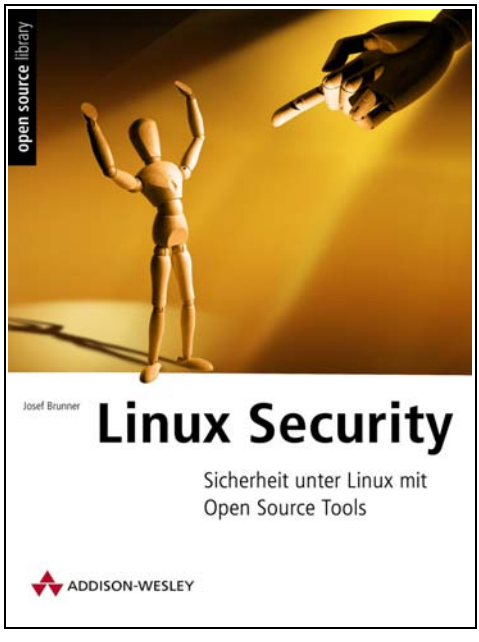
Linux-Systemadministration

open source library

Open Source Software wird gegenüber kommerziellen Lösungen immer wichtiger. Addison-Wesley trägt dieser Entwicklung Rechnung mit den Büchern der **Open Source Library**. Administratoren, Entwickler und User erhalten hier professionelles Know-how, um freie Software effizient einzusetzen. Behandelt werden Themen wie Betriebssysteme, Netzwerke und Sicherheit als auch Programmierung.

In Vorbereitung:



Sicherheit ist ein Problem aller Betriebssysteme, und meist ist es teuer, eine Installation wirklich sicher zu machen. In diesem Buch zeigt der Autor, dass dies auch ohne einen größeren finanziellen Aufwand möglich ist.
Hier erfahren Sie, wie Sie Linux mit Hilfe von Open-Source-Tools sicher machen.

Linux Security
Josef Brunner
ca. 700 Seiten, 1 CD-ROM
EUR 49,95 [D]/sFr39,50
ISBN 3-8273-1999-4

PEARSON Education

ADDISON-WESLEY

Jochen Hein

Linux-Systemadministration

Einrichtung, Verwaltung, Netzwerkbetrieb

ADDISON-WESLEY

An imprint of Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich.

Umwelthinweis:
Dieses Produkt wurde auf chlorfrei gebleichtem Papier gedruckt.
Die Einschrumpffolie – zum Schutz vor Verschmutzung – ist aus umweltverträglichem und recyclingfähigem PE-Material.

10 09 08 07 06 05 04 03 02 01

ISBN 3-8273-1992-7

ein Imprint der Pearson Education Deutschland GmbH
Martin-Kollar-Straße 10–12, D-81829 München/Germany

Einbandgestaltung: Marco Lindenbeck (mlindenbeck@webwo.de)
Lektorat: Sylvia Hasselbach, shasselbach@pearson.de
Korrektorat: Petra Kienle, Fürstenfeldbruck
Satz: reemers publishing services gmbh, Krefeld, www.reemers.de
Druck: Bercker, Kevelaer
Printed in Germany

Kapitelübersicht

Inhaltsverzeichnis

Vorwort

Linux ist ein Unix-ähnliches Betriebssystem. Damit weist ein Linux-System die gesamte Flexibilität, aber auch die Komplexität von Unix auf. Mit diesem Buch möchte ich dazu beitragen, dass Sie diese Komplexität in den Griff bekommen und die Flexibilität effektiv nutzen können.

Linux ist ein freies Betriebssystem. Das gilt nicht nur für den Kernel, sondern auch für fast alle anderen Programme. Frei heißt hier, analog zur GNU General Public License (GPL), dass der Quellcode frei verfügbar ist, und zielt nicht auf die teilweise kostenlose Verbreitung von Linux.

In der Freeware-Szene gibt es im Wesentlichen vier unterschiedliche Lizenzen, die alle ihre Berechtigung für bestimmte Zwecke haben:

- Das BSD- oder X-Copyright, das die Verwendung des Codes auch in kommerziellen und proprietären Systemen erlaubt
- Die GPL, die fordert, dass die Programme oder daraus entstandene verbesserte Versionen weiterhin frei sind
- Programme, deren kommerzielle Nutzung oder Verbreitung untersagt ist
- Programme, die keinerlei Copyright unterliegen, so genannte Public-Domain-Programme

Durch diese liberalen Lizenzbedingungen wurde Linux zu dem, was es heute ist. Insbesondere spielte die Verfügbarkeit der GNU-Programme eine entscheidende Rolle, da hiermit die Utilities eines Unix-Systems bereits frei verfügbar waren. Weiterhin war die Portierung des X-Window-Systems wichtig, das Grundbausteine für grafische Benutzeroberflächen liefert.

Freie Software ist bessere Software, weil diese dem Anwender die Möglichkeit gibt, die Programme anzupassen, zu verbessern, zu verstehen oder einfach nur Fehler zu korrigieren. Diese Möglichkeit hat der Anwender von proprietärer Software nicht, er ist auf Gedeih und Verderb seinem Software-Lieferanten ausgeliefert. Dies ist, neben der Möglichkeit, das System vollständig untersuchen zu können, das ausschlaggebende Argument für Linux oder andere Freeware-Systeme wie NetBSD oder FreeBSD. Ein weiterer Vorteil für Anwender ist, dass verschiedene Anbieter Support für das System leisten können.

Neben den Programmen selbst ist auch die zugehörige Dokumentation frei verfügbar. Das Linux Documentation Project hat es sich zur Aufgabe gemacht, sowohl HowTo-Dokumente als auch vollständige Handbücher zu schreiben. Zum Teil können Sie diese Bücher auch im Buchhandel kaufen. Es existiert ein Projekt von Marco Budde, die HowTos in die deutsche Sprache zu übersetzen.

Man muss jedoch nicht programmieren können, um Linux effektiv einzusetzen, denn Fehler werden von den Programmierern oft binnen weniger Stunden oder

Tage gefunden und korrigiert. Insbesondere, wenn man eine »stabile« Distribution von Linux installiert hat, wird das System von Anfang an stabil und zuverlässig laufen. Die Neuinstallation von neuen Programmen oder neuen Versionen beschränkt sich auf das Ausführen weniger Befehle.

Dieses Buch wurde fast vollständig mit frei verfügbarer Software unter Linux geschrieben. Als Editor verwendete ich GNU-Emacs bzw. XEmacs zusammen mit dem Paket `auctex`, das eine leistungsfähige Umgebung zum Setzen von Texten mittels TeX oder LaTeX darstellt. Der Satz erfolgte in den ersten Auflagen mit LaTeX, dabei wurden eine Reihe von zusätzlichen Styles verwendet. Die aktuelle Auflage wurde in DocBook/XML konvertiert. Als Editor kommt wieder Emacs zum Einsatz, diesmal mit dem `psgml`-Mode.

Die Stabilität und Geschwindigkeit von Linux, gemeinsam mit der leistungsfähigen Unix-Umgebung haben einen erheblichen Anteil am Gelingen dieses Werks. Dieses Buch ist allen Programmierern (und Benutzern) von freier Software gewidmet, ohne die heute nicht diese Vielfalt an freier Software existieren würde.

Typographische Konventionen

Wie in beinahe jedem Computerbuch möchte ich hier die verwendeten typographischen Konventionen vorstellen. Dies soll zu einer besseren Übersichtlichkeit beitragen, da dann einige Voraussetzungen nicht immer wieder neu erläutert werden müssen.

- Ausgaben des Rechners sind in `Courier` (Typewriter) gesetzt.
- Eingaben des Benutzers sind ebenfalls in Courier gesetzt, aber zusätzlich durch **`Fettdruck`** hervorgehoben.
- Variable Eingaben, die vom Anwender passend zu ersetzen sind, werden mit einer *`kursiven`* Schrift dargestellt.
- Zur Unterscheidung zwischen verschiedenen Shells und privilegierten und nicht privilegierten Benutzern verwende ich verschiedene Shell-Prompts:
  - # steht für den Systemadministrator, der die `bash` benutzt.
  - > steht für einen Benutzer mit der `tcsh`.
  - $ steht für einen Anwender mit der `bash`.

Zur Erläuterung folgt das Listing 1, das ein `tcsh`-Benutzer durchgeführt hat. Der Benutzer `root` wird in diesem Buch nur an den Stellen verwendet, an denen es unbedingt notwendig ist. Die Trennung zwischen Benutzer und Systemadministrator ist eines der grundlegenden Unix-Konzepte und dient der Sicherheit der Benutzer und Systemdaten sowie der Systemstabilität.

```
(linux):~> echo 'Das ist eine Eingabe' | sed s/Ein/Aus/
Das ist eine Ausgabe
```

Listing 1 Beispiel für den Einsatz verschiedener Schriften

Weitere spezielle Formatierungen verwenden wir für Shell- oder Umgebungsvariablen (`VARIABLE`). Die Namen von Programmen und Kommandos sind in Courier gesetzt (`programm`), Dateien werden zusätzlich mit dem Pfad angegeben (`/Pfad/Datei`). Listings oder Auszüge aus Konfigurationsdateien werden in Courier (`Typewriter`) gesetzt, genauso wie Rechnernamen. Die Verwendung von Tastenkombinationen wird in der folgenden Form dargestellt:

- A für die Taste »A«, Entf für die Taste »Entf«
- Strg+C für das gleichzeitige Drücken von »Strg« und »C«
- Esc x für die Taste »Escape« gefolgt von »x«

Mailing-Listen

Das Betriebssystem Linux und die zugehörigen Programme wurden von verschiedenen Programmierern im Internet entwickelt. Die Kommunikation erfolgt dabei über verschiedene Kanäle:

- Die Entwickler arbeiten mit privater E-Mail untereinander. Das ist so lange sinnvoll, wie nur eine sehr begrenzte Anzahl von Programmierern an einem Projekt arbeiten.
- Es werden (öffentliche oder private) Mailing-Listen verwendet. Ein spezieller Rechner verteilt jede Nachricht an alle Mitglieder der Mailing-Liste.
- Ein Teil der Diskussionen findet in News-Gruppen statt.

Auf dem Rechner `vger.kernel.org` sind derzeit eine Reihe von Mailing-Listen eingerichtet, die in der Regel jedem Interessenten offenstehen. Zur Verwaltung der Listen wird das Programmpaket `majordomo` verwendet. Wenn Sie eine Mailing-Liste abonnieren möchten, senden Sie eine Mail an den Benutzer *mailto: majordomo@vger.rutgers.edu* mit dem Text `help` im Textteil der Nachricht. Danach erhalten Sie eine Anleitung zur Bedienung des Verwaltungsprogramms.

Danksagung

Zunächst möchte ich mich bei allen Linux-Hackern und -Anwendern sowie bei allen Programmierern bedanken, die dieses Buch erst möglich gemacht haben. Ebenso möchte ich den Korrekturlesern danken, die viele Fehler und Inkonsistenzen aufgespürt haben und das Buch zu dem gemacht haben, was es heute ist.

Wenn Sie Fragen oder Anregungen haben, so können Sie mich über E-Mail unter der Adresse *mailto:jochen@jochen.org* erreichen.

Vorwort zur vierten Auflage

Ich verfolge die Entwicklung von Linux bereits seit 1992 – und immer wenn ich zurückblicke, schaue ich auf eine stürmische Entwicklung und eine interessante Zeit zurück. Denn trotz allen technischen Fortschritten ist die entscheidende Komponente von Linux immer noch vorhanden: eine Gemeinschaft von Anwendern und Entwicklern, die sehr hilfsbereit ist. Man spürt einfach, dass ein sehr wichtiges Element der Motivation der Spaß an der Entwicklung und am Miteinander ist.

Dieses Miteinander färbt, erzwungen von den Bestimmungen der GPL, auch auf kommerzielle Entwickler ab, wenn diese sich an Open-Source-Projekten beteiligen. Für Firmen ist das eine große Chance, denn vielfach können auch sie »auf den Schultern von Giganten stehen«. Auf der anderen Seite gibt es aber auch einen Interessenskonflikt zwischen dem Wunsch nach Freiheit, der das GNU-Projekt antreibt, und Firmen, die aus den frei verfügbaren Projekten kommerzielle und proprietäre Entwicklungen machen wollen. Unix ist in diese Falle geraten, bei Linux bestehen gute Aussichten, dass dies nicht passiert.

Das Besondere an Linux ist aber immer noch, dass es eine Gemeinschaft aus Programmierern, Anwendern und Systemverwaltern ist, die es an ihre Bedürfnisse anpasst. Ich hoffe, dass Sie sich in dieser Gemeinschaft genauso wohlfühlen wie ich. Der Zusammenhalt dieser Gemeinschaft und die Möglichkeiten der Anwender, sich gegenseitig zu helfen, haben Linux erst möglich gemacht.

Die Linux-Anwender haben die Computer-Welt verändert – in mancherlei Hinsicht sicherlich revolutionär. Wenn die Benutzerschar von Linux weiter so wächst, dann könnte sich der (scherzhafte) Plan von Linus Torvalds erfüllen: »World domination. Fast«.

Wiesbaden, im Mai 2002.
Jochen Hein

1 Linux – das Betriebssystem der Zukunft?

1.1 Linux-Distributionen

Linux – im engeren Sinne – ist »nur« ein Betriebssystem, also der Teil der Software eines Computersystems, der die Hardware kontrolliert oder ansteuert. Anders und vereinfacht gesagt: Das Betriebssystem ist die Komponente, die der Anwendersoftware (z. B. Textverarbeitung oder Newsreader) die Möglichkeiten der Hardware (z. B. Drucker, CD-ROM-Laufwerk) zugänglich macht. Andere Betriebssysteme, die die weit verbreitete PC-Hardware kontrollieren, sind z. B. OS/2, Novell Netware, SCO oder Windows 2000. DOS bzw. DOS/Windows sind zwar auch Betriebssysteme, da diese Systeme jedoch den Anwendungsprogrammen den direkten Zugriff auf die Hardware gestatten, sind Vorbehalte angebracht, DOS neben die obigen Systeme zu stellen.

Üblicherweise steht Linux aber für mehr. Es ist ein System, das eine umfangreiche Kollektion der Unix-Tools und unterschiedlicher Anwenderprogramme anbietet; an eine solche Kollektion hat man im Allgemeinen zu denken, wenn von einer *Linux-Distribution* die Rede ist. Zugeschnitten auf die individuellen Bedürfnisse eines Users oder einer User-Gruppe, werden unterschiedliche akzentuierte Distributionen von verschiedenen Seiten zusammengestellt: Die Bandbreite reicht von Distributionen, die auf wenigen Disketten die notwendigsten Tools versammeln (z. B. XLINUX), bis zu Distributionen im Umfang von mehreren hundert Megabyte, die eine komplette Entwicklungsumgebung und eine große Anzahl »freier« Anwendungssoftware zur Verfügung stellen (z. B. die Debian-Distribution). Eine sehr ausführliche Übersicht über Linux-Distributionen finden Sie unter *http://lwn.net/Distributions*.

Diese und weitere Distributionen werden auf zahlreichen `ftp`-Servern bereitgehalten, können oftmals aber auch über andere Medien (Disketten, CD-ROMs, DVDs oder Magnetbänder) bezogen werden. Angesichts des Umfangs der Distributionen ist der Erwerb einer CD einem womöglich tagelangen Download mittels `ftp` oder Modem vorzuziehen. Bei vielen Distributionen erwirbt man auch einen Anspruch auf Support durch den Distributor und ein Handbuch, das die Installation und die ersten Schritte beschreibt.

1.2 Linux-Features im Überblick

Linux ist ein leistungsfähiges Betriebssystem, das sich in den letzten Jahren von einem »Hacker-System« zu einem vielfach auch kommerziell eingesetzten System gewandelt hat. Die Gründe für diese Entwicklung sind vielfältig. Einige Gründe möchte ich hier vorstellen und zudem einen Überblick über die Einsatzmöglichkeiten von Linux geben:

- Linux ist frei, d. h., praktisch der gesamte Quellcode des Systems ist jedermann zugänglich. Damit können Interessierte Verbesserungen implementieren und Fehler beseitigen oder ganz einfach lernen, wie das System funktioniert.
- Linux ist ein Unix-ähnliches System mit der ganzen Flexibilität von Unix. Als grafische Oberfläche kommt X11 zum Einsatz. Motif und CDE sind genau wie OpenLook (XView) für Linux verfügbar.
- Linux ist zu internationalen Standards wie POSIX oder ANSI-C konform. Außerdem werden viele BSD- oder System-V-Erweiterungen unterstützt. Damit können viele Unix-Programme einfach portiert werden.
- Linux ist auf andere Hardware-Architekturen wie DEC-Alpha, MIPS oder Sun Sparc-Rechner portabel. Damit ist es eines der portabelsten Systeme, das auf dem Markt verfügbar ist.
- Linux ist flexibel in TCP/IP-Netzen einsetzbar. Daneben werden aber auch noch andere Protokolle (IPX, Appletalk, SMB) unterstützt.
- Linux ist auf älteren PCs mit erstaunlich wenig Hauptspeicher lauffähig. Ein System mit 16 MB ist als Router oder Firewall gut zu verwenden. Im Vergleich zu Windows NT sind die Hardware-Anforderungen deutlich niedriger.
- Linux nutzt Speicher (durch Shared Libraries und Demand Loading) und moderne Prozessoren (Pentium und Pentium Pro) effizient.
- Linux kann einen PC ohne große Kosten in eine Workstation, ein X-Terminal oder einen Server verwandeln.
- Linux-Support wird von den Distributoren genauso wie von den Entwicklern und den Anwendern untereinander geleistet. Prinzipiell kann jeder Interessierte kommerziellen Support anbieten, so dass der Anwender die Wahl zwischen vielen Anbietern hat.
- Linux ist gut an die übliche PC-Hardware angepasst. Es sind viele Treiber verfügbar und das System ist vergleichsweise schnell.
- Es existieren viele Anwendungen im Bereich der Programmentwicklung, Office-Pakete, Internet-Server und -Clients sowie Emulatoren für andere Systeme wie Windows oder iBCS2 (PC-Unix).

- Das System ist durch die mitgelieferten Handbücher (`/usr/doc` oder `/usr/share/doc`), Manpages und viele Bücher zu diesem Thema sehr gut dokumentiert. Sollte die Dokumentation einmal eine Frage offenlassen, so kann man schlimmstenfalls immer noch im Quellcode nach der Lösung suchen.

Diese Liste ist sicher nicht vollständig und lässt sich daher noch beinahe beliebig fortsetzen. Dabei werden von verschiedenen Anwendern jeweils andere Schwerpunkte gesetzt. Es existiert kein System, das jeden möglichen Anwender zufrieden stellen kann.

1.3 Linux-Distributionen im Vergleich

Wie oben schon erwähnt, existiert eine Vielzahl von Linux-Distributionen. Hieraus eine »gute« Distribution für den eigenen Bedarf auszuwählen, ist nicht einfach. Im Folgenden werde ich einige Distributionen, die ich im Laufe der Zeit ausprobiert habe, kurz vorstellen und die jeweiligen Vor- und Nachteile aufzeigen. Diese Liste kann nicht vollständig sein, da ich nicht jede Distribution zur Verfügung habe und ein vernünftiger Test mit hohem Zeitaufwand verbunden ist. Auch sind viele Erkenntnisse mit einer neuen Version der entsprechenden Distribution wertlos geworden.

Zu jeder Distribution gehört normalerweise eine entsprechende Installationsanleitung. Diese kann in Form einer `README`-Datei, eines Booklets zur CD oder eines Handbuchs vorliegen. Dort wird in der Regel die Installation des Systems sehr detailliert beschrieben, so dass ich hier auf eine derartige Beschreibung verzichte. Vielmehr werde ich einige Unterschiede zwischen den Distributionen aufzeigen. Dabei werden sich je nach den Ansprüchen des Anwenders unterschiedliche Distributionen als besonders geeignet erweisen.

Aufgrund der GNU General Public License (GPL), die für viele der unter Linux verwendeten Programme und den Kernel gilt, ist jeder Distributor verpflichtet, entweder den Quellcode der entsprechenden Programme mitzuliefern oder auf Anfrage nachzureichen. Zu vielen Distributionen findet man daher auch die Quellen der Programme auf CD. Bei Problemen kann man dann einzelne Programme neu übersetzen oder nachsehen, mit welchen Optionen diese Programme installiert wurden. Gerade in diesem Bereich findet man deutliche Unterschiede zwischen den Distributionen.

Ein weiterer Unterschied besteht im Support durch die Entwickler. Manche CDs werden ohne Support angeboten, andere enthalten einen Anspruch auf Support (via E-Mail, Fax oder Telefon) durch den Distributor. Oft wird vom Hersteller auch eine Mailing-Liste betrieben, in der speziell für die eigene Distribution Support geleistet wird. Genauere Informationen dazu finden Sie in der Dokumentation zu Ihrer Distribution.

Um zu große Unterschiede zwischen den Distributionen zu vermeiden, wurde die Linux Standard Base (siehe Abschnitt 2.2, »Linux Standard Base«) entwickelt. Basierend auf dem Filesystem Hierarchie Standard werden nun auch Bibliotheken, Programmierschnittstellen und Programme definiert. Die meisten der hier vorgestellten Distributionen halten sich weitgehend an diese Standards, weichen aber in Einzelfällen auch davon ab. Diese Abweichungen sind normalerweise im Handbuch der Distribution dokumentiert.

Weitere Informationen zu den hier vorgestellten und anderen Distributionen, wie z. B. die Adressen der Distributoren, finden Sie in der *Distributions-HowTo*. Zu praktisch jedem vorstellbaren Thema unter Linux existiert eine `HowTo`-Datei, die in dieses Thema einführt und stellenweise die Funktion einer *FAQ-Liste* (Frequently Asked Questions) übernimmt. Außerdem arbeitet eine Gruppe von Freiwilligen am Linux Dokumentation Project (LDP, *http://www.tldp.org/*), das Manpages und Handbücher schreibt. Diese Bücher sind von guter Qualität und teilweise auch im Buchhandel erhältlich. Eine Übersicht über die bereits verfügbaren Titel finden Sie in Anhang C, »Literaturverzeichnis«.

Bevor wir zur Vorstellung einiger verbreiteter Distributionen kommen, möchte ich verschiedene Anforderungen an eine Distribution genauer erläutern. Dabei geht es sowohl um Installationsroutinen und das System an sich als auch um technische Hintergründe.

Keine der verfügbaren Distributionen erfüllt bisher alle hier gestellten Anforderungen. Dennoch lernen alle Distributoren im Laufe der Zeit dazu und implementieren gelungene Funktionen der anderen Distributionen. Dabei erweist sich der Konkurrenzdruck durch die vielen anderen (guten) Distributionen als wichtiger Antrieb für die Entwickler.

- Die Installationsroutine sollte ohne oder mit nur einer Diskette auskommen, wenn von CD, Festplatte oder Netz installiert wird. Es ist für Einsteiger oder PC-Benutzer, deren PC noch nicht mit einem Betriebssystem ausgestattet ist, sehr hilfreich, wenn eine solche Diskette bereits beiliegt. Dies setzt voraus, dass auch nur eine Diskette benötigt wird, also keine Auswahl aus vielen vorkonfigurierten Boot-Disketten getroffen werden muss.
- Das Installationsprogramm sollte ein integriertes Hilfesystem enthalten. Damit ist es möglich, während der Installation zusätzliche Informationen anzuzeigen und den Benutzer zu unterstützen.
- Bei Eingabefehlern sollte man einzelne Einstellungen erneut vornehmen können, ohne neu beginnen zu müssen. Das sollte für möglichst viele Eingaben des Benutzers gelten.
- Die Paketauswahl sollte vor, nicht während der Installation stattfinden. Dabei soll dann auch der zur Verfügung stehende Platz geprüft werden. Damit ist es möglich, den zeitaufwendigsten Teil ohne Benutzerinteraktion ablaufen zu lassen.

