnisse nicht sichtbar und deshalb auch nicht erreichbar.

Aus diesem Grund müssen einige Dateien manuell in dieses Gefängnis kopiert werden, damit ein Betrieb möglich ist. Bei Bind9 sind es nur noch wenige Dateien, wie die Konfigurationsdatei /etc/named.conf und natürlich alle bereits angelegten Zoneninformationsdateien. Eine genauere Dokumentation ist im Anhang D zu finden.

ACL - Zugriffslisten

Ein weitverbreitetes Feature von Bind sind die ACLs (access control list). Bei diesen ACLs wird unter einem Arbeits- oder symbolischen Name, Systeme mit gleichen Berechtigungen oder Eigenschaften zusammengefasst, die später unter diesem Namen angesprochen werden. Diese Zusammenfassung wird Address-Match List genannt.

```
acl telekom {
      194.25.0.125;
      194.25.0.121;
      194.25.1.113;
      194.25.15.217;
      194.246.96/24;
      129.70.132.100;
      195.244.245.27;
};

acl forwarder (
      194.25.0.68;
      194.25.0.52;
      194.25.0.60;
};
```

Neben der eigenen Vergabe noch ACLs gibt es noch einige vorgegebene Schlüsselwerte wie all (alle Systeme), none (kein System), localhost (alle internen Interfaces) und localnets (alle Netze, zu denen der Server ein direktes Interface besitzt).

Mit Hilfe dieser ACLs können nun die Berechtigungen sehr feinkörnig definiert werden.

```
zone zoneA.de IN {
   type master;
   allow-transfer { none; };
   allow-query { all; };
   allow-notify { forwarder; localnets; };
   allow-update { telekom; localhost; };
   file master/db.zoneA ;
};
```

In diesem Beispiel sind alle vier möglichen Berechtigungen kurz aufgeführt, die für die Zone zoneA.de gelten. Sollen diese Berechtigungen global gelten, so müssen sie unter die option -Option gestellt werden, da alle dort gemachten Einstellungen für sämtliche folgenden Zonen Gültigkeit besitzen.

Views - DNS Split

Der nächste Schritt, nachdem die ACLs definiert sind, ist eine Aufteilung in Sichten (Views). Dies bedeutet, verschiedene Systeme bekommen bestimmte zugeteilte Zoneninformationen zu sehen, andere jedoch nicht. Besonders in einer DMZ ist dies sehr hilfreich, da einerseits externe Nameserver und Clients aus dem Internet, als auch interne Systeme auf einige Zonen zugreifen dürfen, die nicht unbedingt die gleichen sein müssen.. Bei dieser Konstellation ist es nicht förderlich, wenn die interne Struktur von außen einsehbar ist. Gerade aus diesem Grund dürfen externen Anfragern lediglich die öffentlichen Zoneninformationen zur Verfügung stehen. Vor Bind9 wurde dieses Problem durch eine Aufteilung auf 2 Nameserver gelöst, wobei der eine für interne Zonen und der andere für externe Abfragen zuständig war. Nun, da dies mit internen Funktionen möglich ist, können gezielt Szenarien aufgebaut werden, die einen weiteren Schutzmechanismus bilden.

```
view intern {
...
};

view extern {
match-clients { DMZ-Proxy; SMTP; ...};
zone zoneA.de IN {
type master;
allow-transfer { Reg-DNS; };
allow-query { all; };
allow-notify { forwarder; localnets; };
allow-update { telekom; localhost; };
file /extern/master/db.zoneA ;
}; };
```

Die Option match-clients gibt an, für welche Systeme überhaupt die nachfolgenden Zonen zugänglich, bzw. sichtbar sind. Zur besseren Verwaltbarkeit sind diese meist als ACL angegeben. Der nun folgenden Rest ist äquivalent zu den bereits in vorigen Kapiteln beschriebenen Zonendefinitionen. Vorsichtig sollte der Administrator bei der Rangfolge sein, denn eine ANY-Zusage zu Beginn der Definition kann später nicht mehr zurückgenommen werden. Aus diesem Grunde werden zuerst beschränkten Zugriffe definiert, damit auf diese Zonen auch wirklich nur ausgewählte Systeme zugreifen können und erst danach wird der Nameserver für alle anderen geöffnet.

Ligthweight-Resolver

Laufzeitkonfiguration und Diagnose

Laufzeitkonfiguration mit Rndc

Rndc (Remote Name Daemon Control) ist ein Administrationstool, das dem Administrator eine Eingriffsmöglichkeit bietet mit der Befehlszeile den Daemon zu steuern. Es ist die Weiterentwicklung des Tools ndc, dass mit Bind8 zur Verfügung stand, um auch über ein bestehendes TCP-Netz die Kontrolle zu behalten. Mit ihm können Befehle wie Neuladen von Konfigurationsdateien oder Zonen (reload), Laden von neuen oder veränderten Zoneninformationen (reconfig), Statistiken (stats), Erstellen von Cache-Dumps (dumpdb), Stoppen nach gesicherten Veränderungen (stop), sofortiger Stop ohne Sicherung (halt), Erhöhen des Debug-Levels (trace), Leeren des Caches (flush) oder Anzeigen des Status des Nameservers (status).

Die Sicherheit der Netzadministration wird durch einen signierten Schlüssel gewährleistet, der mit einigen anderen Angaben in einer Steuerungsdatei namens /etc/rndc.conf steht.

```
# /etc/rndc.conf

key "rndc-key" {
    algorithm hmac-md5;
    secret "reOtiWFGpPN7wzlk5OEE7Q==";
};

options {
    default-server localhost;
```

```
    default-key "rndc-key";
};
```

Diagnose

named-checkconf

Mit **named-checkconf** wurde ein Tool entwickelt, dass die Syntax der Konfigurationsdatei überprüft und gegebenenfalls interveniert. Sollte sich ein Fehler eingeschlossen haben so erscheint eine Fehlermeldung und die Zeilenangabe. Bei einer richtigen Konfigurationsdatei erhält man einen Return-Code von 0 und keine weitere Meldung. Standardmässig wird die /etc/named.conf als Konfigurationsdatei eingelesen. Sollten Sie diese also irgendwo anders abgelegt haben, so müssen Sie diese explizit angeben, damit die Überprüfung stattfinden kann.

named-checkzone

Ein weiteres Tool ist **named-checkzone**. Mit diesem Tool können Sie Ihre Zonen auf Fehler testen.
Bsp: *named-checkzone zoneA.de /var/named/master/db.zoneA*
Bei Beendigung wird ebenfalls ein Return-Code ausgegeben und evtl. eine kleine Zusammenfassung, ob die Überprüfung etwas ergeben hat.
Bsp: *zone zoneA.de/IN: loaded serial 2003041108*
ok

Dynamisches DNS

Dynamisches DNS - DDNS

Bis jetzt wurden die IP-Adressen und die zugehörigen Systemnamen lediglich von Hand eingetragen, was bei jeder Neuaufnahme Arbeit nötig machte. Dies mag bei 10-20 Systemen noch recht überschaubar sein, jedoch können darüberhinaus doch immer wieder Flüchtigkeitsfehler, wie doppelte IP-Adressen, vergessene Punkte, usw, passieren, die den ganzen Netzwerkbetrieb lahmlegen könnten. Aber keine Angst, auch hier legen Ihnen die Netzwerkdienste einige Handwerksgeräte zur Verfügung. Eines davon ist das dynamische DNS. Dies bedeutet, dass Sie in Zusammenarbeit mit DHCP die IP-Adressen völlig frei zuteilen, diese Adressen automatisch dem DNS-System hinzugefügt werden und somit im Netzwerk bekannt und ansprechbar sind.

Um dies bewerkstelligen zu können, bedarf es jedoch einiger weniger kleiner Änderungen in der Konfigurationsdatei.
Der wichtigste Eintrag sollte bereits vorhanden sein - der Key-Eintrag, also der Schlüssel, der zur Authorisation und Authentifizierung herangezogen wird. Ebenfalls muss den RNDC erklärt werden, wer dem Nameserver Steueranweisungen und wie zukommen lassen darf. Dies geschieht mit dem Signalwort **controls**.

```
# /etc/named.conf

controls {
inet * allow { any; } keys { "rndckey"; };
};

key "rndckey" {
    algorithm hmac-md5;
    secret "reOtiWFGpPN7wzlk5OEE7Q==";
};

oder

include "/etc/rndc.key";
```

Natürlich ist der Schlüssel nur zur Sicherheit gedacht, d.h. es würde auch ohne ihn funktionieren, allerdings sollte ein gewisser Schutz und Sicherheit stets gewahrt bleiben um nicht später als Opfer dazustehen. Die zweite Änderung, die Sie machen müssen ist die allow-update-Option in den Zone-Definitionen.

```
# /etc/named.conf

zone "zoneA.de" in {
type master;
allow-update { key rndckey; };
file "master/db.zoneA";
};
zone "0.1.10.in-addr.arpa" in {
type master;
allow-update { key rndckey; };
file "rev/master/db.10.1.0";
};
```

Mit diesen beiden Optionen dürfen nun nur Systeme Veränderungen vornehmen, die auch im Besitz des Schlüssels sind. Damit wäre die Konfiguration für den Nameserver abgeschlossen. Den Rest muss nun der DHCP-Server erledigen, indem er die nur noch durch die geöffneten Türen treten muss und seine Veränderungen festschreiben. Um zu überprüfen, ob das Zusammenspiel der beiden Server richtig funktioniert, können Sie mit den mitgelieferten Tools **host** oder **dig** sicherstellen, dass die Auflösung richtig ist. Bei der Auflösung müssten nun die neuen IP-Adressen erscheinen.
Bei Bind9 wird im Master- bzw. Reverse-Verzeichnis eine Datei angelegt mit dem Namen der Zonendatei und der Endung .jnl (Journal). Diese ist leider eine Binärdatei (bei Bind8 war es noch ein ASCII-File) und für den Benutzer nicht lesbar, jedoch muss diese Datei vorhanden sein, da hier die neuen DHCP-Adressen den Namen zugeordnet sind und darüber auch abgefragt werden. Angelegt werden diese beiden Dateien beim ersten Aufruf.

Booten und Konfiguration im Netz

Übersicht

Die etwas »schwammige« Formulierung »Booten und Konfiguration im Netz« versucht die nachfolgend vorgestellten - und doch recht unterschiedlichen - Techniken auf einen gemeinsamen Nenner abzubilden.

Genau genommen, umschreibt »Konfiguration einiger Netzwerkparameter während des Netzwerkstarts« trefflich das Ansinnen von *Dynamic Host Configuration Protocol* DHCP oder *Bootstrap Protocol* BOOTP. Aber als Titel schien uns das schlicht weg zu lang.
I.d.R. wird man das Netzwerk im Rahmen des Bootvorgangs aktivieren, weshalb DHCP und BOOTP oft in einem Atemzug mit »Booten« und »Netzkonfiguration« genannt werden. Wichtigstes Ansinnen beider Protokolle ist es, einen Clientrechner mit einer IP-Adresse zu versehen. Ursprünglich entsprang der Bedarf eines solchen Verfahrens dem Einsatz festplattenloser Clients, die ihre IP-Adresse während des Starts nicht kennen können (Wo sollten sie diese auch speichern?). Die heutigen Einsatzfälle sind weit reichender und sollen mit der Vorstellung des jeweiligen Protokolls genannt werden.

Der älteste (verbreitete) Mechanismus zur IP-Adress-Vergabe ist das Reverse Address Resolution Protocol (**RARP**), quasi das »umgekehrte« ARP. Es basiert auf einer Broadcast-Anfrage, wobei der Client in seinem Subnetz die seiner Hardwareadresse zugeordnete IP-Adresse erfragt. Ein RARP-Server wird anhand einer Datenbank mit der gewünschten Information antworten.

RARP ist jedoch zu sehr auf die zugrunde liegende Netzhardware fixiert und auf den bloßen Austausch der IP-Adresse beschränkt. Der Rechnername selbst oder bspw. der Namen eines DNS-Servers kann hierüber nicht in Erfahrung gebracht werden. U.a. der Wunsch nach solchen Anforderungen führte zur Entwicklung des BOOTP, das allgemein als Nachfolger des RARP bezeichnet wird. Der evolutionären Linie folgend, tritt DHCP mit einem nochmals gesteigerten Funktionsumfang das Erbe des BOOTP an.

Bei »Booten übers Netz« handelt es sich da um etwas gänzlich anderes, dient dieses Verfahren doch zum Start von »plattenlosen« Clients (Terminals), die alle notwendigen Daten - einschließlich des Kernels und des Rootdateisystems - von einem Server beziehen. Da einem solchen Vorgehen eine Konfiguration des Netzwerks vorausgehen muss - und DHCP bzw. BOOTP die genutzten Techniken sind - gliedert sich auch dieses Thema unter »Booten und Konfiguration im Netz« ein...

BOOTP - Internet Bootstrap Protocol

Das Bootstrap Protocol dient zur Verteilung von IP-Konfigurationsparametern an einen Rechner. Für gewöhnlich handelt es sich um festplattenlose Clients, die während des Bootvorgangs alle notwendigen Informationen von einem **Bootp-Server** beziehen. Auch wenn das neuere DHCP BOOTP in vielen Bereichen verdrängt hat, gelangt BOOTP zumeist in Zusammenhang mit Booten übers Netz zum Einsatz, da der Kernel BOOTP direkt unterstützt.

Die Funktionsweise ist einfach: Ein konfigurationswilliger Client sendet eine Anforderung (BOOtrEQUEST) als Broadcast in das lokale (Sub)Netz. Ein Bootp-Server antwortet mit einem BOOtrEPLY-Paket, das eventuell mehrere Konfigurationsparameter für das Netzwerk, mindestens aber die IP-Adresse des Clients beinhaltet. Der Bootp-Server muss nicht zwingend im lokalen Subnetz liegen, dann erfordert das Protokoll aber ein **Bootp-Gateway**.

Bootp-Gateways dienen der Weiterleitung von BOOtrEQUESTS. Der zuständige Daemon **bootpgw** wird hierzu mit der Adresse eines Bootp-Servers als Argument gestartet. Dieser Daemon lauscht nun im Netz nach Anfragen von Clients. Trifft eine Bootp-Anfrage ein, wartet das Gateway noch drei Sekunden, ob eine zugehörige Antwort von einem Server eintrifft. Wenn nicht, trägt er in das BOOtrEQUEST-Paket seine Adresse ein und leitet es an den ihm bekannten Bootp-Server weiter. Bootp-Gateways könnten somit die Folgen des Ausfalls eines Servers lindern.

Serverstart

Nur in seltenen Fällen wird ein Bootp-Server entscheidend mehr als einige hundert Clients bedienen. Typisch für einen Rechnerpool sind wohl gar nur 10-50 Clients. Auch fordert ein Client nur ein einziges Mal während seiner Systemlaufzeit eine Netzkonfiguration an, sodass letztendlich eher selten Bootp-Kommunikationen im Netz zu verzeichnen sind. Es bietet sich daher an, den Bootp-Server nur bei Bedarf zu aktivieren und dies ist die Aufgabe von inetd:

```
user@sonne> grep bootp / etc/ inetd.conf
bootps  dgram  udp  wait  root  /usr/sbin/bootpd  /etc/bootptab
```

Wird anstatt des »inetd« der xinetd verwendet, ist die Datei /etc/xinetd.conf der entsprechende Anlaufpunkt:

```
user@sonne> cat / etc/ xinetd.conf
...
bootps
{
        socket_type     = dgram
        wait            = no
        user            = root
        server          = /usr/bin/bootpd
        server_args     = /etc/bootptab
}
...
```

Neben dem »(x)inetd-Modus« lassen sich Bootp-Server und Bootp-Gateway ebenso als Daemon betreiben (»Standalone-Modus«). Die Aufrufe besitzen folgende Syntax:

```
bootpd [ -t Timeout -d Debuglevel -c chdir-path ] [ bootptab [ dumpfile ] ]
bootpgw [ -t Timeout -d Debuglevel ] Bootp-Server
```

Die Kommandozeilenparameter bedeuten (auf einige veraltete Optionen haben wir bewusst verzichtet):

-t Timeout

Zeitspanne (Minuten), nach der sich der Server selbsttätig beendet, falls keine weiteren Bootp-Anfragen eintrafen. Im (x)inet-Modus sind 15 Minuten voreingestellt; im Standalone-Modus arbeiten beide Server unbegrenzt (entspricht 0 Minuten).

-d

Debugmodus (-d, -d1 .., -d4); bestimmt die Flut der Statusmeldungen.

-c chdir-path

bootpd wechselt in das angegebene Arbeitsverzeichnis. Die Option ist nur in Verbindung mit TFTP (Booten übers Netz) sinnvoll, da der Bootp-Server dann für die Überprüfung der Bootdateien der Clients zuständig is Das Verzeichnis sollte dasselbe sein, das der Tftp-Server verwendet.

bootptab

Konfigurationsdatei, aus der der Bootp-Server die Informationen für die Clients bezieht. Voreinstellung ist /etc/bootptab.

dumpfile

Name einer Datei, in die der Bootp-Server bei Empfang des Signals »SIGUSR1« seine interne Datenbank ab

Server

Name des Bootp-Servers, an den ein Bootp-Gateway BOOtrEQUEST-Pakete weiterleitet.

Obwohl die Datei /etc/services in den von Distributionen ausgeliefertem Zustand meist komplett bestückt ist,

sollten Sie sich vergewissern, dass die entsprechenden Ports freigeschalten sind:

```
user@sonne> grep bootp / etc/ services
bootps          67/udp    # Bootstrap Protocol Server
bootps          67/tcp    # Bootstrap Protocol Server
bootpc          68/udp    # Bootstrap Protocol Client
bootpc          68/tcp    # Bootstrap Protocol Client
```

Da alle verbreiteten Bootp-Implementierungen UDP als Transportprotokoll verwenden, genügt die Existenz der ersten Zeile. Die Zeilen 3 und 4 betreffen einen Bootp-Client und sind auf Server-Seite überflüssig.

Die Syntax der Konfigurationsdatei

Typisch erfolgt die Konfiguration in der Datei /etc/bootptab; über die Angabe des Datenbanknamens per Kommandozeilenoption kann die Datei prinzipiell beliebig benannt werden.

Die Fülle der Parameter, die ein Bootp-Server anbieten kann, lassen sich grob in zwei Katagorien einordern. Das sind zum einen Daten, die spezifisch für einen einzelnen Client sind, wie Rechnername, Hardware- und IP-Adresse und es sind Parameter, die für eine Reihe von Clients gleichermaßen gelten (bspw. Domainname, DNS-Server oder Netzwerkmaske). Aus diesem Grund unterstützt das Datenbankformat eine Art »Makro«-Mechanismus, der die Definition von »globalen« Datensätzen erlaubt. Doch zunächst betrachten wir den allgemeinen Aufbau eines Datenbankeintrags:

```
Rechnername:tg[=Wert...]:tg[=Wert...]:tg[=Wert...]
```

Rechnername ist dabei der aktuelle Name des Clients oder ein Dummy-Eintrag. Um einen Dummy-Eintrag von einem gültigen Rechnernamen zu unterscheiden, beginnt der Bezeichner mit einem Punkt. Solch ein Dummy definiert eine Liste von Parametern, die als eine Art Makro in anderen Einträgen eingebunden werden können.

tg steht für ein zweistelliges Symbol, deren wichtigste folgende Liste erläutert:

bf

Name der Bootdatei (siehe Booten übers Netz)

bs

Größe der Bootdatei in 512-Byte-Blöcken (siehe Booten übers Netz)

dn

Domainname

ds

Liste von DNS-Servern

gw

Liste von Gateways

ha

Hardwaredresse des Clients

hd

Verzeichnis, in dem die Bootdatei liegt (siehe Booten übers Netz)

hn

Rechnername, den der Client nachfolgend verwenden soll (überschreibt auf Clientseite den aktuellen Name

ht

Type der Netzhardware, so wie sie im *Assigned Number RFC* spezifiziert ist. Die Angabe kann numerisch od symbolisch erfolgen. Wichtige Symbole sind:

ethernet bzw. ether	10..100 Mb Ethernet
ieee802 bzw. tr bzw. token-ring	IEEE 802 networks
arcnet	ARCNET
ax.25	AX.25 Amateur Radio networks

ip

IP-Adresse des Clients

lp

Liste von Printservern

ms

Weist den Server an, die Antwort-Pakete mit der angegebenen Fragmentgröße zu versenden

rp

Verzeichnis, das als Root zu mounten ist (siehe Booten übers Netz)

sa

Adresse des TFTP-Servers, den der Client verwenden soll (siehe Booten übers Netz)

sm

Subnetzmaske des Netzwerks, in dem der Client sich befindet

T*n*

Eine generischer »Tag«, der Hersteller spezifische Erweiterungen ermöglicht. *n* ist ein numerischer Wert un bezeichnet die konkrete Aktion. Debians Fai macht von einer solchen Eigenschaft regen Gebrauch. Aktuelle Bootp-Implementierungen verfügen über Erweiterungen, die die DHCP-Verbesserungen integrieren. Zustän sind hierfür die Tags T250 (optionale Konfigurationsparameter), T243 (Modus der Adressübernahme in die bootptab) und T253 (Anzahl der dynamisch zuweisbaren Adressen in hexadezimaler Notation). Wir werden ihren Einsatz nicht eingehen.

tc

Eintrag, mit dem auf ein Makro (Dummy-Eintrag) Bezug genommen wird

td

Wurzelverzeichnis auf dem TFTP-Server

ts

Adresse eines Zeitservers

yd

ys

Adresse eines NIS-Servers

Die relative Reihenfolge der Symbole ist unerheblich mit einer Ausnahme, dass **ht** zwingend unmittelbar vor **ha** stehen muss!

Mit Ausnahme des Symbols **ip** darf anstelle der IP-Adresse auch der Name des Servers stehen. Allerdings muss dieser mit den dem Client zur Verfügung stehenden Mitteln in die entsprechende IP-Adresse auflösbar sein. Für einen Client, der sein Rootverzeichnis über das Netzwerk bezieht, könnten die wichtigsten Adressen bspw. in der Datei /etc/hosts aufgeführt sein. Bei der Zuweisung mehrerer IP-Adressen (nicht bei ip, sa, sw, sm und ys) an ein Symbol sind diese durch Leerzeichen voneinander zu trennen. Symbolische Rechnernamen werden dagegen durch Kommata separiert.

Die einzelnen Symbole werden durch den Doppelpunkt voneinander getrennt, tritt ein Sonderzeichen (Doppelpunkt, Komma, Gleichheitszeichen, Leerzeichen) als Bestandteil eines Parameters auf, muss dieser in Anführungszeichen gesetzt werden. Im Falle des Hardwareadresse ist anstatt der Angabe von "7F:F8:10:00:00:AF" auch kurz 7FF8100000AF möglich.

Für einige der Symbole ist die bloße Angabe dieses erlaubt (:tg:) bzw. einzig zulässig. Dies beutet: »Der Wert ist gesetzt«. Ein Beispiel ist :hn:, was bedeutet, dass der Rechnername an den Client zu senden ist.

Im Zusammenhang mit dem Makromechanismus ist das Löschen einzelner Felder nützlich, was durch :tg=@: realisiert wird.

Schließlich lassen sich lange Zeilen aufsplitten, indem an die einzelnen Bestandteile ein Backslash »\« angefügt wird.