- Die Distribution sollte verschiedene »Standardinstallationen« (z. B. mit/ohne X, mit/ohne TeX) bereits zur Auswahl anbieten. Diese Vorauswahlen können für einen Neu-Einsteiger sehr hilfreich sein, da die Masse an Programmen sehr abschreckend sein kann.
- Einzelne Pakete sollten später gelöscht oder hinzugefügt werden können. Dabei sind Abhängigkeiten der Pakete untereinander zu beachten, da viele Pakete auf anderen aufbauen oder sich gegenseitig ausschließen.
- Ein leistungsfähiges Tool zur Systemadministration kann den Einstieg vereinfachen. Nicht jeder Anwender hat Interesse daran, sich die passenden Konfigurationsdateien zusammenzusuchen und mit einem Texteditor anzupassen. Gerade Windows zeigt hier, wie einfach die Systemkonfiguration mit Hilfe von grafischen Tools sein kann.
- Viele Anwendungspakete für Mathematik, Grafik, Textverarbeitung usw. sollten in der Distribution enthalten sein. Dabei ist eine Trennung in verschiedene Anwendungsgruppen oder Serien sinnvoll.
- Bei der Installation muss eine gute Dokumentation (Beschreibung) der Pakete vorliegen, die auch für Einsteiger verständlich ist. Nach der Installation sollte eine möglichst vollständige Dokumentation (README-Dateien, Handbücher oder Beispieldateien) der Pakete auf der Festplatte zu finden sein.
- Der gesamte Quellcode der Programme sollte verfügbar und auf Knopfdruck installierbar sein.
- Die Programme sollten mit einem Befehl genauso neu übersetzt werden können, wie sie für die Distribution erstellt wurden.
- Die Original-Sourcen des Programms (z. B. für die Verwendung unter anderen Systemen) und die Anpassungen für die Distribution sollten getrennt vorliegen.
- Die Distribution sollte der Linux Standard Base entsprechen. Sind einzelne Dinge dort nicht festgelegt, so sollte man sich von den Prinzipien der Übersichtlichkeit und der geringsten Überraschung für den Anwender leiten lassen. Die Abweichungen zum Filesystem-Hierarchie-Standard (FHS) sollten dokumentiert werden.
- Das System sollte auf einem modularisierten Kernel basieren, damit zunächst mit dem mitgelieferten Kernel weitergearbeitet werden kann.
- Der Distributor sollte sich in der Linux-Gemeinde durch Mitarbeit an und Unterstützung von Projekten engagieren. Dazu gehört zum Beispiel die intensive Zusammenarbeit mit den Programmierern, um Fehler zu beheben.

Auch diese Liste erhebt keinen Anspruch auf Vollständigkeit. Sie zeigt jedoch das weite Spektrum an Anforderungen, die an die Distributionen gestellt werden. Der Aufwand für die Erstellung und Wartung einer Distribution ist recht hoch, so dass der Markt in dieser Hinsicht hoffentlich nicht weiter wächst.

1.3.1 Debian-(GNU/Linux)-Distribution

Die Debian GNU/Linux-Distribution (*http://www.debian.org/*) wurde zunächst von Ian Murdock entwickelt. Im Laufe der Zeit haben sich viele Entwickler ebenfalls für Debian engagiert. Dadurch wurde diese Distribution zu einem weltweiten Projekt, das eine zeitlang auch von der Free Software Foundation (FSF) unterstützt wurde. Die FSF betrachtete diese Distribution als »Testfall« für das eigene Betriebssystem Hurd.

Das wesentliche Merkmal ist, dass diese Distribution frei im Sinne der GPL ist. Sie kann im Rahmen der GPL beliebig kopiert und kommerziell eingesetzt werden, auch ohne dass eine entsprechende CD oder ähnliches erworben wurde. Viele Entwickler haben die Copyright-Bedingungen ihrer Programme geändert, damit sie problemlos in dieser Distribution verwendet werden können.

Programme, deren Lizenz nicht den »Debian Free Software Guidelines« entspricht, können nicht Bestandteil von Debian werden. Dazu hat sich das Projekt in seinem »Social Contract« verpflichtet – damit wird die Distribution immer frei (im Sinne einer Weitergabe ohne Restriktion) verfügbar sein. Obwohl diese »unfreien« Programme nicht Teil von Debian sind, können diese auf Debian-Systemen benutzt werden. Im `contrib` oder `non-free` Bereich werden viele Programme im Debian-Paketformat bereitgestellt. Auch das Bugtracking-System wird hier zum Support verwendet. Durch diese strikte Trennung wird sichergestellt, dass jedermann die Debian-Distribution benutzen und verteilen kann.

Zur Installation und für Updates von Programmpaketen wurden ein eigenes Paketformat (`.deb`) und ein eigenes Tool (`dpkg`) entwickelt. Ein wesentlicher Vorteil gegenüber anderen Distributionen ist, dass das System oder einzelne Programme problemlos auf den aktuellen Stand gebracht werden können, ohne dass die Festplatte formatiert und die Software komplett neu installiert werden muss. Die hierzu entwickelten Strategien haben in neuester Zeit auch Einzug in andere Distributionen (z. B. S.u.S.E. und Red Hat) gehalten. Als Frontend zu `dpkg`, das nur mit einzelnen Paketen umgeht, stehen `dselect` und `apt` zur Verfügung.

Ein weiterer Vorteil ist das konsequente Bugtracking, das über Mailing-Listen von den Entwicklern durchgeführt wird. Die Liste der noch offenen Fehlermeldungen ist öffentlich und wird regelmäßig nach zu lange unbearbeiteten Fehlern durchsucht. Dadurch zählt diese Distribution zu den stabilsten und fehlerfreiesten. Die Entwickler legen ebenfalls großen Wert darauf, dass die Debian-Distribution konform zum Filesystem-Hierarchie-Standard ist.

Alle Quellen sind, inklusive der notwendigen Patches, Bestandteil der Distribution. Damit ist es auch möglich, das System komplett selbst neu zu übersetzen. Die Debian-Policy enthält genaue Vorschriften, wie Debian-Pakete zu erstellen sind, das Programm `lintian` prüft Pakete auf die Einhaltung der Policy-Vor-

gaben. Damit kann sich ein Anwender in der Regel auf die erstellten Pakete verlassen – andere Distributionen bieten dieses nicht.

Außerdem gehört zur Distribution ein ausführliches Installationshandbuch. Das Debian-Projekt bietet, außer dem Bugtracking und verschiedenen Mailinglisten, keinen Support an. Dieser kann jedoch von verschiedenen Dienstleistern bezogen werden.

Die Debian-Distribution ist derzeit mein persönlicher Favorit – stabil, gut gepflegt und auch bei Upgrades sehr verlässlich. Neben der stabilen organisatorischen Basis, wie der Debian-Policy, ist die Unterstützung vieler verschiedener Hardware-Architekturen erwähnenswert. Hier ist Debian den anderen Distributionen einen großen Schritt voraus.

1.3.2 Mandrake Linux

Im Vergleich zu den anderen hier vorgestellten Distributionen ist Mandrake (*http://www.mandrake.com/*) relativ neu. Begonnen wurde mit der Red Hat-Distribution, die um KDE-Pakete erweitert wurde. Im Laufe der Zeit hat sich eine vollständige, auch für Einsteiger benutzbare Distribution entwickelt.

1.3.3 Die Red Hat-Distribution

Eine weitere kommerzielle Distribution ist die Red Hat-Distribution. Red Hat verwendet das `rpm`-Format für Pakete und hat zu deren Verwaltung ein eigenes Programm (`rpm` Red Hat Package Management) entwickelt. Das macht die Verwendung der entsprechenden Pakete auf anderen Linux-Systemen eher schwerer, allerdings hat sich dieses Format als Standard durchgesetzt. Mehr Informationen zum `rpm`-Format finden Sie unter *http://www.rpm.org*.

1.3.4 Die Slackware-Distribution

Im Internet wird die Slackware-Distribution per `ftp` verteilt. Sie können sie aber auch auf CD erwerben. Die Slackware war einmal etwas wie die »Standarddistribution« und weltweit sehr oft installiert. Slackware gilt immer noch als eine Art »Geheimtipp« für Bastler, im Vergleich zu den anderen Distributionen wird sie aber scheinbar nicht mehr so häufig eingesetzt.

1.3.5 Die S.u.S.E.-Distribution

Die S.u.S.E. GmbH (*http://www.suse.de/*) hat zunächst damit begonnen, die internationale Slackware-Distribution für den deutschen Markt anzupassen. Vor einiger Zeit haben sich die Entwickler aber von Slackware gelöst und eine eigenstän-

dige Distribution entwickelt. Diese basierte zunächst auf dem Slackware-Paketformat (`tar.gz`), verwendet jetzt aber das von Red Hat entwickelte Format. Dadurch hat sich die technische Seite in der Distribution wesentlich verbessert.

Die SuSE-Distribution enthält Programmbeschreibungen in deutscher Sprache und ein gutes Installationsprogramm (`yast`), das auch zur Systemverwaltung verwendet werden kann. Alle Einstellungen wurden bisher zentral in der Datei `/etc/rc.config` gespeichert und von dort aus maschinell in die entsprechenden Konfigurationsdateien konvertiert. Für viele »alte Hasen« war dieses Verfahren ungewohnt, bei vielen Einsteigern ist es aber genauso beliebt. Heute speichert SuSE die Konfiguration LSB-konform unter `/etc/sysconfig`.

Weiterhin sind viele Programme für deutschsprachige Benutzer vorkonfiguriert, so dass man in der Regel sofort vernünftig mit dem System arbeiten kann. Erwähnenswert sind auch das ausführliche Installationshandbuch und die gute Ausstattung an Paketen.

Die Installation ist einfach und komfortabel. Wenn Sie die Einsteiger-Distribution erwerben, so leistet der Distributor Support. Die Distribution ist wohl in Deutschland die am weitesten verbreitete, es wird jetzt auch aktiv eine amerikanische Version vermarktet.

Neben der eigentlichen Distribution werden nun auch Pakete zum Einsatz als Mail-Server oder Firewall angeboten. Außerdem bietet S.u.S.E kommerziellen Support und Beratung an. Neben Red Hat ist S.u.S.E. die einzige Distribution, die für den Einsatz von SAP R/3 freigegeben ist.

1.3.6 Caldera Open Linux

Basierend auf der LST-Distribution präsentiert Caldera (*http://www.caldera.com/*) sein Caldera Open Linux. Diese Distribution enthält je nach Version zusätzlich zu den bekannten freien Programmen eine Reihe von kommerziellen Paketen.

Caldera will für andere Software-Unternehmen eine stabile Linux-Basis liefern. Diese Basis soll andere Unternehmen dazu animieren, ihre Programme nach Linux zu portieren. Caldera will damit Linux auch in kommerziellen Umgebungen einsatzfähig machen, dazu dient auch das Support-Angebot.

Vor einiger Zeit hat Caldera SCO übernommen und damit auch die Rechte am originalen Unix-Quellcode erworben. Im Laufe der Zeit hofft man, einige Teile des Quellcodes veröffentlichen zu können – für die anderen Linux-Distributionen ist das aber eher uninteressant. Da Caldera sowohl Linux als auch OpenUNIX vertreibt, ist es schwer, beide Produkte voneinander abzugrenzen, insbesondere da OpenUNIX eine »Linux Kernel Personality« anbietet, mit der auch Linux-Programme ausgeführt werden können.

1.3.7 Andere Distributionen

Neben den hier erwähnten Distributionen gibt es noch viele andere. Eine recht vollständige Liste finden Sie unter *http://lwn.net/Distributions*. Besonders erwähnenswert finde ich dabei folgende Distributionen:

- Tombsrt, erhältlich unter *http://www.toms.net/rb/* passt auf eine Diskette und stellt zum Beispiel einen leistungsfähigen Router bereit.
- Fli4l (*http://www.fli4l.de/*) ist ein Floppy-Router für ISDN- oder DSL-Zugänge.
- Eigentlich keine Distribution ist »Linux From Scratch«, sondern eher eine Anleitung, wie man ein Linux-System selbst aufbauen kann. Sehr lehrreich, aber ziemlich zeitaufwendig.

Eine zentrale Anlaufstelle für den Download der verschiedenen Distributionen ist *http://www.linuxiso.org/*. Dort erhalten Sie CD-Images für praktisch alle verbreiteten Distributionen. Wenn Sie also keine Schachtel eines Distributors erworben haben, aber dennoch eine CD benötigen – hier finden Sie die Zutaten dazu.

1.4 Die Zukunft von Linux

Linux hat sich in wenigen Jahren von einem Hacker-System zu einem leistungsfähigen Unix-System gewandelt, das auch kommerziellen Unix-Systemen in vielen Dingen ebenbürtig ist. Aber wie wird Linux sich weiterentwickeln? Diese Frage ist praktisch nicht zu beantworten, da die Antwort von der Zeit und den persönlichen Interessen der Entwickler abhängt. Dennoch kann man heute einige Dinge bereits absehen:

- Neben NetBSD ist Linux wohl auf die größte Anzahl Hardware-Plattformen portiert worden. In der Zukunft wird für viele Systeme ein Linux-Port zur Verfügung stehen, bevor andere Systeme portiert werden – so wie es beim Itanium-Prozessor war.
- Auch für diese Architekturen wird es freie und kommerzielle Distributionen wie Debian und Red Hat geben. Einige kommerzielle Produkte wie Motif wurden bereits portiert, es wird aber auf absehbare Zeit bei vielen Produkten bei einem Linux/Intel-Port bleiben. Hier ist freie Software klar im Vorteil – jeder kann diese auf jede Plattform portieren (oft auch einfach nur kompilieren).
- Der Linux-Kernel unterstützt symmetrisches Multiprocessing (SMP). Ein weitgehend abgeschlossenes Projekt ist die Einführung von »Fine grained Locking«, damit möglichst viele Prozessoren gut genutzt werden können. Dennoch wird hier noch viel Arbeit notwendig sein, um auf viele (mehr als 30) Prozessoren skalierbar zu sein.

- Die Entwickler werden weiter versuchen, die aktuellen Standards, wie z. B. POSIX oder ANSI-C, zu implementieren. Einige Linux-Entwickler sind in den Standardisierungsgremien vertreten. Außerdem ist die Dynamik hier besonders groß, so dass Linux in vielen Dingen selbst Standards setzen kann.
- Die Installation wird durch die Konkurrenz der verschiedenen Distributionen noch einfacher werden. Möglicherweise setzt sich dieser Trend auch in den Anwendungen fort.
- Mit dem Erscheinen von mehr kommerzieller Software wird Linux noch mehr Einsatzgebiete für sich erschließen können. Trotzdem wird es ebenfalls immer mehr freie Software geben.
- Für den kommerziellen Einsatz ist oft ein guter technischer Support notwendig. Dieser wird von verschiedenen Firmen angeboten.
- Linux wird mehr User gewinnen, mehr Programme werden verfügbar sein und es wird mehr Hardware unterstützt werden.
- Linux wird durch gute Desktop-Oberflächen auch auf die Schreibtische unerfahrener Anwender vordringen können, aber durch Projekte wie Linux-HA und viele andere Verbesserungen auch im Server-Umfeld weiter an Einfluss gewinnen.
- Aktuelle Netzwerk-Protokolle wie IPv6, IPsec oder CIFS werden implementiert. Für viele Forschungsprojekte wird heute Linux verwendet – und viele der Ergebnisse werden veröffentlicht werden.

Linux hat die Welt des »Personal Computing« bereits verändert, es ist zu einer akzeptierten Alternative zu kommerziellen Betriebssystemen geworden. Das kann man an verschiedenen Dingen feststellen:

- SCO verschenkte sein Betriebssystem an Privatanwender.
- Sun ermöglicht allen Forschungseinrichtungen einen sehr preiswerten Zugang zum Quellcode von Solaris – allerdings nicht zu denselben Bedingungen, wie es bei Linux der Fall ist.
- Selbst Microsoft produziert Linux-Software und hält Linux für die zurzeit größte Gefahr für das Unternehmen.
- Viele Hard- und Software-Hersteller betrachten Linux bereits als Massenmarkt.
- Immer mehr »Nur-Anwender« zeigen Interesse an dem einstigen »Hacker-System«.
- Firmen wie IBM erklären Linux zu einem zentralen Bestandteil ihrer Strategie – und arbeiten aktiv an Verbesserungen.

Trotz all der Erfolge drohen der weiteren Entwicklung von Linux auch Gefahren. Einige werden immer mal wieder unter den Entwicklern und Anwendern diskutiert, manche betreffen Linux nur indirekt:

- Ist ein derartig komplexes Projekt überhaupt auf Dauer machbar? Insbesondere weil viele Entwickler nicht bei Linux-Firmen angestellt sind, sondern in ihrer freien Zeit daran arbeiten, kommt diese Befürchtung immer wieder auf. Dieses Problem kann die Entwicklung zwar verzögern, aber wenn der Leidensdruck zu groß wird, findet sich ein weiterer Entwickler, der aushilft. Problematisch ist dies aber bei Projekten wie Debian – hier ist es sehr schwierig, die nächste Version der Distribution im »Freeze« von den bekannten Fehlern zu befreien. Dazu ist ein erheblicher, koordinierter Zeitaufwand erforderlich, der die Grenzen der Entwickler aufzeigt.
- Open Source macht es einfach, die Entwicklung zu »forken« – also basierend auf einem Programm eine zweite Entwicklungslinie zu eröffnen. Ein Beispiel dafür ist z. B. Emacs und XEmacs. Da aber beide Entwicklungen öffentlich sind und Entwickler recht genau wissen, wie viel Arbeit ein Fork ist, kommt es nur sehr selten dazu, dass die Entwicklung dauerhaft parallel stattfindet. Viele Features werden ausgetauscht, manchmal die Entwicklungen wieder zusammengeführt oder ein Zweig wieder aufgegeben. Ja, ich halte diese Möglichkeiten für einen großen Vorteil.
- Ein großes Problem können Software-Patente werden. Für einen Entwickler ist es praktisch unmöglich, eine Patent-Recherche zu machen – jeder kann aber den Quellcode untersuchen, ob z. B. sein Patent dort verletzt wird. Aus meiner Sicht werden Software-Patente nicht zum Schutz des geistigen Eigentums eingesetzt, sondern von Firmen als Waffen im Konkurrenzkampf gesehen und eingesetzt.
- Freie Software hat, seit sie auch außerhalb der Entwicklerkreise bekannt geworden ist, auch Probleme mit eingetragenen Markenrechten. Auch hier sind Entwickler freier Software stark benachteiligt, insbesondere wenn die Software weltweit verteilt wird – eine Recherche ist dann sehr aufwändig und damit für Privatleute praktisch unmöglich. Es bleibt zu hoffen, dass sich die bisherigen Vorfälle im Rahmen des Markenrechts nicht häufen, sonst könnten möglicherweise die Distributoren gezwungen sein, ihre Arbeit einzustellen oder in der Aufnahme von Software in die Distribution deutlich vorsichtiger zu sein.
- In welcher Art die Diskussion um geistiges Eigentum, Kopierschutz und digitale Rechte auch freie Software betrifft, ist derzeit nicht absehbar. Sollte es dazu kommen, dass jede Software einen Kopierschutz enthalten muss, dann kann der Einsatz von freier Software schnell illegal sein – trotz aller sonst damit verbundenen Vorteile.

Wir leben in interessanten Zeiten – mal sehen, was die Zukunft bringt. Wenn Sie an der Linux-Entwicklung der letzten Jahre interessiert sind, dann schauen Sie sich einmal die Linux-Timeline unter *http://lwn.net/2001/features/Timeline/* an. Dort finden Sie auch Verweise auf die Timelines älterer Jahre.

2 Linux-Standards

The nice thing about standards is that you have so many to choose from; furthermore, if you do not like any of them, you can just wait for next year's model.

Andrew S. Tanenbaum

2.1 Der Turmbau zu Babel

Einer der wichtigsten Aspekte bei der Linux-Entwicklung (neben dem Spaß, den man hat) ist die Auswahl aus verschiedenen Programmen, Implementationen und Distributionen. Für viele scheint die Chance eher gefährlich, die sich mit der Wahlmöglichkeit bietet. Firmen beispielsweise werden intern nicht Dutzende von verschiedenen Window-Managern supporten, sondern nur eine oder sehr wenige Umgebungen. Aber in der Auswahl, welche Umgebung dies ist, ist man frei.

Linux wurde von vielen verschiedenen Entwicklern und Distributoren implementiert, jeder hat eigene Vorstellungen, wie das System aussehen soll. Das hat am Ende die Folge, dass die verschiedenen Versionen divergieren und sich immer deutlicher unterscheiden. Die Distributoren wollen auf der einen Seite Alleinstellungsmerkmale haben, um Kunden zu finden und zu halten, auf der anderen Seite arbeiten sie mit allen anderen gemeinsam am Markt. Hier einen sinnvollen Mittelweg zu finden ist, nicht einfach.

Eine Zersplitterung des Linux-Marktes ist sowohl für Linux-Anwender als auch Software-Entwickler sicher nicht wünschenswert. Entwickler möchten ein einheitliches System, um Software möglichst nur einmal »auf Linux« zu portieren – und nicht für jede Distribution einzeln. Anwender möchten jedes Programm, das »für Linux« verfügbar ist, auch auf der eigenen Installation einsetzen können.

Eine ähnliche Entwicklung gab es vor vielen Jahren schon: Unix ist ein Beispiel, wie man es besser nicht machen sollte. Jeder Unix-Anbieter wollte sich mit eigenen, proprietären Erweiterungen von den anderen abheben. Im Grund nichts Verdammenswertes, aber aufgrund der Lizenzbestimmungen konnte jeder Anbieter seine Erweiterungen für sich behalten. Damit wurden Kunden auf eine Plattform festgelegt und, was noch viel schlimmer ist, der Software-Markt wurde zersplittert.

Die GNU General Public License, unter der viele der unter Linux verwendeten Programme stehen, verlangt, dass ein Anbieter der entsprechenden Programme den Quellcode (mindestens auf Anfrage) weitergeben muss. Damit ist sichergestellt, dass auf Dauer alle Erweiterungen in den gemeinsamen Pool zurückfließen. Für Anwender hat das den Vorteil, dass sie nicht auf einen Anbieter festgelegt sind. Entwickler hingegen benötigen eine stabile, möglichst einheitliche Basis für ihre Programme.

Aus dieser Erkenntnis heraus wurde zunächst der Filesystem-Hierarchie-Standard (FHS, *http://www.pathname.com/fhs/*) entwickelt. Dieser hat geholfen, eine Zersplitterung zu vermeiden, wie sie bei Unix stattgefunden hat. Dennoch hat sich im Laufe der Zeit gezeigt, dass der FHS zu viele Punkt offen lässt. Aus dieser Erkenntnis hat sich das Projekt Linux Standard Base (*http://www.linuxbase.org/*) entwickelt, das wesentlich genauer festlegen will, was ein Linux-System eigentlich ausmacht. Die Free Standards Group (*http://www.freestandards.org/*) bietet dem FHS und der LSB ein gemeinsames Dach.

2.2 Linux Standard Base

In den letzten Jahren haben sich praktisch alle Distributionen an den Filesystem-Hierarchie-Standard gehalten – der Erfolg gibt den Machern Recht. Dennoch hat sich gezeigt, dass bei weitem nicht alles festgelegt war, was für Anwender, Entwickler und Lieferanten von (kommerzieller bzw. vorkompilierter) Software wünschenswert ist. Ausgehend von den verschiedensten Anforderungen hat sich eine große Gruppe von Entwicklern und Distributoren zusammengefunden, die an einem viel ausführlicheren Standard arbeitet.

Einige Problemfelder, die in den letzten Jahren offensichtlich geworden sind:

- Die Zuordnung von Benutzern und Gruppen zu deren Nummern. Sind diese überhaupt angelegt? Wie werden Benutzer bei der Installation von Software neu angelegt, wenn es denn notwendig ist?
- Welche Funktionen sind in den Systembibliotheken enthalten? Welche Bibliotheken (in welcher Version) darf man voraussetzen?
- Welche Kommandos stehen für portable Skripte zur Verfügung?
- Welche Fonts sind verfügbar und wie zugreifbar?
- In welchem Format können Fremdhersteller Software ausliefern?
- Und viele Dinge, die man bisher immer als gegeben hingenommen hatte, werden als Anforderung dokumentiert.

Der LSB besteht aus einem allgemeinen Teil und verschiedenen architekturabhängigen Teilen. Außerdem wird vielfach Bezug auf andere Standards genommen (Filesystem-Hierarchie-Standard, X, Single Unix Specification, System V ABI etc.). Damit ist ein LSB-konformes Linux-System in weiten Teilen kompatibel zu anderen Unix-Systemen.

Zusätzlich zum LSB werden Testprogramme entwickelt, die die Einhaltung des Standards prüfen. Für den FHS existiert bereits eine entsprechende Testsuite. Für weitere Teile der Linux Standard Base kann die POSIX-Testsuite verwendet werden, andere Bereiche sind neu implementiert worden. Die Hoffnung ist, dass

ISVs (Independent Software Vendors) einfacher Software auf Linux portieren können und der Austausch zwischen den Distributionen einfacher wird.

Viele Teile des LSB sind nur für Entwickler interessant, wir werden hier den für Anwender und Systemverwalter relevanten Teil betrachten.

2.2.1 LSB-Paketformat

Es wird das unter *http://www.rpm.org/* beschriebene RPM-Format verwendet. Dabei werden nicht alle RPM-Funktionen unterstützt, so dass man auch mit einer älteren Programmversion von `rpm` LSB-Pakete installieren kann. Dieses Format gilt nicht für die Installation distributionseigener Pakete, so dass zum Beispiel Debian weiterhin das `.deb`-Format verwenden kann.

2.2.2 Kommandos

Es werden eine Reihe von Kommandos genauer spezifiziert. Im Wesentlichen gelten der POSIX-Standard bzw. die Single Unix Specification, diese werden eventuell präzisiert. Als Anwender werden Sie nur selten in die Verlegenheit kommen, hier nachsehen zu müssen. Distributoren und ISVs können sich in Zukunft (zum Beispiel in Skripten) auf die Funktionalität verlassen – ein großer Gewinn.

2.2.3 Bibliotheken

Für Entwickler interessant sind die Namen und der Inhalt der verwendeten Bibliotheken. Der Name korrespondiert mit dem Dateinamen, unter dem die Bibliothek zu finden ist. Die nächste Frage ist dann, ob die benötigten Funktionen in der Bibliothek enthalten sind. Damit kann ein Entwickler sich darauf verlassen, dass sein Programm auf einem LSB-konformen System laufen wird. Spezielle Bibliotheken werden unter `/usr/lib/lsb` installiert, die nur die LSB-spezifizierten Funktionen enthalten. Zusätzlich soll ein Skript `lsbcc` implementiert werden, das dem Entwickler einige Kompiler-Optionen erspart.

Derzeit werden nur Schnittstellen zu stabilen und frei verfügbaren Bibliotheken festgelegt und keine C++-Bibliotheken wie Qt oder die KDE-Bibliotheken standardisiert. Noch ist das Application Binary Interface (ABI) zwischen den verschiedenen GCC-Versionen nicht stabil, so dass absehbar ist, dass ein Standard keinen Bestand haben kann.

Auch hier wieder bezieht man sich auf verschiedene, bereits etablierte Standards. Damit erspart man sich eine Menge Arbeit und ist in vielen Punkten kompatibel zu anderen Unix-Systemen.

2.2.4 Systemadministration

Außerdem werden einige Standardbenutzer und -gruppen vorgegeben, die auf jedem System vorhanden sein sollten. Diese werden in der Regel durch die Distribution mitgebracht.

Die Benutzernummern von 0 bis 99 sind statisch durch den Distributor festgelegt. Der Bereich von 100 bis 499 ist für automatisch angelegte Benutzer (Skripte des Systemverwalters oder Installationsskripte) reserviert. Der Systemverwalter kann die anderen Nummern frei vergeben.

Eine Anwendung sollte keine besonderen Schreibrechte außerhalb des Home-Verzeichnisses, `/tmp`, `/var/tmp` oder `/var/opt/`*`anwendung`* verlangen und die Rechte von anderen Anwendungen oder Dateien nicht ändern.

Anwendungen sollten möglichst keine `setuid`-Programme benötigen, genauso sollte dokumentiert und begründet werden, wenn spezielle Privilegien benötigt werden. Das soll es dem Systemverwalter ermöglichen, die Risiken zu bewerten und geeignete Maßnahmen zu ergreifen.

Bei der Installation werden häufig Programme oder Skripte als `root` ausgeführt. Diese sollten im Quellcode vorliegen, um ein Audit zu ermöglichen. Andernfalls muss man sich auf die Kompetenz des Software-Lieferanten in Sachen Sicherheit verlassen – und häufig ist man dann verlassen.

Weiterhin werden Hinweise zu `cron`-Jobs und `init`-Skripten gegeben. Diese Hinweise werden in den betreffenden Kapitel wieder auftauchen.

2.3 Der Filesystem-Hierarchie-Standard

Der Filesystem-Hierarchie-Standard (FHS) ist ein Dokument, das die Verzeichnisstruktur und die Platzierung einzelner Dateien unter Unix-artigen Betriebssystemen beschreibt. Der FHS hat sich aus dem Linux-Filesystem-Standard entwickelt, der für eine Vereinheitlichung der Linux-Distributionen sorgen wollte.

Dieses Kapitel soll ein kleiner Wegweiser auf dem Weg durch den Dschungel der vielen Dateien und Unterverzeichnisse eines Linux-Systems sein. Es geht jedoch darüber hinaus und versucht, die Hintergründe dieses Standards sichtbar zu machen. Dadurch soll ein Systemadministrator in der Lage sein zu entscheiden, wie er ein Paket installieren kann, das (noch) nicht dem Standard entspricht. Viele Gründe, die zur Erstellung des Standards in dieser Form führten, werden hier näher erläutert. Der Standard selbst enthält diese Begründungen oft nicht mehr, da sie als »allgemein anerkannte Regeln der Systemadministration« angesehen werden.

Zunächst erscheint der Verzeichnisbaum eines Linux-Systems komplex und unübersichtlich. Er ist jedoch gut strukturiert. Der Standard soll diese Struktur festschreiben und erläutern. Das Verständnis dieses Standards hilft weiter, wenn eine spezielle Konfigurations- oder Datendatei gesucht wird.

Programme wie `find` oder `locate` (Abschnitt 9.2.2, »Suchen mit `locate`«) helfen dabei, einzelne Dateien aufzuspüren. Um einen schnellen Überblick zu gewinnen, kann z. B. ein Datei-Manager wie der Norton Commander-Clone `mc` (Midnight Commander) oder die GNU-Interactive Tools `git` sowie der `dired`-Mode des Editors Emacs (siehe auch Abschnitt 6.13.3, »Der `dired-mode`«) nützlich sein.

2.3.1 Problembereiche

Der Filesystem-Hierarchie-Standard versucht nicht nur, die Dateisystemstruktur der einzelnen Distributionen zu vereinheitlichen, sondern möchte auch eine Reihe von Problemen bei der Verwaltung von Linux-Systemen lösen, die im Laufe der Zeit deutlich wurden.

Zwischen den Verzeichnissen `/bin` und `/usr/bin`, in denen die meisten Programmdateien gespeichert sind, wurde nicht deutlich unterschieden, so dass man viele Programme, je nach Distribution, mal in dem einen, mal in dem anderen Verzeichnis fand. Außerdem war das `/bin`-Verzeichnis oft mit vielen Programmen gefüllt, die nicht unbedingt dort gespeichert werden müssen. Andere Unix-Systeme wie Solaris speichern alle Programme in `/usr/bin`, in `/bin` finden sich nur symbolische Links zur Kompatibilität mit anderen Systemen bzw. Skripten.

Das Verzeichnis `/etc` enthielt sowohl Konfigurationsdateien als auch Programme zur Systemadministration und Netzwerkkonfiguration. Dadurch wurde dieses Verzeichnis, besonders auf vernetzten Rechnern, sehr groß und unübersichtlich.

Die `/usr`-Hierarchie konnte in den gängigen Distributionen nicht schreibgeschützt werden. Damit war es nicht möglich, in einem Netzwerk das `/usr`-Dateisystem schreibgeschützt via NFS zur Verfügung zu stellen. Dadurch würde ein Netzwerk insgesamt verlässlicher und der File-Server muss nicht Update-Operationen verschiedener Rechner bearbeiten. Auch die Verwendung eines Filesystems auf einer CD war nur mit großem Aufwand möglich. Insbesondere hier wurden zur Lösung der Probleme oft symbolische Links verwendet. Das führt jedoch sehr schnell dazu, dass man die Übersicht durch so genannte Link-Farmen (Verzeichnisse, die viele Links enthalten) verliert.

Es gab keine Unterscheidung zwischen Daten, die maschinenbezogen gespeichert werden müssen, und solchen, die von verschiedenen Rechnern gemeinsam benutzt werden können. Insbesondere bei Dokumentationen, Manpages und großen Paketen wie X11 oder TeX kann in einem Netzwerk viel Plattenplatz gespart werden.

Es wurde nicht zwischen maschinenbezogener Konfiguration und netzweiter Konfiguration unterschieden, so dass die Wartung eines großen Linux-Netzwerks recht kompliziert war. So ist z. B. die Konfiguration des Kernels typischerweise maschinenabhängig, die Konfiguration des `man`-Programms kann jedoch für alle Rechner gleich sein.

Daten können statisch sein, d. h. nach der Installation unverändert bleiben, andere Daten sind variabel (verändern sich ständig). Statische Daten können problemlos auf einem schreibgeschützten Medium gespeichert werden, variable Daten nicht. Eine tabellarische Übersicht mit einigen Verzeichnissen finden Sie in Tabelle 2.1.

	gemeinsam verwendbar	**nicht gemeinsam verwendbar**
Statische Daten	`/usr`	`/etc`
	`/opt`	`/boot`
Variable Daten	`/var/mail`	`/var/run`
	`/var/spool/news`	`/var/lock`

Tabelle 2.1 Unterteilung statischer und variabler Daten nach Verwendbarkeit

2.4 Das root- oder /-Dateisystem

Ein Unix-Dateisystem ist hierarchisch organisiert. Dabei geht man vom Wurzel-Verzeichnis (auch Stamm- oder Root-Verzeichnis genannt) aus. Dort findet man, wie man es z. B. von MS-DOS kennt, eine Reihe von Unterverzeichnissen, die selbst wieder Dateien und weitere Unterverzeichnisse enthalten können. Unter Unix werden jedoch keine Laufwerksbuchstaben verwendet, sondern alle Partitionen in den Verzeichnisbaum eingehängt (»gemountet«). Ein Benutzer muss nicht wissen, auf welcher Partition sich welche Dateien befinden, es genügt, den Pfad zu den Dateien zu kennen.

Das erste Dateisystem, das vom Kernel beim Systemstart angehängt wird, ist das `root`- oder `/`-Dateisystem. Der Kernel hat keine weiteren Informationen über andere Dateisysteme des Rechners, so dass alle Informationen und Programme, die zum weiteren Systemstart erforderlich sind, auf dieser Partition vorhanden sein müssen. Folgende Funktionen sind für den Systemstart oder für Reparaturen notwendig und müssen unbedingt von der `root`-Partition aus möglich sein:

Einhängen von Dateisystemen
: Es wird mindestens das Programm `mount` benötigt, um z. B. die `/usr`-Partition einzuhängen. Bei einem `/usr`-Dateisystem, das mittels NFS (Network File System) eingehängt wird, müssen zusätzlich alle notwendigen Netzwerkprogramme wie `ifconfig`, `route` oder `ping` verfügbar sein. Auf der

einen Seite will man möglichst wenige Programme auf der root-Partition speichern, daher könnte man ping auch unter /usr/bin speichern, aber zur Fehlersuche ist es wiederum oft sinnvoll einsetzbar.

Reparatur des Dateisystems

Im Falle eines Systemabsturzes werden die Dateisysteme beim nächsten Start automatisch überprüft. Das geschieht von einem root-Dateisystem aus, das nur im Lesezugriff verfügbar ist. Alle notwendigen Programme zur Reparatur von Dateisystemen müssen daher auf der root-Partition gespeichert werden. Genaueres zum Systemstart finden Sie im Kapitel 3, »Ablauf eines Systemstarts«.

Als Alternative können Sie das System von einer zweiten root-Partition aus starten oder Boot-Disketten oder eine Installations-CD verwenden. In der Regel sollte man aber ohne derartige Hilfsmittel auskommen.

Wiederherstellung einer Datensicherung

Oft ist es sinnvoll, die Programme und Konfigurationsdateien zur Datensicherung auf der root-Partition zu speichern, da man so im Notfall bereits mit einem minimalen System die verlorenen Daten wiederherstellen kann. In Kapitel 8, »Datensicherung«, wird dieses Thema ausführlicher behandelt.

Der Linux-Neuling wird nun vielleicht auf eine eigene /usr-Partition verzichten oder sogar insgesamt nur eine Partition für das Linux-System verwenden. In diesem Fall sind alle notwendigen Programme immer verfügbar, es gibt jedoch eine Reihe von guten Gründen, die root-Partition möglichst klein zu halten und die Daten auf mehrere Partitionen zu verteilen. Am Ende liegt es aber in der Verantwortung des Systemverwalters, eine sinnvolle Aufteilung gemäß den Anforderungen und den eigenen Vorstellungen zu bestimmen.

Fehleranfälligkeit

Fehler in der root-Partition sind ein größeres Problem als Fehler in anderen Dateisystemen. Dabei könnten die Programme zur Dateisystemprüfung oder andere wichtige Programme beschädigt werden und die Benutzung einer Boot- oder Notfalldiskette erforderlich sein.

Dadurch, dass die /-Partition ständig im Schreibzugriff ist, sind bei einem Absturz hier häufiger Inkonsistenzen zu beobachten. Dies ist besonders dann der Fall, wenn das /tmp- oder /var-Verzeichnis nicht auf einer eigenen Partition untergebracht ist und Programme gerade temporäre Dateien angelegt hatten.

Sollte das Einhängen einer Partition mit einer Kernel-Fehlermeldung quittiert werden, so kann ein Extended-2-Dateisystem möglicherweise mit der Option check=none angehängt werden. Bei derartigen Fehlern sollten jedoch nur Lesezugriffe durchgeführt werden, z. B. zur Rettung von Daten, und das Dateisystem schnellstens überprüft oder gegebenenfalls neu angelegt werden.

Zur Beruhigung sei hier erwähnt, dass auch nach einem Systemabsturz nur sehr selten Daten verlorengehen. Die Programme zur Prüfung von Dateisystemen, insbesondere des Extended-2-Filesystems, sind sehr leistungsfähig und zuverlässig. Die aktuell verfügbaren Journaled File Systems sind ebenfalls recht zuverlässig und ersparen möglicherweise einen langwierigen Filesystem-Check.

Die Aufteilung des Linux-Systems auf mehrere Partitionen hat jedoch auch Nachteile. So bleibt in der Regel auf jeder Partition ein Teil des Speicherbereichs ungenutzt. Dieser »Verschnitt« kann sich dann letzten Endes zu dem Platz summieren, den man noch benötigt, um ein weiteres Programmpaket zu installieren. Da jedoch die Größe der Partitionen nur durch Löschen und Neuanlegen verändert werden kann, ist eine Änderung der Partitionsgrößen aufwändig. Mit der Aufnahme des *Logical Volume Manager* (siehe Abschnitt 4.4.2, »Der Logical Volume Manager«) in den Kernel können Sie aber die logischen Laufwerke einfach verändern.

Verzeichnis	Beschreibung
`/`	Das `root`-Verzeichnis
`/bin`	Zum Systemstart notwendige Programme
`/boot`	Dateien für Boot-Lader (z. B. LILO)
`/dev`	Devices (Geräte)
`/etc`	Maschinenlokale Systemkonfiguration
`/home`	Home-Verzeichnisse der Benutzer (optional)
`/lib`	Shared Libraries (libc.so.*, libm.so.* und ld.so)
`/mnt`	Temporäre Mount-Möglichkeit
`/opt`	Zusätzliche Applikationen
`/proc`	Pseudo-Dateisystem mit Prozessinformationen (optional)
`/root`	Home-Verzeichnis für den Systemverwalter `root` (optional)
`/sbin`	Zum Systemstart notwendige Systemprogramme
`/tmp`	Temporäre Dateien
`/usr`	Die `/usr`-Hierarchie (siehe Tabelle 2.3)
`/var`	Variable Daten

Tabelle 2.2 Übersicht über das `root`- oder `/`-Verzeichnis

2.4.1 /home – Home-Verzeichnisse der Benutzer

Ein Benutzer hat außer in seinem Home-Verzeichnis nur an sehr wenigen Stellen des Dateisystems Schreibberechtigung. Oft bietet es sich an, die Home-Verzeichnisse auf einer eigenen Partition unterzubringen, damit bei einer Neuinstallation

diese Daten erhalten bleiben. Auch bietet diese Aufteilung bei der Datensicherung eine gute Trennung zwischen Systemdaten und Benutzerdaten.

Ein weiterer Vorteil ist die »Hardware-Quota«, die dafür sorgt, dass Benutzer nur die /home-Partition bis zum Rand mit Daten füllen können. Die Lauffähigkeit des Systems insgesamt bleibt dennoch erhalten. Bei einer vollen root-, /tmp- oder /var-Partition kann die Lauffähigkeit des Systems beeinträchtigt werden. Das Extended-2-Dateisystem legt für diesen Zweck eine root-Reserve an, die nur vom Systemadministrator benutzt werden kann. Mit dem Programm tune2fs können der Anteil an reservierten Blöcken und die Benutzer, die diese Blöcke verwenden dürfen, festgelegt werden.

Mit der Kernel-Version 2.0 ist der lang ersehnte Quota-Support in den Standard-Kernel aufgenommen worden. Damit ist es möglich, einzelnen Benutzern oder Benutzergruppen nur einen bestimmten Teil des Festplattenspeichers zur Verfügung zu stellen. Dabei unterscheidet man zwischen einem Soft-Limit, das kurzzeitig überschritten werden darf, und dem Hard-Limit, das weitere Schreibversuche unterbindet. Auf vielen Rechnern (mit vernünftigen Benutzern) sind diese Funktionen nicht notwendig, aber bei vielen Anwendern und nur begrenztem Plattenplatz kann der Einsatz von Quotas sinnvoll sein.

Die Verwendung des Verzeichnisses /home ist eine Linux-Spezialität, aber weit verbreitet, und die Bedeutung ist offensichtlich. Andere Unix-Systeme verwenden oft auch /users, /usr/users oder ähnliche Verzeichnisse. Diese Verzeichnisse sind oft auch weiter untergliedert, z. B. in verschiedene Benutzergruppen wie Studenten, Mitarbeiter oder Systemadministratoren. Die genaue Aufteilung ist jedoch eine Entscheidung, die jeder Systemadministrator nach lokalen Erfordernissen treffen sollte. Der FHS beschreibt dieses Verzeichnis als optional und in der Verantwortung des Systemverwalters.

Programme oder auch Benutzer sollten das Home-Verzeichnis nicht direkt (namentlich) referenzieren, sondern die Shell-Variable HOME verwenden oder je nach Shell die Kurzform mit dem Tilde-Zeichen (~). In C-Programmen sollte die Funktion getpwent() verwendet werden.

Traditionell hatte der Benutzer root das Home-Verzeichnis /. Dies führte praktisch unvermeidbar dazu, dass einige Dateien in / angelegt wurden, was das Wurzelverzeichnis unübersichtlich groß machte. Um dieses Verzeichnis auch von der Anzahl der Einträge her klein zu halten, verwendet man heute oft das Verzeichnis /root als Home-Verzeichnis des Systemadministrators.

2.4.2 Zusätzliche Software-Pakete in /opt

Das Verzeichnis /opt dient zur Speicherung von größeren Programmpaketen, die zusätzlich zum eigentlichen System installiert werden. Für jedes Software-Paket sollte ein eigenes Verzeichnis /opt/*Paket* eingerichtet werden, das die

statischen Daten dieses Pakets enthält. Die ausführbaren Programme werden unter `/opt/Paket/bin` gespeichert, Manpages im entsprechenden `man`-Verzeichnis.

Lokal kann der Systemverwalter die Verzeichnisse `/opt/bin`, `/opt/man` usw. (analog zu `/usr/local`) einrichten. Dort können z. B. Skripten zum Aufruf der installierten Pakete oder andere Frontends installiert werden.

Variable Daten eines Pakets sollten unter `/var/opt` abgelegt werden, Konfigurationen unter `/etc/opt`. Lock-, PID- und Device-Dateien werden in den Unix-üblichen Verzeichnissen gespeichert.

In `/opt` werden normalerweise Programme gespeichert, die nicht mit der Distribution mitgeliefert werden und aus einer Quelle stammen, die der Systemverwalter nicht unbedingt unter eigener Kontrolle hat. So installiert man hier Pakete wie Star-Office oder andere extern entwickelte Software. Programme, die der Systemverwalter selbst kompiliert und installiert, wird man weiterhin in `/usr/local` unterbringen.

2.4.3 Die /usr-Hierarchie

Das `/usr`-Dateisystem belegt auf vielen Linux-Systemen den größten Anteil an Plattenplatz. Dort werden die meisten Programme, die mit einer Distribution mitgeliefert werden, installiert. Dieselbe Struktur, die hier beschrieben wird, kann (und sollte) unter `/usr/local` erneut eingerichtet werden. Dort installiert der Systemadministrator alle Programme, die er lokal für diesen Rechner oder Rechnerverbund zusätzlich zu dem benötigt, was die Distribution bereits mitbringt. Direkt nach der Erstinstallation sollte `/usr/local` keine Dateien enthalten.

Das `/usr`-Dateisystem enthält von verschiedenen Rechnern benutzbare Daten, die nicht im Schreibzugriff zugänglich sein müssen. Damit ist es möglich, das `/usr`-Dateisystem über NFS schreibgeschützt anzuhängen oder ein »Live«-Dateisystem von einer CD zu benutzen. Dies ist insbesondere zum »Reinschnuppern« in Linux und bei nur wenig verfügbarem Festplattenplatz eine Alternative zum Kauf einer neuen Festplatte. Nachteile sind langsame Zugriffe auf die Daten und Programme und gewisse Komforteinbußen, da die CD permanent eingelegt sein muss.

Für große Programmpakete wie z. B. TeX oder Emacs sollten keine Verzeichnisse direkt unter `/usr` angelegt, sondern ein neues Verzeichnis unterhalb von `/usr/lib` verwendet werden. Damit soll verhindert werden, dass das `/usr`-Verzeichnis zu groß und damit unübersichtlich wird. Tabelle 2.3 enthält eine Übersicht über die normalerweise unter `/usr` enthaltenen Verzeichnisse.

/usr	**Die /usr-Hierarchie**
X11R6	Das X-Window-System (optional)
bin	Beinahe alle Programme
games	Spiele (optional)
include	Headerdateien für C-Programme
lib	Bibliotheken und Programmpakete
local	Lokale Installationen (zunächst leer)
sbin	Programme für den Systemadministrator
share	Architekturunabhängige Daten
src	Quellcode verschiedener Programme (optional)

Tabelle 2.3 Übersicht über die /usr-Hierarchie

/usr/bin – Programme

Das Verzeichnis /usr/bin ist das primäre Verzeichnis für Programme in einem Linux- oder Unix-System. Ausgenommen sind nur diejenigen Programme, die ausschließlich für den Systemadministrator gedacht sind (diese Programme werden in den entsprechenden sbin-Verzeichnissen gespeichert), die Programme, die vom Systemadministrator lokal in /usr/local für das betreffende System installiert wurden, und die zum Systemstart notwendigen Programme, die sich in /bin befinden müssen.

Bei der Erstellung von Skripten, die über den #!-Mechanismus gestartet werden, muss der komplette Pfad zu dem entsprechenden Interpreter angegeben werden. Der Kernel erkennt anhand der Zeichenfolge #!, dass das Programm nicht direkt gestartet werden kann, sondern von einem Interpreter abgearbeitet werden muss. Dazu wird der Interpreter geladen und diesem das Skript übergeben.