Beispiele

Die nachfolgende Konfigurationsdatei definiert einige Parameter für vier Clients. Neben IP-Adresse sollen die Rechner den Domainnamen, das Gateway und die Adressen zweier DNS-Server vom Bootp-Server beziehen. Zur Demonstration wird der letzte Eintrag auf mehrere Zeilen aufgeteilt:

```
user@sonne> cat / etc/ bootptab
erde:dn=galaxis.de:ds=192.168.100.1 211.97.100.1:ht=ethernet:ha=005056AA6464:ip=192.168.100.10:sm=255.255.255.0
venus:dn=galaxis.de:ds=192.168.100.1
211.97.100.1:ht=ethernet:ha="00:52:52:EA:01:64":ip=192.168.100.11:sm=255.255.255.0
mars:dn=galaxis.de:ds=192.168.100.1 211.97.100.1:ht=ethernet:ha=77630A0B0104:ip=192.168.100.12:sm=255.255.255.0

merkur:\
        dn=galaxis.de:\
        ds=192.168.100.1 211.97.100.1:\
        ht=ethernet:\
        ha=205252EA01F4:\
        ip=192.168.100.13:\
        sm=255.255.255.0
```

Verringert wird der Schreibaufwand durch die Zusammenfassung der gemeinsamen Parameter in ein Makro. Eine andere Darstellung derselben Konfiguration wäre:

```
user@sonne> cat / etc/ bootptab
.default:\
        dn=galaxis.de:\
        ds=192.168.100.1 211.97.100.1:\
        sm=255.255.255.0:\
        ht=ethernet

erde:tc=.default:ha=005056AA6464:ip=192.168.100.10
```

```
venus:tc=.default:ha="00:52:52:EA:01:64":ip=192.168.100.11
mars:tc=.default:ha=77630A0B0104:ip=192.168.100.12
merkur:tc=.default:ha=205252EA01F4:ip=192.168.100.13
```

Zum Testen sollten Sie den Bootp-Server zunächst im Standalone-Modus mit maximalem Debuglevel betreiben. Bez. obiger Beispielkonfiguration startet der Server bei korrekter Dateisyntax mit folgenden Statusmeldungen:

```
root@sonne> bootpd -d4
bootpd: info(6):   bootptab mtime: Mon Jun 4 09:54:14 2001
bootpd: info(6):   reading "/etc/bootptab"
bootpd: info(6):   read 5 entries (4 hosts) from "/etc/bootptab"
```

Die weiteren Schritte zur Verifzierung der Konfiguration erfolgen auf Seite der Clients. Im entsprechenden Abschnitt finden Sie weitergehende Informationen. Hinweise zum Auslesen der Hardwareadresse eines Rechners erfahren Sie im folgenden Text zum DHCP-Server.

DHCP

Die Möglichkeiten von BOOTP stießen aus (mind.) zwei Gründen bald auf ihre Grenzen. Den ersten Schwachpunkt offenbarte die zunehmende Verbreitung von tragbaren Computern. »Mal eben schnell« einen solchen in ein bestehendes Netz zu integrieren, scheiterte, weil hierfür die Kenntnis der Hardwareadresse unbedingt erforderlich ist. Problem Nummer 2 erwuchs mit der zunehmenden Vernetzung, wobei frei verfügbare IP-Adressen immer mehr zur Mangelware wurden. Sind mehr Rechner miteinander verbunden, als es IP-Adressen im lokalen Netzwerk gibt, schließt die statische Zuordnung einige Rechner zwangsläufig für immer von der Kommunikation aus. Nun ist es in der Praxis selten der Fall, dass alle Rechner eines Netzwerks gleichzeitig laufen, sodass »meist« genügend Adressen für die aktiven Clients vorhanden sind. Mit der dynamischen Adressverteilung kann somit i.d.R. jeder Rechner mit einer Adresse versehen werden.

Drei Arten der Adresszuteilung

Maximale Flexibilität erlangt DHCP durch drei verschiedene Verfahrensweisen bei der Zuteilung von IP-Adressen an einen Client:

Statische Zuordnung

Die auch als »manuelle Zuweisung« bezeichnete Methode erlaubt die feste Vergabe konkreter IP-Adressen konkrete Clients. So sollte bspw. ein Server immer unter ein und derselben Adresse zu erreichen sein. Das Prinzip ähnelt dem Verfahren des BOOTP.

Automatische Zuordnung

Der Server hält einen Pool von IP-Adressen, aus der er einem Client eine freie Adresse zuweist. Der Client e die Adresse für unbegrenzte Zeit.

Dynamische Zuordnung

Wie »Automatische Zuordnung«, wobei der Client die Adresse nur für einen bestimmten Zeitraum verwende darf. Nach Ablauf der »Lease-Zeit« wird dem Client die Adresse entzogen. Ein Clientrechner kann in einem solchen Fall versuchen, sein Abonnement zu verlängern, wobei weder die Verlängerung noch die erneute Zuordnung derselben IP-Adresse gesichert ist (ein anderer Client hat sich ggf. »vorgedrängelt«).

Der gesteigerte Variation der Kommunikation spiegelt sich ebenso in der Anzahl der Anfragen und Antworten wider, die Client und Server miteinander austauschen (können):

DHCPDISCOVER

Broadcast-Anfrage eines Clients zur Lokalisierung eines DHCP-Servers

DHCPOFFER

DHCPREQUEST

Anfrage eines Clients an einen konkreten Server. Hierunter fallen sowohl die Anfrage nach der Erstaustattu mit IP-Adresse und den weiteren Parametern als auch die Bitte um Verlängerung der Lease-Zeit als auch di Anforderung nach Bestätigung der bisherigen Parameter (u.a. nach einem Reboot des Clients).

DHCPACK

Serverantwort mit IP-Adresse und den geforderten Parametern

DHCPNACK

Ablehnung eines DHCPREQUEST durch den Server

DHCPDECLINE

Ablehnung des Clients, da die IP-Adresse schon verwendet wird

DHCPRELEASE

Ein Client gibt seine Konfiguration frei (dClient an Server, Adresse ist schon benutzt.amit steht die IP zur erneuten Vergabe zur Verfügung)

DHCPINFORM

Clientanfrage nach Parametern ohne IP-Adresse

Konfiguration

Seine Konfiguration liest der Server aus der Datei **/etc/dhcp.conf**. Sie enthält alle Anweisungen über zu bedienende Netzwerke, Rechner und die zu verteilenden Konfigurationsdateien.

```
user@sonne> cat /etc/dhcpd.conf
subnet 192.168.100.0 netmask 255.255.255.0 {
# --- default gateway
      option routers 192.168.100.1;
      option subnet-mask 255.255.255.0;

#      option nis-domain "nisdomain.de";
      option domain-name "galaxis.de";
      option domain-name-servers 192.168.100.100;

      option time-offset 1;
#     option ntp-servers 192.168.1.1;
#      option netbios-name-servers 192.168.1.1;
# --- Selects point-to-point node (default is hybrid). Don't change this unless
# -- you understand Netbios very well
#      option netbios-node-type 2;

#      range dynamic-bootp 192.168.100.10 192.168.100.99;
      range 192.168.100.10 192.168.100.99;
      default-lease-time 21600;
      max-lease-time 43200;

# we want the nameserver to appear at a fixed address
#      host ns {
#        next-server marvin.redhat.com;
#        hardware ethernet 12:34:56:78:AB:CD;
#        fixed-address 207.175.42.254;
#      }
}
```

Auslesen der Hardwareadressen

Für den Fall, dass bestimmte IP-Adressen an konkrete Rechner gebunden werden müssen (bei Servern macht dies sicherlich Sinn), ist das Ermitteln der Hardwareadresse erforderlich. Ein Weg führt über das Kommando **ifconfig**, wozu allerdings der Gang zu jedem in Frage kommenden Rechner (bez. eine entfernte Sitzung) erforderlich wird:

```
user@sonne> /sbin/ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:00:CC:E9:1E:37
          inet addr:192.168.100.11 Bcast:192.168.100.255 Mask:255.255.255.0
          UP brOADCAST NOtrAILERS RUNNING MTU:1500 Metric:1
          RX packets:65 errors:0 dropped:0 overruns:0 frame:0
          TX packets:149 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:6884 (6.7 Kb) TX bytes:22860 (22.3 Kb)
          Interrupt:11 Base address:0xa000
```

Effektiver ist wohl das Auslesen des **arp-Caches**, also des Puffers, der die Hardware-Adressen zu jedem in »letzter Zeit« (die Dauer ist abhängig von der Gültigkeit eines solches Eintrags) kontaktierten Rechner enthält.

```
user@sonne> /sbin/arp -a
erde.galaxis.de (192.168.100.99) at 00:00:1E:D9:4C:A5 [ether] on eth0
mars.galaxis.de (192.168.100.22) at 00:00:3F:D0:4D:D4 [ether] on eth0
```

Um alle derzeit im lokalen Netz befindlichen Rechner zu erfassen, ließe sich ein **ping** in einer Schleife über alle IP-Adressen absetzen.

Eine weitere Möglichkeit bietet das Kommando **tcpdump**, wozu die Netzwerkkarte allerdings im so genannten PROMISC-Modus laufen muss. Folgender Aufruf findet die Hardwareadressen aller aktiven Rechner des lokalen Netzes:

```
root@sonne> tcpdump -qte broadcast and port bootpc
```

Start des Daemons

I.d.R. genügt der bloße Aufruf des Kommandos **dhcpd** zum Start des Servers, womit der Serverprozess sich automatisch in den Hintergrund begibt und alle Parameter in der Voreinstellung übernimmt. Über mehrere Kommandozeilenoptionen lässt sich das Verhalten beeinflussen:

```
dhcpd [ -p port ] [ -f ] [ -d ] [ -q ] [ -cf config-file ]
    [ -lf lease-file ] [ if0 [ ...ifN ] ]
```

-cf config-file

Der Server liest die Konfiguration aus der angegebenen Datei anstatt aus /etc/dhcpd.conf

-d

Die Fehler landen auf der Standardfehlerausgabe anstatt beim syslogd

-f

Der Serverprozess startet im Vordergrund

-lf lease-file

Der Server liest die Datei zur Lease-Zeiten aus der angegeben Datei anstatt aus /var/lib/dhcp/dhcpd.leases. Diese Datei wird vom Server selbst verwaltet und sollte - obwohl sie im ASCII-Format vorliegt - nicht verän werden.

-p port

Der Server verwendet den angegebenen Port anstatt 67

-q

Unterdrückt die Ausgabe einer Copyright-Meldung beim Programmstart (sinnvoll bei Verwendung in Startskripten)

```
root@sonne> dhcpd
Internet Software Consortium DHCP Server 2.0pl3
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.

Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html

Listening on Socket/eth1/172.16.45.0
Sending on  Socket/eth1/172.16.45.0
Listening on Socket/eth0/192.168.100.0
Sending on  Socket/eth0/192.168.100.0
```

Dynamischer DNS-Eintrag von DHCP-zugewiesenen Adressen

Anmerkung: Der folgende Abschnitt basiert auf einer Ausarbeitung von Bernd Reimann.

Langer Titel - kurze Rede.
Wenn Sie in Ihrem Netzwerk mit DHCP arbeiten, ist es leider so, dass keinerlei Namensauflösung funtioniert. Um Ihre Clients mit Namen ansprechen zu können, bedarf es einiges an Handarbeit und auch einiges an konsequenter Update-Arbeit, woran es meist scheitert.

Mit der Kombination von DNS und DHCP gehört dies alles der Vergangenheit an. Diese beiden Server bieten Ihnen die Möglichkeit, die Adressen dynamisch vergeben zu lassen und auch dynamisch in Ihrem Nameserver einzupflegen. Einige wenige Handgriffe und Änderungen in der Datei »dhcp.conf« genügen und schon können Sie sich ruhig zurücklehnen und die Arbeit anderen - nämlich den Servern - überlassen.

```
# /etc/dhcpd.conf

server-identifier ns1.zoneA.de;
authoritative;
# Dies ist die wichtigst Zeile. Sie spezifiziert die Methode, wie Updates
# an den Server übertragen werden.
ddns-update-style interim;
ddns-updates on;

# Dies ist die gleiche Key-Definition, wie sie
# bereits in der /etc/named.conf zu finden ist
key rndckey {
algorithm hmac-md5;
secret "secret_md5_hash";
};

# oder
include "/etc/rndc.key";

# Hier werden die Zonen und die Schlüssel festgelegt
zone zoneA.de. {
primary ns1.zoneA.de;
key rndckey;
}

zone 0.1.10.in-addr.arpa. {
primary ns1.zoneA.de;
key rndckey;
}
```

```
# Hier verbieten wir noch den Clients selbstständig
# Updates an den DNS-Server zu geben, da allein der
# DHCP-Server diese Aufgabe übernehmen soll
deny client-updates;
allow unknown-clients;
```

Sie sehen lediglich die Art der Kommunikaton muss festgelegt werden und schon können die beiden Server kommunizieren.
Genauere Informationen zu den Schlüsseln und DNS finden Sie hier.
Clientseitig müssen nun noch der Eintrag send host-name "mars" in die /etc/dhclient.conf eingetragen werden, damit der DHCP-Server den Hostnamen einer IP-Adresse zuordnen kann und diese beiden beim DNS registrieren kann

```
user@sonne> / sbin/ arp
          Address       HWtype HWaddress         Flags Mask   Iface
          mars.zoneA.de  ether  xx:xx:xx:xx:xx:xx  C           eth0
          ns1.zoneA.de   ether  xx:xx:xx:xx:xx:xx  C           eth0
user@sonne> tail / var/ log/ messages
    ns1 dhcpd: DHCPREQUEST for 10.1.0.208 from xx:xx:xx:xx:xx:xx (mars) via eth0
    ns1 dhcpd: DHCPACK on 10.1.0.208 to xx:xx:xx:xx:xx:xx (mars) via eth0
                       ^ ^ ^ ^  send host-name "mars"
```

Booten übers Netz

Im 2-Jahres-Turnus stellt man mit Bedaueren fest, dass der einst so moderne Computer so gar nicht mehr zu den Reißern zählt; die eine oder andere Software stottert träge vor sich hin; das aktuellste Actionspiel erreicht die Dynamik einer Blitzschach-Partie...

Es wird Zeit für die Aufrüstung. Der Gang zum Händler beginnt mit einer Belehrung des besserwissenden Angestellten, dass unser Mainboard für die neuen Prozessoren nicht geeignet ist. Das Netzteil ist unterdimensioniert und mit dem langsamen Speicher dürfe man das System auf gar keinen Fall ausbremsen! Ach ja, die alte Grafikkarte passt selbstverständlich auch nicht zum performanten Aufrüstpaket, ganz zu schweigen vom Kopfschmerz provozierenden Festfrequenzmonitor.

Als Resultat wird man zum Besitzer eines Zweitrechners, denn für die paar Kröten, die der Händler für das gute Stück noch anrechnen wollte, stellt man ihn dann doch lieber in die Besenkammer...

WWW Server

Übersicht

Apache

Tux

Tux ist ein von RedHat entwickelter, kernel-basierter Webserver. Gerade diese Kernelrealisierung erlaubt dem Server, angeforderte Seiten direkt aus dem Seitencache auf die Netzwerkschnittstelle zu schreiben, ohne die zwischen »Userspace« und »Kernelspace« sonst notwendigen Kopieroperationen tätigen zu müssen (dieses Verfahren ist als »Zero Copy« bekannt). Im Wesentlichen resultiert aus diesem eingesparten Transfer die deutlich erhöhte Geschwindigkeit gegenüber anderen Webservern, mit der Tux die Seiten ins Web entlässt.

Statische Webseiten kann Tux vollständig mit boardeigenen Mitteln behandelt, relativ problematisch ist die Verarbeitung dynamischer Inhalte, die ja bekanntlich mehr und mehr die Webseiten durchdringen. Tux versucht auch solche dynamischen Seiten in seinen Seitencache aufzunehmen. Hierzu erzeugt ein (i.d.R.) im Userspace laufendes Tux-Modul (solche Module liegen nicht jedem Paket bei; dafür eine ausführliche Anleitung, wie sie zu schreiben sind) ein Objekt, dass anstelle der Webseite in den Seitencache platziert wird. Werden dynamische Daten vom Server angefordert, liefert das Objekt eine Mischung aus dynamischen erzeugten Inhalten und bereits vorbereiteten statischen Objekten (nahezu jede dynamische Webseite beinhaltet statische Elemente, bspw. Bilder). Diese statischen Objekte profitieren wiederum vom Zero-Copy-Protokoll, sodass letztlich auch die Auslieferung dynamischer Seiten eine Beschleunigung erfährt.

Weiß Tux einmal nichts mit einer Seite anzufangen - weil bspw. kein entsprechendes Modul existiert - leitet er die Anforderung an einen »normalen« Webserver weiter (bspw. an den Apache).

Installation

Ab Version 2.4 enthalten die Kernel offiziell die Unterstützung für Tux. Die relevanten Optionen finden Sie unter »Netzwerkoptionen« und lauten:

Threaded LinUX application protocol accelarator layer (TUX)

Integriert den TUX-Webserver-Code in den Kernel; die Realisierung als als Modul wird empfohlen

External CGI module

Aktiviert die Unterstützung um CGI-Programme im Kernel zu starten.

extended TUX logging format

Erweitert die Satusausgaben, die TUX an den Syslogd absetzt.

debug TUX

Aktiviert die Ausgabe von Debugmeldungen und dient zur Kontrolle der einzelnen Schritte während der Arb von TUX. Die Option ist nur für TUX-Entwickler sinnvoll.

Bevor Sie an die Kernelerstellung gehen, lohnt ein Blick in /proc/config.gz (existiert nur bei entsprechend konfiguriertem Kernel), ob die notwendigen Fähigkeiten nicht schon im laufenden Kernel enthalten sind:

```
user@sonne> zcat / proc/ config.gz |  grep CONFI G_HTTP
```

Des Weiteren benötigen Sie das Paket Tux; für die aktuellen Distributionen sollten vorkompilierte Versionen verfügbar sein.

Konfiguration

Vermutlich haben Sie bislang einen anderen Webserver eingesetzt, der seine Arbeit an Port 80 verrichtete. Da Tux's Fähigkeiten nicht allen denkbaren Anforderungen genügen, wird im der »alte« Webserver als Assistent zur Seite gestellt. Was Tux nicht kann, deligiert er fortan an den Gehilfen weiter.

Im Falle des Apache müssen Sie den Eintrag »Port 80« aus seiner Konfigurationsdatei »httpd.conf« ändern. Als Portnummer können Sie eine beliebige nicht genutzte Nummer verwenden, also bspw. »Port 8080«. Außerdem empfiehlt es sich, den Eintrag »Bindaddress *« aus selbiger Datei in »Bindaddress 127.0.0.1« zu modifizieren, um direkten Verbindungen am Port 8080 von anderen Rechnern aus vorzubeugen.

RedHat **/ etc/ sysconfig/ tux** SuSE **/ etc/ rc.config.d/ tux.rc.config**

```
user@sonne> cat / etc/ rc.config.d/ tux.rc.config
# /etc/sysconfig/tux

# TUXTHREADS sets the number of kernel threads (and associated daemon
# threads) that will be used. $TUXTHREADS defaults to the number of
# CPUs on the system.
# TUXTHREADS=1

# DOCROOT is the document root; it works the same way as other web
# servers such as apache. /var/www/html/ is the default.
#DOCROOT=/var/www/html/
DOCROOT=/usr/local/httpd/htdocs/

# CGI_UID and CGI_GID are the user and group as which to run CGIs.
# They default to "nobody"
# CGI_UID=nobody
CGI_GID=nogroup

# DAEMON_UID and DAEMON_GID are the user and group as which the daemon runs
# They default to "nobody"
# This does not mean that you should execute untrusted modules -- they
# are opened as user/group root, which means that the _init() function,
# if it exists, is run as root. This feature is only designed to help
# protect from programming mistakes; it is NOT really a security mechanism.
# DAEMON_UID=nobody
DAEMON_GID=nogroup

# CGIs can be started in a chroot environment by default.
# Defaults to $DOCROOT; set CGIROOT=/ if you want CGI programs
# to have access to the whole system.
# CGIROOT=/var/www/html

# each HTTP connection has an individual timer that makes sure
# no connection hangs forever. (due to browser bugs or DoS attacks.)
# MAX_KEEPALIVE_TIMEOUT=30

# TUXMODULES is a list of user-space TUX modules. User-space TUX
# modules are used to serve dynamically-generated data via tux.
# "man 2 tux" for more information
# TUXMODULES="demo.tux demo2.tux demo3.tux demo4.tux"

# MODULEPATH is the path to user-space TUXapi modules
# MODULEPATH="/"
```

Erster Start und Test

Um Problemen auf die Schliche zu kommen, empfiehlt sich in einer zweiten Konsole die Datei /var/log/messages zu überwachen. Meldungen wie

```
root@sonne> tail -f / var/ log/ messages
...
Sep 3 18:29:25 feld kernel: TUX: could not look up documentroot: "/var/www/html/"
...
Sep 3 18:34:45 feld kernel: TUX: error -98 binding socket. This means that probably some other process is (or was a short
```

```
time ago) using addr 00000000, port 80.
Sep 3 18:34:45 feld kernel: TUX: could not start worker thread 0.
```

fokussieren die Fehlerquelle ziemlich exakt. Im ersten Fall versucht Tux von einer nicht existierenden - oder mit falschen Rechten versehenen - Dokumenten-Root zu lesen ("/var/www/html/" sollte bei RedHat-basierten Distributionen der richtige Eintrag sein; bei SuSE ist es "/usr/local/httpd/htdocs/"). Die zweite Meldung resultiert aus einem Programm, das bereits den Port 80 belegt hat. Vermutlich wurde der Apache zuvor gestartet, obwohl Tux zwingend vor diesem den Port belegen muss.

```
user@sonne> lynx -head -dump http:/ / 127.0.0.1/
HTTP/1.1 200 OK
Content-Type: text/html
Date: Mon, 03 Sep 2001 16:36:48 GMT
Server: TUX/2.0 (Linux)
Content-Length: 55
```

Kernel-Httpd

Mail Server

Übersicht

Postfix

Sendmail

Qmail

FTP Server

Übersicht

Der BSD-FTP-Daemon

Der Pure-FTP-Daemon

Der Very Secure FTP Daemon

Allgemeines

Very Secure im Namen des FTP-Servers unterstreicht das Anliegen der Entwickler, bei Design und Implementierung die Sicherheit zum obersten Prinzip zu erheben. Der Einsatz des Servers ist zwar keine Garantie für absolute Sicherheit. Bislang jedoch sind keine gravierenden Sicherheitslücken des *Vsftpd* bekannt geworden.

Das recht einfache Aufsetzen des Servers und nicht zuletzt seine gute Skalierbarkeit und Geschwindigkeit prädistenieren den Server für den Einsatz als FTP-Server in kleinen und mittleren Netzwerken.

Konfiguration

Die allgemeine Konfiguration des Daemons erfolgt in der Datei **/ etc/ vsftpd.conf**. Belange der Sicherheit werden bei Verwendung von Pluggable Authentication Modules in der Datei **/ etc/ pam.d/ vsftpd** eingestellt.

Die Datei /etc/vsftpd.conf

Mit dem Doppelkreuz beginnende Zeilen der Konfigurationsdatei sind Kommentare. Einträge besitzen stets die Form:

```
<Option> = <Wert>
```

Boolsche Optionen kennen den Status *YES* (aktiviert) und *NO* (deaktiviert). Ist eine solche Option nicht explizit aufgeführt, gilt ihr voreingestellter Wert.
Folgende boolsche Optionen (Auswahl) kennt *vsftpd*; die Voreinstellung ist in Klammern angegeben:

anon_mkdir_write_enable (NO)

Gestattet dem anonymen Zugang das Anlegen neuer Verzeichnisse. Hierzu müssen sowohl *write_enable* aktiviert als auch die entsprechenden Rechte im übergeordneten Verzeichnis entsprechend gesetzt sein.

anon_other_write_enable (NO)

Umbenennen und Löschen von Dateien/Verzeichnissen ist auch den anonymen Benutzern möglich.

anon_upload_enable (NO)

anon_world_readable_only (YES)

Anonyme Benutzer dürfen nur Dateien runterladen, auf die »Leserechte für alle« bestehen.

anonymous_enable (NO)

Ist diese Option gesetzt, ist der anonyme FTP-Zugang zugelassen.

chown_uploads (NO)

Hochgeladene Dateien gehören in edr Voreinstellung dem Benutzer, der das Hochladen vornahm. Bei *YES* v
stattdessen der unter *chown_username* angebene Benutzer zum neuen Eigentümer.

chroot_list_enable (NO)

Bei Aktivierung werden lokale Benutzer, die in einer Datei **/ etc/ vsftp.chroot_list** aufgeführt sind, vom
chroot-Wechsel in ihr Heimatverzeichnis ausgenommen (in Verbindung mit der Option **chroot_local_usr**
sinnvoll). Die zu verwendende Konfigurationsdatei kann mittels der Option **chroot_list_file** geändert werd

chroot_local_user (NO)

Steht die Option auf *YES*, landen lokale Benutzer nach der Anmeldung via **chroot** in ihrem Homeverzeichni
der Voreinstellung verfügen die Benutzer den vollen Zugriff auf das Dateisystem analog zur lokalen Anmeld

dirmessage_enable (NO)

Ist die Option aktiv, erhalten die Benutzer beim erstmaligen Wechsel in ein Verzeichnis den Inhalt der
Datei **.message** aus dem Verzeichnis (falls vorhanden) angezeigt. Der Name der anzuzeigenden Datei ist ü
die Option **message_file** konfigurierbar.

guest_enable (NO)

Anonyme Anmeldungen werden bei aktivierter Option auf den in **guest_username** benannten Zugang
gemappt.

listen (NO)

Die Option ist zu Setzen, wenn der FTP-Daemon nicht über einen der Internet-Daemonen inetd oder xinetd
gestartet wird. Erst somit überwacht er selbstständig die Ports auf eintreffende Verbindungen.

local_enable (NO)

Erst bei Aktivierung dürfen sich lokale Benutzer (mit Zugang in der Datei /etc/passwd) via FTP anmelden.

log_ftp_protocol (NO)

Bei aktiver Option werden alle FTP-Anforderungen und -Antworten protokolliert.

port_enable (YES)

text_userdb_names (NO)

Eigentümer/Gruppen werden beim Dateilisting als Namen anstatt als numerische ID's dargestellt.

userlist_deny (YES)

Die Option ist nur bei gesetzem **userlist_enable** relevant. Dann wird lokalen Benutzern das Anmelden nur
ermöglicht, wenn sie explizit in der in **userlist_file** benannten Datei aufgeführt sind.

userlist_enable (NO)

Bei Aktivierung ist das Anmelden nur für die localen Benutzer möglich, die in der in **userlist_file** benannte Datei aufgeführt sind.

write_enable (NO)

Bei Aktivierung sind FTP-Kommandos, die Änderungen am Dateisystem vornehmen, gestattet.

xferlog_enable (NO)

Ermöglicht die detaillierte Protokollierung von Downloads und Uploads. Die Daten landen in der in **xferlog_** benannten Datei (Voreinstellung */var/log/vsftpd.log*).

Numerische Optionen konfigurieren im Wesentlichen das Timeout-Verhalten und die zu verwendenden Ports (Auswahl):

accept_timeout (60)

Nach so vielen Sekunden nach Aufbau der Verbindung wird diese abgebrochen, falls der Client sich noch ni authentifiziert hat.

anon_max_rate (0)

Anzahl Bytes pro Sekunde, mit denen Daten vom/zum anonymen Clients erfolgen. 0 bedeutet keine Einschränkung der Transfergeschwindigkeit.

connect_timeout (60)

Nach dieser Zeitspanne wird die Verbindung zu einem Client gekappt, falls keine Kommunikation stattfand.

data_connection_timeout (300)

Werden laufende Datenübertragungen unterbrochen, wird ein Client nach Ablauf dieser Zeitspanne automa rausgeworfen.

ftp_data_port (20)

Der Datenport für die Datenübertragung.

idle_session_timeout (300)

Nach Ablauf dieser Zeit ohne jeglicher Kommunikation zwischen Server und Client wird die Verbindung zum Client geschlossen.

local_max_rate (0)

Analog zu *annon_max_rate* nur für lokale Benutzer.

max_clients (0)

Maximale Anzahl gleichzeitig akzeptierter Verbindungen. Nur im Stand-alone-Modus relevant. 0 bedeutet »unbegrenzt«.

Des Weiteren existieren eine Fülle von Zeichenkettenoptionen (Auswahl):

anon_root (none)

Bei anonymen Zugang erfolgt eine Wechsel in das angegebene Verzeichnis (via *chroot*).

banner_file (none)

chown_username (root)

Der Eigentümer, dem hochgeladene Dateien anonymer Benutzer zugeordnet werden, insofern auch die Opt *chown_uploads* gesetzt ist.

chroot_list_file (/ etc/ vsftpd.chroot_list)

Existiert die angegebene Datei und sind die Optionen *chroot_list_enable* aktiv bzw. *chroot_local_user* nicht aktiv, so werden die in der Datei benannten lokalen Benutzer via *chroot* bei Anmeldung in ihr Heimatverzeic verbannt.

guest_username (ftp)

Der Benutzername für den anonymen Zugang, falls dieser auf ein spezielles »Gastlogin« gemappt ist. Die Option wird nur betrachtet, wenn *guest_enable* gesetzt ist.

ftp_username (ftp)

Der Benutzername für den anonymen Zugang. Das Heimatverzeichnis ist i.d.R. ein spezielles FTP-Verzeichn das mittels eines *chroot*-Umgebung betreten wird. Der Unterschied zum Gast-Zugang (vergleiche *guest_username*) ist im Wesentlichen, das letzterer nicht zwingend in einer *chroot*-Umgebung gefangen ist.

ftpd_banner (none)

Der Begrüßungstext bei erstmaligem FTP-Zugang. Ist die Option nicht gesetzt, wird ein Vsftpd-eigener Text angezeigt.

listen_address (none)

Bei Rechnern mit mehreren Schnittstellen kann der Vsftpd engeweisen werden, eine andere als die erste Schnittstelle auf einkommende Verbindungen zu überwachen. Einzutragen ist hier die numerische IP-Adress der zu überwachenden Schnittstelle.

local_root (none)

Wenn gesetzt, landen lokale Benutzer nach erfolgreicher Anmeldung in diesem Verzeichnis (sonst in ihrem Heimatverzeichnis).

message_file (.message)

Der Inhalt dieser Datei wird angezeigt, wenn ein Verzeichnis erstmals betreten wird und eine solche Datei c existiert. Des Weiterin muss *dirmessage_enable* gesetzt sein.

pam_service_name (ftp)

Bezeichner, den der Vsftpd bei Verwendung der Pluggable Authentication Modules wählt.

user_config_dir (none)

Ermöglicht eine Benutzer abhängige Konfiguration. Existiert im angegebenen Verzeichnis eine gleichnamige Datei wie ein sich anmeldender Benutzer(name), so gelten für dessen Zugang neben den allgemein gesetzt alle darin aufgeführten Optionen. Die Benutzer spezifischen Optionen überschreiben ggf. die globalen!

userlist_file (/ etc/ vsftpd.user_list)

Siehe *userlist_enable*.

xferlog_file (/ var/ log/ vsftpd.log)

Eine typische Konfigurationsdatei für einen einfachen Server könnte folgende Optionen enthalten:

```
user@sonne> cat / etc/ vsftpd.conf
# Beispielkonfiguration /etc/vsftpd.conf
#
# Anonymes FTP gestatten
anonymous_enable=YES
#
# Lokale Anmeldung gestatten
local_enable=YES
#
# Veränderungen am Dateisystem prinzipiell zulassen
write_enable=YES
#
# Maske für Rechte auf hoch geladene Dateien und neu angelegte Verzeichnisse setzen
local_umask=022
#
# Anonyme Benutzer dürfen nichts am Dateisystem ädern (beide Optionen sind in der Voreinstellung auf NO gesetzt; wir führen sie hier nur zur
Demonstration explizit auf)
anon_upload_enable=NO
anon_mkdir_write_enable=NO
#
# Verzeichnis-Nachrichten aktivieren
dirmessage_enable=YES
#
# Den Datentransfer protokollieren
xferlog_enable=YES
#
# Zur Datenübertragung muss der Port 20 frei geschaltet werden
connect_from_port_20=YES
#
# Die Begrüßungstext
ftpd_banner=Willkommen auf dem Linuxfibel-FTP-Server
#
# Als PAM-Dienst verwenden wir nicht die Standard-FTP-Konfiguration
pam_service_name=vsftpd
```

Die Datei /etc/pam.d/vsftpd

Wir verbleiben an dieser Stelle einzig bei einer kommentierten Beispieldatei. Was es mit dem Verfahren der Pluggable Authentication Modules auf sich hat, erläutert detailliert der gleichnamige Abschnitt im Kapitel Systemadministration→Sichere Systeme.

```
root@sonne> cat / etc/ pam.d/ vsftpd
#%PAM-1.0

#
# Die Anmeldung bedarf der Existenz des PAM-Moduls »pam_listfile.so«
# Die weiteren Einrtäge auf der Zeile steuern das Vorgehen des Moduls:
#    Ist der anmeldende Benutzer (item=user)
#    nicht (sense=deny)
#    in der angegebenen Datei (file=/etc/ftpusers)
#    fahre dennoch mit der Authentifizierung fort (onerr=succeed)
auth     required      pam_listfile.so item=user sense=deny file=/etc/ftpusers onerr=succeed
# Anonymous ftp wird zugelassen
#    Ist die folgende Authentifizierung erfolgreich, werden nachfolgende Regeln nicht mehr beachtet (sufficient)
auth     sufficient    pam_ftp.so
# Für lokales Login werden die weiteren Regeln befolgt
auth     required      pam_unix.so
auth     required      pam_shells.so
account  required      pam_unix.so
password required      pam_unix.so
session  required      pam_unix.so
```

Start des Servers

In den meisten Anwendungsfällen wird ein FTP-Server nur sporadisch benötigt, sodass sich dessen Start erst bei Bedarf, also über einen der Internet-Daemons inetd oder xinetd anbietet.

Start via inetd

```
root@sonne> vi / etc/ inetd.conf
...
# These are standard services.
#
# ftp   stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd
ftp   stream  tcp     nowait  root    /usr/sbin/tcpd  vsftpd
...
```

Der im Beispiel zwischen geschaltete TCP-Wrapper ist nicht zwingend notwendig. Zumindest bei Authentifizierung mittels Pluggable Authentication Modules ist der zusätzlich Gewinn an Sicherheit gleich Null.

Vergessen Sie nach Änderungen in der Konfigurationsdatei nicht, den inetd neu zu starten!

Start via xinetd

```
root@sonne> vi / etc/ xinetd.conf
...
service ftp
  {
  socket_type = stream
  wait        = no
  user        = root
  server      = /usr/sbin/vsftpd
  server_flags = -a
  log_on_success += DURATION
  instance    = 4
  }
```

Nach Modifikation ist der xinetd neu zu starten.

Start als eigenständiger Server

Der *Vsftpd* ist nicht für den Stand-Alone-Betrieb vorgesehen. Versuchten Sie dennoch den Start, ernten Sie eine Abfuhr:

```
root@sonne> vsftpd
500 OOPS: vsftpd: does not run standalone, must be started from inetd
```

Trivial FTP Deamon

Übersicht

News Server

Übersicht

Übersicht

Samba

Übersicht

Sicherheit

Überblick

Ziel des Kapitels

Inhalt des Kapitels

Sicherheit unter Linux

Übersicht

Sie sind der einzige Benutzer in Ihrem System? Ihr System ist in keinem Netzwerk integriert und besitzt auch keinerlei Verbindung zum Internet? Sie arbeiten auch nur unter dem Root-Zugang, wenn es unabdingbar ist?

Dann mögen die hier besprochenen Themen Sie nicht betreffen, aber vielleicht verleitet Sie ja die Neugier, unseren Ausführungen dennoch zu folgen.

Die Sicherheit in einem Linuxsystem ist viel gestaltig. Dies beginnt schon beim Booten des Rechners. Wer ist dazu berechtigt? Wer darf das System herunterfahren? Was wäre, wenn ein Eindringling Ihren Webserver stoppen würde?

Alle Befugnisse einer Person zuzusprechen, ermöglicht dieser die volle Kontrolle über das System. Die Sicherheit des Systems steht und fällt mit den Fähigkeiten des Administrators, Schwachpunkte zu erkennen und Schlupflöcher zu schließen. Die Achillesferse jener Methodik ist in der Stärke des Rootpasswortes verborgen. Gelingt einem Angreifer dieses zu umgehen, hält er alle Instrumente in seiner Hand.

Ein Angreifer kann sich mehrere Techniken bedienen, um seine Befugnisse in einem System zu erweitern. Zu liberale Berechtigungen verhelfen ihm ebenso zur Implementierung trojanischer Pferde, wie die leichtfertige und dennoch verbreitete Aufnahme des aktuellen Verzeichnisses in die Suchpfade für Kommandos.

Vorab aber etwas Entwarnung: Solche Horrorszenarien, wie sie immer wieder von Viren unter einigen populären Systemen herauf beschworen werden, sind unter einem Unixsystem kaum möglich.

Sichere Passwörter

Zunächst sollten Sie sich einmal in die Lage eines Hackers versetzen. *Wer ist er und wie geht er vor?* Sie können davon ausgehen, dass es sich um einen absoluten Spezialisten handelt, der aufmerksam die Hinweise auf neueste Sicherheitslücken von Programmen verfolgt und sie zu seinen Gunsten zu nutzen weiß. Er ist ebenso ein guter Programmierer wie auch ein Administrator. Zum Glück versteht er seine Handlungen zumeist nur als »kleine Fingerübung« und wird nach gelungenem Einbruch keinen wirklichen Schaden anrichten. Aber darauf sollten Sie nicht vertrauen.

Das primäre Ziel des Hackers ist der Zugang zu einem Rechner. Idealerweise handelt es sich um den Rootzugriff. Jedoch sind sich Administratoren der Sicherheitsproblematik bewusst und verwenden daher als sicher (?) einzustufende Passwörter. Aber der »normale« Benutzer? Er ist wenig geneigt, kryptische Buchstabenkombinationen zu pauken, wechselt selten das Passwort und beschränkt sich bei der Wahl auf Begriffe aus seinem sozialen Umfeld, die er, um sie zu verkomplizieren, rückwärts buchstabiert. Für einen Hacker schwerer zu knacken? Mitnichten, wie Sie noch lesen werden.

Eine gänzlich andere Methode ist die Ausnutzung von Programmfehlern. Hierfür benötigt ein Angreifer detailliertes Wissen über die Anordnung der Daten eines Programms im Hauptspeicher und über die Schwachstelle des Programms selbst. Er provoziert bspw. in Programmen, die ihre Eingaben ungenügend auf Korrektheit testen, durch anscheinend unsinnige (aber exakt geplante) Eingaben Speicherüberläufe, sodass der ursprüngliche Speicherbereich des Programms überschrieben wird (*Buffer overflow*). Wenn es dem Angreifer gelingt, den Speicherinhalt gezielt durch eigene Befehle zu ersetzen (z.B. mit dem Aufruf eines eigenen Programms), dann wird der nachfolgend ausgeführte Programmcode mit denselben Rechten ausgestattet, über die das kompromittierte Programm verfügte. Und damit kristallisieren sich die Programme heraus, auf die es ein Angreifer vornehmlich abgesehen hat: auf Programme, die unter privilegierten Rechten laufen (suid-Bit!). Gelingt ihm dies gar bei einem Programm, das mit Rootrechten ausgeführt wird, könnte der eingeschleuste Code bspw. die Passtwortdatei

manipulieren und dem Angreifer vollkommene Macht über den Rechner verschaffen.

Wie viele Programme lagern in Ihrem System? Wie viele davon sind Entwickler-Versionen oder relativ neu? Sie können sich niemals sicher sein, dass die eingesetzte Software keinen Risiken in sich birgt. Für einen Angreifer ist das erste Ziel der Zugang zum Rechner - egal unter welcher Benutzerkennung - um so die Schwächen der Programme für weitere Maßnahmen ausnutzen zu können. D.h. für Sie, als Administrator, dass Sie auch die Passwörter aller Benutzerkennzeichen einer regelmäßigen Kontrolle auf ihre Güte hin unterziehen sollten.

Wie Unix das Passwort ermittelt

Um die Güte abschätzen zu können, muss man die Art und Weiße kennen, wie unter Unix Passwörter erzeugt und gespeichert werden.

Wenn Sie am Login-Prompt Ihr Passwort eintippen, so berechnet eine Funktion *crypt* die verschlüsselte Zeichenkette zum eingegebenen Klartext. Genau diese Zeichenkette vergleicht Unix mit einer in einer Passwortdatei gespeicherten Zeichenkette. Stimmen beide überein, ist das Passwort gültig (auch wenn modernere Verschlüsselungsverfahren eingesetzt werden, ändert das nichts am hier am Beispiel von *crypt* demonstrierten Prinzip).

Es mag verwundern, dass das verschlüsselte Passwort in einer Klartext-Datei abgelegt ist. Allerdings ist es extrem aufwändig, vom verschlüsseltem Text auf das Klartextpasswort zu schließen, während die Berechnung der Gegenrechnung (Klartext ➔ verschlüsselt) in wenigen Schritten möglich ist. Eine Funktion, die diese Eigenschaft besitz, nennt man daher auch **Einwegfunktion**. D.h. die Hinrechnung ist einfach, die Gegenrechnung aber unmöglich (es gibt (noch) keinen effizienten Algorithmus). Im Falle der meisten Unix-Systeme, kommt zum Verschlüsseln der *Data Encryption Standard* (DES) zum Einsatz.

Beispiel: Ich stelle Ihnen die Aufgabe die Primzahlen 113*119 ohne Zuhilfenahme eines Taschenrechners zu ermitteln. Sicherlich kein Problem. Die Gegenrichtung wäre, wenn ich Sie beauftrage, die Primfaktoren der Zahl 13447 zu finden. Für ein Computerprogramm ist das ein Pappenstil, aber Sie werden eine zeitlang darüber brüten. Wählen Sie nun anstatt dreistelliger Primzahlen welche mit 10000 Ziffern, so wird auch ihr Gigahertz-Prozessor ein paar Jahre am Limit werkeln...

Wie sicher ist das Passwort?

Sie könnten jetzt meinen, der DES wäre so ausgeklügelt, dass vom verschlüsselten Text niemals auf den Klartext geschlossen werden könne. Sicher ist die Aussage richtig, wenn Sie mit »darauf schließen« die Anwendung einer Berechnungsvorschrift meinen.

Aber der Hacker geht anders vor. Er nimmt ein elektronisches Wörterbuch, verschlüsselt der Reihe nach jedes enthaltene Wort und vergleicht das Resultat mit dem verschlüsselten Vorbild. Als nächstes kodiert er alle Wörter rückwärts, dann modifiziert er die Klein- und Großschreibung, lässt verschiedene Wörter kombinieren, baut Ziffern und Sonderzeichen ein... Klar, dass er ein Programm verwendet, das all die Modifikationen nach einem bestimmten, der menschlichen Logik nachgeahmten Schema vornimmt. Die heutigen PC's verschlüsseln bis zu mehreren Millionen Wörter binnen einer Minute! Wetten, dass die Trefferquote ganz akzeptabel ist?

Aber ganz so einfach ist es dann doch nicht mehr, da die verschlüsselten Passwörter in aktuellen Distributionen wohl kaum noch in der Datei /etc/passwd zu finden sind, sondern meist die shadow-Suite zum Einsatz gelangt Das Passwort steht nun in der Datei /etc/shadow, die nur noch für Root lesbar ist. Damit ist ein Angreifer am Lesezugriff auf die Shadow-Datei interessiert. Um hier seine Tricks ausspielen zu können, benötigt er Zugang zum System. Womit wir wieder beim Passwortgebahren der »normalen« Benutzer wären...

Der sicherste Weg, schwache Passwörter zu vermeiden, ist, sie von vornherein auszuschließen.

Die Programmpakete zur so genannten proaktiven Passwortprüfung liegen den hier beschriebenen Distributionen nicht bei. Programme wie »anlpasswd«, »npasswd« und »passwd+ « sind ein Ersatz für »passwd« und prüfen das vom Benutzer eingegebene Passwort nach bestimmten Regeln (Telefonnummern, Passwortlänge, Groß- und Kleinschreibung, Wörterbuchlisten...) und lehnen es ggf. ab.

Programme, die die verschlüsselten Passwörter in Passwortdateien verifizieren, finden Sie im Lieferumfang fast

aller Distributionen. **crack** und die zugehörigen Wortlisten sind die traditionellen Bestandteile eines Passwortprüfsystems. Nachfolgend möchten wir auf das moderne (und zumindest der SuSE-Distribution beiliegende) »John the Ripper« eingehen.

John the Ripper

John vermag mit verschiedenen Verschlüsselungsschemen umzugehen, u.a. mit DES, Blowfish, MD5 und WinNT LM Hashes.

Des Weiteren versteht es verschiedene Modi:

- **Wortliste:** Es wird versucht, das Passwort anhand einer Wortliste und optionaler Regeln zu entschlüsseln. Solche Regeln können Buchstabendreher, Kombinationen aus Groß- und Kleinschreibung, zusammen gesetzte Wörter usw. beinhalten. Die maximal zulässige Passwortlänge kann angegeben werden.
- **Einfacher Modus:** John überprüft das Passwort gegen das Nutzerkennzeichen, Informationen des GECOS-Feldes und »verbreitete« Passwörter (aus Datei /var/lib/john/password.lst)
- **Inkrementeller Modus:** Alle erdenklichen Zeichenkombinationen werden getestet. Die Passwortlänge kann wiederum beschränkt werden. Mit dieser Methode kann jedes Passwort entschlüsselt werden - vorausgesetzt, man bringt die nötige Geduld mit;-)
- **Externer Modus:** Hierbei handelt es sich um eine Schnittstelle, die den Einsatz anderer Crackmethoden ermöglicht. Auf diesen Modus soll nicht weiter eingegangen werden.

Einfacher Modus

Zur Demonstration wurden drei Accounts in die Passwortdatei aufgenommen:

- »user1« mit dem passwort »Nobody«
- »user2« mit dem Passwort »2resu«
- »user3« mit dem Passwort »foobar«

john wird im einfachsten Fall mit dem Namen der zu prüfenden Passwortdatei aufgerufen. Im Beispiel schränken wir die Suche auf die oben genannten Benutzerkennzeichen ein.

```
root@sonne> john -users:user1,user2,user3 / etc/ shadow
Loaded 3 passwords with 3 different salts (Standard DES [24/32 4K])
2resu          (user2)
foobar         (user3)[Ctrl]-[C]
Session aborted
```

Binnen Bruchteilen einer Sekunde wurden »foobar« und »2resu« gecrackt. »Nobody« ist mit dieser einfachen Methode nicht beizukommen. Durch Eingabe von [Ctrl][C] wird »john« abgebrochen; den Status der Bearbeitung speichert »john« in einer Datei »john.pot«. Durch Eingabe von:

```
root@sonne> john -restore
Loaded 1 password (Standard DES [24/32 4K])
```

wird an der Abbruchstelle mit der Suche fortgefahren.

Wortlistenmodus

Die Stärke des Modus hängt zum einen von den verwendeten Wortlisten (Textdatei mit einem Wort pro Zeile) und zum anderen von den Regeln der Datei »john.ini« ab. Wir setzen die Wortlistenprüfung auf das Passwort von »user1« an:

```
root@sonne> john --wordfile= / usr/ dict/ words -rules -users:user1 / etc/ shadow
Loaded 1 password (Standard DES [24/32 4K])
Nobody         (user1)
guesses: 1  time: 0:00:00:04 100%  c/s: 15008  trying: Nobleman - Notation
```

Das - zugegeben recht simple - Passwort entdeckte »john« nach 0.04 Sekunden.