Der absolute Pfad ist kein Problem für sh- oder csh-Skripten, da diese Programme auf jedem Unix-System im Verzeichnis /bin zu finden sind. Problematischer ist es für Interpreter wie perl, tcl oder python, die oft an unterschiedlichen Stellen installiert werden. Dadurch können Skripten, die diese Interpreter verwenden, oft nicht unverändert von einem System zu einem anderen übertragen werden, weil der Pfad in der ersten Zeile des Skripts modifiziert werden muss. Der Filesystem-Hierarchie-Standard verlangt die Installation dieser Interpreter im Verzeichnis /usr/bin oder die Einrichtung symbolischer Links zu den Programmen. Dann können Skripten die Interpreter in /usr/bin referenzieren – unabhängig davon, in welchem Verzeichnis die Programme tatsächlich installiert sind.

/usr/local – lokal installierte Software

Unter `/usr/local` kann und soll im Wesentlichen dieselbe Hierarchie wie unter `/usr` eingerichtet werden (Tabelle 2.3). Hier kann der Systemadministrator lokal Programme installieren, die nicht Bestandteil einer Distribution sind. Nach der Installation einer Distribution soll diese Hierarchie bis auf die Verzeichnisse leer sein, so dass Programme der Distribution und nachträglich installierte Programme unterscheidbar sind. Diese Trennung erleichtert die Systemwartung sowohl bei der Installation oder dem Update der Distribution als auch bei der Installation von anderen Programmen.

Beim Upgrade der Distribution muss man damit rechnen, dass Programmdateien in `/usr` oder `/` überschrieben werden. Das Verzeichnis `/usr/local` liegt definitionsgemäß in der Verantwortung des lokalen Systemverwalters und wird daher nicht verändert. Zur Sicherheit kann man dieses Verzeichnis auch einfach aushängen (und aus der Datei `/etc/fstab` entfernen). Nach der Installation muss das Verzeichnis `/usr/local` die notwendigen Unterverzeichnisse enthalten, ansonsten aber leer sein.

/usr/share – architekturunabhängige Daten

In der Anfangszeit von Linux wurden nur Intel-basierte PCs unterstützt. Daher gab es keine Unterscheidung zwischen prozessorabhängigen Daten und prozessorunabhängigen Daten. Heute sind Portierungen für praktisch alle Hardware-Plattformen verfügbar oder in Arbeit, so dass sich diese Situation geändert hat. Im Verzeichnis `/usr/share` findet man Daten, die statisch sind und von verschiedenen Rechnern mit verschiedenen Prozessorarchitekturen gemeinsam benutzt werden können.

`/usr/share`	**Die `/usr/share`-Hierarchie**
`dict`	Wörterbücher aller Art (optional)
`doc`	Dokumentationen zu Paketen und FAQs (optional)
`games`	Statische Daten für Spiele (optional)
`info`	Online-Doku der GNU-Programme (optional)
`locale`	Daten für den National Language Support (optional)
`man`	Online-Handbuch (optional)
`nls`	National Language Support (optional)
`sgml`	Daten für SGML und XML (optional)
`terminfo`	Die Terminal-Datenbank (optional)
`tmac`	`troff`-Makros, die bei `groff` fehlen (optional)
`zoneinfo`	Zeitzoneninformationen (optional)

Tabelle 2.4 Übersicht über die `/usr/share`-Hierarchie

Viele der hier gespeicherten Daten hat man früher unter /usr/lib, /usr/man oder /usr/doc gefunden. Aus Gründen der Abwärtskompatibilität können einzelne Verzeichnisse weiterhin unter /usr/lib eingerichtet sein; das liegt am Distributor. Tabelle 2.4 enthält eine Übersicht über die möglichen Verzeichnisse.

Viele der Konzepte, die für das /usr/share-Verzeichnis verwendet wurden, stammen aus den BSD-Projekten. Diese ebenfalls frei verfügbaren Systeme waren schon recht früh auf verschiedene Hardware-Plattformen portiert, so dass hier schon einiges an Erfahrung gesammelt wurde. Die Zeit wird zeigen, wie nützlich dieses Verzeichnis unter Linux sein wird.

/var – variable Daten

Das /var-Verzeichnis enthält variable Daten. Da es erst mit FHS-kompatiblen Distributionen weite Verbreitung gefunden hat, ist auf vielen älteren Systemen immer noch keine saubere Trennung zwischen /usr und /var realisiert. Die Trennung ist sinnvoll, damit die oft größte Linux-Partition (/usr) nicht im Schreibzugriff eingehängt werden muss. Moderne Distributionen entsprechen auch hier in der Regel dem FHS.

Viele der hier angegebenen Verzeichnisse wurden eingeführt, damit einzelne Programme oder Programmpakete keine beschreibbare /usr-Partition erfordern. Daher ist eine Reihe der hier vorgestellten Verzeichnisse für viele Systeme optional, andere müssen für neue Software vom Systemadministrator angelegt werden. Solange noch nicht alle Programme völlig an diesen Standard angepasst sind, wird der Systemadministrator immer wieder überlegen müssen, wie einzelne Programme sinnvoll konfiguriert und installiert werden.

/var	**Variable Daten**
account	Abrechnungsdaten (Prozess-Accounting, optional)
cache	Cache für Applikationsdaten
games	Variable Daten von Spielen (optional)
lib	Variable Statusinformationen
lock	Lock-Dateien
log	Log-Dateien und Verzeichnisse
mail	Postfächer der Benutzer (optional)
opt	Variable Daten aus /opt
run	PID-Dateien von Prozessen
spool	Verzeichnisse für Warteschlangen
tmp	Temporäre Dateien
yp	Network Information Service (optional)

Tabelle 2.5 Übersicht über die /var-Hierarchie

Prozess-Accounting (/var/account)
Dieses Verzeichnis enthält die variablen Daten des Prozess-Accounting, sofern die Distribution dies unterstützt. Mit Hilfe des Accounting kann festgehalten werden, wann welcher Benutzer welches Kommando ausgeführt hat und welche Ressourcen benötigt wurden. Für viele private Systeme ist dies nicht notwendig, aber viele kommerzielle Rechenzentren rechnen z. B. nach verbauchter CPU-Zeit ab.

Wenn Ihre Distribution die entsprechenden Tools nicht mitliefert, dann finden Sie diese auf jedem GNU-Mirror als `acct-6.3.tar.gz`. Beachten Sie, dass die Rohdaten einen sehr großen Umfang annehmen. Daher sollten Sie regelmäßig diese Daten in eine Summenform überführen.

/var/cache – Cache für Applikationen
Mit der Version 2.0 des FHS wurde diese Verzeichnishierarchie neu eingeführt. Hier werden Daten gespeichert, die durch zeitintensive Berechnungen oder andere aufwändige Operationen erzeugt wurden. Wenn diese Daten bei späteren Läufen zur Verfügung stehen, dann können diese Aufgaben entfallen, stattdessen werden die bereits bestehenden Daten verwendet.

Denkbar sind hier WWW-Proxy-Caches, TeX-Fonts oder vorformatierte Manpages. Die Applikationen können das Löschen veralteter Daten selbst organisieren, aber der Administrator kann z. B. bei knappem Plattenplatz diese Daten (fast) bedenkenlos löschen.

Eingeführt wurde dieses Verzeichnis, damit der Administrator z. B. speziell für diese Daten eigene Sicherungsregeln definieren kann. Für viele private Systeme ist dies von eher untergeordneter Bedeutung, aber für öffentlich zugängliche Rechner, auf denen viel mit TeX gearbeitet wird, lohnt sich das Speichern der entsprechenden Fonts.

/var/lock – Lock-Dateien
Alle Lock-Dateien, die zum Sperren von Geräten gegen mehrfache Benutzung verwendet werden, sollten im Verzeichnis `/var/lock` angelegt werden. Um die `/usr`-Partition schreibgeschützt anhängen zu können, sollten dort keine Lock-Dateien angelegt werden.

Der Name einer Lock-Datei beginnt mit `LCK..`, gefolgt von dem Namen des zu sperrenden Geräts. Um das Gerät `/dev/ttyS0` zu sperren, wird die Datei `/var/lock/LCK..ttyS0` verwendet.

Alle Lock-Dateien sollen für jeden Benutzer lesbar sein und die Prozessnummer des sperrenden Prozesses im Klartext enthalten. Dadurch kann jedes Programm prüfen, ob der Prozess noch existiert, der den Lock angelegt hat.

Es ist sinnvoll, auf den symbolischen Link /dev/modem zu verzichten und stattdessen das echte Device zu verwenden. Lock-Dateien können dann ausschließlich für das echte Gerät erzeugt werden. Zum einen vermeidet man damit, dass ein Gerät unter zwei verschiedenen Namen angesprochen (und gesperrt) werden kann, zum anderen entspricht die Semantik des chown-Systemaufrufs unter Linux bei symbolischen Links nicht dem von anderen Systemen.

Die Standardisierung dieses Verzeichnisses sollte die Interoperabilität zwischen verschiedenen Programmen, bei denen die Synchronisation über Lock-Dateien stattfindet, deutlich verbessern. Das liegt zum einen daran, dass alle Programme hier ihre Lock-Dateien anlegen, zum anderen daran, dass im FHS ein Standardformat für den Inhalt der Datei festgelegt wird.

/var/run – PID-Dateien von Prozessen

In /var/run werden für Dämonen die PID (process id)-Dateien abgelegt, so dass die Prozessnummer eines Hintergrundprozesses einfach festzustellen ist. Diese Dateien sollen *Programm*.pid heißen und die Prozessnummer im Klartext enthalten. Damit ist es einfach, einem Hintergrundprozess, wie z. B. dem syslog-Dämon, ein Signal zu schicken (siehe auch Listing 2.1).

```
(linux):~# kill -HUP $(cat /var/run/syslog.pid)
```

Listing 2.1 Verwendung von Lock-Dateien am Shell-Prompt

Dieses Verzeichnis soll auch für Unix-Domain-Sockets verwendet werden, die zur Kommunikation zwischen Programmen dienen. Da Dämonen unter verschiedenen Benutzer-IDs gestartet werden (die News-Dämonen z. B. laufen unter dem Benutzernamen news), muss dieses Verzeichnis für alle Benutzer beschreibbar sein. Aus Sicherheitsgründen sollte dann jedoch, wie bei den tmp-Verzeichnissen auch, das t-Bit gesetzt werden. Dadurch kann nur der Benutzer eine Datei löschen, die ihm selbst gehört. Eine Alternative dazu könnte sein, hier entsprechende Unterverzeichnisse einzurichten, die dann nur für den entsprechenden Benutzer oder ein spezielles Programm beschreibbar sind.

/var/spool – Verzeichnisse für Warteschlangen

Das /var/spool-Verzeichnis dient zur Zwischenspeicherung von uucp-Jobs, Druckausgaben oder anderen Aufgaben, die das System in der Zukunft zu erledigen hat. Oft werden die Daten automatisch gelöscht, nachdem der Auftrag durchgeführt wurde. Die genaue Aufteilung dieses Verzeichnisses bestimmt der Systemadministrator anhand der zur Verfügung stehenden Hard- und Software-Umgebung. Tabelle 2.6 enthält eine Übersicht über in /var/spool möglicherweise vorhandene Unterverzeichnisse.

/var/spool	Spool-Verzeichnisse
lpd	Warteschlangen des Drucksystems (optional)
mqueue	Warteschlange für ausgehende elektronische Post (optional)
news	Verzeichnis zur News-Speicherung und -Auslieferung (optional)
rwho	Arbeitsdateien des rwho-Dämons (optional)
uucp	Warteschlangen für UUCP (optional)

Tabelle 2.6 Das Verzeichnis /var/spool im Überblick

/var/tmp – temporäre Dateien
Das Verzeichnis /var/tmp ist wie /tmp ein Verzeichnis für temporäre Dateien. Es wird benutzt, um die root-Partition, die auch das /tmp-Verzeichnis enthält, möglichst klein zu halten. Auch hier ist es die Entscheidung des Systemadministrators, wie lange Daten in /var/tmp bestehen bleiben. Die Daten sollten jedoch mindestens so lange erhalten bleiben wie im Verzeichnis /tmp.

Dateien in /var/tmp überstehen einen Reboot. Das Verzeichnis /tmp könnte in einer RAM-Disk liegen oder beim Booten gelöscht werden.

2.4.4 Die weitere Entwicklung des FHS

Der Filesystem-Hierarchie-Standard ist in der heute vorliegenden Form unter den Linux-Anwendern im Allgemeinen als sinnvolle und vernünftige Absprache anerkannt. Dennoch gibt es weiterhin (und es wird diese Unterschiede vermutlich immer geben) deutliche Unterschiede zwischen den einzelnen Distributionen.

Immer wenn diese Unterschiede zu Problemen führen (können), dann wird in der entsprechenden Mailing-Liste darüber gesprochen. Zurzeit gibt es eine Diskussion über die Integration von (kommerziellen) Paketen in den Standard und die Verwendung als Filesystem-Hierarchie-Standard auch für die *BSD-Systeme.

Obwohl zu vielen Fragen verschiedene Ansichten geäußert werden, die auch häufig zu »Flames« Anlass geben könnten, ist die Mailing-Liste eine sehr produktive Einrichtung. Aufgrund der teilweise sehr ausführlichen Diskussionen ist das Nachrichtenaufkommen dort im Vergleich zu anderen Mailing-Listen, sowohl was die Anzahl als auch den Umfang der Nachrichten betrifft, sehr hoch.

2.5 Linux Internationalization Initiative (Li18nux)

Die Linux Internationalization Initiative (Li18nux, *http://www.li18nux.net/*) standardisiert APIs und Programme, um Programme internationalisieren und lokalisieren zu können. Dabei werden C-Funktionen festgelegt, die verfügbar sein müssen, wiederum wird dabei auf bereits vorhandene Standards aufzubauen.

Die Namen der unter Linux verwendeten Lokale-Umgebungen werden festgelegt, genauso wie zu verschiedenen Programmen spezifiziert wird, wie diese an lokale Besonderheiten anzupassen sind. Ein weiterer großer Teil sind Methoden zur Eingabe von beliebigen Sprachen. Mit etwas Geduld wird Linux auch über eine durchgängige Unicode-Implementation verfügen – das ist derzeit noch nicht der Fall.

2.6 Die Free Standards Group

Die Free Standards Group (*http://www.freestandards.org/*) bildet ein Dach für verschiedene Gruppen, die an Linux-Standards arbeiten. Beteiligt sind als Arbeitsgruppen die Linux Standard Base, die Linux Internationalization Initiative und die Linux Assigned Names and Numbers Authority (*http://www.lanana.org/*). Assoziierte Gruppen sind der FHS und die X Desktop Group.

2.7 Standards, Standards, Standards

In der Anfangszeit von Linux war man froh, wenn etwas funktionierte. Dann kam die Zeit, als die ersten Distributionen und die ersten Binärpakete erschienen. Und damit begann die Suche nach Standards, um genau diese Pakete austauschbar zu machen. Auch wenn die GNU General Public License verlangt, den Quellcode zugreifbar zu machen, ist es doch sehr bequem, ein vorgekochtes Paket installieren zu können. Dass heute eine Reihe von Standards verfügbar sind, ist ein Zeichen für das Alter und die Stabilisierung der Linux-APIs und ABIs.

Anwender, Entwickler und Distributoren profitieren von Standards. Allerdings ist ein wichtiger Aspekt der freien Software die Wahl, die der Anwender (und auch der Anbieter) hat. Wo bleibt bei den Standards die Individualität? Immer noch kann jeder eine Linux-Distribution entwickeln, man muss sich dabei an keinerlei Standards halten. Aber die Anwender werden auf die Dauer dafür sorgen, dass die einzelnen Systeme nicht allzu weit auseinander laufen. Bisher hat das sehr gut funktioniert und die Aussichten für die Zukunft sind gut.

3 Ablauf eines Systemstarts

It's not so hard to lift yourself by your bootstraps once you're off the ground.
Daniel B. Luten

3.1 Überblick über einen Systemstart

Dem Start eines Computers ist etwas Magisches eigen. Üblicherweise wird der Vorgang auch als »boot strapping« bezeichnet. Dabei kommt einem im deutschen Sprachraum das Bild eines sich selbst an den Stiefelschäften aus dem Sumpf ziehenden Baron von Münchhausen in den Sinn. Aber es finden sich (in diesem Fall) logische Erklärungen, die zeigen, dass jede Stufe des Vorgangs ganz bestimmte Ergebnisse der vorausgegangenen Phasen voraussetzt und keinesfalls irgendwie in der Luft hängt. Der Start eines Linux-PCs findet in den im Folgenden aufgeführten Phasen statt:

- Ausführen des Power-On-Self-Tests (POST) des Basic-Input/Output-Systems (BIOS). Dieser ist oft in einem ROM oder Flash-EPROM gespeichert.
- Ausführen eines Boot-Laders von einer Diskette, Festplatte oder von einem CD-ROM-Laufwerk aus. Viele Unix-Workstations können auch von einem angeschlossenen Band gebootet werden, die PC-Architektur gibt das aber nicht her.
- Start des Kernels durch den Boot-Lader.
- Abarbeiten der individuellen Systemkonfiguration via `init`.

In den folgenden Abschnitten konzentriere ich mich auf die Initialisierung der PC-Hardware durch das BIOS. Andere Systeme führen ähnliche Funktionen aus, allerdings möglicherweise in anderer Reihenfolge oder mit anderen Verfahren.

3.2 Das Basic-Input/Output-System (BIOS)

An erster Stelle im Bootvorgang steht das BIOS der Hauptplatine. Diese zumeist in EPROMS (neuerdings auch Flash-EPROMS) befindliche Software enthält eine Reihe von Funktionen, die BIOS-Interrupts, so z. B. `0x10` für die Bildschirmansteuerung oder `0x13` für Disketten- und Festplattenzugriffe. Damit war es möglich, kleinere Abweichungen in der Hardware-Ansteuerung durch entsprechende Anpassungen im BIOS auszugleichen – das war einer der Gründe für den Erfolg der IBM-kompatiblen PCs.

Jede Erweiterungskarte bzw. Komponente, die IBM-kompatibel sein will, muss auf entsprechende Aufrufe, z. B. zur Initialisierung durch das BIOS in definierter Art und Weise, reagieren. Hierdurch wurde eine gewisse Unabhängigkeit von

spezifischen Peripheriegeräten geschaffen, was wesentlich zur Herausbildung des Marktes für IBM-kompatible Komponenten beigetragen hat. Eigentlich handelt es sich aber um eine Intel-Kompatibilität, genauer um eine Kompatibilität zum Intel-8088/86, da die im BIOS fest eingebrannte Software aus Instruktionen für diesen Prozessor besteht.

Es gibt eine Vielzahl unterschiedlicher BIOS, nämlich für jeden Typ von Motherboard eine eigene BIOS-Version. Wie einzelne Funktionen realisiert wurden und in welcher Reihenfolge sie abgearbeitet werden, liegt zu einem gewissen Maß im Ermessen des jeweiligen Herstellers. Die folgende Übersicht über den Ablauf eines BIOS-Starts hat insofern exemplarischen Charakter.

Nach dem Einschalten der Stromversorgung wird das BIOS hardwaremäßig in den Speicherbereich von `0xF000:E000` bis `0xF000:FFFF` eingeblendet und der Programmzähler auf die Adresse `0xF000:FFF0` gesetzt, womit die Ausführung des BIOS-Codes beginnt. Der Prozessor startet im so genannten Real-Mode, daher steht nur der Speicherbereich unterhalb von 1 MB zur Verfügung. Üblicherweise enthält das BIOS auch nur Code für diesen Modus, so dass es von Protected-Mode-Systemen wie Linux nicht verwendet werden kann (einige Funktionen des modernen PCI-BIOS können von Linux verwendet werden). Neue Linux-Versionen sind teilweise so groß, dass sie nicht mehr in das erste Megabyte Speicher geladen werden können. Ein Ausweg sind hier Kernel-Module oder die Erzeugung eines `bzImage`, das bereits im Protected Mode geladen wird.

Bevor das BIOS damit beginnt, sich selbst und die Peripherie des Computers zu testen, untersucht es den Speicherbereich zwischen dem Grafikadapter an der Adresse `0xA000` und seiner eigenen Startadresse nach BIOS-Erweiterungen anderer Steckkarten, z. B. von Netzwerkkarten. Karten mit eigenem BIOS werden an der BIOS-Signatur `0xAA55` bzw. `0x55AA` am Ende eines 32K-Blocks erkannt. Bereits vor dem endgültigen Booten des Systems werden hier Initialisierungen durchgeführt.

Danach führt das BIOS den Power-On-Self-Test (POST) durch: Die Funktionen des Prozessors selbst, die Register und einige Befehle werden überprüft. Falls der Prozessor defekt ist, wird der PC, logischerweise ohne Angabe von Gründen, angehalten. Bei manchen Fehlern geben einige BIOS-Versionen noch akustische Signale aus, die je nach Anzahl der Piepser Aufschluss über die Art des Fehlers geben. Danach werden Prüfsummen über das ROM selbst gebildet und mit den darin festgelegten Werten verglichen, um mögliche Defekte im ROM-BIOS oder, wenn es in den Hauptspeicher kopiert wurde, des betreffenden RAM-Bausteins zu finden.

Anschließend werden weitere Bauteile auf der Hauptplatine, wie Interrupt- und DMA-Controller, sowie der Hauptspeicher getestet und initialisiert. Fortgesetzt wird der POST mit dem Überprüfen von Tastatur, Disketten- und Plattenlaufwerken. Die Ergebnisse dieser Tests werden in BIOS-Variablen im Hauptspei-

cher abgelegt und die Interrupt-Vektor-Tabelle wird initialisiert. Mittels des BIOS-Interrupts `0x19` wird das eigentliche Booten eingeleitet bzw. der »boot strap loader« aufgerufen.

Durch das frühe Initialisieren der Erweiterungskarten ist es möglich, dass eine Netzwerkkarte im Laufe des weiteren Boot-Vorgangs die Funktionen des Boot-Laders übernehmen kann. Diese Funktion wird verwendet, wenn man einen Rechner über das Netz bootet. Dies hat einige Vorteile:

- Auch wenn der Rechner ausgeschaltet ist, kann man auf dem Boot-Server Anpassungen in der Konfiguration vornehmen.
- Der Rechner braucht keine oder nur eine kleine Festplatte für eine Swap-Partition. Er ist damit ausgesprochen leise.
- Man kann einen Rechner relativ einfach umkonfigurieren, z. B. auf eine neue IP-Adresse oder einen neuen Kernel.

Natürlich hat diese Funktion auch Nachteile: Die Netzlast wird höher und Dateizugriffe sind via NFS langsamer als auf eine lokale Festplatte. Wenn Sie diese Funktion nutzen möchten, dann finden Sie weitere Informationen in der Datei `Documentation/nfs-root.txt` in den Kernel-Quellen und in den Mini-HowTos `NFS-Root` und `NFS-Root-Client`.

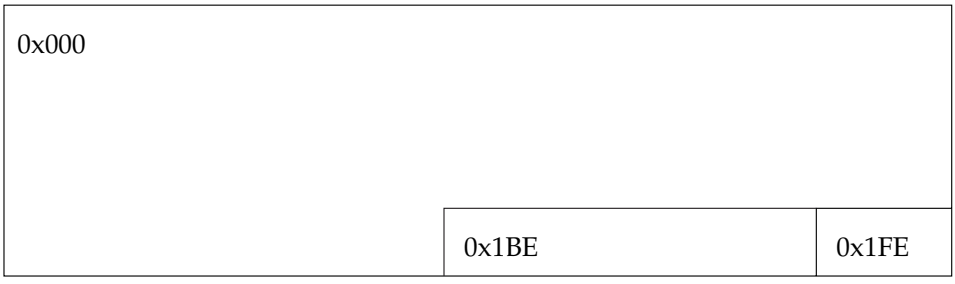
Bei Systemen, die nicht auf Intel-Prozessoren basieren, nennt man das BIOS und die entsprechenden Initialisierungs- und Laderoutinen häufig PROM, Bootmonitor oder Console. Vielfach sind hier auch interaktive Funktionen zur Hardware-Prüfung oder Initialisierung untergebracht, die auch ohne jegliches Betriebssystem verwendet werden können. Außerdem sind in vielen Fällen leistungsfähige Boot-Lader implementiert, mit denen man beliebige Kernel oder via Netzwerk booten kann.