Inkrementeller Modus

Die Laufzeit von »john« ist unbegrenzt. Im inkrementellen Modus läuft es, bis das letzte Passwort erkannt oder der Vorgang explizit abgebrochen wurde. Bei Verwendung dieser Prüfart ist es deshalb ratsam, die Suche mit verringerter Priorität zu starten, um die »normale« Arbeit mit dem System nicht auszubremsen. Das nachfolgende Beispiel startet »john« mit einem »Nicelevel« von 20:

```
root@sonne> nice -n 20 john -i /etc/shadow
Loaded 2 passwords with 2 different salts (Standard DES [24/32 4K])
```

Regelmäßige Prüfung

Es spricht nichts dagegen, hin und wieder die Passwortsicherheit im System zu überprüfen. Mit entsprechendem Nicelevel gestartet, stört der Testlauf die tägliche Arbeit nur unmerklich. **john** kann jederzeit abgebrochen (z.B. wenn die Systemlast einen bestimmten Schwellwert übersteigt) und im letzten Zustand gestartet werden. Ein nettes Skript ist das dem Paket beiliegende **mailer** (liegt meist unter /var/lib/john), das automatisch Benutzer mit schwachem Passwort per Mail zum Wechsel auffordert.

Dateischutz

Unter Unix geht der Schutz der Dateien mit den Zugriffsrechten Hand in Hand. Zu liberale Rechte ermöglichen das Lesen von vertraulichen Informationen, das Modifizieren von Daten oder gar das Implementieren von trojanischen Pferden.

SUID - Mit den Rechten des Eigentümers

Programme, die das **suid**-Flag gesetzt haben, waren (und sind) potentielle Sicherheitslöcher. Leider ist es nach wie vor zwingend erforderlich, dem Benutzer Rootrechte beim Start von Programmen wie »passwd« einzuräumen (sonst könnte niemand sein Passwort ändern, weil die Passwortdatei nur für Root schreibbar ist), aber es finden sich in den meisten Installationen derartige Programme, die einfach mit der Standardinstallation auf der Platte landeten, aber niemals benötigt werden.

Als Systemadministrator gilt es, solche Programme aufzufinden und - bei Nichtgebrauch - von der Platte zu entfernen.

Die nachfolgende Anwendung des Kommandos find findet alle Programme, die das suid-Bit (Eigentümer, Gruppe) gesetzt haben:

```
root@sonne> find / -perm +6000
```

Aus der Liste gilt es nun, die Programme heraus zu finden, die nicht benötigt werden. Hier hilft in erster Linie wohl nur ein Schmökern der Literatur, was es mit dem Kommando so auf sich hat. Hat man seine Kandidaten zusammen, sollte man sich vor der Deinstallation noch vergewissern, dass das Kommando nicht von anderen Kommandos/Paketen benötigt wird. Dazu sucht man zunächst das Paket, indem das Programm enthalten ist und fragt nachfolgend die Abhängigkeiten ab (im Beispiel werden RPM-Pakete verwendet):

```
root@sonne> rpm -qf /usr/sbin/suexec
apache-1.3.12-97
root@sonne> rpm -q --whatrequires apache-1.3.12-97
kein Paket verlangt apache-1.3.12
```

Anmerkung: Ist man sich nicht absolut sicher, ob das eine oder andere Programm nicht vielleicht doch benötigt wird, so kann man das »suid«-Flag auch entfernen. Anschließend werden nur noch der Besitzer bzw. die besitzende Gruppe sinnvoll mit dem Kommando arbeiten können.

Die »richtigen Rechte«

Was als kritisch erachtet wird, liegt letztlich im Ermessen des Administrators. Rechte, die leicht zu verifizieren sind, sind:

- Dateien mit Schreibrechten für »Alle«
- Homeverzeichnisse mit Schreibrechten für »Alle«
- Konfigurationsdateien mit Schreibrechten für »Nicht-Root-Gruppen«
- Leserechte auf sicherheitsrelevante Dateien, z.B. Mailboxen, Protokolldateien

Bei neu installierten Paketen kann man davon ausgehen, dass die voreingestellten Rechte in irgend einer Art und Weise sinnvoll gewählt sind. Häufig modifiziert man im Laufe der Zeit die Rechte bestimmter Dateien und noch häufiger entfallen dem Gedächtnis die getroffenen Modifikationen. Zum Glück sind die Rechte in den Paketen selbst gespeichert und lassen sich problemlos rekonstruieren:

```
# Beispiel anhand des Paketes "bash"
root@sonne> rpm --setperms bash
```

Aufspüren modifizierter Dateien

Auf Anhieb fallen einem die zu jeder Datei gespeicherten Modifikationszeiten ein. Doch leider sind diese ebenso vertuschbar wie die Dateigröße. Der einfache Ansatz, das Dateisystem nach Dateien mit diesen Parametern zu durchsuchen, stellt die Integrität keinesfalls sicher (zumindest falls ein Profi-Cracker am Werk war).

Neben Werkzeugen wie Tripwire, Hobglobin, sXid u.a, die leider keiner Distribution beiliegen, kann hier eine Bestandsaufnahme des System Abhilfe schaffen. Da die MD5-Verschlüsselung unterdessen allgemein verfügbar ist, kann zu jeder Datei ein MD5-Fingerabdruck erzeugt werden. Speichert der Administrator Dateiname und Fingerabdruck in einer separaten Datei (außerhalb des Dateisystems!), so kann jederzeit die Unversehrtheit überprüft werden:

```
root@sonne> for i in $(find /  -type f -print): do md5sum $i > > database.txt; done
```

Das obige Schema auf alle Dateien anzuwenden, scheitert wohl an der damit verbundenen Aktualisierung der »Datenbank«. Zumindest verlangt sie eine manuelle Kontrolle, ob die Änderung rechtens war.

Unveränderbare Dateien

Es gibt sicherlich zahlreiche Dateien, deren Inhalte sich niemals oder nur sehr selten ändern. Warum sollte man diese nicht als »nicht änderbar« markieren? Das Dateisystem ext2 hält hierfür das Konzept der Dateiattribute bereit. Informationen zum Setzen und Löschen solcher Attribute findet der interessierte Leser im Abschnitt Zugriffsrechte.

Systemschutz

Was nützt die sicherste Konfiguration, wenn jederman das System starten oder herunterfahren darf? Die sicherste Methode, Gefahren durch Cracker aus dem Weg zu gehen, wäre, den Rechner in einem gesicherten Raum zu installieren und auf jegliche Netzwerkverbindung zu verzichten. Aber in welchen Situation ist das schon realisierbar?

Nehmen wir also für die folgenden Aussagen an, dass der potentielle Übeltäter legitimen Zugang zu Ihrem Rechner erhält. Wie könnte er meine Sicherheitsstrategie hintergehen und wie schütze ich mein System davor?

Zugang zum System

Auch wenn Sie die kuriosesten Passwörter schlechthin verwenden, so nützt es gar nichts, wenn Sie nicht schon Möglichkeiten des Rechners außerhalb ihres Betriebssystems schützen.

Als erstes sollten Sie sich vergewissern, dass tatsächlich nur die vorgesehenen Betriebssysteme auf Ihrem Rechner bootbar sind. Stellen Sie also im BIOS die Bootreihenfolge so ein, dass ein Start von einem Wechselmedium (CDROM, Floppy) ausgeschlossen wird. In dem Zusammenhang sollten Sie das Passwort im BIOS aktivieren. Da dieses keinesfalls sicher ist (manche BIOS-Implementierungen kennen eine Art Super-Passwort), sollte in kritischen Bereichen die Bootfähigkeit solcher Geräte deaktiviert werden. Entweder hängen Sie das CDROM/Floppy an einen Controller, von dem nicht gebootet werden kann oder sie bauen es ganz aus.

Die nächste Option steht Ihnen bei Verwendung von Bootmanagern wie Lilo oder Chos offen. Diese ermöglichen den Zugang zu einzelnen Betriebssystemen über ein Passwort zu schützen.

Haben Sie mehrere Betriebssysteme parallel installiert, so stellen Sie sicher, dass wichtige Partitionen des einen System aus dem anderen heraus nicht sichtbar sind. Solche Partitionen sollten niemals vom »normalen« Anwender gemountet werden können, eventuell kann die Unterstützung nicht benötigter Dateisystemtypen aus der Konfiguration entfernt werden. In dieser Hinsicht sind leider alle Windows-Betriebssysteme eine große Gefahr, da von diesen aus jeder Nutzer mit entsprechenden Programmen (bspw. explore2fs) Zugriff auf Linuxpartitionen erlangen kann!

Herunterfahren des Systems

Das Booten einer Unix-Workstation bleibt wohl immer dem Administrator vorbehalten. Da Linux aber zunächst für die i368 Architektur - also für Heim-PCs und Heimanwender - entwickelt wurde, existiert mit dem »Affengriff« [Ctrl][Alt][Del] eine jedem Benutzer zugängliche Methode, den Rechner zu booten.

Verhindern können Sie dieses, wenn Sie die entsprechende Zeile aus der Datei /etc/inittab entfernen:

```
root@sonne> cat / etc/ inittab
...
# Wirkung des Affengriffs abschalten:
#ca::ctrlaltdel:/sbin/shutdown -r -t 4 now
...
```

Die Pfadvariable

Immer wieder wundern sich Linux-Neulinge, warum die Shell beim Start eines Kommando verärgert reagiert:

```
user@sonne> ls -l doit
-rwxr-xr-x  1   user  users    177 Sep 23 10:52 doit

user@sonne> doit
bash: doid: command not found
```

Der Grund wird sofort klar, wenn man sich den Inhalt der Shellvariablen PATH betrachtet:

```
user@sonne> echo $PATH
/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin: /usr/lib/java/bin:/usr/games/bin:/usr/games: /opt/gnome/bin:/opt/kde/bin:/usr/ope
```

Jeweils durch den Doppelpunkt getrennt, enthält die Variable alle Verzeichnisse, in denen die Shell nach einem Kommando sucht (das Kommando ist in unserem Fall kein Alias, built-in oder Funktion). Das aktuelle Verzeichnis ist nicht enthalten!

Die Reihenfolge der Pfadangaben in PATH entspricht der Suchreihenfolge der Bash. Existiert z.B. ein Kommando »rm« unterhalb von »/usr/local/bin« und in »/bin«, so wird immer ersteres ausgeführt (es sei denn, /bin/rm wird mit vollständigem Pfad angegeben).

Vorsicht mit »./«!

Aus reiner Bequemlichkeit werden die meisten Anwender ihre Pfadvariable dahin gehend anpassen, dass das aktuelle Verzeichnis mit berücksichtigt wird. In der Bash ließe sich das wie folgt realisieren:

```
user@sonne> export PATH= .:$PATH
# oder
user@sonne> export PATH= $PATH:.
```

Gibt es einen funktionellen Unterschied zwischen den beiden Angaben? Ja! Die erste Angabe ist **dringend zu vermeiden**!

Fallstudie

Früher oder später befindet sich jeder Benutzer einmal im Verzeichnis /tmp. Dieses Verzeichnis hat die Eigenschaft, dass jeder seine (temporären) Daten darin ablegen darf. Überlegen Sie sich, was passieren würde wenn sie das Kommando »rm -r .my_temp_dir« aufrufen, jemand aber folgendes Skript unter diesem Namen in »/tmp« abgelegt hat?

```
user@sonne> cat / tmp/ rm
#!/bin/sh

cd $HOME
/bin/rm -rf *
```

Pfutsch sind all die Daten, da das Programm mit **ihren Rechten** ausgestattet wurde!

Die Alternative, den aktuellen Pfad als letzten zu betrachtenden Pfad anzufügen, schützt in den meisten Fällen. Aber gerade wer des öfteren an verschiedenen Rechnern arbeitet, wird automatisch Kommandos aufrufen, die womöglich gar nicht installiert sind. Er tut es, weil er das Kommando von einer anderen Umgebung her gewöhnt ist. Sicher wäre es Zufall, wenn just dieses Kommando sich als »trojanisches Pferd« enttarnen würde, aber es ist genauso Zufall, dass ausgerechnet Ihr Rechner Ziel eines Crackers wird.

Die oben beschriebenen Szenarien sind der Grund, warum zumindest die Pfadvariable des Administrators niemals das aktuelle Verzeichnis einschließt.

Restricted Shells und die Pfadvariable

Haben Sie den Zugang für einen Benutzer auf eine restricted Shell beschränkt, so darf die PATH-Variable für diesen Benutzer **niemals** das aktuelle Verzeichnis enthalten.

Die Problematik mag einem gar nicht bewusst sein, darf der Benutzer doch ohnehin in einer restricted Shell sein Homeverzeichnis nicht wechseln. Verfügt er jedoch über einen Editor (bspw. vi) und befindet sich in seinem Heimatverzeichnis eine ausführbare Datei bzw. darf er »chmod« verwenden, so ist folgendes Szenario realisierbar:

1. Der Benutzer startet den Editor
2. Der Benutzer lädt eine Shell in den Editor (im vi »:r /bin/bash«)
3. Der Benutzer schreibt den Inhalt des Editors in die ausführbare Datei (im vi »:w my_script«)
4. Der Benutzer startet »my_script«

Der Benutzer verfügt nun über eine "normale" Shell!

Ressourcen beschränken

So genannte **Denial of Service**-Angriffe zielen darauf ab, die Hard- und Software eines Rechners auf irgend eine Art und Weise zu beeinträchtigen. Im Netzwerkbereich ist es nach wie vor ein beliebtes Angriffsziel, durch zeitgleiche massive Zugriffe auf irgend welche Server, einen Ausfall dieser zu provozieren. Eine nicht minder schwerwiegende Situation können auch lokale Benutzer herauf beschwören, indem sie - wir müssen ihnen nicht einmal eine Absicht unterstellen - die beschränkten Ressourcen (Rechenleistung, Plattenplatz, geöffnete Dateien, usw.) übermäßig beanspruchen, so dass andere Aufgaben langsamer oder überhaupt nicht mehr erledigt werden.

Je nach Art der Ressourcen lassen sich gezielt Schranken fest setzen, die ein »normaler« Benutzer keinesfalls

übertreten kann. Plattenplatz kann jedem Benutzer mittels den nachfolgend beschriebenen Quotas zugeteilt werden; andere Limits verwaltet das bash-interne Kommando ulimit.

Eine weitere Variante der Ressourcen-Zuteilung wird mit dem Einsatz von PAM ermöglicht.

Quotas

Die Möglichkeiten der Befugnisverteilung sind weitreichender, als es das Benutzer- und Gruppenkonzept mit den Zugriffsrechten vermögen.

Im folgenden Abschnitt lernen Sie kennen, wie sie den Anwendern ein bestimmtes und beschränktes Kontingent an Plattenplatz zur Verfügung stellen. Somit halten Sie die Benutzer an, nicht allzu verschwenderisch Datenschrott in ihren Verzeichnissen zu sammeln. Der Bedarf an Speicherkapazität wird kalkulierbar und ein volles Dateisystem wird nicht so schnell Ihr System in die Knie zwingen. Leider unterstützt das **Reiserfs keine Quotas**.

Quota- Voraussetzungen

Vorraussetzung ist neben dem installierten Paket die Unterstützung von Quotas durch den Kernel:

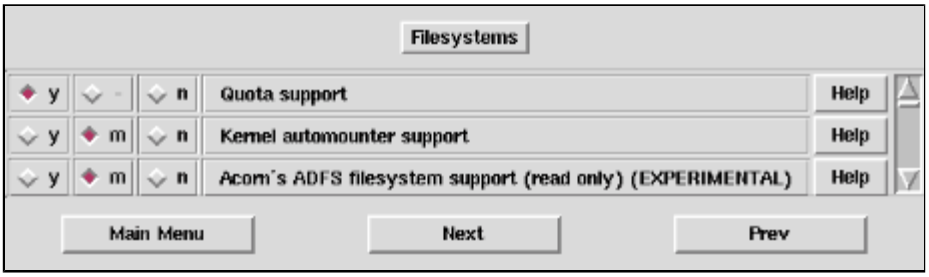


Abbildung 1: Quota-Option im Kernel

Quotas auf einem Dateisystem einrichten

Das weitere Vorgehen werden wir anhand eines Beispiels erläutern. Zwar wird das von uns gewählte Beispielmedium Diskette aus praktischen Gründen niemals Gegenstand von Quota-Vergaben sein; aber als Strickmuster sollte es durchaus dienlich sein. Für die weiteren Schritte gehen wir davon aus, dass ein ext2-Dateisystem auf der Diskette vorhanden ist.

Ein Dateisystem, auf dem Quotas aktiv sein sollen, muss beim Mounten zusätzlich mit den Optionen **usrquota** (Quotas auf Benutzerebene) und **grpquota** (Quotas auf Gruppenebene) versehen werden. Da Quotas sicher erst auf permanent gemounteten Medien einen Sinn ergeben, sollten diese Optionen in der /etc/fstab Aufnahme finden:

```
user@sonne> cat / etc/ fstab
...
/dev/fd0    /floppy    ext2    defaults,usrquota,grpquota   0    0
...
```

Zum Nachvollziehen der folgenden Schritte muss das Dateisystem gemountet sein (ggf. als Root »mount /floppy« aufrufen). Jetzt geht es an das Einrichten der für die Ressourcenverteilung wichtigen Dateien **quota.user** und **quota.group**:

```
root@sonne> touch / floppy/ quota.user
root@sonne> touch / floppy/ quota.group
root@sonne> chmod 600 / floppy/ quota.user
root@sonne> chmod 600 / floppy/ quota.group
```

Root sollte Eigentümer beider Dateien sein und als Einziger Rechte darauf besitzen.

Eine Überprüfung der bisherigen Vorgänge kann zu diesem Zeitpunkt nicht schaden. Hier gelangt **quotacheck** zum Einsatz, das entweder die von Quota betroffenen Dateisysteme aus der Datei /etc/fstab betrachtet oder das auf der Kommandozeile angegebene Device. Ein Testlauf über unser Dateisystem auf der Diskette würde - falls es das einzige Dateisystem mit Quotas ist - folgende Ausgaben hervorrufen:

```
root@sonne> quotacheck -avug
Scanning /dev/fd0 [/floppy] done
Checked 2 directories and 2 files
Using quotafile /floppy/quota.user
Using quotafile /floppy/quota.group
```

Die Optionen bewirken dabei (wenn nicht anderes angegeben, verwenden die nachfolgend aufgführten Kommandos dieselben Optionen mit identischer Bedeutung):

a

Überprüfen der /etc/fstab nach Einträgen usrquota und grpquota

g

Überprüfen der Dateien und Verzeichnisse für eine Gruppe

u

Überprüfen der Dateien und Verzeichnisse für den Benutzer (Voreinstellung)

v

Erweiterte Ausgabe von Informationen (verbose)

Zeigte obiger Kommandoaufruf keinerlei Fehlermeldungen, steht dem Aktivieren der Quotas nichts mehr im Wege:

```
root@sonne> quotaon -avug
  /dev/fd0: group quotas turned on
  /dev/fd0: user quotas turned on
```

Die beiden letzten Schritte sollten zukünftig bei jedem Systemstart ausgeführt werden. Die sauberste Lösung ist, beide Aufrufe in ein Runlevelskript aufzunehmen. Beim Herunterfahren sollten Quotas deaktiviert werden. Der folgende Aufruf findet am besten im selben Runlevelskript seinen Platz:

```
root@sonne> quotaoff -avug
  /dev/fd0: group quotas turned off
  /dev/fd0: user quotas turned off
```

Zuweisung der Schranken

Das An- und Abschalten allein nützt noch gar nichts, solange der Administrator den Plattenplatz noch nicht zugeteilt hat. Er bedient sich hierzu des Kommandos **edquota**. Sollen nun Quotas eines (oder mehrerer) Benutzer editiert werden, so ist die Option **-u** mit anschließendem Benutzernamen (oder Liste der Namen) zu wählen:

```
root@sonne> edquota -u user
Quotas for user user:
  /dev/fd0: blocks in use: 0, limits (soft = 50, hard = 60)
  inodes in use: 0, limits (soft = 0, hard = 0)
```

Wer sich bislang dem Editor **vi** versagte, sollte sich die Grundlagen dessen Bedienung schleunigst aneignen oder zuvor die Shellvariable »EDITOR« mit dem von ihm beherrschten Instrumentarium belegen.

Die Beschränkung des verfügbaren Speicherplatzes geschieht nun auf zwei Arten:

1. Beschränkung des Gesamtspeicherplatzes (blocks; in kByte)
2. Beschränkung der maximalen Anzahl von Dateien (inodes)

D.h. aus Sicht des Benutzers (bzw. einer Gruppe) können Anforderungen für neuen Speicherplatz fehlschlagen, weil entweder zu viele Dateien angelegt wurden oder die Summe der einzelnen Dateigrößen die maximal zulässige Speicherkapazität überschritten hat.

Der Administrator setzt nun die Soft- und Hardlimits, wobei letztere auf keinen Fall überschritten werden dürfen. Dies impliziert, dass die Angabe des Softlimits kleiner oder gleich der des Hardlimits sein muss.

Übersteigt der verbrauchte Speicherplatz eines Benutzers das Softlimit, so darf er eine gewisse Zeit lang über die Ressourcen verfügen, sofern sie noch unterhalb des Hardlimits liegen. Die Zeitspanne legt Root mit der Option **-t** fest:

```
root@sonne> edquota -t
  Time units may be: days, hours, minutes, or seconds
  Grace period before enforcing soft limits for users:
  /dev/fd0: block grace period: 7 days, file grace period: 7 days
```

Im Beispiel wurden 7 Tage fest gelegt; die Zeit kann aber feingranularer definiert werden (mit »seconds« auf Sekundenbasis).

Quotas auf Gruppenbasis werden analog definiert (Option »-g«); wir überlassen die Schritte dem Leser und kommen zum abschließenden Test (die Ausgabe entspricht dem Stand nach dem weiter unten exemplarisch ausgeführten »dd«-Kommando):

```
root@sonne> repquota -a
                    Block limits                File limits
User        used    soft    hard  grace    used  soft  hard  grace
root    --     14       0       0              4     0     0
users   --     60       0       0              2     0     0

                    Block limits                File limits
User        used    soft    hard  grace    used  soft  hard  grace
root    --     18       0       0              4     0     0
user    +-     60      50      60  7days       2     0     0
```

Hinweis: Um ein und dieselben Quota-Einstellungen für mehrerer Benutzer fest zu legen, editiert der Administrator den Eintrag für einen Benutzer und verteilt diesen mit nachfolgendem Kommandoaufruf auf die anderen:

```
root@sonne> edquota -p tux newbie sam
```

Anzeige des aktuellen Status

Jeder Benutzer kann zu jeder Zeit mit Hilfe des Kommando **quota** Auskunft über den ihm verfügbaren Speicherbereich erlangen:

```
user@sonne> quota
Disk quotas for user user (uid 502):
Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
  /dev/fd0       1      50      60               1       0       0
```

Zum Abschluss provozieren wir einen Fehler, indem wir eine »zu große« Datei erzeugen:

```
user@sonne> dd if=/dev/zero of=Datei bs=1k count=60
/floppy: write failed, user disk limit reached.
dd: Datei: Der zugewiesene Plattenplatz (Quota) ist überschritten
59+0 Records ein
```

```
58+0 Records aus
```

Pluggable Authentication Modules

Flexible Benutzerauthentifizierung

Wenn Sie in Ihrem System das Passwortverhalten vom ursprünglichen »Speichere das verschlüsselte Passwort in der passwd« auf die »Verwende die Shadow-Suite« ummünzen wollten, so war es nicht mit dem einfachen Ersatz des Programms »passwd« und Anlegen der Datei »shadow« getan. Nahezu jedes Programm im System, das eine Authentifizierung des Benutzers vorsieht, musste in der neuen Variante installiert werden. Eine Variante, die den Mechanismus der Shadow-Passwörter beherrscht.

Kurzum: Eine Anpassung an einen neuen Authentifizierungsprozess zog weite Kreise, die kaum mehr manuell durchzuführen waren. Zum Glück schnürten die Distributoren ein diesbezügliches Paket, sodass das Einspielen dessen alle notwendigen Änderungen transparent erledigte.

Nun schreitet die Technik immer weiter fort und bisherige Passwort-Authentifizierungs-Modelle sehen sich durch so genannte Brute-Force-Angriffe massiv gefährdet. Schnelle Rechner oder Rechnerverbunde sind durchaus in der Lage, durch systematische Suche im gesamten Schlüsselraum binnen vertretbarer Zeit (Tage) jedes mittels herkömmlichen DES kodierte Passwort zu knacken. Dennoch ist der Faktor Mensch die Achillesferse eines jeden Passworts.

Neue Authentifizierungsverfahren wie Einmail-Passwörter, biometrische Zugangskontrollen oder Chipkarten ersetzen schon heute in sicherheitskritischem Umfeld die früheren Passworte. Selbst kombinierte Verfahren finden durchaus Anwendung.

Es galt nun, Unix für derartige Methoden zugänglich zu machen. Der Ansatz analog zum Shadowpasswort mit seinen angepassten Programmen wäre zwar machbar gewesen, hätte aber dessen Probleme nicht beseitigt. Ein andersartige Authentifizierung hätte zu tiefgreifenden Eingriffen im System geführt. Die Firma SUN entwickelte die **Pluggable Authentication Modules**. Hintergrund ist die strikte Trennung aller authentifizierenden Programme vom eigentlichen Verfahren. Die Umsetzung ist denkbar einfach. Anstatt jedes Programm starr gegen eine Authentifizierungsroutine zu linken, kennt das Programm nur noch die Schnittstellen. Die Authentifizierung wird durch Module einer Bibliothek realisiert. Und welches Modul wann zum Einsatz gelangt, ist hochgradig konfigurierbar. PAM wird vermutlich eines Tages die antiquaren Verfahren in die Geschichtsbücher verbannen.

Die Konfigurationsdateien

SUN's Implementierung konfigurierte PAM in einer einzelnen Datei **/ etc/ pam.conf**. Aufgrund des enormen Umfangs dieser Datei splittete man in Linux diese in einzelne Einheiten auf, die die Authentifizierung für jeweils einen Dienst beschreiben. Diese Dateien tragen den Namen des Dienstes, den sie konfigurieren und finden sich unterhalb des Verzeichnisses **/ etc/ pam.d/** :

```
user@sonne> ls / etc/ pam.d
chfn  chsh  login  other  passwd  ppp  rexec  rlogin  rsh  samba  su
```

other ist kein eigenständiger Dienst, sondern kommt immer dann zum Einsatz, wenn für einen Dienst, der auf die PAM-Bibliotheksfunktionen zurück greift, keine eigene Konfigurationsdatei existiert.

Um kompatibel zu SUN's PAM zu bleiben, kann die Konfiguration unter Linux dennoch in der Datei /etc/pam.conf erfolgen. Diese wird allerdings ignoriert, sobald das Verzeichnis /etc/pam.d existiert.

Eine zu den Dateien in /etc/pam.d äquivalente Datei pam.conf müsste somit alle Zeilen der Dateien aus /etc/pam.d enthalten, wobei jeder Zeile der Namen des Dienstes voransteht. Wir diskutieren nachfolgend die Konfiguration anhand der Datei /etc/pam.d/login. Beachten Sie, dass einige Einstellungen die Vorgaben aus der Datei /etc/login.defs überschreiben.

```
user@sonne> cat / etc/ pam.d/ login
```

```
# Anzuzeigender Text (aus /etc/issue) vor dem Login-Prompt:
auth       required    pam_issue.so issue=/etc/issue

# fontRoot darf sich nur an Konsolen aus /etc/securetty anmelden:
auth       requisite  pam_securetty.so

# Existiert /etc/nologin, darf sich einzig Root anmelden
auth       required  pam_nologin.so

# Setzen von Umgebungsvariablen aus /etc/environment (meist nur bei Debian verwendet)
auth       required  pam_env.so

# Passwort-Abfrage; Null-Passwörter sind erlaubt
auth       required  pam_unix.so nullok

# Ein Benutzer gehört zusätzlich einigen "extra"-Gruppen an (diese Gruppen werden in /etc/security/group.conf fest gelegt)
# auth       optional  pam_group.so

# Zeitgesteuerte Zugangsberechtigungen (definiert in /etc/security/time.conf)
# account    requisite  pam_time.so

# Eingeschränkte Zugangsberechtigungen (definiert in /etc/security/access.conf)
# account    required  pam_access.so

# Standard Un*x Zugangs- und Sitzungsberechtigungen
account    required  pam_unix.so
session    required  pam_unix.so
# Limits (definiert in /etc/security/limits.conf)
# session    required  pam_limits.so

# Informationen zum letzten Login anzeigen
session    optional  pam_lastlog.so

# "Nachricht des Tages" ausgeben (Siehe Datei /etc/motd)
session    optional  pam_motd.so

# Benachrichtigung bei neuer Mail
session    optional  pam_mail.so standard noenv

# Bedingungen an ein Passwort
password   required  pam_unix.so nullok obscure min=4 max=8

# Prüfung neuer Passwörter
#
# password required      pam_cracklib.so retry=3 minlen=6 difok=3
# password required      pam_unix.so use_authtok nullok md5
```

Jede Zeile einer solchen Datei besteht offensichtlich aus 3-4 Feldern. Das erste Feld beschreibt den **Authentifizierungstyp**, beim zweiten handelt es sich um ein **Kontrollflag**. Feld Nr. 3 beinhaltet den **Namen des Moduls** (eventuell mit vollständigem Pfad). Das optionale 4. Feld kann weitere **Argumente** enthalten.

Authentifizierungstyp

Die 4 Typen sind:

account

Steuert den Zugang zu den Diensten, indem die Berechtigungen für einen Benutzer überprüft werd
Weiterhin können zeitabhängige Zugänge ermöglicht und veraltete Passworte erkannt werden.

Anmerkung: Beim »normalen« (Shadow-)Passwortverfahren entspräche der Schritt der Feststellung
Benutzer Zugang zum System hat (also sein Benutzerkennzeichen bekannt ist).

auth

Hierbei wird gesteuert, wie der Benutzer sich dem System gegenüber ausweisen muss (Passwort, Fingerabdruck, Spracherkennung usw.).

password

Diese Module werden sowohl zu Beginn als auch bei Beendigung einer Sitzung aufgerufen. Sie dien Protokollierung der Sitzung und/oder zum Setzen von nutzerspezifischen Systemschranken.

Kontrollflag

Das Verhalten beim Scheitern einer Prüfung wird hier fest gelegt. Die zulässigen Schlüsselworte sind:

optional

Es wird versucht, das Modul auszuführen. Das Ergebnis spielt aber keine Rolle für die Auswertung Module.

required

Die Authentifizierung schlägt fehl, sobald eines der mit »required« gekennzeichneten Module schei weiteren Module desselben Typs werden allerdings dennoch abgearbeitet.

requisite

Wie »required«, nur führt das Scheitern der Authentifizierung in einem Modul zum sofortigen Abbru

sufficient

Es genügt, wenn ein so gekennzeichnetes Modul erfolgreich durchläuft, um die Authentifizierung al erfolgreich zuzulassen. Weitere Module werden bei Erfolg nicht bearbeitet. Bei Nichterfolg eines »s Moduls wird mit dem nächsten Modul fortgefahren, so als wäre das »sufficient«-Modul nicht vorhar gewesen.

Wenn Sie sich unter Unix auf der Konsole anmelden und ein falsches Benutzerkennzeichen angegeben habe so werden Sie dennoch zur Angabe eines Passwortes aufgefordert. Eigentlich scheint dies sinnlos, wird eine nicht-existenten Benutzer doch ohnehin der Zugang verwehrt. Aber was ist das erste Ziel eines Crackers? D Kenntnis der Benutzerkennzeichen auf einem Rechner. Und mit Ausnahme des allgegenwärtigen Root lässt gescheiterter Anmeldevorgang keinen Rückschluss zu, ob ein Benutzerkennzeichen existiert (aus diesem Gr wird in manchen Konfigurationen des kdm auf die Anzeige der Benutzer verzichtet).

Im Zusammenhang mit **PAM** lässt sich das beschriebene Abfrageverhalten realisieren, indem das Benutzerkennzeichen mit »required« versehen wird. Meist wird sogar die Passwortabfrage per »required« vorgenommen, um eine anschließende Protokollierung (»session«) vorzunehmen.

Modul(pfad)

Hier genügt der Name des Moduls, wenn dieses im Verzeichnis /lib/security liegt. Module in anderen Pfader müssen mit vollständigem Zugriffspfad angegeben werden.

Argumente (optional)

Jedes Modul kann zumeist durch zahlreiche Optionen in seiner Arbeitsweise gesteuert werden, d.h. die hier möglichen Argumente sind stark vom Modul selbst abhängig und in der Dokumentation zu diesem beschriel (oft befinden sich die Textdateien unter /usr/[share/]doc/[packages/]/pam). In der obigen Beispieldatei leg bspw. **nullok** des Moduls »pam_unix.so« die Zulässigkeit leerer Passwörter fest. **min= 4** und **max= 8** beschränken die Länge eines Passworts und **obscure** veranlasst einfache Regeln über das Aussehen des Passworts (nicht ausschließlich (Klein)Buchstaben...).

Die Programmierschnittstelle

Jedes Programm, das die Möglichkeiten von PAM zu nutzen gedenkt, muss entsprechende Aufrufe tätigen.

Zunächst muss ein Programm PAM initialisieren. Vor Ende des Programm gehört es zu guten Programmierstil, die reservierten Ressourcen wieder frei zu geben, sodass die PAM-Funktionsaufrufe in folgenden Rahmen eingebettet sind:

```
...
pam_start (...);
# PAM-Funktionsaufrufe u.a.m.
pam_end(...);
...
```

Ein Programm wie **login** wird nun mit dem Aufruf von **pam_authenticate(...)** den Start aller Einträge vom Typ **auth** veranlassen. Die Reihenfolge entspricht den »auth«-Zeilen der Konfigurationsdatei, wobei das Verhalten im Fehlerfall durch die Kontrollflags beeinflusst wird.

Ist »auth« erfolgreich durchgelaufen, wird **login** mit dem Funktionsaufruf **pam_acct_mgmt(...)** fortfahren, womit der Reihe nach die »account«-Einträge abgearbeitet werden.

Die Sitzung schließlich wird **login** mit **pam_open_session(...)** beginnen und mit **pam_close_session(...)** beenden. Jeder Aufruf führt zur Bearbeitung der unter "session" aufgeführten Module.

Mit »password« beginnende Zeilen rufen PAM-basierte Programme auf, die die Authentifzierungs-Token ändern (i.A. ein Passwort). Ein Kommando wie **passwd** beinhaltet zu diesem Zweck den Funktionsaufruf **pam_chauthtok(...)**.

passwd ist somit ein Beispiel dafür, dass ein auf PAM zugreifendes Programm nicht alle Authentifizierungstypen brücksichtigen muss.

Zusätzliche Konfigurationsdateien

Anstatt die zahlreichen Argumente zu einem Modul in jeder Zeile explizit anzugeben, kann für einige Module der Satz voreingestellter Argumente in Konfigurationsdateien niedergeschrieben werden. Diese liegen unter /etc/security:

access.conf

Die (auch leere) Datei wird vom Modul **pam_access** benötigt. Sie ermöglicht eine detaillierte Konfiguration welchem Benutzer an welchem Terminal der Zugang zum System verwehrt/gestattet wird. Eine Konfigurationszeile besteht aus drei, durch Doppelpunkte getrennten Feldern.

Das erste Feld definiert die folgenden Angaben als Verbot (Minus) oder Erlaubnis (Plus).

Das zweite Feld enthält die Liste der Benutzer, die davon betroffen sind. Neben Benutzerkennzeichnen steh **ALL** für alle Benutzer, **LOCAL** für lokale Benutzer und mittels EXCEPT lassen sich Ausnahmen festlegen.

Das dritte Feld umfasst die Quelle des Zugriffs, also den Namen des Terminals, einen Rechnernamen oder -adresse, einen Netzwerknamen, oder -adresse (Domain), sowie **ALL** und **LOCAL** bzw. **EXCEPT**.

Um bspw. die Anmeldung als Root einzig an der lokalen Konsole zuzulassen, ist folgender Eintrag denkbar:

```
...
- : root : ALL EXCEPT console
...
```

group.conf

Diese (auch leere) Datei wird vom Modul **pam_group** benötigt. In Erweiterung der von Gruppen her bekannten Zugriffsrechte vermag ein Benutzer einer Gruppe nur unter bestimmten Voraussetzungen (»zwis 8 und 18 Uhr«, »wenn er sich im Terminal XXX anmeldet«,...) zugeordnet werden.

Die fünf Felder eines Eintrags sind hier durch Semikola getrennt. Das 1. Feld enthöält eine Liste der PAM-Dienste, für die die Zeile in Frage kommt. Der Name des PAM-Dienstes entspricht häufig dem Namen des Programms, kann aber vom Programm selbst »umgebogen« werden, um den Zusammenhang zwischen Konfigurationsdatei und Dienst zu verschleiern.

Das 2. Feld, die Liste der Terminals, verfügt, dass die Gruppenzuordnung nur erfolgt, wenn der Benutzer si am betreffenden Terminal anmeldet.

Feld Nr.3 enthält die Liste der Benutzer.

Zu welchen Zeiten die Gruppenzugehörigkeit gestattet ist, schreibt das 4. Feld vor. Die Syntax lautet "TagStartzeit-Endzeit". "Tag" ist ein zweibuchstabiges Symbol, wobei "Mo", "Tu", "We", "Th", "Fr", "Sa" und "Su" für die einzelnen Tage stehen. "Wk" sind die Wochentage Mo-Fr, "Wd" das Wochenende und "Al" steh "alle Tage". Kombinationen wie "AlMo" oder "WdMo" sind erlaubt. Da im ersteren Fall "Montag" doppelt erfa ist, wird dieser Tag aus der Liste entfernt, während im zweiten Beispiel der Montag hinzuaddiert wird. Die S und Endzeit wird im Format HHMM angegeben (HH-Stunden, MM-Minuten).

Das 5. Feld enthält die Liste der Gruppen, denen ein Benutzer zugeordnet wird, falls alle 4 Bedingungen de vorherigen Felder erfüllt sind.

Das folgende Beispiel ordnet alle Benutzer der Gruppe »games« zu, falls sie sich am Wochenende nach 8.0 Uhr an einem lokalen Terminal anmelden:

```
...
login;tty*&!ttyp*;*;Wd0800-2400;games
...
```

limits.conf

Die vom Modul **pam_limits** benötigte Datei beschränkt die einem Benutzer zur Verfügung stehenden Ressourcen. Die vier Felder werden durch Leerzeichen oder Tabulatoren getrennt.

Das erste Feld legt den/die Benutzer fest, denen die Schranken auferlegt werden. Neben Benutzerkennzeich sind die Angabe von Gruppen (»@games«) und Wildcards (»*«) zulässig.

Im 2. Feld darf **soft**, **hard** oder ein Minus stehen, je nachdem, ob das Limit zum Abbruch einer Anforderun (hard) oder nur zu einer Warnung (soft) führen soll. Ein Minus wirkt wie »hard«.

Feld 3 ist der Name eines Limits und Nr 4. beinhaltet den zugeordneten Wert.

Wichtige Limits sind **data**, **rss**, **stack**, **priority** und **as**, die unmittelbar die Ressourcen beschränken, die e einzelnes Programm beanspruchen darf. Wie viele Dateien ein Benutzer öffnen darf, besagt **nofile** und die Anzahl Prozesse, die er starten darf **nproc**. Verfügbare Prozessorzeit wird in Minuten per **cpu** angegeben. W oft sich ein Benutzer maximal gleichzeitig anmelden darf, beschränkt **maxlogins**.

```
...
*         soft    nproc    20
*         hard    nproc    30
ftp       hard    fsize  1024
@games    hard    cpu      20
@users    -       maxlogins 4
...
```

pam_env.conf

Die Datei wird vom Modul **pam_env** benötigt und ermöglicht den Export von Umgebungsvariablen. Jede Z beginnt mit dem Variablennamen, dem zwei Optionen folgen können. **DEFAULT=** ***Wert*** weist einer Variabl einen default-Wert zu. **OVERRIDE=** ***Wert*** wird verwendet, um diesen voreingestellten Wert zu überschreib Als **Wert** kann auch auf den Inhalt von anderen Variablen zurück gegriffen werden, so setzt die folgende Z DISPLAY auf »0.0«, falls die Variable nicht schon gesetzt war. War sie bereits gesettt, bleibt ihr alter Wert

erhalten:

```
...
DISPLAY     DEFAULT="0.0"    OVERRIDE=${DISPLAY}
...
```

time.conf

Die Datei ist analog zu »group.conf« aufgebaut mit Ausnahme des 5.Feldes, das hier entfällt. Durch diese Konfiguration kann der Zugang zu den PAM-Diensten zeitlich beschränkt werden.

Die Datei wird vom Modul **pam_time** benötigt.

Verschlüsselung

Übersicht

Kryptologie, Kryptografie, Kryptanalyse

In knappe Worte gefasst ist die K r y p t o l o g i e die Wissenschaft von algorithmischen Methoden, die Informationen derart aufbereiten, so dass sie nur für authorisierte Personen verständlich werden. Die K r y p t o g r a f i e als Teilgebiet der Kryptologie umfasst die Verschlüsselung der Informationen während sich die K r y p t (o) a n a l y s e mit dem Brechen der in der Kryptologie angewandten Methoden beschäftigt.

Die Aufgaben der Kryptografie

Im Zuge der Datenübertragung über öffentliche Kanäle kommt der Verschlüsselung der Informationen eine enorme Bedeutung zu. Sind es die Zugangsdaten zum Bankkonto, die vertrauliche Daten über Firmeninterna oder einfach nur die persönliche Email-Korrespondenz, in den meisten Fällen ist die G e h e i m h a l t u n g der Informationen erwünscht.

Dieser Schutz der Daten vor unbefugter Verwendung durch Dritte definiert gleichzeitig zwei weitere Anforderungen an die Kryptografie. Zum Einen muss sie die A u t h e n t i z i t ä t der Nachricht gewährleisten, d.h. sie muss Mechanismen anbieten, um zu überprüfen, ob eine Nachricht auch tatsächlich vom angegebenen Absender stammt. Und sie muss die I n t e g r i t ä t der Daten - also ihre Unverfälschtheit - garantieren.

Digitale Signaturen garantieren die V e r b i n d l i c h k e i t, d.h. einem Absender wird das nachträgliche Leugnen seiner Urheberschaft unmöglich.

Historisches

Die ältesten bekannten Überlieferungen zu Geheimschriften datieren ca. 3000 Jahre zurück; dementsprechend vielgestaltig und umfangreich zeigt sich die Geschichte der Kryptografie (zum Vergleich: Schriften blicken auf 6000 jährige Evolution zurück).
Dies in wenigen Sätzen zu umreißen, ist nur möglich, indem wir einzig die (vermeintlich) markanten Ergeignisse erwähnen und beiläufigen aber keineswegs minder interessante Entwicklungen keinerlei Beachtung schenken.

Dem interessierten Leser möchte ich das Buch »Abenteuer Kryptologie« von Reinhard Wobst empfehlen.

Atbash

Die vielleicht älteste Verschlüsselungstechnik ist unter dem Namen »Atbash« überliefert, welcher jüdischen Ursprungs ist. Es handelt sich um ein einfaches Substitutionsverfahren, wobei als Schlüssel das Alphabet in umgekehrter Reihenfolge dient:

```
Klartextalphabet   : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Geheimtextalphabet : Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
```

Im zu verschlüsselnden Text wird also jedes A durch ein Z, jedes B durch ein Y usw. ersetzt. *WRV ORMFCURYVO DZVI V HXSDVI AF OVHVM ZYVI VRMUZXS AF VMGHXSOFVHHVOM.*

In der Bibel unter Jeremia 25,26 wird die mit Atbash verschlüsselte Stadt Sheshak erwähnt. Dabei handelt es sich um die Stadt Babel.

Cäsar-Verfahren

Julius Cäsar war nachweislich der erste, der seine Nachrichten verschlüsselte. In seinem Briefwechsel mit Cicero verwendete er allerdings ein sehr einfaches Substitutionsverfahren, in dem er jeden Buchstaben des Klartextes um

den im Alphabet drei Stellen folgenden Buchstaben ersetzte:

```
Klartextalphabet   : A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Geheimtextalphabet : D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

Obwohl diese Verfahren mit Atbash verwandt ist, taucht hier etwas Neues auf. Die Sicherheit bei Atbash und anderen alten Geheimschriften beruht darauf, dass das Verschlüsselungsverfahren geheim ist.

Prinzipiell nützt die Kenntnis des Verfahrens bei der Cäsar-Chiffrierung allein nichts. Erst mit dem Schlüssel (hier »Ersetze jeden Buchstaben durch seinen dritten Nachfolger«) kann der Klartext aus dem Geheimtext gewonnen werden.

Simples Durchprobieren der 25 möglichen Schlüssel würde allerdings schnell zum Brechen des Verfahrens führen.

RVAR I NEVNAGR QRF PNRFNE-I RESNUERAF VFG EBG13 (EBGNGVBA HZ 13 CBFVGVBARA). QVR NAJRAQHAT QRF I RESNUERAF NHS QRA TRURVZGRKG SHRUEG JVRQRE LHZ XYNEGRKG.

Das Vigenère-Verfahren

Dieses Verfahren geht auf den französischen Diplomaten *Blaise de Vigenère* (1523 bis 1596) zurück. Hierbei handelt es sich um ein verändertes Cäsar-Verfahren. Beim ursprünglichen Cäsar-Verfahren wird mit einem Geheimtextalphabet gearbeitet, bei dem Vigenère-Verfahren mit mehreren, die wiederholt angewandt werden.

Cäsar verwendete eine einmalige Verschiebung der Buchstaben seiner Klartexte um 3 Positionen. Das Vigenère-Verfahren verwendet hingegen ein Schlüsselwort, das in Wiederholungen unter den Klartext geschrieben wird. Jeder Buchstabe des Klartextes wird nun um die durch den unter ihm stehenden Buchstaben des Schlüssels verschoben. Lautet der Schlüssel bspw. »GEHEIM«, so würde jeder siebte Klartextbuchstabe um denselben Wert rotiert. Die Buchstaben an den Positionen *x+1* (also der 1., 7., 14.,...) würden um 6 Positionen (Position von G im Alphabet) verschoben werden, die an den Positionen *x+2* bzw. *x+4* um 4 Positionen (E) usw.:

```
Klartext     : DASIST DERZUVERSCHLUESSELN DETEXT
Schlüsselwort : GEHEIM GEHEIM GEHEIM GEHEIM GEHEIM G
Geheimtext   : JEZMAFJI YDCHKVZGPXAI ZWMXTHLXMJZ
```

Das Vigenère-Verfahren ist also eine mehrfache Anwendung der Cäsar-Chiffrierung. So verzwickt der Geheimtext auch erscheint, so einfach ist das Verfahren mit Hilfe von Computern zu brechen. Immerhin überstand das Verfahren fast 300 Jahre lang jeder Kryptanalyse (gebrochen 1854 durch *Mr. Charles Babbage*). Für einen erfolgreichen Angriff auf das Verfahren ist einzig die Kenntnis der Sprache des Klartextes notwendig (und ein hinreichend umfangreicher Geheimtext). Zunächst bestimmte Babbage die verwendete Schlüssellänge. Hierzu suchte er im Geheimtext nach sich wiederholenden Zeichenfolgen (mit mindestens 3 Zeichen). Mit der Annahme, dass gleiche Geheimtextzeichenfolgen durch gleiche Klartextpassagen erzeugt wurden (was mit längeren Zeichenfolgen immer wahrscheinlicher wird), beschreibt der Abstand der Zeichenfolgen ein Vielfaches der Schlüssellänge. Aus mehreren solcher Abstände kann letztlich die wahrscheinliche Schlüssellänge berechnet werden. Mit bekannter Länge können die Buchstaben des Geheimtextes, die durch die gleiche Substitution ersetzt wurden, in Mengen eingeteilt werden. Auf diesen Mengen erfolgt letztlich eine Häufigkeitsanalyse der Buchstaben (E ist bspw. im Deutschen der mit Abstand häufigste Vokal).

Enigma

Die Enigma wurde im zweiten Weltkrieg von der deutschen Wehrmacht zur Verschlüsselung ihrer Funksprüche verwendet. Dabei handelte es sich um eine Maschine mit 3 oder 4 Scheiben (Rotoren). Auf beiden Flächen der Scheiben waren je 26 Schleifkontakte angebracht, die die Buchstaben des Alphabets repräsentierten. Jeder Kontakt auf der linken Seite eines Rotors war mit genau einem Kontakt auf der rechten Seite verbunden, wobei die interne Verdrahtung bei den verschiedenen Scheiben differierte. Eine Chiffierung erfolgte durch Anlegen einer Spannung an einem der linken Schleifkontakte. Je nach Stellung der Rotoren zueinander und der Auswahl der Rotoren selbst lag auf der rechten Seite die Spannung an einem anderen Kontakt an. Nach jedem Schritt der Verschlüsselung wurde zunächst der Eingangsrotor um eine Position weiter rotiert. Nach einer vollen Umrundung (26 Schritte) rückte der zweite Rotor eine Position weiter. Vollendete dieser eine Umdrehung, änderte sich die Stellung des dritten Rotors...

Es handelt sich um ein angepasstes Vigenère-Verfahren mit fast unendlich langen Geheimwort (genauer »Anzahl der Buchstaben« hoch »Anzahl der Rotoren«: 17576 bei 3 Rotoren) Das Schlüsselwort entspricht dabei der Anfangsstellung der Rotoren.

Die Enigma galt als sehr sicher. Sie war es aber nicht. Erst 1975 wurde es öffentlich, dass britische und polnischen Mathematiker deutsche Funksprüche im zweiten Weltkrieg entschlüsseln konnten. Dies geschah mit Hilfe einer der ersten Rechenmaschinen, die extra dafür hergestellt wurde.

Weiterentwicklungen der Enigma waren lange nach dem zweiten Weltkrieg noch in Gebrauch und auch heute basiert das crypt(1) Kommando unter einigen UNIX'en auf diesem Algorithmus (Linux *crypt* verwendet das DES-Verfahren). Gerade durch das Aufkommen schneller Computer wurde es notwendig, neue und sicherere Verschlüsselungsverfahren zu finden.

Moderne Verfahren

»Moderne Verfahren« ist natürlich ein dehnbarer Begriff. Die nachfolgend vorgestellten Algorithmen sind in dem Sinne »modern«, als dass zum jetzigen Zeitpunkt kein Programm bekannt ist, das einen Angriff auf ein solches Verfahren ermöglicht. Allerdings steht und fällt die Sicherheit eines solchen Verfahrens oft mit der verwendeten Schlüssellänge, da die Leistung heutiger Rechner(verbunde) den *Brute-Force-Angriffen* in endlicher Zeit zum Erfolg verhelfen.

Symmetrische Verfahren

Die so genannten »symmetrischen Verfahren« verwenden zum Ver- und Entschüsseln jeweils den gleichen Schlüssel. Die Symmetrie bezieht sich i.d.R. allein auf den Schlüssel, da für Chiffrierung und Dechiffrierung bis auf wenige Ausnahem stets verschiedene Verfahren zum Einsatz gelangen (eine Ausnahme ist das oben erähnte ROT13). Unter Linux werden symmetrische Verfahren durch das Kommando *mcrypt(1)* realisiert.

Nachfolgend benennen wir die wichtigsten Vertreter symmetrische Verfahren.

Lucifer

Bei Lucifer handelt es sich um den ersten veröffentlichen Algorithmus (Veröffentlichung 1973 in der Zeitschrift *Scientific American*), der die Blockchiffrierung verwendete. Nach seinem Erfinder *Horst Feistel* (1970, IBM) werden derartige Verfahren auch als *Feistelnetzwerke* bezeichnet.

Die »einfachen« Blockchiffrierungen unterteilen den (bitweisen) Eingabestrom in gleichgroße Blöcke. Jeder Block wird separat mit dem gleichen Verfahren verschlüsselt. In Feistels Verfahren wird nun ein solcher Block in zwei Hälften unterteilt. Jeder Hälfte wird für sich mit einer vom Schlüssel und dem aktuellen Durchlauf abhägigen Funktion verschlüsselt und anschließend mit der jeweils anderen Hälfte verknüpft. Das Verfahren wiederholt sich für den neu entstandenen Block, wobei sich für jeden Durchlauf die Verschlüsselungsfunktion ändert.

Block- Schlüssellänge betragen beim Lucifer-Verfahren jeweils 128 Bit.

DES und DES3

Eine erste Ausschreibung des US-Wirtschaftsministeriums zum Entwurf eines sicheren Verschlüsselungsverfahren brachte zwar enorme Resonanz aber keinen brauchbaren Algorithmus hervor. Erst die wiederholte Ausschreibung im Folgejahr gewann ein Team von IBM mit einer Weiterentwicklung des Lucifer-Verfahrens.

Mit auf 56 Bit reduziertem Schlüssel wurde das Verfahren 1977 vom *U.S. National Bureau of Standards* als *Data Encryption Standard* (DES) genormt. Der Grund der Reduzierung ist bis heute ungeklärt und steter Kritikpunkt an dem Verfahren, da hier der Einfluss und eine Hintertür der NSA (National Security Agency) vermutet wird, die die Entwicklungsunterlagen des Kernstücks von DES lange Zeit unter Verschluss hielt.

1997 fanden 14000 über das Internet verbundene Rechner den Schlüssel zu einer mittels DES verschlüsselten Nachricht mit Hilfe eines Brute-Force-Angriffs. Im Jahr darauf demonstrierte die *Electronic Frontier Foundation* einen 25000$ teure Spezialhardware, die jeden DES-Schlüssel binnen dreier Tage brechen konnte. Spätesten ab diesem Zeitpunkt galt DES als unsicher, wobei die Unsicherheit einzig durch den zu kleinen Schlüsselraum

resultiert. Mit heutige Technik dürfte ein DES-Schlüssel binnen weniger Minuten zu finden sein...

Heute werden in kritischen Bereichen deshalb Weiterentwilckungen von DES eingesetzt. Wichtigster Vertreter ist *TripleDES* (DES3), der den DES-Algorithmus dreifach anwendet mit einem auf 112 Bit erweiteren Schlüssel.

AES

Der *Advanced Encryption Standards* (AES) resultierte aus einer vom *US National Institute for Standards and Technology* (NIST) initiierten Ausschreibung (1977) zur Ablösung von DES als amerikanischen Standard. 2001 wurde das auf einen Kryptologie-Algorithmus von *Rijndael* zurückgehende Verfahren offiziell als Nachfolger von DES bestätigt.

Im Unterschied zu DES unterstützt das neue Verfahren wahlweise Schlüssellängen von 128, 192 oder 256 Bit, so dass Brute-Force-Angriffe auch lange Sicht hin unwahrscheinlich werden. Außerdem unterliegt das Verfahren keinerlei Lizenzbestimmungen und kann von jedem implementiert werden.

Asymmetrische Verfahren

Ein offensichtlicher Nachteil symmetrischer Verfahren besteht im Austausch des Geheimwortes, der oft ungesichert erfolgt und damit eine potentiell Gefahrenquelle darstellt.

Eine Lösung hierfür bieten die so genannten *Public-Key-Verfahren*, die zwei Paare von Schlüssel verwenden. Der eine **öffentliche** Schlüssel dient zur Chiffrierung der Nachricht und nur mit dem zugehörigen **privaten** Schlüssel lässt sich aus der Nachricht wieder der Klartext gewinnen. Alle Verfahren nach diesem Schema werden als **asymmetrisch** bezeichnet.

Das Prinzip der Public-Key-Verfahren beruht auf mathematische Einwegfunktionen. Ein oft bemütes Beispiel ist die Faktorisierung. Angenommen, Sie multiplizierten zwei relativ große Primzahlen miteinander. Die Rechnung dürfte relativ schnell vonstatten gehen. Nähmen Sie jetzt jedoch ein beliebiges Resultat eines Primzahlenmultiplikation her und versuchten ohne Kenntnis der beiden Faktoren dieselbigen zu berechnen, dann müssten Sie trotz Computerhilfe vermutlich recht viel Zeit investieren. Sind die Faktoren nur groß genug, würden Sie an der Faktorisierung letztlich scheitern. Und genau solche Berechnungsvorschriften, wo die »Hinrechnung« einfach, die »Rückrechnung« allerdings schier unmöglich ist, werden zur Schlüsselerzeugung für asymmetrische Verfahren eingesetzt.

RSA

Das von *Rivest*, *Shamir* und *Adleman* entwickelte RSA-Verfahren ist wohl der bekannteste Vertreter asymmetrischer Verfahren und ein Beispiel der Anwendung der Faktorierung zur Erzeugung der Schlüssel.

Vorab wird die Schlüssellänge fest gelegt. Je länger dieser ist, desto sicherer ist das Verfahren. Allerdings wird es auch langsamer, weshalb bspw. die Secure Shell 1024 Bit als Voreinstellung vorschlägt (ssh-keygen). Dieser Wert gilt als sicher und stellt bei heutige Rechengeschwindigkeit noch keine Bremse dar.

Die gewählte Schlüssellänge beeinflusst die Länge der beiden zu erzeugenden Primzahlen p und q. Jede muss mindestens halb so lang wie die Schlüssellänge sein. Die Erzeugung solcher Primzahlen wird per Zufallsgenerator mit anschließendem Primzahltest vorgenommen. Einige Testverfahren erreichen Trefferraten von nahezu 100%, d.h. nur in ganz ganz seltenen Fällen verkennt ein solches Verfahren eine Nicht-Primzahl als Primzahl.

Eine weitere kleine Primzahl > 1, an die als einzige Bedingung gestellt wird, dass sie zu $p-1$ und $q-1$ teilerfremd ist, dient als Exponent e.

Der private Schlüssel d berechnet sich aus

$$de = 1 \bmod (p-1)(q-1)$$

Der öffentliche Schlüssel besteht aus dem Exponenten e und dem Produkt der beiden Primzahlen pq

Zur Verschlüsselung wird der Klartext in Abschnitte unterteilt, die um eins kürzer sind als die Schlüssellänge. Ggf.

muss der letzte Block aufgefüllt werden. Die Bitfolge eines Blocks wird als Zahl *k* interpretiert. Der Geheimtextblock ist der Rest der Teilung von k^e durch *pq*.

Die Rückgewinnung des Klartextes erfolgt über die blockweise Zerlegung des Geheimtextes. Der Klartextblock ist der Rest der Teilung von c^d durch *pq* (*c* bezeichnet den Geheimtextblock)..

Hier lässt sich ein entscheidender Nachteil der Public-Key-Verfahren erahnen. Diese sind im Vergleich zu dem symmetrischen Verfahren sehr langsam. In der Praxis wird daher häfig mit einer Kombination aus beiden Verfahren gearbeitet. Der Sitzungsschlüssel wird hierzu per asymmetrischen Verfahren chiffriert und zwischen den Partnern ausgetauscht. Anschließend erfolgt die Verschlüsselung nach einem symmetrischen Verfahren mit dem zuvor getauschten Schlüssel. Genau dieses Vorgehen realisiert die Secure Shell.

ElGamal

Der nach seinem Erfinder *Taher ElGamal* benannten Algorithmus verwendet ein anderes zahlentheoretisches Problem, die Berechnung des diskreten Logarithmus modulo einer »großen« Primzahl.

ElGamal soll uns hier nicht weiter beschäftigen. In Zusammenhang mit OpenSSL werden Sie mit *Digital Signature Algorithm (DSA)* eine Weiterentwicklung des Verfahrens kennen lernen.

OpenSSL

OpenSSL ist eine freie Implementierung der Protokolle *Secure Socket Layer (SSL)* und *Transport Layer Security (TLS)*. Das Paket umfasst kryptografische Algorithmen zum Erzeugen von Schlüsseln und Zertifikaten sowie zur Verschlüsselung. Sämtliche Eigenschaften sind in einem einzigen Programm **openssl** implementiert. Das Programm wird hierzu i.d.R. mit nachfolgendem Kommando aufgerufen:

```
openssl Kommando [Optionen] [Argumente]
```

Der folgende Abschnitt beschreibt nur die wichtigsten Anwendungsszenarien mit den gebräuchlichen Optionen.

Erzeugen von Schüsseln

OpenSSl beherrscht die Erzeugung von *RSA*-, *DH*- und *DSA*-Schlüsseln.

RSA-Schlüssel

Erzeugung: Hierzu müssen Sie das Kommando **genrsa** bemühen. Wenn Sie das so einfach versuchen, wird der erzeugte Schlüssel (mit einer Schlüssellänge von 512 Bit) auf der Konsole ausgegeben. Allerdings sollten Sie es vielleicht doch nicht so einfach machen...

...denn nach obigen Schema erscheint der Schlüssel unverschlüsselt. D.h. jeder, der ihn in die Finger bekäme, könnte ihn auch verwenden. Besser also, Sie verschlüsseln den Schlüssel, indem Sie die Option **-des3** verwenden. Sie werden nun nach einer Passphrase gefragt, die zur Chiffrierung mit dem TripleDES-Verfahren verwendet wird.

Des Weiteren gilt eine Schlüssellänge von 512 Bit nicht mehr als zeitgemäß. Zwar wird kaum eine Privatperson über ausreichend Rechenkapazität verfügen, um einen solchen Schlüssen anzugreifen, aber für Rechnerverbünde oder Großrechner liegt das wohl im Bereich des Möglichen. Geben Sie einfach einen höheren Wert am Ende der Optionsliste an und Sie brauchen sich über die Sicherheit fortan keine Gedanken mehr zu machen.

Mit der Option **-out** leiten wir die Ausgabe in eine Datei um, so dass sich letztlich folgender Aufruf ergibt:

```
user@sonne> openssl genrsa -des3 -out my_rsa_key.pem 1024
Generating RSA private key, 1024 bit long modulus
.............++++++
.....................++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
```

```
Verifying password - Enter PEM pass phrase:
```

OpenSSL arbeitet intern mit einem Zufallsgenerator, der bei jedem Aufruf initialisiert wird. Wie jeder Zufallsgenerator arbeitet auch dieser nicht wirklich zufällig sondern könnte unter Umständen identische Zahlenfolgen liefern. Misstrauische Naturen sollten daher selbst die Initialisierung in die Hand nehmen. Hierzu leiten Sie die Ausgabe aus /dev/random in eine Datei um:

```
user@sonne> cat / dev/ random > random.bin
```

Klimbern Sie heftig auf der Tastatur herum und wackeln Sie an der Maus. Wenn Sie nach einigen Sekunden mittels [Ctrl]-[C] die Generierung abbrechen, wird das Ergebnis wirklich zufällig sein. Die entstandene Datei können Sie mit der Option *-rand random.bin* bei der Schlüsselerzeugung verwenden.

Einen RSA-Schlüssel erkennen Sie am typischen Beginn:

```
user@sonne> head -3 my_rsa_key.pem
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,8E68A1647C316139
```

Überprüfung des Schlüssels: Hierzu verwenden Sie das Kommando **rsa** mit der Option **-check**. Die Eingabedatei geben Sie mittels **-in *RSA-Datei*** an, **-noout** unterdrückt die Ausgabe des Schlüssels:

```
user@sonne> openssl rsa -check -noout -in my_rsa_key.pem
read RSA key
Enter PEM pass phrase:
RSA key ok
```

DSA-Schlüssel

Erzeugung: Ein Schlüssel nach dem *Digital Signature Algorithm* wird in zwei Schritten erzeugt. Zunächst benötigen wir die passenden Parameter. Da deren Erzeugung ein extrem rechenintensiver Vorgang ist und die Parameter oft mehrfach verwendet werden, werden sie separat erzeugt. Wiederum können Sie die Werte für die Schlüssellänge und die Ausgabedatei angeben:

```
# Die Ausgaben des Kommandos sind stark verkürzt wiedergegeben
user@sonne> openssl dsaparam -out mydsaparam.pem 2048
Generating DSA parameters, 2048 bit long prime
This could take some time
.................+++++++++++++++++++++++++++*
.....+.....................+.........+.......+......+.....+....+..
+++++++++++++++++++*
```

Die Erzeugung des eigentlichen Schlüssels mit dem Kommando **gendsa** geht dann relativ flott vonstatten. Geben Sie mindestens den Namen der zuvor erzeugten Parameterdatei an:

```
user@sonne> openssl gendsa -out myprivkey_dsa.pem mydsaparam.pem
Generating DSA key, 2048 bits
```

Notorische Sicherheitsfanatiker können wieder das oben (RSA) erwähnte Verfahren der Zufallszahlengenerierung einsetzen.

GNU Privacy Guard

Einleitung

GnuPG ist eine freie Implementierung der *OpenPGP-Spezifikation* (RFC 2440) und vollständig kompatibel zu *PGP*

5.x (Privat Good Privacy) der Firma *NAI*. GnuPG ist ein Programm zum Verschlüsseln und Signieren digitaler Daten, wobei es sowohl asymmetrische (Public-Key) als auch symmetrische Verfahren unterstützt.

Das Verschlüsseln und Signieren erfolgt dennoch fast ausschließlich mittels Public-Key-Verfahren (Signieren sogar immer). Eine Nachricht verschlüsselt der Sender mit dem öffenlichen Schlüssel des Empfängers und einzig der Empfänger vermag die Nachricht mit Hilfe seines privaten Schlüssels in Klartext zu verwandeln. Das Signieren von Nachrichten verläuft genau anders herum. Der Unterzeichner verwendet hierzu seinen privaten Schlüssel und ein Empfänger kann mittels des öffenlichen Schlüssels des Absenders die Echtheit der Signatur überprüfen.

Vorbereitungen

Symmetrisches Verschlüsseln

Erzeugung und Schlüsseln

Textkonsole

GUI: gpa (GNU Privacy Assistant)

Der TCP-Wrapper

Übersicht

»Vertrauen ist gut - Kontrolle ist besser«. Jeder Netzwerker dürfte mittlerweile genügend Beispiele kennen, wie Dienste ausgenutzt, Sicherheitsbarrieren unterlaufen und Schäden durch böswillge Zugriffe von »außen« verursacht wurden. Wenn hoffentlich nicht am eigenen Netzwerk, so doch zumindest durch Artikel und Mundpropaganda.

Ein einfach zu verwendende und dennoch recht effektive Vorsichtsmaßnahme gegen unberechtigten Zugriff auf lokale Dienste bietet der TCP-Wrapper, der mittlerweile etliche Internetdienste von der **Außenwelt** abschirmen sollte.

Die Arbeitsweise des TCP-Wrappers

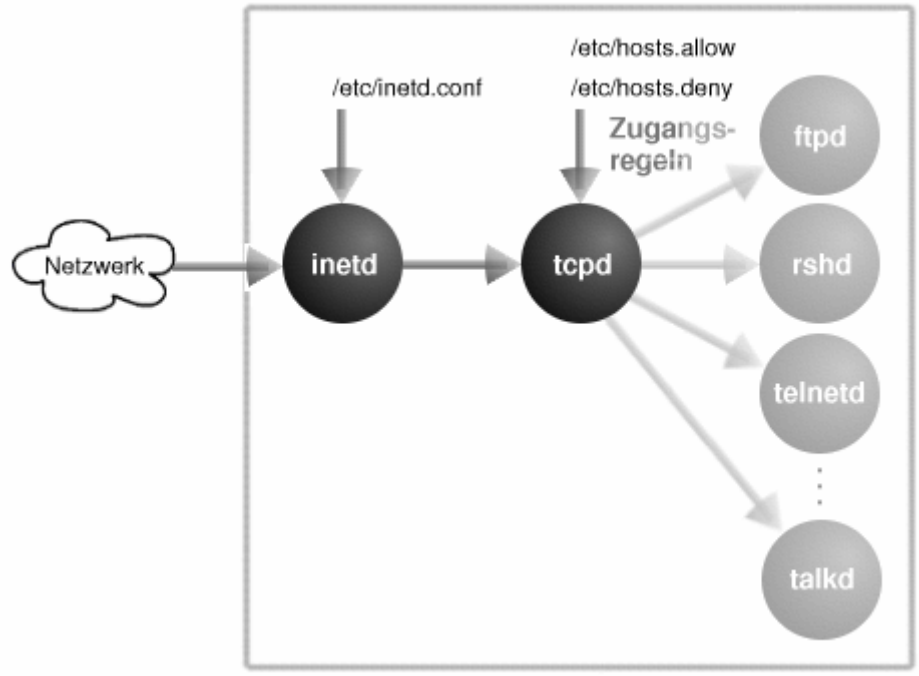


Abbildung 1: Arbeitsweise des TCP-Wrappers

Die meisten Netzwerkdienste, die der lokale Rechner anbietet, werden nicht unmittelbar mit den Start des Systems ins Leben berufen, sondern erst, wenn tatsächlich eine Anforderung für sie eintrifft.

Damit kein Verbindungswunsch eines Clients verschlafen wird, überwacht ein Internet-Dämon, meist der **inetd**, stellvertretend alle Ports der Dienste. Welche das konkret sind, erfährt der **inetd** aus den Dateien /etc/inetd.conf und /etc/services.

Der **inetd** vermag nun den jeweiligen Dienst zu starten, sobald eine Anforderung an dessen Port anliegt und für einige Dienste, die zumeist als »sicher« gelten, wird dies auch so gehandhabt. Sobald der entsprechende Service aktiv ist, und der **inetd** ihm die bereits geöffnete Verbindung vererbt hat, legt der **inetd** sich zur Ruhe und wartet auf neue Arbeit...

Im **inetd** selbst sind keinerlei Sicherheitsmechanismen implementiert. Deswegen wurde ein Filter geschaffen, der sich zwischen den **inetd** und den zu startenden Dämonen spannt. Der **inetd** startet nun nicht mehr den für den Port zuständigen Dienst, sondern den so genannten **TCP-Wrapper** »**tcpd**«, der den Programmnamen des Dienstes als Argument erhält. Der Name TCP-Wrapper ist etwas irreführend, da der Wrapper ebenso UDP-Ports

überwachen kann.

Der Wrapper arbeitet für Server und Client transparent, er verschwindet aus dem Hauptspeicher, sobald die Verbindung zwischen den beiden besteht. Zuvor jedoch protokolliert (syslogd) und überprüft er die Zulässigkeit des Zugriffs und betrachtet zunächst den Inhalt der Datei **/ etc/ hosts.allow**. Hier steht, welcher Dienst für welchen Rechner gestattet ist. Findet er dort eine den Dienst betreffende Zeile, kann er entscheiden, ob der Start des Serverdienstes erlaubt ist oder nicht.

Fehlt ein passender Eintrag oder existiert die Datei nicht, schaut der Wrapper als nächstes nach einer Datei **/ etc/ hosts.deny**. Ist dort ein entsprechender Eintrag vorhanden, verfährt der Wrapper analog zur /etc/hosts.allow, nur dass, wenn der zugreifende Rechner hier genannt wird, er den Serverstart verweigert.

Ist kein Eintrag zum Dienst vorhanden, gilt der Zugriff als zulässig, der Wrapper wird den Dämon starten und sich selbst beenden. Den Aufbau der beiden Dateien /etc/hosts.allow und /etc/hosts.deny schauen wir uns weiter unten an.

Das Problem der resistenten Server

Ein einmal aktivierter Serverprozess kann durchaus so implementiert sein, dass er von sich aus weitere Verbindungen annimmt. Solange er seine Ports nun selbst im Auge behält, bleiben der **inetd** und damit auch der TCP-Wrapper außen vor, wenn es um die Annahme neuer Client-Wünsche geht.

Auch schließen manche Server aus Effzienzgründen ihre Ports nicht unverzüglich, sondern lauschen bis zu einer Zeitüberschreitung weiter. Kommt in dem Augenblick eine Anforderung herein geschneit, wird sie ebenso akzeptiert werden.

Und Dienste, die den **inetd** umgehen (z.B. RPC-Dienste), lassen sich niemals durch den TCP-Wrapper überwachen.

Die Dateien / etc/ hosts.allow und / etc/ hosts.deny

Der Aufbau beider Dateien ist identisch. Bevor wir uns diesem zuwenden, soll nochmals die **Reihenfolge der Bearbeitung** verdeutlicht werden:

1. Existiert in der /etc/hosts.allow das gesuchte Server-Client-Paar, wird der Zugriff sofort gewährt
2. Existiert in der /etc/hosts.deny das gesuchte Server-Client-Paar, dann wird der Verbindungswunsch sofort abgeleht
3. In allen anderen Fällen ist der Zugriff erlaubt

Server ist dabei der Dienst, den der Client in Anspruch zu nehmen wünscht und **Client** ist der Rechner, von dem die Anforderung ausging.

Eine Zeile in den beiden Dateien hat nun folgenden Aufbau:

```
Dienst : Rechner : [Kommando]
```

An Stelle von **Dienst** kann entweder der **Programmname** dessen stehen oder das Schlüsselwort **ALL**, falls die Zeile alle Dienste betreffen soll. Einem **ALL** kann ein **EXCEPT Dienstname** folgen, dann sind alle Dienste mit Ausnahme der benannten gemeint. Per Komma getrennt, lassen sich mehrere Dienste angeben.

Die möglichen Einträge sind Rechnernamen, IP-Adressen oder die folgenden Schlüsselworte:

ALL

Alle Rechner

KNOWN

LOCAL

Alle Rechner, deren Namen keinen Punkt enthalten (Rechner, die in der /etc/hosts unter einem Kurznamen aufgeführt sind).

UNKOWN

Rechner, deren Namen der **tcpd** nicht ermitteln kann.

PARANOI D

Alle Rechner, deren Namens- und Adressauflösung über DNS widersprüchliche Angaben ergibt (so etwas ist bei Rechnern mit mehreren Netzwerkkarten möglich, da diese auch mehrere IP-Adressen besitzen).

Schließlich kann ein optionales **Kommando** angegeben werden, dass immer dann ausgeführt wird, wenn diese Zeile für eine Anforderung zutrifft. Falls auf diese Möglichkeit zurückgegriffen wird, dann meist zum Zwecke des detaillierten Protokollierens. Im Argument des Kommandos können einige Sonderzeichen verwendet werden, die folgende Bedeutung haben:

% a

Liefert die IP des rufenden Rechners

% c

Gibt den Namen des Nutzers und Rechners zurück, sofern die Informationen vom entfernten Rechner geho werden können

% d

Name des gewünschten Dienstes

% h

Name des zugreifenden Rechners oder dessen IP

% n

Name des zugreifenden Rechners (oder unknown oder paranoia)

% p

Prozessnummer des Dienstes

% s

Name des Serves in Verbindung mit dem Rechnernamen

% u

Name des Nutzers auf Clientseite, sofern er ermittelt werden kann

Konkrete Beispiele sollen den Aufbau der Dateien /ect/hosts.allow und /etc/hosts.deny und den Gebrauch der Schlüsselwörter verdeutlichen:

```
user@sonne> cat / etc/ hosts.allow
# Mail ist jedem gestattet
#
in.smtpd: ALL
```

```
# Telnet und FTP wird nur Hosts derselben Domain und dem Rechner "melmac" erlaubt.
#
in.telnetd, in.ftpd: LOCAL, melmac.outside.all

# Finger ist jedem erlaubt, aber root wird per Mail darüber informiert
#
ALL: ALL: spawn (/usr/sbin/safe_finger -l @%h | mail -s "finger from %h" root)

user@sonne> cat / etc/ hosts.deny
ALL: ALL
```

Erklärung: In der letzten Zeile der Beispieldatei /etc/hosts.allow wird »finger« gestattet, wobei gleichzeitig versucht wird, Information vom rufenden Rechner zu gewinnen. Nun könnte es durchaus sein, dass der Gegenüber seinen »finger«-Zugang mit eben diesem Mechanismus umgeben hat. Die Folge wäre eine Endlosschleife sich gegenseitig initiierender finger-Aufrufe. Um so etwas auszuschließen, existiert das Kommando »safe_finger«, das den einmaligen Aufruf sicher stellt. »spawn« erzeugt einen neuen Kindprozess, der die nachfolgenden Kommandos ausführt.

Überprüfen der Konfiguration

Erstes Hilfsmittel ist das Kommando **tcpdchk**. Dieses Programm findet Unstimmigkeiten in der Datei /etc/inetd.conf, wie z.B. fehlende Netzwerkdienste. Des Weiteren werden Syntaxfehler der Dateien /etc/hosts.allow und /etc/hosts.deny und unbekannte Rechnernamen aufgespürt. Zur Identifizierung von logischen Fehlern hilft das Kommando allerdings nicht.

Jetzt kommt **tcpdmatch** zum Zuge. Das Kommando erwartet als Argumente den zu verifizierenden Dienst (eventuell in der Form »Dienst@Server«) und den Nutzer oder Host (»user@host« bzw. nur »host«). Anschließend berichtet das Kommando, wie der **tcpd** die Anforderung erfüllen würde:

```
# Es ist unwahrscheinlich, dass folgender Rechner existiert:
user@sonne> / usr/ sbin/ tcpdmatch in.telnetd melmac.outside.all
warning: melmac.outside.all: host not found

user@sonne> / usr/ sbin/ tcpdmatch in.fingerd sonne
warning: sonne: hostname alias
warning: (official name: sonne.galaxis.de)
client:   hostname sonne.galaxis.de
client:   address  194.180.239.232
server:   process  in.fingerd
matched:  /etc/hosts.allow line 11
access:   granted

user@sonne> / usr/ sbin/ tcpdmatch in.rshd www.gnu.org
client:   hostname gnudist.gnu.org
client:   address  198.186.203.18
server:   process  in.rshd
matched:  /etc/hosts.deny line 1
access:   denied
```

Übersicht

Firewall

Übersicht

Anhang

Übersicht
Inhalt des Anhangs

Übersicht

Der Anhang umfasst neben Verweisen auf Webseiten mit linuxrelevantem Inhalten und einem Glossar weitere Informationen, die wir in die Struktur der Linuxfibel nicht recht unterbringen konnten.

So haben wir einige Beschränkungen rund um den Linuxkernel zusammengetragen, die durchaus für den alltäglichen Bedarf relevant sein dürften. Geht es bspw. um die maximale unterstützte Dateigröße, die bei Datenbanken mitunter erreicht werden wird. Oder aber die Anzahl offener Dateien, deren Grenzen auf frequentierten Web- und Dateiservern eine Rolle spielen könnten. Diese Informationen hängen allerdings ebensostark von der Kernelversion ab wie von Konfigurationen. Soweit wir es wissen, unterscheiden wir daher die Versionen und geben ggf. Tipps, wie ein Limit umgangen werden kann.

Gerade die aktuelle Entwicklung im (meiner Meinung nach substanzlosen Rechtsstreit) zwischen SCO und der Linux-Gemeinde lohnt sich doch einmal ein Blick auf die zugrunde liegenden Lizenzmodelle der so genannten freien Software. In den Weiten des Netzes werden Sie die exakten Texte bze, Übersetzungen finden. Wir umschreiben das Ganze hier nur in knappen Worten. Juristen mögen uns verzeihen, wenn die Formulierungen teils wage oder gar völlig daneben sind. In dem Sinne unterscheiden sich unsere Aufassungen allerdings vom heutigem Rechtsverständnis;-)

Eine Shell-Skripte und Konfigurationstipps, die sich im Laufe der Zeit angesammelt haben, und vielleicht doch für den einen oder anderen Leser von Interesse sind, stellen wir ebenso innerhalb dieses Abschnitts vor.

Inhalt des Anhangs

Limits unter Linux

Übersicht

Selten wird man in die Verlegenheit kommen, die voreingestellten Schranken für Systemressourcen ausschöpfen zu müssen. Dennoch stellt sich die Fragen wie: »Wir groß darf eine Datei maximal werden?« oder »Wieviele Prozesse dürfen gleichzeitig gestartet werden?«.

Der nachfolgende Abschnitt soll über einige Schranken bzgl. der x86er Architektur und Kernelversionen ab 2.4.x Auskunft geben. Für ältere Kernel sind die Werte nur angegeben, wenn sie von den aktuellen Schranken abweichen. Ebenso erfahren Sie, wie sich ggf. die Limits ändern lassen.

Prozessoren & Prozesse

Maximale Anzahl Prozessoren: 32

Der Wert kann zwar in den Kernelquellen (Datei include/linux/threads.h) direkt angepasst werden, jedoch eign sich Linux auch dann nicht für massiv parallele Rechner.

Maximale Anzahl Tasks: flexibel

Der Wert wird dynamisch anhand des vorhandenen Hauptspeichers berechnet. Er kann im Prozessdateisystem Laufzeit angepasst werden (/proc/sys/kernel/threads-max).

Maximale Anzahl Tasks (Kernel <= 2.2.x): 4096

Der Wert ist in der Datei include/linux/tasks.h in den Kernelquellen festgelegt; wenn Ihr System einmal Fehlermeldungen bringen sollte, dass sich keine neuen Prozesse starten lassen, obwohl noch reichlich Speicher Verfügung steht, sollten Sie eine Erhöhung des Werts in Betracht ziehen (allerdings nur, wenn ihr Rechner auc der Last gewachsen ist...).

Maximale Anzahl Tasks pro Nutzer: Tasks/ 2

Damit ist klar, dass durch Anpassung der maximalen Anzahl Tasks auch dieser Wert beeinflusst wird.

Maximale Prozess-ID: 32768

Eine Anpassung erfolgt in include/linux/threads.h.

Maximale Anzahl Threads: 16380

Eine Anpassung erfolgt am schnellsten durch Manipulation im Prozessdateisystem (/proc/sys/kernel/tasks).

Speicherverwaltung

Maximale Größe an physischem Speicher: 64 GB

...

Maximale Größe an virtuellem Speicher: 64 GB

64 GB ist die Obergrenze, kleinere Werte (meist 1GB) sind aus Effizienzgrenzen im Kernel konfigurierbar.

Maximale Größe an virtuellem Speicher Kernel 2.2.x: 3 GB

Maximale Anzahl an Shared Memory-Segmenten: 2147483647

...

Dateien & Dateisysteme

Maximale Anzahl verwaltbarer Dateisysteme: 64

include/linux/fs.h

Maximale Dateigröße: 2^{63} Byte

Kernel 2.2.x 2 GB (2^{31} Bytes)

Maximale Anzahl offener Dateien: 8192

Aber maximal 1024 pro Prozess bzw. pro User. Die Gesamtanzahl kann im /proc/sys/fs/file-max angepasst wer Root kann per »ulimit« den Wert erhöhen.

Maximale Dateinamenslänge: 256 Zeichen

...

Maximale Pfadnamenslänge: 4096 Zeichen

...

Partitionen

Maximale Partitionsgröße (ext2): 4TB

Bei Kernel 2.2.x: 2 GB

Maximale Partitionsgröße (reiserfs): 10^{25}Bytes

...

Maximale Größe einer Swap-Partition: 2GB

Bei Kernel < 2.1.117 128MB

Maximale Anzahl von Swap-Partitionen: 8

Mit entsprechendem Patch 63, festgelegt in include/linux/swap.h.

Sonstiges

Maximale Kernelgröße (bzip2): 2.5 MByte

1 MB bei Laden von Floppy oder CD

Lizenzmodelle

Übersicht

Seit den 50er Jahren beherrschte IBM mit seinen Mainframes (Großrechnern) den Hardwaremarkt. Man kann Parallelen zu heutigen Ereignissen sehen... zumindest missbrauchte auch damals IBM seine Vormachtstellung und diktierte das Geschehen rund um Hard- und Software. Die Konsequenz war, dass nahezu jede Hardware und nahezu jedes Stück Programm aus der firmeneigenen Schmiede stammte und andere Anbieter keinen Fuß in diesem Marktsegment fassen konnten.

Auch damals schob erst das Gesetz einen Riegel vor die Machenschaften von IBM. Im so genannten Antitrust-Gesetz wurde IBM 1972 in eine Hardware- und eine Softwarefirma aufgeteilt.

Die Phase gilt als der eigentliche Beginn der Ära der Softwareindustrie.

Lizenzmodelle auf einen Blick

Der simple Vergleich versucht die gängigsten Lizenzmodelle unter Aspekt ihrer Nutzungsbedingungen gegenüber zu stellen. »Nichts kosten« besagt allerdings nur, dass die Software kostenlos erhältlich ist, es sagt nichts über die Folgekosten während der Nutzung aus.

	Kostet nichts	Unbeschränkte Nutzungsdauer	Quellen verfügbar	Ableitungen stets frei	Ableitung fällt automatisch unter dieselbe Lizenz
Kommerziell					
Shareware, Evaluation-Software	teilweise				
Freeware	x	x			
BSD-Lizenz	x	x	x		
Künstlerische Lizenz	x	x	x		
Gnu LGPL	x	x	x	x	
Gnu GPL	x	x	x	x	x

Software und Recht

Wir sind keine Juristen und haben weder die Muße noch die Fähigkeiten hinter jeder Formulierung der Gesetzeshüter die Variationen zur Auslegung zu erkennen und auszunutzen. Was wir hier zum besten geben, sollte als bloße Zusammenfassung der relevanten deutschen Gesetztestexte verstanden werden.

Einige Lizenzen - kurz vorgestellt

GPL und LGPL

Die *Gnu General Public License* und *Gnu Library General Public License* unterscheiden sich nur in Details. So soll nur auf erstere genauer eingegangen und die Änderungen bez. der Software-Bibliotheken im Anschluss kurz genannt werden.

Das Wesen beider, von der Free Software Foundation ins Leben berufener Lizenzformen offenbart sich schon in den einleitenden Worten:

Lizenzen der meisten Softwareprodukte zielen darauf ab, dem Nutzer die Freiheit der Verbreitung und Änderung der Software zu nehmen. Im Unterschied hierzu ist die GNU General Public License eine Garantie der Freiheit, freie Software auszutauschen und zu teilen - um zu sichern, dass diese Software für alle Nutzer frei ist.

Verfolgen wir einmal die **Kernaussagen der GPL**.

- Ihr **Geltungsbereich** erstreckt sich auf alle Programme, deren Entwickler den Copyright-Hinweis ausdrücklich zu diesem hinzugefügt haben, sowie auf alle Programme, die Teile eines GPL-Programmes enthalten.
- Programme sind **frei kopierbar**, jeder Kopie ist der Text der GPL beizulegen. Die Kosten für den Kopiervorgang dürfen erhoben werden. Der Quelltext des Programmes muss mit der Kopie nicht vertrieben werden, jedoch ist jedem Interessenten Zugang zu diesem zu gewähren (für eine Zeitdauer von mindestens 3 Jahren ab Vertrieb).
- Die **Modifzierbarkeit** ist gewähleistet, wobei jede Änderung im Quelltext zu kennzeichnen ist (Datum und Grund der Änderung, Name des Programmierers). Ein Programm, das Teile aus GPL-Software enthält, unterliegt damit automatisch der GPL. Dies gilt nicht für klar abtrennbare Programmteile, die vollständig ohne GPL-Code implementiert sind. Diese können anderen Lizenzbedingungen unterstehen.
- Der Urheber des Programmes trägt **keine Verantwortung** für die Korrektheit der Software und muss für eventuelle Schäden durch den Betrieb dieser nicht aufkommen.

Nach den obigen Bestimmungen wäre es unmöglich, basierend auf einer GPL-Softwareumgebung kommerzielle Software zu entwickeln, da z.B. unter Linux bereits die Compiler und Bibliotheken reine GPL-Software sind. Deswegen wurde die LGPL erschaffen, die folgende Erweiterungen zu GPL definiert:

- Die entstehende Software muss selbst eine Bibliothek sein. Wird innerhalb einer Bibliotheksfunktion auf Teile eines Nicht-GPL-Anwendungsprogrammes zurückgegriffen, so muss sicher gestellt sein, dass die Bibliothek auch ohne dieses Programmes korrekt arbeitet.
- Um die LPGL anstelle der GPL zu benutzen, müssen entsprechende Hinweise in die Software aufgenommen werden
- *Ein Werk, das die Bibliothek benutzt* ist ein Programm, das weder abgeleitete noch andere Teile einer Bibliothek besitzt, aber so hergestelllt wurde, dass es durch Compilieren und Binden mit der Bibliothek mit dieser zusammen arbeitet. Ein solches "Werk" fällt nicht unter die Lizenz.
- In gewissem Umfang dürfen in Software Teile aus GPL-Programmen (numerische Parameter, Datenstrukturvereinbarungen, kleine Makros und Inline Funktionen bis zu 10 Zeilen Länge) verwendet werden, ohne dass das entstehende Werk der Lizenzbedingung unterliegt
- Ein Programm, das mit Teilen einer LGPL-Bibliothek gebunden wurde, unterliegt der LGPL.
- Enthält eine Bibliothek sowohl Funktionen, die auf GPL-Software basieren, als auch eigene Funktionen, so muss der Quelltext der unter die GPL-Lizenz fallenden Funktionen für jederman zugänglich sein.

BSD Lizenz

Die BSD-Lizenz beinhaltet keinerlei Einschränkungen für den Gebrauch und die Weiterverbreitung von Quellcode und Programmen. Einzig einen Copyright-Hinweis, die BSD-Lizenzbedingungen selbst und ein Garantieausschluss sind dem Werk beizulegen. Außerdem darf in einem abgeleiteten Werk nicht mit dem Namen des ursprünglichen Entwicklers geworben werden, es sei denn, dieser erklärt sich ausdrücklich damit einverstanden.

Künstlerische Lizenz

Der Inhaber des Copyrights besitzt als Schöpfer eines Programmpaketes eine gewisse »künstlerische« Kontrolle über die Enwicklung seines Paketes. Damit ist es jedem erlaubt, neben der freien Verteilung »kleinere« Änderungen am Paket eigenmächtig vorzunehmen. Teile eines unter die künstlerische Lizenz fallenden Werkes können in kommerzielle Produkte eingebunden werden.

Skripte, Programme und Anderes

Übersicht

An dieser Stelle sollen nützliche Skripte folgen, auf die teilweise im Buchtext verwiesen wird. Einige der Skripte sind bewusst einfach gehalten und verzichten zu Gunsten der Verständlichkeit auf umfangreiche Fehlerbehandlungen. Der Leser sollte diese leicht selbst ergänzen können.

Auch Themen, die nicht unbedingt in den Inhalt des Buches passen, finden ggf. hier ihren Platz.

crypt - Verschlüsselung von Passwörtern

Das nachfolgend beschriebene Programm ist ein Ersatz für das Unix-Programm **crypt**, das den meisten Linux-Distributionen nur in Form einer Bibliotheksroutine beiliegt. Das Programm erwartet als Eingabe das zu verschlüsselnde Passwort und eine aus 2 alphanumerischen Zeichen bestehende Zeichenkette - das Salz in der Suppe. Mit Hilfe dieser beiden Zeichen wird die Arbeitsweise des Algorithmus auf eine von 4096 möglichen Arten modifiziert. Die Zeichen können beliebig gewählt werden.

Aber nun das Listing des Programmes:

```
#include <unistd.h>
#include <stdio.h>

int main (int argc, char** argv)
{
  if ( argc != 3 )
  {
    fprintf(stderr, "%s need two arguments to encrypt\n", argv[0]);
    return -1;
  }

  printf("%s", crypt(argv[1], argv[2] ));

  return 0;
}
```

Übersetzt wird das Programm mit:

```
user@sonne> cc crypt.c -o mycrypt -lcrypt
```

Ein Aufruf sieht wie folgt aus:

```
user@sonne> ./ mycrypt MyPassword X7
X7xe.8.ZuZ/PE
```

Zur Nutzerverwaltung

Farbe für die Konsole

Abbildung 1: Weiße Schrift auf schwarzem Grund...

Genug von Schwarz und Weiß? Dann gibt es mehrere Möglichkeiten, die Konsole ein wenig aufzupeppen...

Das Kommando **ls** kann mittels der Option **--color** zu einer farbigen Darstellung der Ausgabe überredet werden. Dazu verwendet **ls** die in der Datei /etc/DIR_COLORS angegebenen Informationen.

```
user@sonne> cat /etc/DIR_COLORS
################ gekürzte Datei ############################
# Configuration file for the color ls utility
# This file goes in the /etc directory, and must be world readable.
# You can copy this file to .dir_colors in your $HOME directory to override
# the system defaults.

# COLOR needs one of these arguments: 'tty' colorizes output to ttys,but not
# pipes. 'all' adds color characters to all output. 'none' shuts colorization
# off.
COLOR tty

# Extra command line options for ls go here.
# Basically these ones are:
#  -F = show '/' for dirs, '*' for executables, etc.
#  -T 0 = don't trust tab spacing when formatting ls output.
OPTIONS -F -T 0

# Below, there should be one TERM entry for each termtype that is colorizable
TERM linux
TERM console
TERM con132x25
TERM con132x30
TERM con132x43
TERM con132x60
TERM con80x25

# EIGHTBIT, followed by '1' for on, '0' for off. (8-bit output)
EIGHTBIT 1

NORMAL 00       # global default, although everything should be something.
FILE 00         # normal file
DIR 01;34       # directory
LINK 01         # symbolic link
FIFO 40;33      # pipe
SOCK 01;35      # socket
BLK 40;33;01    # block device driver
CHR 40;33;01    # character device driver
```

```
# This is for files with execute permission:
EXEC 01;31

# List any file extensions like '.gz' or '.tar' that you would like ls
.tar 00;31 # archives or compressed (red)
.tgz 00;31
.taz 00;31
.lzh 00;31
.zip 00;31
.z   00;31
.bz2 00;31
.jpg 01;35 # image formats
.gif 01;35
.bmp 01;35
.xbm 01;35a
```

Mit Hilfe des Kommandos **echo** lässt sich die Farbe für alle folgenden Ausgaben steuern:

Aufruf: echo -e "\033[<Flag>;<Farbe>[;<Farbe>]m"

Als »Flag« sind folgende Codes zulässig:

00	normale Darstellung
01	fette Schrift
04	unterstrichen
05	blinkende Schrift
07	inverse Darstellung
08	verborgene Darstellung (keine Ausgabe)

Als »Farbe« sind folgende Codes zulässig:

	Textfarbe		Hintergrundfarbe
30	schwarz	40	schwarz
31	rot	41	rot
32	grün	42	grün
33	gelb	43	gelb
34	blau	44	blau
35	violett	45	violett
36	kobaltblau	46	kobaltblau
37	weiß	47	weiß

Die Eingabe `user@sonne> echo -e "\033[01;33;41m"` erzeugt folgende Konsolenoptik:

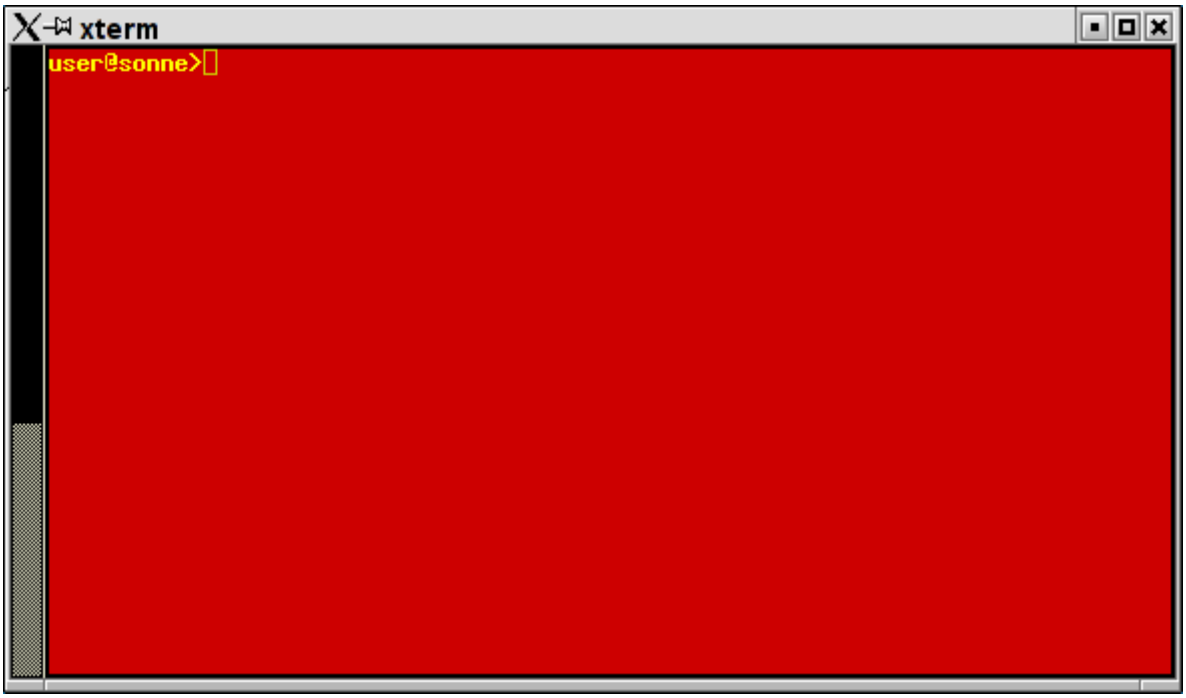


Abbildung 2: Etwas Farbe für die Konsole

Zurück zum Abschnitt Bash.

Druckversion der Linuxfibel

Die Generierung der Druckversion der Linuxfibel übernimmt ein kombiniertes Awk- und Bash-Skript. Eine Diskussion des Skript selbst finden Sie im Abschnitt Bashprogrammierung, Komplexe Anwendungen.