3.3 Laden von Diskette oder Festplatte

In einer für alle Betriebssysteme gleichen Reihenfolge wird versucht, das System zu starten. Die Reihenfolge, in der versucht wird, von den verschiedenen Laufwerken zu booten, kann normalerweise im BIOS-Setup eingestellt werden. Früher wurde versucht, zunächst von `/dev/fd0` (Laufwerk `A:`), danach von `/dev/hda` (Laufwerk `C:`), den ersten Sektor zu lesen. Heute hat man die Wahl, ob das Laufwerk `A:`, Laufwerk `C:`, ein SCSI- oder das CD-ROM-Laufwerk als Boot-Device verwendet werden soll. Bei Disketten wird dieser erste Sektor *Boot-Sektor*, bei Festplatten *Master Boot Record* (MBR) genannt. Er ist jeweils 512 Byte groß.

Viele Unix-Workstations können auch vom Band oder von CD gebootet werden, auch diese Funktionen sind nur in den neueren PC-BIOS-Versionen bzw. im BIOS des SCSI-Controllers implementiert. Wenn Ihr BIOS diese Funktion unter-

stützt, dann können Sie direkt von einer geeignet erstellten Linux-CD booten und das System installieren – ohne Disketten.



Offset:	0x000	Programmcode
	0x1BE	Partitionstabelle
	0x1FE	Magic Number (0xAA55)

Abbildung 3.1 Aufbau eines Master Boot Records

Im Master Boot Record (MBR, siehe Abbildung 3.1) einer Festplatte befinden sich vier jeweils 16 Byte große Einträge, jeweils einer für jede der vier möglichen (primären) Partitionen. Ist eine solche Partition weiter unterteilt, wird sie erweiterte Partition genannt. Pro (erweiterter) Partition können bis zu vier so genannte logische Laufwerke eingerichtet werden, d. h., es sind maximal 16 logische Laufwerke je Festplatte möglich. Der Aufbau des ersten Sektors einer erweiterten Partition entspricht aus Gründen der Einfachheit dem Aufbau des MBR.

Jeder der vier Partitionseinträge des MBR enthält neben Angaben zum Typ der Partition (z. B. Linux = `0x83`, Linux Swap = `0x82`) und Angaben zu Beginn, Ende und Größe der Partition auch ein Bootflag, das anzeigt, ob die Partition »aktiv« ist und damit von ihr gebootet wird. Es kann immer nur eine primäre Partition durch das Bootflag als aktiv markiert sein, die dann vom Standard-MBR gebootet wird.

Der vor der Partitionstabelle des MBR liegende Programmcode lädt den Boot-Sektor der als aktiv markierten Partition und wird deshalb als Boot-Lader bezeichnet. Unabhängig vom benutzten Betriebssystem hat ein Boot-Lader beim Starten von einer Festplatte die Aufgabe, die aktive Partition zu bestimmen und unter Zuhilfenahme der BIOS-Routinen den Boot-Sektor dieser aktiven Partition zu laden. Dies ist auch die Funktion des Standard-MS-DOS-Boot-Sektors.

Wann immer es möglich ist, sollten Sie den MBR unverändert lassen und stattdessen z. B. in den Boot-Sektor der Root-Partition installieren und diese aktivieren. Wenn Sie weitere Betriebssysteme installieren oder modernisieren wollen, dann können diese die Bootfähigkeit der Linux-Installation nicht gefährden.

Am Anfang gab es unter MS-DOS nur primäre Partitionen und der Standard-MBR kann nur von solchen Partitionen das Betriebssystem laden – das MSDOS-`FDISK` kann deshalb nur primäre Partitionen aktivieren. Die MS-DOS-Bootsequenz sieht vor, dass der MBR den Boot-Sektor der aktiven Partition startet. Von diesem werden die Dateien `IO.SYS` und `MSDOS.SYS` geladen, in denen DOS gespeichert ist. Die moderneren Boot-Lader unter Linux bieten demgegenüber viele weitere Möglichkeiten, es kann auch von logischen Laufwerken in erweiterten Partitionen gebootet werden.

3.4 Die Linux Boot-Lader

Gegenüber dem MS-DOS-Verfahren haben die moderneren Boot-Lader wie `lilo` unter Linux viele weitere Möglichkeiten, z. B. das Booten anderer Betriebssysteme, das Wechseln zwischen verschiedenen Versionen eines Betriebssystems und das Übergeben von Parametern beim Start des Kernels.

Die drei wichtigsten Boot-Lader für Linux auf der Intel-Plattform sind: `lilo`, `loadlin` und `syslinux`. Für die anderen Plattformen gibt es jeweils eigene Boot-Lader, z. B. SILO für Linux/Sparc, MILO für Linux/Alpha und `amiboot` für Linux/m68k (Amiga). In den folgenen Abschnitten werden wir uns nur mit der Intel-Plattform beschäftigen.

Allen Boot-Ladern ist die Fähigkeit gemeinsam, schon beim Aufruf oder zu einem bestimmten Zeitpunkt während des Ladens quasi »per Hand« übergebene Parameter oder Argumente zu akzeptieren. Diese Parameter beeinflussen entweder den Boot-Lader selbst, den Kernel, den `init`-Prozess oder werden als Umgebungsvariablen abgelegt und können so zur Konfiguration der zu startenden Prozesse benutzt werden. Diese Angaben werden häufig als Kommandozeilenparameter des Kernels bezeichnet. Die Bezeichnung Boot-Parameter wäre jedoch treffender, da sie vielen Systemadministratoren vorwiegend in speziellen Konfigurationsdateien begegnen.

Die Parameter für die Boot-Lader selbst enthalten Dinge wie den Namen und Pfad der zu ladenden Kernel-Datei und Angaben zur Partition, auf der diese zu finden ist. Diejenigen Parameter, die die Boot-Lader nicht verstehen, werden einfach an den Kernel weitergegeben; was dieser selbst bzw. ein Gerätetreiber auswertet, bezieht sich zumeist auf das Vorhandensein und ggf. die Konfiguration bestimmter Hardware. Der Kernel wiederum übergibt die Parameter, die ihm unbekannt sind, oder das, was erkannt, aber nicht behandelt wird, an `init`. Die Werte, die auch `init` nicht kennt, werden als Environmentvariablen im Format `variable=`*`wert`* abgelegt und können somit von einer Shell oder einem beliebigen anderen Programm ausgewertet werden.

Diese Flexibilität der verschiedenen beteiligten Programme in der Handhabung der Boot-Parameter hat aber auch zur Folge, dass Optionen nicht nur durchgereicht, sondern auch überschrieben werden können. Dies ist z. B. dann beabsichtigt, wenn in den Kernel fest einkompilierte Werte, etwa der Interrupt und der Port einer Netzwerkkarte, sich im Nachhinein als falsch herausstellen.

3.4.1 Der Linux-Lader lilo

`lilo` von Werner Almesberger ist der meistbenutzte und ein recht vielseitiger Boot-Lader für Linux. Die Abkürzung `lilo` steht für Linux-Loader, `lilo` kann aber auch andere Betriebssysteme laden. Bei vielen Linux-Benutzern hat sich `lilo` zum erstenmal nach der Installation einer Linux-Distribution durch das Anzeigen des Textes `LILO` nach einem Neustart des Systems bemerkbar gemacht. Nach dem Drücken einer der Tasten [Umschalt], [Strg] oder [Alt] oder wenn entweder [Umschalt-Fest] oder [Rollen] eingeschaltet sind, zeigt `lilo` den `boot:`-Prompt und wartet auf eine Benutzeraktion, d. h. darauf, dass der Name einer zu ladenden Boot-Konfiguration eingegeben wird. Durch die Eingabe von [Tab] wird eine Liste der verfügbaren Konfigurationen auf den Bildschirm ausgegeben.

Das Paket `lilo` besteht aus zwei Teilen, die in gewisser Weise voneinander abhängen. Diese Teile sind:

- Das Programm `/sbin/lilo`. Es muss, sofern `lilo` als Boot-Lader verwendet wird, nach jeder Änderung an den Kernel-Dateien (meist `/vmlinuz`) aufgerufen werden. Dieses Programm bestimmt die Sektornummern, in denen das Kernel-Image gespeichert ist, und vermerkt diese Liste in der Map-Datei.
- Verschiedene Boot-Sektoren, die andere Betriebssysteme (auch von der zweiten Platte) starten können oder ein Kernel-Image laden. Da diese Boot-Sektoren keine Kenntnisse von der internen Struktur, z. B. des Extended-2-Dateisystems, haben, benötigen sie eine aktuelle Map-Datei.

Dies ist auch der größte Nachteil von `lilo`: Nach jeder Änderung am Kernel muss das Programm `lilo` aufgerufen werden, um die Map-Datei zu aktualisieren. Der Systemverwalter sollte außerdem darauf achten, dass immer ein Backup-Kernel zur Verfügung steht und in die `lilo`-Konfiguration eingebunden ist. Die von den *BSD-Systemen bekannte Funktion, einen beliebigen Kernel aus dem Dateisystem zu booten, ist leider nicht verfügbar.

`lilo` kann bis zu 16 verschiedene Konfigurationen booten und sowohl in den MBR als auch in den Boot-Sektor von primären Partitionen installiert werden. `lilo` kann

- auf einer Linux-Diskette als Boot-Sektor dienen,
- auf der ersten Festplatte als MBR fungieren und

- auf beliebigen primären Partitionen (außer Swap) der ersten Festplatte als Boot-Sektor installiert sein.

Demgegenüber kann es nicht

- auf einer MS-DOS-Diskette (oder irgendeiner anderen Nicht-Linux-Diskette) installiert werden,[1]
- in eine logische Partition installiert werden oder
- auf eine andere als die erste Festplatte installiert werden, wenn nicht ein dazu fähiger Boot-Lader vor `lilo` verwendet wird.

Die zuletzt genannte Beschränkung stellt jedoch keine wirkliche Einschränkung beim Entwurf einer Boot-Konfiguration dar, da `lilo` ein Kernel-Image von anderen als der ersten Festplatte laden kann. Es kann auch beliebige Partitionen, d. h. auch logische Laufwerke, auf weiteren Festplatten als root-Dateisystem benutzen.

Die Installation von `lilo` in den MBR oder den Boot-Sektor erfordert logischerweise `root`-Rechte. `lilo` wird über die Datei `/etc/lilo.conf` konfiguriert. Anschließend sollte vor der endgültigen Installation mit `/sbin/lilo -t` zunächst getestet werden, ob die in der Konfigurationsdatei befindlichen Befehle bekannt und syntaktisch korrekt sind, ob die angegebenen Kernel-Dateien gefunden wurden und die spezifizierten Partitionen existieren.

lilo konfigurieren

Zwar akzeptiert `lilo` auch Parameter, die am Boot-Prompt eingegeben werden, aber im Regelfall eines fertig konfigurierten Systems werden permanent gewünschte Einstellungen über die Konfigurationsdatei `/etc/lilo.conf` übergeben. Eine solche Datei kann z. B. wie in Listing 3.1 aussehen.

```
boot = /dev/hda2
compact
image = /boot/vmlinuz
  label=linux
image = /boot/vmlinuz.old
  label=linux.old
other = /dev/hda1
  table = /dev/hda
  label = win2k
```

Listing 3.1 Eine sehr einfache `lilo`*-Konfiguration*

Bei der in Listing 3.1 dargestellten Konfiguration werden drei Möglichkeiten der Boot-Konfiguration installiert. Der erste `image`-Eintrag definiert den Start des aktuellen Linux-Kernels `/boot/vmlinuz` unter dem Namen `linux`, das ist der

1. Zu diesem Zweck kann SYSLINUX verwendet werden.

Default. Der zweite `image`-Eintrag sorgt dafür, dass der automatisch von `make zlilo` bzw. `make install` erzeugte Backup-Kernel ebenfalls gebootet werden kann. Mit dem Eintrag `other=` kann ein anderes System, hier Windows, gebootet werden. Die ersten beiden Zeilen im Listing 3.1 bestimmen, dass `lilo` im Boot-Sektor der Partition `/dev/hda2` installiert wird und versucht wird, mit möglichst wenigen Befehlen die Kernel von der Platte zu lesen.

Bei den `lilo`-Optionen unterscheidet man in globale und image-spezifische Optionen. Globale Optionen beziehen sich auf alle Kernel-Dateien und wirken unabhängig von der ausgewählten Konfiguration. Die anderen Optionen beziehen sich jeweils nur auf eine von mehreren bootbaren Konfigurationen. Im Beispiel sind die globalen Optionen linksbündig dargestellt, die image-bezogenen sind zwei Zeichen eingerückt. Ein Block image-bezogener Optionen wird entweder durch `image=` oder `other=` eingeleitet. Bis zu 16 verschiedene solcher Blöcke sind möglich – auch mehrere `other=`-Blöcke. `lilo` kennt die folgenden globalen Optionen:

`backup=`*`backup_file`*
: Vom ursprünglichen Boot-Sektor wird eine Sicherungskopie mit dem Namen *`backup_file`* erstellt. Sofern die Datei `/boot/boot.`*`Device`* nicht existiert, wird dort standardmäßig der alte Boot-Sektor gesichert.

`boot=`*`boot_device`*
: Gibt die Festplatte bzw. die Partition an, die den Boot-Sektor von `lilo` enthält. Wenn Sie den NT-Loader verwenden, können Sie auch eine 512 Byte große Datei anlegen und diese in die Datei `boot.ini` aufnehmen. Genauere Informationen dazu finden Sie im Abschnitt »`lilo` und der Windows-Boot-Manager«.

`compact`
: Versucht, Leseanforderungen an benachbarte Sektoren auf dem Laufwerk zu bündeln. Damit wird die Ladezeit, insbesondere beim Laden von Disketten, deutlich reduziert.

`default=`*`name`*
: Der angegebene `image`- oder `other`-Eintrag wird als Standard zum Booten eingetragen. Wenn keine Angabe erfolgt, wird der erste `image`-Eintrag als Default-Wert verwendet.

`delay=`*`tsecs`*
: Wartezeit in Zehntelsekunden, bevor `lilo` das Default-Image lädt. Wenn kein `delay` angegeben ist, dann wird unmittelbar gebootet.

`disk=`*`device_name`*
: Mit diesem Eintrag beginnt ein Abschnitt in der Datei `/etc/lilo.conf`, in dem mit den Schlüsselwörtern `sectors=`, `heads=` und `cylinders=` die Geometrie des Boot-Device bestimmt werden kann. Diese Funktion sollten Sie nur einsetzen, wenn Sie die Platte mit anderen Parametern partitioniert ha-

ben, als diese selbst dem BIOS und damit `lilo` meldet. In der Regel sollten Sie diesen Eintrag nicht benötigen. Lesen Sie hierzu unbedingt die entsprechenden Abschnitte in der `lilo`-Dokumentation.

`fix-table`
Ermöglicht es `lilo`, dreidimensionale Sektor-, Kopf- und Zylinderadressen in Partitionstabellen zu korrigieren. Beachten Sie, dass Sie damit möglicherweise unangenehme Interaktionen mit anderen Betriebssystemen auslösen können.

`force-backup=`*`backup_file`*
Wie die Option `backup`, aber eine bereits existierende Backup-Datei wird überschrieben. Wenn `force-backup` zusammen mit `backup=` angegeben wird, ist nur die Option `force-backup` wirksam.

`ignore-table`
Instruiert `lilo`, defekte Partitionstabellen zu ignorieren.

`install=`*`boot_sector`*
Installiert die angegebene Datei als neuen Boot-Sektor. Wenn dieser Parameter nicht angegeben wird, verwendet `lilo` die Datei `/boot/boot.b` als `lilo`-Boot-Sektor.

`linear`
Es sollen die linearen Sektoradressen anstelle von Zylinder-/Kopf-/Sektor-Adressen benutzt werden. Lineare Adressen werden zur Laufzeit umgesetzt und sind nicht von der Festplattengeometrie abhängig. Boot-Disketten, bei deren Erstellung `linear` benutzt wurde, funktionieren möglicherweise nicht korrekt, weil die BIOS-Funktionen zur Bestimmung der Festplattenparameter für Diskettenlaufwerke nicht verlässlich arbeiten.

`map=`*`map_file`*
Gibt den absoluten Pfad und Namen der zu benutzenden map-Datei an; Standardwert ist `/boot/map`.

`message=`*`message_file`*
Gibt den Namen der Datei an, in der am Bildschirm auszugebende Meldungen stehen. Diese Meldungen werden vor dem Boot-Prompt ausgegeben. Ein `0x0C`-Zeichen (Strg-L, Form-Feed) in dieser Datei löscht den Bildschirm, z. B. eine eventuell dort befindliche BIOS-Meldung. Die maximale Größe der Datei beträgt 64 Kbyte. Wenn die Message-Datei geändert wird, muss die map-Datei durch den Aufruf von `lilo` neu erzeugt werden.

`nowarn`
Warnungen bezüglich möglicher Probleme werden nicht ausgegeben.

`optional`
Macht alle Images optional, so dass keine Prüfungen durchgeführt werden.

`password=`*`password`*
: Setzt ein Passwort, das für alle Images oder `other`-Einträge abgefragt wird. Achten Sie darauf, dass normale Benutzer keine Leserechte auf die Datei `/etc/lilo.conf` erlangen und dieses Passwort ausspähen können. `lilo` gibt eine entsprechende Warnung aus.

`prompt`
: Zeigt den Bootprompt an, ohne dass zuvor eine Taste gedrückt werden muss. Wenn zusätzlich `timeout` nicht gesetzt ist, bleibt der Rechner an dieser Stelle stehen und wartet auf eine Eingabe.

`restricted`
: Hebt einige durch `password` gesetzte Begrenzungen wieder auf. Es kann eine beliebige Konfiguration gebootet werden, aber die Eingabe von Parametern ist nicht erlaubt. Mit dieser Option können Sie z. B. den Aufruf des Single-User-Modus durch Unbefugte verhindern.

`serial=`*`parameter_string`*
: Ermöglicht die Systemkontrolle über eine serielle Leitung, d. h. `lilo` akzeptiert Eingaben sowohl über das Keyboard als auch von dem Terminal, das an der seriellen Leitung angeschlossen ist. Das Schicken eines [Break] über die Leitung entspricht dem Drücken der Taste [Umschalt] an der Konsole und bringt den `boot:`-Prompt auf den Bildschirm.

  Wenn der Zugang über die serielle Leitung eingeschaltet ist, sollten alle Konfigurationen mittels `password` gesichert sein, sofern der Zugang über diese serielle Leitung weniger sicher ist als der zur Konsole selbst, z. B. wenn an der seriellen Schnittstelle ein Modem angeschlossen ist. Das Format des *parameter_string* hat die folgende Syntax: *`port,bpsParityBits`*. Die letzten drei Angaben können weggelassen werden. Wenn eine dieser drei Angaben nicht gemacht wird, müssen die nachfolgenden auch entfallen. Wenn nur der *port* angegeben wird, muss das Komma ebenfalls entfallen. Auf meinem System wird `serial=1,9600n8` für ein 9600-Baud-Terminal an `/dev/ttyS1` verwendet.

  Wenn `serial=` gesetzt ist, wird der Wert für `delay` automatisch auf 20 Zehntelsekunden gesetzt. Mit der Einstellung `serial=0,2400n8` wird die erste serielle Schnittstelle mit den Standardwerten initialisiert. Damit kann ein serielles Terminal als einfache Konsole verwendet werden. Für ältere Kernel-Versionen existieren entsprechende Patches.

`timeout=`*`tsecs`*
: bestimmt, wie lange auf Tastatureingaben gewartet wird. Wenn in der eingestellten Wartezeit keine Taste gedrückt wird, wird das Default-Image geladen. Sofern bei `default=` nichts angegeben wurde, ist dies das erste aufgeführte Image. Dementsprechend wird auch die Passwortabfrage abgebrochen, wenn innerhalb dieser Zeit keine Eingabe gemacht wurde. Der Standardwert für `timeout=` ist unendlich.

`verbose=`*`level`*
Hiermit können Fehler- und Diagnosemeldungen für `lilo` eingeschaltet werden. Je größer der Wert ist, desto mehr Meldungen werden ausgegeben. Dieser Eintrag entspricht der Anzahl der `-v`-Optionen beim `lilo`-Aufruf.

Image-bezogene Optionen

Die global wirksamen Optionen können auch innerhalb der image-bezogenen Konfigurationsblöcke benutzt werden. `lilo` bietet über die global wirksamen Optionen hinaus noch folgende weitere Optionen, die nur innerhalb eines image-bezogenen Blocks benutzt werden können:

`append=`*`string`*
Fügt die angegebene Zeichenkette an die Kernel-Kommandozeile an.

`literal=`*`string`*
Verwendet die angegebene Zeichenkette als Kernel-Kommandozeile.

`ramdisk=`*`size`*
Sorgt für die Erstellung einer RAM-Disk in der entsprechenden Größe (in Kbyte).

`read-only`
Veranlasst den Kernel, das `root`-Dateisystem nur im Lesezugriff anzuhängen. Damit kann auch für dieses Dateisystem ein `fsck` durchgeführt werden. Dies ist Standard bei allen Distributionen. Nach dem Prüfen des Dateisystems wird es erneut, diesmal im Schreibzugriff, gemountet. Die Parameter für die Anzahl und den Abstand der Prüfungen können mit dem Programm `tune2fs` konfiguriert werden.

`read-write`
Das `root`-Dateisystem wird in Lese- und Schreibmodus angehängt.

`root=`*`root_device`*
Das angegebene Device wird als `root`-Dateisystem verwendet.

`vga=`*`mode`*
Ermöglicht die Veränderung der Bildschirmauflösung an der Konsole. Mit dem Parameter `vga=ask` hält der Kernel beim Booten an und fragt nach der gewünschten Auflösung. Der dort eingegebene Wert kann dann für den nächsten Systemstart hier angegeben werden. Danach muss noch `/sbin/lilo` ausgeführt werden. Alternativ steht eine Framebuffer-Konsole zur Verfügung, hier wird die Grafikkarte im Grafikmodus betrieben, die Auflösung können Sie ebenfalls per Kernel-Parameter (z. B. `vesafb=` für den VESA-Treiber) angeben.

`initrd=`*`Image`*
Die Datei *`Image`* wird als RAM-Disk geladen. Ist dies nicht gewünscht, kann am `lilo`-Prompt `noinitrd` angegeben werden.

Es ist möglich, für eine Image-Datei mehrere Konfigurationen vorzunehmen, da `lilo` diese nicht im Image, sondern in so genannten Image-Beschreibungen speichert. Für einen Eintrag kann ein weiterer Name mit `alias=`*`Aliasname`* vergeben werden.

lilo und der Windows-Boot-Manager

Die Konfiguration des Windows-Boot-Managers erfolgt in der Datei `BOOT.INI`. Diese Datei ist möglicherweise mit den Attributen »System«, »Read-Only« und »Versteckt« gekennzeichnet. Bevor Sie diese Datei unter Windows bearbeiten können, müssen Sie diese Attribute möglicherweise mit dem Befehl `ATTRIB` entfernen. Die Datei `BOOT.INI` ist eine Textdatei, die den weiteren Verlauf des Systemstarts beschreibt.

Das Listing 3.2 zeigt eine einfache Boot-Konfiguration für Windows NT, Linux und MS-DOS. Dabei ist auf der `C:`-Platte MS-DOS (hier liegt auch die Datei `BOOT.INI`) und auf der `D:`-Partition Windows NT in einem NTFS-Dateisystem installiert. Der Standard-MBR bootet hierbei zunächst die aktive Partition, das ist die `C:`-Platte. Von dort aus wird der weitere Boot-Vorgang gesteuert.

```
timeout=30
default=scsi(0)disk(1)rdisk(0)partition(3)\WINNT
[operating systems]
scsi(0)disk(1)rdisk(0)partition(3)\WINNT="Windows NT 4.0"
scsi(0)disk(1)rdisk(0)partition(3)\WINNT="NT [VGA-Modus]"
                /basevideo /sos
C:\bootsect.lnx="Linux"
C:\ = "MS-DOS"
```

Listing 3.2 Eine einfache Boot-Konfiguration für Windows

Der Boot-Menü-Eintrag für Linux in der vorletzten Zeile verweist auf die Datei `c:\bootsect.lnx`. Diese enthält den von `lilo` erzeugten Boot-Sektor, der dann den Linux-Kernel lädt. Dafür habe ich in der Datei `/etc/lilo.conf` den Eintrag `boot=/mount/dos/bootsect.lnx` vorgenommen, damit der von `lilo` erzeugte Boot-Sektor in diese Datei geschrieben wird (meine DOS-Partition ist unter `/mount/dos` eingehängt). Vor dem ersten Start von `lilo` muss diese Datei existieren und (mindestens) 512 Byte groß sein, da `lilo` eine Sicherheitskopie des alten Boot-Sektors erstellen will. Man kann die Datei mit `dd if=/dev/zero of=/mount/dos/bootsect.lnx bs=512 count=1` erzeugen.

Vorteilhaft ist hier, dass man die Windows-Konfiguration praktisch unverändert lässt und damit viele Probleme umgeht. Der einzige mir bekannte Nachteil ist, dass man mindestens eine Partition haben muss, auf die beide Systeme zugreifen können. `lilo` muss dort den Boot-Sektor ablegen, der Windows-Boot-Manager muss ihn von dort lesen können.

Resümee

Der Boot-Lader `lilo` ist derzeit weit verbreitet, er hat aber einige gewichtige Nachteile. Der wichtigste ist, dass die Konfiguration sehr statisch ist. Nach jeder Änderung an der Konfigurationsdatei `/etc/lilo.conf`, der Message-Datei oder einem der Kernel-Images muss in jedem Fall das Programm `lilo` aufgerufen werden, das dann die entsprechende Map-Datei neu erstellt.

Vorteilhaft ist, dass `lilo` relativ einfach ist, da der Boot-Sektor keinerlei Informationen über den Aufbau des Dateisystems hat. Nachteilig ist, dass man nicht schnell mal einen beliebigen Kernel booten kann. Wenn man im Falle eines Falles alle in `lilo` eingerichteten Images verloren hat, dann wünscht man es sich, einen beliebigen Kernel aus einem beliebigen Verzeichnis starten zu können.

3.4.2 Der Boot-Lader GRUB

Wenn man GRUB (*http://www.gnu.org/software/grub/grub.en.html*) als Boot-Lader verwendet, dann ist es wie bei Workstations möglich, die verschiedensten Kernel zu booten. Ein weiterer Vorteil ist, dass GRUB nicht nur Linux-Kernel booten kann, sondern auch für *BSD-Kernel verwendet werden kann. Sollte man also beide Systeme auf seinem System haben, dann benötigt man nur einen Boot-Lader. Weitere unterstützte Systeme sind Mach und The Hurd, dabei hält sich GRUB an den Multiboot-Standard, so dass für andere Betriebssysteme diese Infrastruktur bereits bereitsteht.

GRUB besteht im Wesentlichen aus einem Boot-Sektor, der den weiteren Lader startet (Stage1), und dem eigentlichen Lader, der in der Lage ist, verschiedene Dateisysteme (wie Extended-2, BSD FFS oder DOS FAT; das Dateisystem wird automatisch erkannt) zu lesen. Neben dem Booten von Unix-Kerneln können auch beliebige andere Partitionen angesprochen werden, so dass man GRUB auch als Boot-Menü und damit zum Booten von Windows oder OS/2 verwenden kann.

Die Konfiguration von GRUB

Beim Booten versucht GRUB zunächst, seine Konfigurationsdatei zu lesen. Wenn eine solche existiert, dann kann der Anwender das zu startende System aus einem Menü auswählen. Existiert keine Konfigurationsdatei, dann wechselt GRUB in die Kommandozeile (diese kann auch aus dem Menü heraus angesprungen werden).

In der Regel wird man die wichtigsten Systeme und Optionen in einer Konfigurationsdatei festlegen. Der Name dieser Datei ist beliebig, er muss nur bei der Installation des Boot-Sektors angegeben werden. Das Listing 3.3 zeigt eine einfache Konfiguration; eine vollständige Dokumentation der Optionen finden Sie in den Quellen zu GRUB.

```
# Wartezeit bis zum Start des Default-Image
timeout 30

# Booten des Linux-Kernels von /dev/hda2
title Linux Boot
root       (hd0,1)
kernel     (hd0,1)/boot/vmlinuz-2.0.34

# Booten von Windows NT
title Windows NT boot menu
root           (hd0,1)
makeactive
chainloader    +1
```

Listing 3.3 Eine einfache GRUB*-Konfiguration*

Die Konfigurationsdatei für GRUB kann unter Unix genauso bearbeitet werden wie unter DOS oder Windows. Wichtig ist nur, dass es eine Textdatei ist. Die Kodierung des Zeilenendes ist mit CR+LF oder mit LF erlaubt. Kommentare beginnen mit einer Raute (#) in Spalte eins.

GRUB hat, genau wie lilo, globale und lokale Optionen. Globale gelten für GRUB selbst bzw. für alle Images. Die globalen Optionen sind:

`timeout` *Sekunden*
: Dieser Eintrag setzt den Timeout, die Zeit in Sekunden, die GRUB wartet, bevor das Default-Image gestartet wird.

`default` *Nummer*
: Bestimmt das Image, das gestartet wird, wenn der Benutzer keine Eingabe vornimmt. Die Zählung der Images beginnt bei Null, so dass der Default von Null das erste Image bootet.

`fallback` *Nummer*
: Wenn beim Booten des ausgewählten Image ein Fehler aufgetreten ist, dann versucht GRUB, das Fallback-Image zu starten. Das funktioniert natürlich nur, wenn der Fehler noch innerhalb von GRUB auftritt.

`password` *Passwort neue_Konfiguration*
: Ohne diesen Eintrag kann jeder die einzelnen Images verändern oder neue Einträge generieren. Außerdem ist es möglich, im Kommandomodus einen beliebigen Kernel zu wählen und mit beliebigen Parametern zu booten. Dieser Eintrag macht diese Funktionen unzugänglich, solange nicht das korrekte Passwort eingegeben wird. Danach wird eine neue Konfigurationsdatei verwendet, in der man beispielsweise bereits Images für den Single-User-Modus definieren kann. Mit der Option `--md5` wird das Passwort mittels MD5 verschlüsselt gespeichert. Ein neues Passwort können Sie mit dem GRUB-Kommando `md5crypt` verschlüsseln und dann hier eintragen.

Jedes Image beginnt mit dem Eintrag `title` und wird implizit durch den Befehl `boot` beendet. Die Image-bezogenen Optionen sind:

`root` *`Partition`* [*`hdbias`*]
: Verwendet die Partition *`Partition`* als root-Partition. Diese Partition wird von `GRUB` gelesen, um weitere Daten an den nächsten Lader weiterzugeben. Wenn die root-Partition nicht mit BIOS-Mitteln gelesen werden kann, weil sie z. B. hinter der 1-Gbyte-Grenze liegt, dann können Sie den Eintrag `rootnoverify` verwenden, der auf das Lesen der Partition verzichtet. Der optionale Parameter `hdbias` ist nur für BSD-Systeme interessant.

`kernel` *`Kernel-Image ...`*
: `GRUB` lädt das angegebene Kernel-Image, das im Multiboot `a.out`-, ELF-, Linux- oder *BSD-Format vorliegen kann. Das Kommando selbst ignoriert den Rest der Zeile, dieser wird an den Kernel als Kommandozeile übergeben.

`initrd` *`Datei ...`*
: Lädt die initiale RAM-Disk für einen Linux-Kernel und übergibt die weiteren Parameter an den Kernel.

`makeactive`
: Markiert die als `root` angegebene Partition als »aktiv«. Das ist Voraussetzung für einige Betriebssysteme. Es können nur primäre Partitionen als aktiv markiert werden.

`chainloader`
: Lädt die Datei oder den angegebenen Sektor als Chain-Loader. `GRUB` übergibt hierbei die Kontrolle z. B. an einen anderen Boot-Lader wie den von OS/2 oder Windows.

`pause` *`Text`*
: Der angegebene Text wird auf den Bildschirm ausgegeben und `GRUB` hält danach an. Ein `^G` im Text sorgt dafür, dass ein Beep ausgegeben wird.

`uppermem` *`Kilobytes`*
: Normalerweise ist `GRUB` in der Lage, auch mehr als 64 Mbyte Speicher zu erkennen und dem geladenen System mitzuteilen. Nur bei älteren Rechnern sollte es notwendig sein, mit Hilfe dieser Option die korrekte Speichergröße einzustellen.

`boot`
: Startet das angegebene Image.

`module` *`Datei ...`*
: Lädt ein Multiboot-Modul. Module können komprimiert gespeichert werden und werden automatisch dekomprimiert.

`modulenounzip` *`Datei ...`*
: Wie die Option `module`, es wird aber nie die automatische Dekompression versucht.

GRUB hat noch keine Dateisysteme gemountet, kann aber Zugriffe auf Dateien über die Strukturen der verschiedenen Dateisysteme durchführen. Daher muss zu jedem Dateinamen zusätzlich der Name der Partition, auf der diese Datei gespeichert ist, angegeben werden. GRUB unterstützt alle unter Linux verbreiteten Dateisysteme. Die Syntax für die Angabe einer Partition ist:

(*Disk*[,*Partition*[,*BSD-Slice*]])

Als Disk kann `fd0` oder `fd1` für das erste bzw. zweite Diskettenlaufwerk verwendet werden; dann darf keine Partition angegeben werden. Wenn man `hd0`/`hd1` verwendet, kann der MBR angesprochen werden, indem keine Partition angegeben wird. Bei den Partitionen beginnt die Zählung wieder mit Null, so dass die erste Partition der ersten Festplatte als `(hd0,0)` bezeichnet wird. Diese Bezeichnungen sind durch die Device-Namen in den freien BSD-Varianten beeinflusst und daher nicht die unter Linux gewohnten.

Zusätzlich zu den normalen Partitionen, wie sie von DOS, Windows, OS/2 und Linux verwendet werden, kann GRUB auch mit den Slices der *BSD-Systeme umgehen. Dabei wird an die Partitionsnummer der Buchstabe angehängt, der die entsprechende Slice identifiziert.

Wenn GRUB einen Dateinamen erwartet, dann können Sie diesen direkt an die Partition als absoluten Pfad anfügen. Bei einigen Funktionen ist es außerdem möglich, einen Offset oder eine Sektornummer anzugeben (mit `+`*`Sektornummer`*).

Wenn Sie die Kommandozeile von GRUB benutzen, dann werden Sie die Vervollständigung zu schätzen lernen. Der Editor ist an die Funktionen und die Tastenbelegung der `bash` angelehnt, aufgrund der geringen Größe aber nicht ganz so leistungsfähig. Die Tab-Taste zeigt entweder eine Liste der möglichen Befehle an oder mögliche Eingaben.

Die wichtigsten Kommandos für die interaktive Nutzung sind `kernel` zur Auswahl des zu startenden Kernels und `boot` zum Starten des ausgewählten Kernels. Beachten Sie, dass im Gegensatz zu einigen Boot-PROMs an Workstations die Kernel-Parameter nach der `kernel`-Option anzugeben sind. Das Kommando `boot` kennt keine weiteren Optionen.

Die Installation von GRUB

Die Installation von GRUB ist etwas schwieriger als diejenige von `lilo`. Bei `lilo` wird zunächst eine Konfigurationsdatei (`/etc/lilo.conf`) erstellt und diese dann von einem normalen Linux-Programm ausgewertet. Viele Distributionen enthalten ein bei der Installation automatisch startendes Konfigurationstool für `lilo`, so dass der Anwender zunächst wenig mit der Technik zu tun hat.

Das ist bei GRUB anders. Bisher enthält praktisch keine Distribution diesen Boot-Lader, außerdem fehlen Tools zur einfachen Konfiguration. Zudem besteht GRUB

nur aus den entsprechenden Boot-Sektoren, so dass jede Konfiguration ausschließlich beim Booten vorgenommen werden kann. Mit einigen Tricks kann man sich das Leben allerdings wieder etwas vereinfachen.

Am besten beginnen Sie damit, eine Boot-Diskette mit GRUB zu erstellen. Das kann unter Linux mit den in Listing 3.4 dargestellten Befehlen geschehen. Anschließend können Sie von dieser Diskette booten. Sie landen zunächst in der Kommandozeile. Hier können Sie ein Kernel-Image auswählen und booten oder GRUB installieren.

```
(linux):# dd if=bin/stage1 of=/dev/fd0 bs=512 count=1
(linux):# dd if=bin/stage2 of=/dev/fd0 bs=512 seek=1
```

Listing 3.4 Erstellen einer generischen GRUB-Diskette

Die weitere Installation erfolgt mit dem Befehl `install`. Wenn Sie den Rechner mit der generischen Boot-Diskette gestartet haben, dann müssen Sie den vollständigen Befehl eingeben. Wenn Sie bereits eine vorkonfigurierte GRUB-Installation haben, dann können Sie diesen Befehl auch in die Konfigurationsdatei aufnehmen. Die Syntax des `install`-Befehls lautet:

`install` *`Stage1`* `[d]` *`Ziel Datei Adresse`* `[p] [`*`Konfiguration`*`]`

GRUB wird die Datei *`Stage1`* laden und prüfen, ob diese eine gültige GRUB-Datei ist. In diese Datei wird die Block-Liste der Stage2-Datei eingefügt und das Ergebnis als *Ziel* installiert. Die Adresse gibt an, in welchen Hauptspeicherbereich der Stage2 geladen werden soll. Unter Linux wird man 0x8000 verwenden, bei FreeBSD 0x2000, wenn man den Stage1_5-Lader verwendet. Wenn die Option `p` angegeben ist oder eine Konfigurationsdatei angegeben wurde, dann wird in den ersten Sektor der Stage2-Datei der Name dieser Datei aufgenommen.

Das klingt alles sehr kompliziert, ist aber nur halb so wild. Nehmen Sie einen Stift und einen Zettel und schreiben Sie sich den Befehl auf. Die Befehle in Listing 3.5 habe ich verwendet, um GRUB zunächst nur auf einer Diskette zu installieren.

```
install (fd0)+1 (fd0) (hd0,2)/boot/grub/stage2 0x8000 p \
                      (hd0,2)/boot/grub/menu.lst
```

Listing 3.5 Installation von GRUB auf Diskette

Die Parameter im Einzelnen bedeuten:

`(fd0)+1`
: Lese den ersten Sektor vom Diskettenlaufwerk und verwende diesen als Boot-Sektor.

`(fd0)`
: Schreibe den Boot-Sektor wieder auf das Diskettenlaufwerk zurück, nachdem dort die Sektornummern der Stage2-Datei gespeichert wurden. Bei der Installation auf Festplatte geben Sie hier das Ziel an; entweder `(hd0)` für den MBR der ersten Platte oder `(hd0,`*`Partition`*`)` für die entsprechende Partition.

`(hd0,2)/boot/grub/stage2`
: Verwende diese Datei aus der angegebenen Partition als Stage2-Lader. Die Sektornummern dieser Datei werden im Boot-Sektor der ersten Stage vermerkt. Der Programmcode hier liest dann eventuell die Konfigurationsdatei, lädt den gewünschten Kernel usw.

`0x8000`
: Lade die Stage2 an die Adresse 0x8000. Unter Linux wird man keinen anderen Wert verwenden, unter FreeBSD kann es sinnvoll sein, Stage1_5 einzusetzen und die Adresse 0x2000 zu verwenden.

`p`
: Vermerke den Namen der Konfigurationsdatei im Stage2-Lader.

`(hd0,2)/boot/grub/menu.lst`
: Die Datei `/boot/grub/menu.lst` auf der Partition `/dev/hda3` wird als Konfigurationsdatei verwendet.

Das große Problem bei der Installation von GRUB ist, dass kein Linux-Programm die Konfiguration vorbereitet, so dass man zumindest einmal den komplizierten `install=`-Befehl eingeben muss. Später kann man diesen Befehl allerdings in die Konfigurationsdatei aufnehmen, so dass er vor dem eigentlichen Booten ausgeführt wird.

Ein weiteres Feature von GRUB ist, dass zum Booten des Kernels auch BOOTP oder DHCP verwendet werden kann. Genaueres dazu finden Sie in der Info-Dokumentation zu GRUB.

Lohnt sich GRUB wirklich?

GRUB hat einige sehr nützliche Vorteile, vor allem die Verwendbarkeit für Linux und *BSD-Systeme und die sehr flexible Konfiguration zur Laufzeit. Dafür muss man aber auch einige Nachteile in Kauf nehmen, insbesondere die komplexe Konfiguration und die geringe Unterstützung durch Tools und Distributionen. Dennoch kann sich der Einsatz von GRUB lohnen. Im Zweifelsfall installieren Sie GRUB auf einer Diskette und probieren es aus.

3.4.3 Der Boot-Lader LOADLIN

LOADLIN und loadlinX von Hans Lermen sind Programme, die unter MS-DOS laufen und es erlauben, Linux von MS-DOS aus zu starten. loadlinX[2] ist ein Präprozessor für LOADLIN, der eine Umsetzung der Kommandozeilenparameter vornimmt. Zusammen mit dem UMSDOS-Dateisystem ermöglicht LOADLIN ein »Reinschnuppern« in Linux, ohne dass eine Neupartitionierung der Festplatte notwendig wird. Diese Möglichkeit stellt insbesondere für Linux-Neulinge eine einfachere und sicherere Alternative zu lilo dar. LOADLIN ist auch dann nützlich, wenn etwa eine Soundkarte zunächst unter DOS mit einem speziellen Programm initialisiert werden muss, bevor sie unter Linux benutzt werden kann.

Um LOADLIN zu benutzen, muss zunächst Linux installiert werden (manche Distributionen verwenden LOADLIN auch auf den Bootmedien). Der bei den meisten Distributionen folgende Schritt, lilo in den Boot-Sektor zu installieren, wird jedoch übersprungen. Das Erstellen einer Boot-Diskette hingegen muss erfolgen, damit Linux zunächst von der Diskette gebootet werden kann. Der Kernel (/vmlinuz) wird dann auf die DOS-Partition kopiert. Sofern die DOS-Partition noch nicht durch die Installationsroutine gemounted wurde, kann dies z. B. via mount -t msdos /dev/hda1 /mnt per Hand nachgeholt werden.

Das Programm LOADLIN bekommt als Parameter den DOS-Dateinamen des zu startenden Kernel-Image übergeben, z. B. c:\linux\vmlinuz. Das zweite Argument, z. B. root=/dev/hda2, enthält die als root-Dateisystem zu benutzende Partition. Nun wird nur noch die Angabe benötigt, ob die root-Partition read-only (ro) oder read-write (rw) eingehängt wird. Eine typische Kommandozeile sehen Sie in Listing 3.6.

```
C:\> c:\linux\loadlin c:\linux\vmlinuz root=/dev/hdb2 rw
```

Listing 3.6 Ein typischer Aufruf von loadlin

Wenn UMSDOS als Dateisystem verwendet wird, ist zu beachten, dass anstelle von LOADLIN der Präprozessor loadlinX benutzt werden sollte. Bei der Verwendung von loadlinX wird im Parameter root= als Argument nur noch das zu benutzende Laufwerk, z. B. die primäre DOS-Partition C: oder auch eines der Diskettenlaufwerke A: oder B:, übergeben. Eine Kommandozeile unter Verwendung von loadlinX sehen Sie in Listing 3.7.

```
C:\> c:\linux\loadlinX c:\linux\vmlinuz root=c: rw
```

Listing 3.7 Ein typischer Aufruf von loadlinX

2. Auch DOS 6.2 unterscheidet nicht zwischen Groß- und Kleinschreibung bei Befehlen; das große X in loadlinX dient nur zur Hervorhebung der unterschiedlichen Programmnamen.

Zu beachten ist, dass bei der Benutzung von UMSDOS das schreibgeschützte Einfügen des Dateisystems (`ro`/`read-only`) nicht möglich ist. Spezielle Boot-Parameter nur für `LOADLIN` sind nicht vorhanden.

Ein wesentlicher Vorteil von `LOADLIN` ist, dass weder der MBR noch der Startsektor der Linux-Partition verändert werden muss. Damit kann man problemlos zwischen den verschiedenen Systemen wechseln und diese auch aktualisieren. Hat man `lilo` in den MBR installiert, so wird er dort z. B. von der NT- oder Windows 95-Installation überschrieben. Außerdem stört er dort, wenn man Linux wieder entfernt.

Um das Laden mit `LOADLIN` zu automatisieren, können Sie unter DOS einen Boot-Selektor wie `BOOT.SYS` verwenden. Unter Linux sollten Sie ein Skript `/sbin/installkernel` erstellen, das den Kernel beim `make install` auf die DOS-Partition kopiert.

3.4.4 Der Boot-Lader SYSLINUX

`SYSLINUX` von H. Peter Anvin ist ein Boot-Lader, der den MS-DOS-Boot-Sektor bzw. die Dateien `IO.SYS` und `MSDOS.SYS` auf Disketten ersetzt. `SYSLINUX` selbst ist ein MS-DOS-Programm, mit dem aus MS-DOS-formatierten Disketten Linux-Boot-Disketten erstellt werden können. Unter DOS (oder mittels `mcopy`) werden eine oder mehrere Kernel-Dateien auf die MS-DOS-Diskette kopiert. Anschließend werden mit dem DOS-Befehl aus Listing 3.8 das Image in der Datei `a:vmlinux` als zu ladendes Kernelimage und die Boot-Parameter `root=/dev/hda1` als zu übergebendes Argument eingetragen. `SYSLINUX` ändert den Boot-Sektor der Diskette und überträgt die Datei `LDLINUX.SYS` darauf.

```
c:\> syslinux a: vmlinuz root=/dev/hda1
```

Listing 3.8 Ein typischer Aufruf von `syslinux`

Beim Starten von einer mittels `syslinux` erzeugten Boot-Diskette wird, ähnlich wie bei `lilo`, nach dem Drücken einer der Tasten [Umschalt], [Alt], [Umschalt-Fest] oder [Rollen] ein Boot-Prompt angezeigt. An dieser Stelle ist es möglich, andere Parameter als den Default-Boot-Parameter und einen anderen als den fest eingestellten Kernel-Dateinamen anzugeben, sofern er auf der Diskette vorhanden ist.

Falls auf der Diskette eine Datei namens `LINUXMSG.TXT` existiert, wird deren Inhalt beim Starten angezeigt. In diesem Fall wird auch der Boot-Prompt ausgegeben, auch wenn keine der oben genannten Tasten gedrückt wurde.

Mit `syslinux` erstellte Boot-Disketten können ohne Probleme mit dem DOS-Programm `DISKCOPY` kopiert werden. Da jedoch immer mehr Distributionen mit boot-fähigen CDs ausgestattet sind, wird dieses Verfahren nur noch relativ selten verwendet.

3.5 Start des Kernels

Nachdem einer der Boot-Lader das Kernel-Image, z. B. `/vmlinuz`, von der aktiven Partition in den Hauptspeicher geladen hat, muss der Kernel zunächst dekomprimiert werden. Der zusätzliche Aufwand, den die CPU hierbei leisten muss, fällt nicht ins Gewicht, da sie zu dieser Zeit überwiegend »Däumchen dreht« und die meiste Zeit auf das Eintreffen von Daten aus dem »Flaschenhals« DiskIO wartet. Insgesamt ist das Laden komprimierter Daten sogar schneller. Würde demgegenüber in einem System die CPU den Flaschenhals darstellen, so wäre das Benutzen unkomprimierter Daten günstiger; beim gegenwärtigen Stand der Bus- und Festplattentechnik ist dies aber in absehbarer Zeit nicht zu erwarten. Ein weiterer, in diesem Fall aber nicht so gewichtiger Vorteil der Kompression ist die Ersparnis von Plattenplatz.

Die Verwendung komprimierter Kernel wurde notwendig, da sie unkomprimiert größer als 704 Kbyte wurden (640 Kbyte »konventioneller DOS-Speicher« zuzüglich 64 Kbyte ungenutzter Datenbereich der Grafikkarten unter DOS) und der Prozessor sich zu diesem Zeitpunkt noch im Real Mode befindet, in dem nur Speicheradressen unterhalb von 1 Mbyte zur Verfügung stehen.