```
user@sonne> cat printversion.sh
#!/bin/sh

linuxfibel_base=${1:-./}
awk_script=/tmp/`basename $0`.$$

trap 'test -e $awk_script && rm $awk_script' 2 15

test -d $linuxfibel_base || { echo "Verzeichnis $linuxfibel_base existiert nicht"; exit 1; }

 cd $linuxfibel_base
 test -e vorwort.htm          || { echo "Falsches Linuxfibel-Verzeichnis?"; exit 1; }
 test -d printversion         || mkdir printversion
 test -L printversion/images || (cd printversion && ln -s ../images)

cat > $awk_script << EOF
#--------- AWK-SCRIPT BEGINN ----------
#!/usr/bin/awk -f

BEGIN {
    DoPrint="true"
    IGNORECASE=1
}

-->
 /<script language="JavaScript">/        { DoPrint = "false" }
 /<\/head>/                              { DoPrint = "true" }
 /<body bgcolor/                         { print \$0; DoPrint = "false" }
 { getline; print \$0; DoPrint = "false" }
```

Glossar

? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

?

42 Die immense Wichtigkeit der Zahl offenbart sich nur dem Kenner der großen Weltliteratur. Für alle jene, denen der Name *Douglas Adams* genauso wenig geläufig ist, wie die Pläne der intergalaktischen Umgehungstraße, die letztlich zum Verlust der Frage zur Antwort **42** führte, seien dessen tiefschürfende Diskussionen einer uns zu erwartenden Zukunft wärmstens empfohlen.

A

ARP (*Address resolution protocol*) Dieses Protokoll ermöglicht einem Rechner zu einer gegebenen IP-Adresse die Hardwareadresse des Rechners zu ermitteln, dem diese IP-Adresse zugeordnet ist. Er sendet hierzu eine Broadcast-Anfrage im lokalen Netzwerk aus; der Zielrechner wird ihm als Antwort seine Hardwareadresse übermitteln.

Awk Eine von *Aho, Kernighan* und *Weinberger* entwickelte Programmiersprache mit spezieller Ausrichtung auf das Suchen und Bearbeiten von Mustern in Zeichenketten.

B

Backup Unter einem Backup versteht man die Sicherungskopie einer Datei oder einer ganzen Dateiliste, um im Falle eines ungewollten Datenverlustes diese wieder herstellen zu können.

Betriebssystem ist ein Programm, das die Ausführung anderer (Anwendungs-)Programme steuert. Es reguliert den Zugang zur Hardware, organisiert die Datenspeicherung, regelt die Ein-und Ausgaben usw.

BIOS Beim *Basic Input / Output System* handelt es sich um ein kleines Programm, das meist in einem permanenten Speicher (ROM) des Mainboards gehalten wird und nach dem Einschalten des Rechnern die Steuerung der Grundkomponenten des Computers übernimmt.

Bootmanager (auch Bootloader). Ein Bootmanager ist ein kleines Programm, dessen Code an einer dem BIOS bekannten Speicheradresse auf einem bootfähigen Medium liegt. Er dient zum Laden des Betriebssystems in den Hauptspeicher und anschließendem Start dessen.

Broadcast ist i.A. eine an alle Recher eines Netzwerkes gerichtete Nachricht. Eine Nachricht an einen bestimmten Rechner nennt man auch **Unicast**, an eine Gruppe von Rechnern (aus unterschiedlichen Netzwerken) **Multicast**.

C

Chos (*Choose Operating System*) siehe Bootmanager

CUPS (*Common Unix Printing System*) ist eine Implementierung des **Internet Printing Protokolls** (IPP) und stellt eine vollwertige Alternative zum verbreiteten Drucksystem Lpd dar.

D

Dämon Als Dämon (daemon) wird ein Systemprogramm bezeichnet, welches im Hintergrund tätig ist und dort seine Arbeit erledigt (Siehe auch Hintergrundprozess).

DRI *Direct Rendering Infrastructure* koordiniert den konkurrierenden Zugriff von X-Server und OpenGL-Treibern auf die Grafikkarte. DRI ist damit eine Schlüsselkomponente zur Intergration von 3D-Hardware-Unterstützung in

die Architektur von Xfree86 Version 4.x.

E

F

FIFO *First In First Out*, ein Datencontainer, der sicherstellt, dass hinzugefügte Elemente exakt in ihrer Reihenfolge auch wieder entfernt werden können. Entspricht quasi einer Warteschlange: »Wer zuerst kommt, ist auch zuerst dran.« (zumindest solange nicht gedrängelt wird;-).

Filter Unter Unix fasst man Programme unter diesem Begriff zusammen, die ihre Eingabe von der Standardeingabe beziehen und das Ergebnis auf die Standardausgabe schreiben. Die Eingaben werden "gefiltert". Filter besitzen die Eigenschaften, dass man verschiedene Programme verketten kann, d.h. die Ausgabe eines Filters wird die Eingabe eines anderen Filters...

G

GLX ist ein Mechanismus, um 3D-Fähigkeiten auch im Netzwerk verfügbar zu machen. Hierbei werden die Funktionen von OpenGL durch das X-Protokoll hindurch getunnelt.

GNOME Das *GNU Network Object Model Environment* ist eine Ansammlung von Bibliotheken und Programmen, die eine einheitliche Benutzerschnittstelle fü das X Window System bereit stellen. Neben KDE ist GNOME das wichtigste freie Projekt zur Implementierung einer grafischen Oberfläche unter Unix.

Grub (*Grand Unified Bootloader*) siehe Bootmanager

H

Hash(tabelle) ist eine Datenstruktur, in der Werte zusammen mit einem Schlüssel gespeichert werden. Anhand des Schlüssels kann nach einer Berechnungsvorschrift (**Hashfunktion**) extrem schnell [O(1)] der Speicherort eines Wertes ermittelt werden.

Hintergrundprozess ist ein sich in Ausführung befindendes Programm, das keine Verbindung zur Standardein- und -ausgabe besitzt.

I

ICMP *Internet Control Message Protocol* ist ein Protokoll der TCP/IP-Familie und dient vorrängig zur Übertragung von Fehler- und Diagnosenachrichten. Ist z.B. ein Datenpaket in einem Router nicht vermittelbar (weil die Lebensdauer des Paketes abgelaufen ist oder weil der Adressat unbekannt ist...), so wird dieser eine ICMP-Nachricht mit der Fehlerursache an den Absender des Paketes schicken.

IEEE: *Institute of Electrical & Electronic Engineers*.

J

Jokerzeichen sind Sonderzeichen der Shells, die als Platzhalter für Zeichenklassen stehen. Üblicher ist die Bezeichnung **Wildcards**.

K

Küslübürtün ist der Name des guten Geistes der Sprache im Kinderbuch "Die dampfenden Hälse der Pferde im Turm zu Babel" von Franz Fühmann.

L

Lpd

(*Line Printer Daemon*) ist ein ursprünglich für Zeilendrucker entwickeltes Drucksystem unter Unix, das mittels trickreicher Erweiterungen zur Unterstützung modernerer Drucker erzogen wurde.

Lilo (*Linux Loader*) siehe Bootmanager

Leichtgewichtsprozess siehe Thread

M

MAC *Media Access Control* Ein Protokoll aus der OSI-Schicht 2, das den Zugang zum physischen Netzwerk regelt.

Mbr *Master Boot Record* Es handelt sich um den ersten Sektor (512 Bytes) einer Festplatte. Der Aufbau des Mbr ist standardisiert, sodass er von jedem BIOS gelesen werden kann. Er kann den zum Laden eines Betriebssystems notwendigen Bootcode aufnehmen, daher auch seine Bezeichnung. Zu Beginn des Mbr steht die 64 Byte große Partitionstabelle, die die Beschreibung der Aufteilung (oder Teile davon) der Festplatte enthält. Die nächsten 446 Bytes sind für den Bootcode vorgesehen. Die letzten beiden Bytes enthalten eine so genannte *Magic Number* (AA55).

Mounten beschreibt den Vorgang des Verbindens eines Verzeichniseintrags mit einem auf einem Speichermedium liegenden Dateisystem.

N

NFS *Network File System* ist ein System , das die Nutzung von Verzeichnissen über Rechnergrenzen hinweg ermöglicht. Eine solche Nutzung geschieht transparent, d.h. die über das NFS importierten Verzeichnisse verhalten sich als wären sie lokal vorhanden.

O

OpenGL *Open Graphic Library* ist die verbreitetste Bibliothek unter Linux, um 3D-Grafikfunktionen bereitzustellen.

P

Parsen ist die Analyse und Aufbereitung eines Datenstroms nach konkreten syntaktischen Regeln. Ein hierzu verwendetes Programm wird als **Parser** bezeichnet.

Partitionstabelle Sie beschreibt die Verteilung der Partitionen einer Festplatte. Da die Beschreibung einer Partition insgesamt 16 Byte umfasst (u.a. Status, Typ, Start- und Endsektor, Länge) und nur 64 Byte für die Partitionstabelle zur Verfügung stehen, kann die Tabelle nur vier Einträge beherbergen. Um dennoch eine höhere Anzahl von Partitionen auf einer einzigen Festplatte verwalten zu können, darf eine diese so genannten **primären** Partitionen als **erweiterte** gekennzeichnet werden. Dieser Eintrag kann wiederum auf eine so genannte **logische** Partition verweisen, in welcher ggf. die Adresse einer weiteren logischen Partition gespeichert ist. Auf diese Art und Weise lassen sich bis zu 63 (IDE) bzw 15(SCSI) logische Partitionen auf einer einzigen Festplatte unterbringen.

Pipe bezeichnet in einer Shell die Verbindung der Ausgabe eines Kommandos mit der Eingabe eines anderen Kommandos. Die Shells stellen hierfür das Symbol **|** zur Verfügung, wobei das links des Pipesymbols stehende Kommando in diese schreibt und das rechts stehende daraus liest.

POSIX *Portable Operating System Interface for uniX* ist ein Programmierstandard, womit Programme leichter auf andere Plattformen portierbar werden. In der Praxis ist die Portierung POSIX-konformer Programme dennoch nicht so einfach...

Prompt Im Allgemeinen wird so die Eingabeaufforderung eines Kommandozeileninterpreters bezeichnet.

Protokoll bezeichnet eine Sammlung von Regeln, wie eine Kommunikation gesteuert werden kann oder wie Daten zu übertragen sind.

Prozess Ein Programm in Ausführung inklusive einer im System eindeutigen Prozessnummer und der Prozessumgebung (Stack, Daten, ...).

Q

Queue (Warteschlange) ist eine Datenstruktur, in die auf der einen Seite Daten eingefügt und auf der anderen Seite wieder ausgelesen werden können. Daten müssen in der Reihenfolge einer Queue entnommen werden, in der sie eingefügt wurden.

R

Recovery ist das Wiederherstellen des Datenbestandes nach dessen ungewollter Zerstörung mit Hilfe eines Backups.

r-Utilities Sammlung Unix-spezifischer Kommandos, die im Netzwerk analog zu den lokalen Kommandos arbeiten (sollten). Die Utilities wurden in Berkeley entwickelt und die enthaltenen Programme beginnen mit den Buchstaben "r" (remote), deshalb die Namensgebung.

S

Semantik definiert Regeln, wie ein bestimmtes Objekt zu deuten ist; sie ordnet ihm einen Sinn zu (siehe auch Syntax).

Socket Die zentrale Komponente des Berkeley-Entwurfs für die Netzwerkkommunikation ist der Socket (Kommunikationsendpunkt). Analog zu einem Telefon, das den Zugang zum Telefonnetz ermöglicht, bieten Sockets dem Benutzer eine Schnittstelle zum Netzwerk.

Subnetz Der Begriff wird in zwei Zusammenhängen gebraucht. So fasst man die Protokollschichten 1 (Transportschicht) und 2 (Vermittlungsschicht) des OSI-Referenz-Protokolls im TCP/IP-Jargon als Subnetz zusammen, um die enge Verknüpfung beider anzudeuten. Auf der OSI-Ebene 3 (Vermittlungsschicht) bezeichnet ein Subnetz einen logischen Abschnitt einer Netzklasse.

Syntax Ganz allgemein beschreibt die Syntax einen Satz von Regeln, nach denen ein Objekt aufgebaut ist. In der Informatik schreibt die Syntax bspw. vor, wie eine bestimme Konfigurationsdatei aufgebaut ist oder welche Elemente und Struktur ein gültiges Computerprogramm besitzen darf. Im übertragenen Sinne umfasst die Syntax der natürlichen Sprache die Regeln zur Rechtschreibung und zur Grammatik; sie definiert aber in keinster Weise die Bedeutung der resultierenden Wortgefüge. Einem Objekt ordnet erst die Semantik eine konkrete Bedeutung zu.

T

TCP *Transmission Control Protocol* ist ein Protokoll der TCP/IP-Familie, das eine verbindungsorientierte Übertragung von Daten realisiert. Dazu baut das Protokoll zunächst eine Verbindung zum Empfänger auf und sendet erst nachfolgend die Daten. Das Protokoll stellt dabei die Unversehrtheit der übertragenen Pakete sicher.

TELNET Terminal Emulation for Networks

Terminal In Zeiten der Großrechner wurde die gleichzeitige Arbeit mehrerer Benutzer durch Anschluss mehrerer Terminals an einen Rechner ermöglicht. Unter einem Terminal verstand man damals die Kombination aus Bildschirm und Tastatur. Im heutigen Sprachgebrauch steht das Terminal allgemein für eine zusammengehörige Ein- und Ausgabe.

Thread Im Zusammenhang mit Mails und Newsartikeln werden alle Beiträge, die sich auf eine Ausgangsnachricht/Ausgangsartikel beziehen, als Thread bezeichnet.

Thread In der Programmierung eine Möglichkeit, um mehrere Instanzen eines Programms parallel zu starten. Die Threads eines Programms teilen sich mit dem Hauptprogramm den Programmkode, der nur einmalig im Hauptspeicher geladen ist (im Unterschied zur Parallelisierung auf Prozessebene, wo jeder Prozess eine eigene Kopie besitzt). Um alternative Programmabläufe im Thread zu realisieren, besitzen diese einen eigenenen Befehlszähler und threadlokale Variablen u.a.m. Da Threads i.A. weniger Ressourcen verbrauchen als Prozesse, werden sie auch als »Leichtgewichtsprozesse« bezeichnet.

U

UDP *User Datagram Protocol*, ein Protokoll der TCP/IP-Familie, das verbindungslos arbeitet. Die Datenpakete werden dabei im Stil eines Telegramms zum Zielrechner gesendet, ohne den Erfolg der Übertragung zu überprüfen. Eine solche Kontrolle wird dann meist von einem darüber liegenden Protokoll vorgenommen.

V

VFS *Virtual File System* Eine Schnittstelle zwischen Betriebsystem und den physischen Dateisystemen mancher Unix-Systeme. Ein auf Daten zugreifendes Programm verwendet dann immer dieselben Systemrufe, unabhänig vom Dateisystem, auf dem die Daten physisch abgelegt werden. Das VFS ist für die Umsetzung der Systemrufe auf die tatsächlichen Funktionen des Dateisystems zuständig.

W

Whitespace Zusammenfassung für Leerzeichen, Tabulator und Zeilenumbruch, also die Zeichen, die i.A. zwei Wörter voneinander abgrenzen.

WYSIWYG *What You See If What You Get* bezeichnet die Eigenschaft einiger Programme (Schreib-, oder Grafikprogramme, ...), ihre Eingaben so darzustellen wie die spätere Ausgabe.

X

X Kurzbezeichnung für das X Window System, die grafische Oberfläche für Unix-Systeme.

XDR *External Data Representation* ist ein Protokoll der Schicht 6 (Repräsentation) des OSI-Referenz-Protokoll-Stacks und definiert eine einheitliche Darstellung von Daten, so dass diese auch von Rechnern mit unterschiedlichen Datenformaten verarbeitet werden können.

Y

Yellow Pages ist der frühere Name des Network Information Systems, den die Firma Sun im Rechtsstreit um das Markenzeichen der British Telecom freiwillig ablegte. Die zu NIS gehörigen Programme werden dennoch traditionell mit yp*programmname* benannt.

Z

Zombie ist ein Prozess, der seine Arbeit zwar beendet und seinen Elternprozess darüber informiert hat (Signal SIGCHILD), aber noch auf die Behandlung dieses Signals durch den Elternprozess wartet.

Informationen im Netz

Übersicht

Natürlich sind die ganzen Fakten im Buch nicht (allein) »auf meinem Mist« gewachsen. Die einzelnen Quellen im Text zu verzeichnen, ist kaum machbar, denn kein Abschnitt basiert auf einer einzelnen Recherche, sondern beinhaltet dutzende Informationen, die im Laufe der Zeit den Weg in mein Gedächtnis fanden. Wollte ich jedwede Quelle benennen, müsste ich sicher bei meinen Vorlesungen während des Studiums beginnen. Aber als recht typischer Student, dessen häufigste Erscheinungsform die Abweisenheit war, sind schon meine damaligen Aufzeichnungen reichlich lückenhaft.

Deshalb sollen auf dieser Seite reichlich Verweise in die weite Welt des Webs stehen, die mehr oder minder zum meinem Erfahrungsschatz beitrugen und -tragen.

Suche im Netz

Es ist in Foren nicht gern gesehen, wenn wiederholt Probleme angeschnitten werden, deren Lösungen schon mehrfach zum besten gegeben wurden. Deshalb sollte eine jede Fragestellung zunächst im Web recherchiert werden. Als wahre Fundgruben erweisen sich die Suchmaschinen, wobei die Thematik »Linux« wohl am umfassendsten bei http://www.google.de behandelt wird. Weitere nützliche Spürnasen sind:

http://www.altavista.de
http://www.excite.de
http://www.fireball.de
http://www.lycos.de
http://www.witch.de

Umfangreichere Aufzählungen sind im Netz unter http://www.suchmaschinen.de, http://www.suchfibel.de und http://www.suchmaschinen-verstehen.de zu finden.

Hilfe im Netz

Nicht immer verhelfen Suchmaschinen zur erhofften Erleuchtung... In Diskussionsforen besteht am ehesten die Chance, unter Gleichgesinnten gemeinsam eine Thematik zu erörtern. Deutschsprachige Foren mit »gutem Umgangston«, sind u.a.:

http://www.linuxforen.de
http://computerforum.stayhere.de/forum/
http://www.pl-forum.de

Verwenden Sie bitte zuerst die Suchfunktionen der Foren, um Beiträge mit Relevanz zu Ihrem Anliegen aufzuspüren. Das Lesen in Foren ist zumeist ohne vorherige Anmeldung möglich. Erst wenn Sie eigene Diskussionsbeiträge einzubringen gedenken, wird eine (anonyme) Registrierung notwendig. Nehmen Sie sich die Zeit, um die jeweiligen Forenregeln zu studieren!

Weitere Plattformen zur Diskussion stellen diverse Mailingslisten bereit. Nahezu zu jeder Distribution existiert (mind.) eine eigene Liste, der jeder Interessierte beitreten darf. Möglichkeiten zur Anmeldung finden Sie zumeist auf den Webseiten der Linux-Distributoren.

Speziell die Belange der SuSE-Distribution behandelt die Support-Datenbank von SuSE. Aber auch zahlreiche allgemein gehaltene Informationen finden sich darunter.

Linuxseiten

Seiten mit Linux-relevantem Inhalt gibt es zu Hauf. Schon allein mit den Adressen der einzelnen Linux-User-Groups in Deutschland ließe sich die Seite füllen. Ich werde mich daher auf »lebende« Seiten beschränken, deren Inhalte auch in letzter Zeit aktualisiert wurden.

Deutschsprachig

Herausheben möchte ich dennoch ein Projekt, das ein ähnliches Ansinnen verfolgt, wie es die Linuxfibel tut. Auch wenn sie derzeit noch in den Kinderschuhen stecken, so besitzen die Seiten von http://www.linuxhilfen.org das Potenzial, sich zu einer wahren Fundgrube zu mausern.

Die Reihenfolge der weiteren Links spiegeln in etwa den Nutzen wieder, den sie für mich haben.

Pro Linux
Linuxinfo.de
Linux NetMag
64-Bit
Netzmafia

Einige englischsprachige Seiten...

Linux, Homepage
Linuxgazette
Alan Cox
Linux today
Linux newbie
Woven Goods for Linux
Linux Tutorials (IBM)

Linux-Anbieter

Zu den Distributoren gibt es ja wohl nicht viel zu sagen...

Caldera
Corel
Debian
Mandrake
Red-Linux
RedHat
Slackware
Stempede
SuSE

Homepages ausgewählter Programme

Zahlreiche Projekte haben unterdessen auf Sourceforge ihre neue Heimat gefunden; ihre Webadressen sollen nicht extra erwähnt werden.

Fai

Samba

Sonstiges

Gnu is Not Unix;-)

Packer - Shareware zum Entpacken verschiedener Archivformate unter Windows (u.a. bzip2)

Register - Überblick

Überblick
Inhalt des Kapitels

Überblick

Das vorliegende Register der Stichwörter ist leider noch unvollständig. So lange die Linuxfibel noch in der Entstahungsphase steckt, werden wir das Register auch nur stiefmütterlich behandelt. Also bemühen Sie besser eine Volltextsuche in ihrer lokalen Installation der Linuxfibel (grep), dies führt vermutlich schneller zur gewünschten Information.

Besser bestellt ist es um das Bildregister, das automatisch generiert wird und somit den aktuellen Stand repräsentieren sollte.

Inhalt

?

A

B

C

D

E

F

G

H

I

J

K

Konfigurationsdateien

Kwm

L

Lizenzen

M

N

O

P

Q

R

Remote Procedure Call

S

Shellprogrammierung

Shells

Signal

Software

Systemruf

- exec
- exit
- fork
- stat
- wait

Systemzeit

T

tac
talk
tkinfo
tin
top

U

umount
uname
uniq
uptime
users

V

vidmode
Virtuelle Konsole
Virtuelles Dateisystem

W

w
wc
whatis
whereis
which
who
whoami

Windowmanager

- Enligthenment
- Fvwm2
- Kwm

write

X

xinfo

X-Server

Y

Z

Zeit

? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

?

A

B

C

D

E

F

G

H

I

J

K

L

M

Q

R

S

T

U

V

W

X

Y

Z