Die Einführung der komprimierten Kernel hat das eigentliche Problem nur verdeckt. Durch die vielen Treiber, die in den Kernel eingebunden werden können, stößt man jetzt wieder auf die Beschränkungen des Real-Modes, da der Kernel zunächst in den Speicher unterhalb 1 Mbyte geladen werden muss. Früher oder später wird der Boot-Lader im Protected-Mode laufen, so dass die Größe des Kernels nur noch durch den realen Hauptspeicher begrenzt ist.

Daher hat man mit dem `bzImage` und der initialen RAM-Disk `initrd` Möglichkeiten geschaffen, um diese Grenzen zu umgehen. Wenn Ihr Kernel wirklich so groß geworden ist und Sie keine Module verwenden können, so finden Sie die notwendigen Informationen in der Datei `./Documentation/initrd.txt` in den Kernel-Quellen bzw. in der `lilo`-Dokumentation.

Die letzte Amtshandlung des Boot-Laders ist die Übergabe der Programmkontrolle an das entpackende (und logischerweise selbst nicht komprimierte) »Präfix«, das beim Kompilieren des Systems mittels der in dem Verzeichnis `/usr/src/linux/arch/i386/boot/compressed` liegenden Programme `xtract` und `piggyback` angefügt wurde. Der Prozessor wird jetzt in den Protected-Mode umgeschaltet, damit die Dekompressionsroutine das Image auf (virtuelle) Adressen ab drei Gigabyte (`0xC0000000`) entpacken kann. Der Programmzähler wird auf diese Adresse gesetzt. Somit wird die Programmkontrolle ein weiteres Mal, und diesmal an den eigentlichen Kernel, übergeben.

Unter dem Aspekt der Konfigurationsmöglichkeiten betrachtet, können beim Systemstart nach der Auswahl des Betriebssystems zwei Abschnitte unterschieden werden:

- Ermittlung und Überprüfung der Hardware-Ausstattung und
- Abarbeiten der Konfigurationsdateien und automatisches Starten von Programmen.

Im ersten Abschnitt (siehe: `arch/i386/boot/setup.S`) werden bereits vor dem Entpacken des Kernels Teile der Hardware-Ausstattung ermittelt. Dazu werden die entsprechenden BIOS-Variablen (für Disketten und Festplattenparameter sowie die angeschlossene Grafikkarte) ausgelesen und an einem sicheren Platz im Hauptspeicher vermerkt. Weitere Hardware-Tests erfolgen nach der Umschaltung in den 32-Bit-Modus, dazu jedoch später mehr (zur exakten Reihenfolge vgl.: `./linux/init/main.c`). Zunächst werden jedoch interne Funktionen und Tabellen des Systems initialisiert:

- Die Tabelle für die Speicherseitenverwaltung,
- die Belegung der Fehlerinterrupts und der IRQs,
- die Prozesstabelle.

Anschließend wird der Scheduler gestartet und der Bildschirm initialisiert. Ab hier werden Meldungen nicht nur auf den Bildschirm ausgegeben, sondern auch mittels des `syslogd` verwaltet und möglicherweise in einer Datei gespeichert. Mehr zur Verwendung von `syslogd` finden Sie im Abschnitt 4.4.4, »System-Logs«. Diese Meldungen können später mit dem Programm `dmesg` erneut angezeigt werden.

An dieser Stelle erfolgt, im Zusammenhang mit der Initialisierung der entsprechenden Gerätetreiber, die bereits erwähnte zweite Phase des Hardware-Tests. Die Hardware wird mittels verschiedener Tests untersucht (so genanntes Auto-Probing oder Auto-Detect). Anhand der Bildschirmmeldungen können Sie verfolgen, welche Geräte gefunden wurden. Wird hier für eine Controller-Karte keine oder eine falsche Meldung (z. B. andere IO-Ports oder Interrupts) ausgegeben, so werden Sie dieses Gerät später nicht ansprechen können. Eine Ursache für solche Probleme kann darin bestehen, dass mehrere Karten auf einen Interrupt oder dieselben IO-Ports eingestellt sind (entweder mittels Jumper oder Software-Konfiguration). Die automatische Hardware-Erkennung kann mit Hilfe von Kommandozeilenparametern für den Kernel beeinflusst werden. Lesen Sie dazu auch die aktuellen `README`-Dateien in den Kernel-Quellen.

Jetzt wird der Prozessor vom privilegierten (auch Kernel-Mode) in den nicht privilegierten Modus[3] (User-Mode) geschaltet; ab jetzt läuft der erste Linux-Prozess (mit der »Prozessnummer« 0). Dieser Prozess unterscheidet sich nicht grundsätzlich von allen anderen Linux-Prozessen, läuft aber nur dann, wenn kein anderer Prozess aktiv ist. Es handelt sich beim Prozess 0 um den so genannten `idle`-Prozess, der im Fall der Prozessor-Nichtnutzung 100% der Systemressourcen belegt.

3. Der privilegierte Modus ist nicht zu verwechseln mit dem Protected-Mode.

Vom Prozess 0 aus wird mittels des Systemaufrufs `fork()`[4] der Prozess 1 gestartet, der wiederum die weitere Initialisierung des Systems in Gang setzt. Hier ist die Hardware im Prinzip initialisiert, die `root`-Partition angehängt und das Programm `init` wird ausgeführt. `init` wird bei Unix-Systemen häufig als »Vater aller Prozesse« bezeichnet, was für Linux nicht ganz korrekt ist; eigentlich ist es hier der `idle`-Prozess mit der Nummer 0. Es wird noch ein weiterer Prozess gestartet, der für das regelmäßige Update der internen Informationen des Dateisystems auf der Festplatte sorgt. Das Verhalten dieses Prozesses kann mit dem Programm `bdflush` verändert werden.

Gemäß des Filesystem-Hierarchie-Standards (FHS-Version-2.0) sollte sich das `init`-Programm in `/sbin/init` befinden. Die tatsächliche Suchreihenfolge, die in `linux/init/main.c` festgelegt ist, lautet jedoch `/sbin/init`, `/etc/init` und `/bin/init`. Wenn `init` dort nicht gefunden wird, wird versucht, `/bin/sh` zu starten, um die Instandsetzung eines derart »unordentlichen« Systems zu ermöglichen. Mit aktuellen Kernel-Versionen ist es möglich, mit dem Kommandozeilen-Parameter `init=/bin/sh` eine Shell anstelle von `init` zu starten. Achten Sie hier darauf, dass dieser Befehl mit `root`-Privilegien gestartet wird, und verwenden Sie, falls erforderlich, die Option `password=` in der `/etc/lilo.conf`.

Zum Abschluss der Initialisierung meldet sich der Kernel mit seiner Versionsnummer sowie seinem Erstellungsdatum. Zuvor wird jedoch das `init`-Programm gestartet. Damit ist der Systemstart aus der Sicht des Kernels abgeschlossen. Das System ist aber zu diesem Zeitpunkt noch nicht benutzbar, da keine Anmeldung erfolgen kann und keine Hintergrundprozesse (Dämonen) gestartet sind.

Das Verfahren wird etwas komplizierter, wenn ein Kernel mit der `initrd`-Option gestartet wird. In diesem Fall wird vor dem Start von `init` die initiale RAM-Disk gemountet und das dort enthaltene Programm `linuxrc` gestartet. Damit ist es möglich, einen vollständig modularisierten Kernel zu verwenden, der auch die Festplattentreiber dynamisch lädt. In Zukunft wird man versuchen, viele Funktionen des Auto-Probing hier unterzubringen.

3.5.1 Parameter für den Kernel

Viele Initialisierungen (z. B. IO-Adressen für Erweiterungskarten) des Kernels können durch Kommandozeilenparameter beeinflusst werden. Diese Parameter werden dem Kernel vom Boot-Lader übergeben.

4. Dieses fork() arbeitet insofern anders, als der Prozess 0 eine Sonderrolle einnimmt: Er arbeitet mit demselben Speicherbereich wie Prozess 1, insbesondere mit demselben Stack. Deshalb führt Prozess 0 auch nur Operationen aus, die den Stack nicht benutzen. Zitiert nach [Beck2001].

`debug`
: Setzt `console_loglevel` auf 10 und schaltet somit die ausführlichen Konsolenmeldungen ein.

`mem=`*`size`*
: Setzt das Ende des physikalischen Speichers. Dies kann notwendig sein, wenn die RAM-Größe nicht automatisch ermittelt werden kann, z. B. bei Rechnern mit mehr als 64 MB RAM. Es kann auch nützlich sein, um defekten Speicher nicht zu verwenden, ohne diesen auszubauen oder die Performance eines Systems mit weniger Speicher zu prüfen.

`no387`
: Ein eventuell vorhandener mathematischer Coprozessor wird nicht benutzt.

`no-hlt`
: Verhindert das Benutzen bzw. Ausführen der HLT-Instruktion neuerer Intel-(kompatibler-)Prozessoren, wenn das System idle ist. Durch die Verwendung dieses Befehls wird die CPU weniger heiß und Laptops verbrauchen deutlich weniger Strom. Allerdings kann diese Einstellung bei manchen Rechnern dazu führen, dass diese beim Booten einfach hängen bleiben.

`reserve=`*`port1,num1[,portn,numn]`*
: Nimmt IO-Ports und -Bereiche vom Auto-Probing aus. Dies ist zum Beispiel dann notwendig, wenn das Auto-Probing zum »Hängen« des Systems führt. *port1* stellt die erste Adresse dar, *num1* gibt die Anzahl der ab *port1* auszublendenden Adressen an.

`root=`*`device`*
: ändert das Gerät, das als `root`-Dateisystem verwendet wird. Hiermit werden die Angaben im Kernel-Image überschrieben. *device* ist entweder ein Gerätename wie etwa `/dev/hda3` oder eine hexadezimale Gerätenummer. Das Root-Device kann auch mit Hilfe des Programms `rdev` in das Kernel-Image geschrieben werden.

`ro` bzw. `rw`
: Die root-Partition wird read-only oder read-write in den Verzeichnisbaum eingefügt.

3.6 Tipps und Tricks zur Boot-Konfiguration

Es ist leicht, das System unbenutzbar zu machen, wenn man an der Boot-Konfiguration herumbastelt. Leider sind derartige Fehler nicht einfach zu beheben, so dass man hierbei wirklich vorsichtig sein sollte. Auf den folgenden Seiten möchte ich einige Hinweise geben, wie man derartige Fehler verhindern bzw. wie man sein System retten kann.

Der erste und wichtigste Tipp ist, dass man immer ein zweites Boot-Medium greifbar haben sollte. Das kann entweder die Boot-Diskette (oder CD-ROM) der Distribution sein oder eine selbsterstellte Boot-Diskette. Wenn Sie einen eigenen Kernel erstellen (siehe auch Abschnitt 4.2.1, »Konfigurieren des Kernels«), dann können Sie den neuen Kernel zunächst mittels `make bzdisk` auf eine formatierte Diskette schreiben und von dieser booten.

Wenn Sie `lilo` verwenden, dann sollten Sie zusätzlich zu den beiden Kerneln `/boot/vmlinuz` und `/boot/vmlinuz.old`, die bei der Kernel-Installation mit `make install` bzw. `make bzlilo` automatisch erzeugt werden, eine zusätzliche Version aufheben. Wichtig ist, dass nach jeder Änderung an der Datei `/etc/lilo.conf` oder einem Kernel-Image unbedingt neu aufgerufen werden muss. Warum? `lilo` merkt sich beim Aufruf die Sektoren, die das Kernel-Image enthalten, in der Map-Datei und diese Sektoren ändern sich.

Bevor Sie `lilo` tatsächlich einsetzen, können Sie mit der Option `boot=/dev/fd0` in der Datei `/etc/lilo.conf` dafür sorgen, dass `lilo` nur den Boot-Sektor auf der Diskette verändert. Wenn Sie damit zufrieden sind, dann können Sie später den Boot-Sektor an eine geeignete Stelle auf der Festplatte kopieren (bzw. die `lilo`-Konfiguration ändern und `lilo` erneut aufrufen). Ein wichtiger Vorteil ist hier, dass Sie nicht in die Konfiguration Ihres laufenden Systems eingreifen, von Nachteil ist allerdings, dass Sie von Diskette booten müssen (mit der üblichen Gefahr von Boot-Sektor-Viren). Ich persönlich setze diese Funktion gerne zum Test einer neuen Konfiguration ein.

Wenn Sie die Wahl haben, wo Sie den `lilo`-Boot-Sektor installieren möchten, dann verwenden Sie möglichst den Boot-Sektor einer Linux-Partition und markieren Sie diese als »aktiv« (boot-fähig). Damit lassen Sie den Master-Boot-Record unverändert und können mittels `fdisk` schnell ein anderes System aktivieren. In der `lilo`-Konfiguration können dann andere Systeme zur Auswahl angeboten werden.

Wenn Sie bereits Windows auf Ihrem System haben und den Boot-Manager dieser Systeme verwenden wollen, dann geht das natürlich auch. Das hat den Vorteil, dass man das bisherige Boot-Verfahren praktisch unverändert lassen kann. Mehr zu diesem Thema finden Sie im nächsten Abschnitt.

Wenn Sie Boot-Probleme haben, dann ist das wichtigste Ziel, keine Daten zu verlieren und möglichst schnell wieder ein lauffähiges System zu erhalten. Je nach Fehlerquelle und Situation sind verschiedene Strategien erforderlich. Die folgende Liste versucht, einen Überblick über die notwendigen Schritte zu geben – ein Kochrezept für jede Situation kann es aber nicht sein. Seien Sie vorsichtig!

- Bei Fehlern in den Boot-Skripten, die von `init` aufgerufen werden, kann es ausreichen, im Single-User-Mode zu booten und dann das Skript zu korrigieren oder zu deaktivieren.

- Wenn das Programm `init` nicht mehr startet, dann kann man möglicherweise mit der Kernel-Option `INIT=/bin/sh` eine Shell starten und mit Reparaturmaßnahmen beginnen. Vielleicht haben Sie die Stand-Alone-Shell `sash` installiert – diese enthält viele Befehle als Builtins, die man zur Rettung gebrauchen kann.
- Falls der aktive Kernel nicht bootet, so kann man möglicherweise eine ältere Kernel-Version (`/boot/vmlinuz.old`) verwenden. Achtung: Wenn Sie den neuen Kernel einfach mittels `make bzlilo` neu übersetzen, überschreiben Sie auch die vorherige Sicherungskopie.
- Das System bootet gar nicht: Verwenden Sie eine Boot-Diskette oder die CD-ROM der Distribution.

In vielen Fällen werden nicht alle Partitionen eingehängt sein, dann können Sie diese manuell mit einhängen. Wenn die Root-Partition nur im Lesezugriff eingehängt ist, so müssen Sie diese erneut mit den richtigen Optionen einhängen. Auf meinem System geht das mit `mount -n -o remount,rw /dev/`*`root`*, wobei ich als Root-Device `sda2` verwende.

3.7 Der init-Prozess

Das gestartete `init`-Programm arbeitet seine Konfigurationsdatei `/etc/inittab` ab und startet die darin eingetragenen Programme und Prozesse. Wenn kein `init`-Programm gefunden wird, versucht der Kernel, eine Shell zu starten und `/etc/rc` darin auszuführen. Das Format der `/etc/inittab` ist bei `simpleinit` und System-V-`init` unterschiedlich.

Das Mischen verschiedener `init`s und `inittab`s kann das System unbrauchbar machen. Es ist in jedem Fall äußerst wichtig, vor der Umstellung einer `init`-Methode auf eine andere Backups der betroffenen Dateien und Programme zu machen sowie eine Boot- und Root-Diskette einschließlich Editor zu erstellen. Eine Umstellung ist nur dann sinnvoll, wenn Sie sich einen deutlichen Gewinn versprechen. Ein guter Grund ist die Verwendung von `simpleinit` auf einem Rechner mit sehr wenig Speicher, allerdings wird man dieses nur sehr selten wirklich tun. Aus diesem Grund werde ich mich hier auf das System-V-`init` konzentrieren.

3.7.1 Parameter für init

Folgende Parameter werden von allen unter Linux verwendeten `init`-Varianten erkannt. Diese Parameter können vom Kernel an `init` weitergereicht werden.

`single`
: Startet das System im Single-User-Modus. Alle für Linux verfügbaren `init`-Programme verstehen mindestens den Parameter `single`.

`auto`
Startet das System so, wie es in der Konfigurationsdatei `/etc/inittab` festgelegt wurde.

3.7.2 Die init-Konfiguration in der Datei /etc/inittab

Das Programm `init` wird mit Hilfe der Datei `/etc/inittab` konfiguriert. Ein `inittab`-Eintrag sieht sowohl beim echten System-V-`init` als auch beim System-V-ähnlichen `init` schematisch wie folgt aus:

id:*runlevel*:*action*:*process*

`id`
Steht für einen eindeutigen Code zur Identifizierung eines Eintrags. Dieser Code darf bis zu vier Zeichen enthalten.

`runlevel`Runlevel
Kann Werte von 1 bis 6 sowie S und A bis C annehmen (Groß- bzw. Kleinschreibung wird ignoriert). S steht z. B. für den Single-User-Modus, in dem der Systemadministrator üblicherweise Systemarbeiten durchführt. Wenn `init` als `telinit` aufgerufen wird, stehen darüber hinaus auch noch die Runlevel A, B und C zur Verfügung, mit denen spezielle Einstellungen via `inittab` aufgerufen werden können. Dies kann zum Starten bzw. Stoppen von verschiedenen Diensten genutzt werden.

`action`
Gibt an, was in Bezug auf den Prozess bei einem Neustart (des Systems) oder seiner Beendigung geschieht (z. B. Neustart des Prozesses). In Tabelle 3.1 finden Sie eine Übersicht über die möglichen Aktionen.

`process`
Enthält den Namen des zu startenden Programms mit den beim Start zu übergebenden Parametern.

Aktion	Beschreibung
`respawn`	Der Prozess wird nach Beendigung erneut gestartet.
`wait`	`init` wartet auf das Ende des Prozesses.
`once`	Der Prozess wird beim Wechsel in diesen Runlevel gestartet.
`boot`	Der Prozess wird beim Systemstart erzeugt.
`bootwait`	Wie `boot`, aber `init` wartet auf das Ende des Prozesses.
`off`	Eintrag ohne Funktion.
`ondemand`	Start bei der Anforderung des Runlevel, aber kein Wechsel des Runlevel.
`initdefault`	Default-Runlevel.
`sysinit`	Start vor `boot` bzw. `bootwait`.

Aktion	Beschreibung
`powerwait`	Signal der UPS über Fehler in der Stromversorgung. Der `powerd`-Dämon muss laufen.
`powerfail`	Wie `powerwait`, aber `init` wartet nicht auf das Prozessende.
`powerokwait`	Stromversorgung ist wieder ok.
`ctrlaltdel`	An der Konsole wurde Strg+Alt+Entf gedrückt.
`kbrequest`	Bearbeitung einer speziellen Tastenkombination.

Tabelle 3.1 Aktionen in der `inittab`

Das Listing 3.9 zeigt ein Beispiel für die Datei `/etc/inittab`. Eine ausführliche Dokumentation des Formats finden Sie auch in der Manpage zu `inittab(5)`.

Das Programm `telinit` ist ein Link auf `init` und bietet über die zusätzlichen Argumente `a`, `b` und `c` die erweiterte Möglichkeit zur Kontrolle des Systems: Es werden jeweils nur die Einträge mit dem Runlevel `a`, `b` und/oder `c` angesprochen. Es können z. B. nur die Programme des speziellen Runlevel `a` neu gestartet werden. Zwischen den numerischen Runleveln (1 bis 6) kann ebenfalls mit dem Programm `telinit` gewechselt werden. Mit dem Parameter `q` wird das Programm `init` veranlasst, die Datei `/etc/inittab` neu einzulesen.

Dabei sind einige Runlevel reserviert, andere haben eine historisch gewachsene Bedeutung. In Tabelle 3.2, finden Sie eine entsprechende Übersicht, so wie die Runlevel in der »Linux Standard Base« definiert sind.

Durch die differenziertere Verwendung der verschiedenen Runlevel ist die Initialisierung des Systems einfacher und flexibler zu steuern. Außerdem ist es wesentlich übersichtlicher, welche Dämonen in welchem Runlevel gestartet bzw. gestoppt werden sollen. Zum Editieren der Runlevel kann das Programm `tksysv` verwendet werden, das Bestandteil einiger Distributionen ist.

Runlevel	Bedeutung
0	Halt
1	Single-User-Modus (reserviert)
2	Multi-User-Modus ohne Netzwerkdienste
3	Multi-User-Modus als Netz-Server
4	reserviert für den Systemverwalter
5	X-Workstation (`xdm`)
6	Reboot

Tabelle 3.2 Die verschiedenen Runlevel und deren Bedeutung

Der Ablauf eines Systemstarts ist zunächst ähnlich, wie in den vorigen Kapiteln beschrieben. Zunächst werden, nach dem Laden des Kernels und dem Start von

init, die Dateisysteme geprüft und die stets notwendigen Dämonen gestartet. Anschließend wird das Skript /etc/rc gestartet. Dieses Skript erhält als Parameter den gewünschten Runlevel. Den entsprechenden Aufruf in der Datei inittab finden Sie in Listing 3.9.

```
# Default runlevel.
id:5:initdefault:


# Boot-time system configuration/initialization script.
# This is run first except when booting in emergency (-b) mode.
si::sysinit:/etc/init.d/rcS

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

Listing 3.9 Änderungen in der Datei /etc/inittab

Für jeden Runlevel existiert ein Verzeichnis /etc/rc.d/rc*Runlevel*.d. Jedes dieser Verzeichnisse enthält Links auf so genannte Start- und Stop-Skripten. Die Namen der Start- bzw. Stop-Skripten beginnen mit einem S bzw. K, gefolgt von einer zweistelligen Nummer. Die Skripten werden aufsteigend bezüglich dieser Nummer sortiert gestartet. Damit ist es möglich, zunächst das Netzwerkinterface zu aktivieren und dann die Client- und Server-Prozesse zu starten. Der letzte Teil des Namens beschreibt normalerweise den Typ des Services, der mit diesem Skript gesteuert wird.

Die Skripten selbst sind im Verzeichnis /etc/init.d gespeichert. In den Verzeichnissen für die entsprechenden Runlevel werden nur symbolische Links auf die Skripten gelegt, die gestartet bzw. gestoppt werden sollen. Das hat den Nachteil, dass die Konfiguration der gestarteten Dämonen nicht übersichtlich in einer Datei vorgenommen wird, sondern durch Links in einem Dateisystem. Ein Vorteil ist, dass dieses System sehr flexibel ist und auch auf anderen Unix-Systemen eingesetzt wird.

Die Skripten werden mit dem Parameter start bzw. stop aufgerufen und starten oder stoppen die entsprechenden Subsysteme. Damit ist es sehr einfach möglich, einen einzelnen Dienst manuell anzuhalten und neu zu starten (z. B. weil man die Konfiguration geändert hat oder den Dienst temporär nicht benötigt). In letzter Zeit sind bei einigen Distributionen weitere mögliche Parameter hin-

zugekommen, wie beispielsweise `restart`, `status` oder `list`. Im Zweifelsfall gibt das Skript eine entsprechende Hilfe aus, wenn man es ohne Parameter aufruft.

In Listing 3.10 finden Sie eine Übersicht über die auf meinem System verwendeten Skripten. Eine besondere Bedeutung haben die Stop-Skripten, die beim Wechsel in einen Runlevel bestimmte Services stoppen. Die Namen der Stop-Skripten beginnen mit `K`. Innerhalb eines Runlevels werden die Skripten nach aufsteigender Nummer abgearbeitet.

```
(linux):/etc> ls -F rc*
rc0.d:
S20halt@

rc1.d:
S20single@

rc2.d:
K51syslog@  K60lpd@     S51syslog@  S60lpd@
K55cron@    K95gpm@     S55cron@    S95gpm@

rc5.d:
K51syslog@  K60lpd@     S51syslog@  S60lpd@
K55cron@    K95gpm@     S55cron@    S95gpm@

rc6.d:
S20reboot@
```

Listing 3.10 Skripten für das System-V-`init`

In der Regel wird sich der Umstieg auf ein anderes `init`-Verfahren nicht lohnen. Andererseits ist die Kompatibilität mit anderen (kommerziellen) Systemen sicherlich sinnvoll, insbesondere wenn man ständig zwischen verschiedenen Systemen wechselt. Dann ist es besonders sinnvoll, die Runlevel so zu belegen, wie man es von anderen Systemen her gewohnt ist.

Bei Rechnern mit wenig Hauptspeicher kann es sinnvoll sein, nur möglichst wenige und kleine Dämonen zu starten. In diesem Fall eignet sich z. B. das `simpleinit` in Kombination mit dem abgespeckten `bdflush` (ein erweiterter `update`-Dämon) statt eines System-V-`init` mit dem normalen `bdflush`. Beide Programme sind im Paket `util-linux` enthalten.

Wenn Sie Ihr System von einer Variante des `init`-Programms auf eine andere umstellen wollen, so müssen Sie für alle Fälle eine Boot- und Root-Diskette bereithalten. Nur mit einer Boot-Diskette kommen Sie nicht weiter, da möglicherweise das Programm `init` auf der Festplatte nicht funktionsfähig ist. Im schlimmsten Fall können Sie das System auch nicht im Single-User-Modus starten und müssen von Diskette booten. Glücklicherweise müssen Sie beim Um-

stieg vom System-V-ähnlichen `init` auf das System-V-`init` nur die Datei `/etc/inittab` verändern (und möglicherweise diese Änderungen zurücknehmen). Alle übrigen betroffenen Dateien werden durch gänzlich andere Dateien mit völlig anderen Namen ersetzt.

Das unter Linux verwendete `init` weicht in einer Beziehung von denen anderer Systeme ab: Es wird direkt in den angeforderten Runlevel gewechselt und nicht der Reihe nach alle kleineren Runlevel durchlaufen.

3.7.3 Stoppen des Systems

Unter Linux laufen, wie bei jedem anderen Unix-System, viele Prozesse im Hintergrund. Vor dem Ausschalten des Systems müssen diese Prozesse beendet werden, da sonst möglicherweise Daten verlorengehen. Ein weiterer Grund für das geordnete Stoppen des Systems ist, dass Linux auch Schreibzugriffe im Hauptspeicher puffert. Nach dem Ende eines Programms sind also die Daten noch nicht auf der Festplatte verewigt. Dies erfolgt erst nach einem `sync`, der aber regelmäßig vom `update`-Dämon durchgeführt wird.

Das Standardverfahren zum Stoppen eines Unix-Systems ist der Aufruf des Befehls `shutdown`. Je nach Kommandozeilenparametern kann die Reaktion des Systems unterschiedlich aussehen. Als Parameter muss eine Zeit angegeben werden, nach der das System heruntergefahren wird. Der Zeitpunkt `now` steht für das sofortige Stoppen des Systems, andernfalls wird eine Zeit in Minuten angegeben. Vor dem Shutdown werden alle angemeldeten Benutzer gewarnt, dass eine Systemwartung bevorsteht.

Mit dem zusätzlichen Parameter `-h` (Halt) wird das System angehalten. Der Parameter `-r` (Reboot) sorgt für einen Neustart des Systems. Wird der Parameter `-f` angegeben, so wird ein fast-Reboot ohne Überprüfung der Dateisysteme durchgeführt. Dies ist in den `rc`-Skripten oft nicht mehr implementiert, da das Extended-2-Dateisystem über ein eigenes Flag für ein ordnungsgemäß heruntergefahrenes System verfügt.

Zur Vereinfachung existieren für häufig vorkommende Aufgaben entsprechende Abkürzungen. So steht je nach `init` z. B. `reboot` oder `init 6` für einen Neustart, `halt` oder `init 0` für Halt. Diese Programme sind Bestandteil des jeweiligen `init`-Systems. Weitere Informationen hierzu finden Sie in den entsprechenden Manpages.

Viele DOS/Windows-Benutzer haben sich an die Tastenkombination Strg+Alt+Entf zum Neustart ihres Systems gewöhnt. Diese Tastenkombination kann auch unter Linux verwendet werden. Mit dem Befehl `ctrlaltdel` kann das Verhalten des Systems eingestellt werden. Die Option `hard` (Default) beim Drücken der magischen drei Tasten startet das System ohne `sync` neu

(`hard_reset_now()` im Kernel). Bei der Option `soft` wird das Signal `SIGINT` an `init` gesendet, das den weiteren Shutdown steuert.

Mit Windows NT und den Nachfolgesystemen wird diese Tastenkombination als eine Art »Secure Attention Key« verwendet. Ein unbedarfter Anwender wird diese Tastenkombination früher oder später auch auf einem Linux-Rechner ausprobieren und den Rechner versehentlich booten. Wenn Sie damit rechnen müssen, ist diese Einstellung vielleicht nicht allzu praktisch.

Damit nicht jeder Anwender das System stoppen kann, lässt sich die Funktion des Neustarts davon abhängig machen, dass ein autorisierter Benutzer an der Konsole angemeldet ist. Die Namen der berechtigten Benutzer werden dazu einfach in die Datei `/etc/shutdown.allow` aufgenommen. Eine Einschränkung auf einzelne Benutzer ist nur dann sinnvoll, wenn das System in keinem Fall einfach ausgeschaltet werden kann, also der Reset- und der Strom-Schalter nicht erreichbar sind.

3.8 Die init-Skripte im LSB

Aktion	Beschreibung
`start`	Starten des Dienstes
`stop`	Stoppen des Dienstes
`restart`	Stoppen und erneutes Starten des Services. Läuft der Dienst nicht, so wird dieser gestartet.
`reload`	Konfiguration erneut einlesen (optional)
`force-reload`	Wie `reload`, wenn der Dienst das unterstützt. Wenn nicht, dann wird der Dienst gestoppt und neu gestartet.
`status`	Ausgeben des Status dieses Dienstes

Tabelle 3.3 Standardisierte Aktionen für `init`-Skripte

Bei der Debian-Distribution wird zur Verwaltung der symbolischen Links in den Runlevel-Verzeichnissen das Programm `update-rc.d` verwendet. Red Hat verwendet das Programm `chkconfig`. Im LSB ist das Programm `/usr/lib/lsb/install_initd` definiert, das spezielle Kommentare in den Skripten auswertet.

```
# Provides: boot_facility_1 [ boot_facility_2 ...]
# Required-Start: boot_facility_1 [ boot_facility_2 ...]
# Required-Stop: boot_facility_1 [ boot_facility_2 ...]
# Default-Start: run_level_1 [ run_level_2 ...]
# Default-Stop: run_level_1 [ run_level_2 ...]
# Short-Description: short_description
# Description: multiline_description
```

Listing 3.11 Kommentare in `init`-Skripten

Für den Entwickler ist es wichtig, dass die Datei /lib/lsb/init-functions in den Skripten eingelesen werden muss und damit verschiedene Funktionen zum Starten, Stoppen und Steuern von Dämonen zur Verfügung stehen. Damit wird es erstmals möglich, auch init-Skripte zwischen den Distributionen auszutauschen.

3.9 init-Konzepte ohne symbolische Links

So bequem und bekannt wie das Verwalten der symbolischen Links auch ist, es hat auch Nachteile. So kann man die Links nur schwer mittels einer Versionsverwaltung bearbeiten oder eine alte mit einer neuen Version vergleichen. Die Programme file-rc (*http://packages.debian.org/file-rc*) und r2d2 (*http://www.ibiblio.org/pub/Linux/system/daemons/init/*) implementieren die bekannte init-Struktur mit Hilfe einer Konfigurationsdatei. Diese enthält Runlevel, die Sortierung darin und den Skriptnamen zum Starten bzw. Stoppen der Dienste (Listing 3.12).

```
# Format:
# <sort> <off-> <on-levels>     <command>
01       0,1,6   -              /etc/init.d/xdm
05       -       1              /etc/init.d/single
20       0,1,6   2,3,4,5        /etc/init.d/gpm
...
```

Listing 3.12 Die Konfigurationsdatei /etc/runlevel.conf

Für Debian-Benutzer ist die Verwendung von file-rc fast vollkommen transparent. Die üblichen symbolischen Links werden von den einzelnen Paketen nicht selbst erstellt, sondern das Programm update-rc.d wird aufgerufen. Wenn man dieses Programm austauscht, wie es zum Beispiel das Paket file-rc macht, so braucht kein einziges Paket angepasst zu werden. Ein Nachteil ist allerdings, dass die Konfigurationsdatei jeweils neu geschrieben wird, eventuell könnte aber ein Verfahren wie das /etc/cron.d-Verzeichnis Abhilfe schaffen. Obwohl die Idee schon einige Jahre alt ist, werden diese Programme nur selten eingesetzt.

4 Konfiguration und Administration

4.1 Anpassungen und Nachvollziehbarkeit

Die Anpassung eines Linux-Systems an die verwendete Hardware und die Bedürfnisse des Benutzers, auch Konfiguration genannt, umfasst eine Reihe von völlig unterschiedlichen Aspekten. Für jeden größeren Bereich, der nicht in einem eigenen Abschnitt dieses Buches angesprochen wird, werden hier die wichtigsten Konfigurationsmöglichkeiten vorgestellt.

Viele Dinge können sowohl vom Systemadministrator für alle Benutzer des Systems als auch von jedem Anwender für sich selbst konfiguriert werden. Gerade diese Möglichkeit ist ein großer Vorteil gegenüber Ein-Benutzer-Systemen wie DOS oder OS/2. Dort ist eine Konfiguration normalerweise für das gesamte System gültig, unabhängig davon, ob andere Benutzer dieses Rechners dadurch beeinträchtigt werden oder nicht. Wer hat sich nicht schon über die verstellten Farben unter Windows oder die unsinnig angeordneten Icons unter OS/2 geärgert?

Bevor Sie jedoch mit der Konfiguration Ihres Systems beginnen, lesen Sie zunächst die Kapitel über Versionverwaltungen (Kapitel 11, »Source- und Konfigurations-Management«) und Emacs (Kapitel 6, »Der Editor Emacs«), damit Sie nicht nur die Änderungen durchführen können, sondern auch die Möglichkeit haben, Ihre Änderungen nachvollziehbar zu dokumentieren. Gerade die einfache Nachvollziehbarkeit der Änderungen ist ein Vorteil gegenüber DOS- und Windows-Systemen, wo praktisch jedes Programm hinter dem Rücken des Anwenders systemweite Einstellungen in den Dateien `CONFIG.SYS`, `AUTOEXEC.BAT` und den Windows `*.INI`-Dateien vornimmt. Verschenken Sie diese Möglichkeit nicht durch übereilte Änderungen, bevor Sie mehr über Versionsverwaltungen gelernt haben.

4.2 Kernel- und Hardware-Konfiguration

Zunächst wird beim Systemstart (siehe auch Kapitel 3, »Ablauf eines Systemstarts«) vom Kernel die Hardware untersucht und gegebenenfalls initialisiert. Dies ist in der Regel problemlos, da viele Treiber ein so genanntes Auto-Probing durchführen. Darunter versteht man die Überprüfung, ob sich an einer bestimmten IO-Adresse oder einem Interrupt eine entsprechende Schnittstelle oder Karte befindet.

Geräte, für die kein Treiber im Kernel vorhanden ist, werden nicht initialisiert und können nicht angesprochen werden. Treiber, bei denen das zugehörige Ge-

rät nicht installiert ist, benötigen unnötig Speicherplatz im Hauptspeicher. Da der Kernel bei nur wenig verfügbarem Speicher nicht wie normale Anwendungsprogramme aus dem realen Speicher auf eine Festplatte ausgelagert werden kann, ist dieser Speicher für Anwendungen oder Puffer nicht verfügbar.

Bei der großen Anzahl von Treibern, die früher in den Kerneln der Distributionen eingebunden waren, konnte eine beträchtliche Menge Speicher belegt werden. Auch wird die Zeit, die das Betriebssystem zum Booten benötigt, zum Teil wesentlich verlängert. Mit der Verwendung von dynamisch ladbaren Modulen, die auch wieder aus dem Speicher entfernt werden können, wird dieser Aspekt immer weniger interessant. In vielen Fällen kann also der Standard-Kernel der Distribution ohne Probleme dauerhaft verwendet werden.

Einige NE2000-kompatible ISA-Netzwerkkarten sind dafür bekannt, dass sie den Rechner (genauer gesagt den ISA-Bus) anhalten, wenn andere Treiber auf diesen Adressen nach Geräten suchen. Daher sollten bei älteren Rechnern nur die notwendigen Treiber in den Kernel eingebunden werden. Auf diese Weise werden auch Konflikte bei der Initialisierung der Karten verhindert oder zumindest vermindert. Daher ist es auch heute manchmal noch sinnvoll, einen neuen, speziell an die eigene Hardware angepassten Kernel zu erstellen.

Zu vielen Funktionen, Optionen oder Treibern finden Sie in den Kernel-Quellen zusätzliche Informationen. In den meisten Fällen sind diese im Verzeichnis `./Documentation` abgelegt. Dort finden Sie auch Informationen über zusätzlich notwendige Programme und deren Bezugsquellen sowie möglicherweise über weiterführende Dokumentationen. Außerdem gibt es die Kernel-HowTo des Linux Documentation Project.

4.2.1 Konfigurieren des Kernels

Der Quellcode des Kernels befindet sich nach der Installation der Kernel-Quellen im Verzeichnis `/usr/src`. Früher wurde das Verzeichnis `/usr/src/linux` verwendet, heute hat man häufig noch die Kernel-Release im Verzeichnisnamen. Dort wird zunächst der Kernel konfiguriert und dann übersetzt. Die Konfiguration des Kernels erfolgt interaktiv und besteht aus einer Reihe von Fragen zur verwendeten Hard- und Software. Sie starten die Konfiguration mit dem Befehl `make config` (im Textmodus), `make menuconfig` (Menüs im Textmodus) oder `make xconfig` (unter X, Sie benötigen Tcl/Tk). Sie werden durch eine Reihe von Fragen geführt. Zu jeder Frage steht auch online ein Hilfetext zur Verfügung, den Sie durch die Eingabe von `?` oder die Anwahl des Help-Buttons erhalten.

Als Ausgangspunkt für eine eigene Konfiguration können Sie die Konfiguration Ihrer Distribution verwenden. Häufig finden Sie diese im Verzeichnis `/boot` neben dem installierten Kernel. Bei manchen Distributionen ist die Konfiguration im Verzeichnis `configs` unterhalb der Kernel-Quellen zu finden. Sie können die-

se Datei als `.config` in das Kernel-Verzeichnis kopieren und einmal `make oldconfig` aufrufen. Mit den oben angegebenen Aufrufen können Sie dann weitere Anpassungen vornehmen.

Im folgenden Abschnitt wird zunächst nur auf die allgemeine Konfiguration des Kernels eingegangen, Informationen zu speziellen Einstellungen finden Sie in den entsprechenden Kapiteln (Kapitel 14, »Linux in einer vernetzten Umgebung«, Abschnitt 4.4.1, »Dateisysteme und Datenträger«). Weitere Informationen über den Kernel und die Installation finden Sie in den `README`-Dateien in den Kernel-Quellen. Daneben existiert noch eine HowTo für die Kernel-Konfiguration, die ebenfalls zu Rate gezogen werden kann.

Zur Übersetzung des Kernels sind einige Development-Tools notwendig. Zunächst benötigt man den GNU-C-Compiler GCC mit den zugehörigen `binutils` (den Assembler `gas` und den Linker `gld`). Für die Erstellung des Boot-Sektors im Kernel wird der 16-Bit-Assembler `as86` benötigt. Die Ablaufsteuerung der Übersetzung übernimmt das Programm `make`, das im Abschnitt 10.2, »Das Programm `make`« genauer erläutert wird.

Eine deutsche Übersetzung vieler Hilfetexte steht unter *http://www.suse.de/~ke/kernel/* zur Verfügung. Diese Hilfe ist in den Kernel der SuSE-Distribution integriert.

4.2.2 Übersetzen des Kernels

Nach der Konfiguration muss nun der Quelltext des Kernels übersetzt werden. Zur Sicherheit sollte zunächst keine Installation mit LILO (dem Boot-Lader) erfolgen, sondern eine Boot-Diskette erstellt werden. Funktioniert das System auch mit dem neuen Kernel einwandfrei, so kann dieser dann mit `make install` installiert werden. Weitere Informationen zur Verwendung von `make` finden Sie in Abschnitt 10.2, »Das Programm `make`«. Die zur Übersetzung des Kernels notwendigen Befehle finden Sie im Listing 4.1. Weitere Informationen bietet auch die Datei `README`. Das Übersetzen des Kernels dauert je nach Prozessor und Speicherausbau zwischen wenigen Minuten (der veröffentlichte Rekord sind allerdings 7,52 Sekunden) und mehreren Stunden.

```
(linux):/usr/src/linux# make dep
(linux):/usr/src/linux# make clean
(linux):/usr/src/linux# make
(linux):/usr/src/linux# make zdisk
(linux):/usr/src/linux# make modules
(linux):/usr/src/linux# make modules_install
```

Listing 4.1 Übersetzung des Kernels

Anschließend sollte mit der eben erstellten Diskette neu gebootet und nach einem erfolgreichen Test der Befehl `make install` zur Installation des neuen Kernels ausgeführt werden. Falls in Ihrem System der Boot-Lader LILO installiert ist, ist dieser Befehl äquivalent zu dem Kommando `make zlilo`. Verwenden Sie jedoch einen anderen Boot-Lader wie z. B. `loadlin`, dann können Sie ein Skript `/sbin/installkernel` anlegen, das alle Operationen zur Installation des neuen Kernels automatisch durchführt.

4.2.3 Update des Kernels

Ein Update des Kernels kann auf zwei Wegen erfolgen. Es kann entweder ein komplett neuer Kernel installiert werden oder die Änderungen am Quellcode können mit dem Programm `patch` (siehe auch den Abschnitt »Patches erzeugen und einspielen«) auf die Kernel-Quellen übertragen werden.

Komplettinstallation des Kernels

Die Neuinstallation eines kompletten Kernels ist recht einfach. Zunächst sollten die alten Kernel-Quellen gesichert werden, so dass im Fehlerfall wieder der alte Kernel verwendet werden kann. Bei ausreichend Platz auf der Festplatte genügt es, den kompletten Pfad umzubenennen. Andernfalls kann es sinnvoll sein, den Kernel nach `make clean` in einem (komprimierten) Archiv (z. B. mit `tar`) zu speichern. Anschließend können die neuen Kernel-Quellen ausgepackt werden und das bekannte Verfahren der Kernel-Konfiguration und Übersetzung wird durchlaufen. Viele systemnahe Programme sind darauf angewiesen, dass zumindest der Schritt `make config` durchgeführt wird. Die hier notwendigen Befehle finden Sie in Listing 4.2.

```
(linux):/usr/src/linux# make clean
(linux):/usr/src/linux# cd ..
(linux):/usr/src# mv linux linux.old
(linux):/usr/src# tar zxvf /tmp/linux-1.2.2.tar.gz
(linux):/usr/src# cd linux
(linux):/usr/src/linux# make config
...
```

Listing 4.2 Komplett-Update des Kernels

Wenn Sie die Datei `.config` des alten Kernels aufgehoben haben, so können Sie diese in das neue Kernel-Verzeichnis kopieren und anschließend `make oldconfig` aufrufen. Dabei werden alle alten Einstellungen übernommen, bei neu hinzugekommenen Fragen hält das Programm an und Sie können die passende Einstellung vornehmen. Damit ist ein Upgrade auf eine neue Kernel-Version nicht erneut mit einer vollständigen Kernel-Konfiguration verbunden.

Patchen des Kernels

Die zweite Methode, den Kernel zu aktualisieren, ist das so genannte Patchen des Kernels. Da sich der Quellcode von zwei Kernel-Versionen oft kaum unterscheidet, müssen bei einem Update erheblich weniger Daten übertragen werden. Ein Patch enthält genau die Änderungen von einer Version zur nächsten. Möchte man also einen um mehrere (viele) Versionen neueren Kernel verwenden, so ist es oft einfacher, die neuen Quellen zu besorgen, als alle Patches in der richtigen Reihenfolge anzuwenden.

Werden diese Voraussetzungen beachtet, so ist ein Update des Kernels recht einfach. Zur Veranschaulichung folgt die Darstellung des Updates von Kernel 1.2.1 auf die Version 1.2.2. Der wesentliche Vorteil von Patches ist, dass man alle Änderungen am Quelltext relativ übersichtlich vorfindet. Es empfiehlt sich bei etwas Programmiererfahrung unbedingt, einen Blick auf die Patches zu werfen, bevor man sie installiert. Beispielhaft finden Sie die entsprechenden Befehle im Listing 4.3. Der Befehl `make mrproper` löscht neben allen Arbeitsdateien auch die alte Kernel-Konfiguration. Sie können die beiden Befehle `make mrproper` und `make config` wieder durch `make oldconfig` ersetzen.

```
(linux):/usr/src/linux# zmore /tmp/patch-1.2.2.gz
(linux):/usr/src/linux# zcat /tmp/patch-1.2.2.gz|patch -p1
(linux):/usr/src/linux# make mrproper
(linux):/usr/src/linux# make config
...
```

Listing 4.3 Update des Kernels mittels `patch`

Nach einem Patch sollte nach eventuell fehlgeschlagenen Änderungen gesucht werden. Dazu kann mit dem Programm `find` (siehe auch den Abschnitt 9.2.1, »Suchen von Dateien mit dem Kommando `find`«) nach den entsprechenden Sicherungsdateien gesucht werden. Diese Dateien heißen je nach `patch`-Version `*.rej` oder `*#`. Fehlgeschlagene Änderungen sind ein schlechtes Zeichen, hier sollten je nach Erfahrung entweder ein komplett neuer Kernel eingespielt oder die Änderungen mit einem Editor von Hand durchgeführt werden. Wenn Sie selbst den Kernel modifiziert oder zusätzliche Patches installiert haben, dann kann das durchaus zu solchen Problemen führen.

Ebenfalls werden Sicherungskopien der erfolgreich veränderten Dateien angelegt. Diese Dateien heißen je nach `patch`-Version `*.orig` oder `*~`. Sie können später gelöscht werden. Weitere Informationen zu dem Programm `patch` finden Sie im Abschnitt »Patches erzeugen und einspielen«.

4.2.4 Konfiguration der Hardware

Viele Treiber verwenden Auto-Probing, um die entsprechende Hardware zu finden und dann zu initialisieren. Dabei wird an verschiedenen IO-Adressen (Ports), Interrupts oder DMA-Kanälen nach charakteristischen Signaturen gesucht. Dies kann zum Absturz des Rechners führen, eventuell werden aber auch installierte Geräte nicht gefunden. In solchen Fällen können Sie einmal die Liste der zu prüfenden Adressen bzw. Interrupts im Kernel ändern, häufig aber auch die entsprechenden Parameter mittels der Kernel-Kommandozeilen einstellen. Dies setzt allerdings voraus, dass Sie einen entsprechenden Boot-Lader (siehe auch Kapitel 3, »Ablauf eines Systemstarts«) verwenden. Die Syntax der Kommandozeilenoption für den Kernel hängt von den einzelnen Treibern ab, daher finden Sie diese in den entsprechenden HowTo- bzw. `README`-Dateien dokumentiert.

Oft werden Multi-Serial-Karten nicht korrekt erkannt. Die entsprechenden Parameter (IO-Adresse und IRQ), die der Kernel verwendet, können mit `setserial` eingestellt werden. Entsprechende Beispiele findet man in der Datei `/etc/init.d/serial`.

Einige Treiber, z. B. CD-ROM-Treiber, unterstützen die Konfiguration des Verhaltens der Geräte über die `ioctl()`-Funktion. Hier kann z. B. der automatische Auswurf der Lade beim Abhängen des CD-ROM-Dateisystems ein- oder ausgeschaltet werden. Ein Beispiel, das direkt eingesetzt werden kann, ist das Paket `cdeject`.

4.3 Kernel-Module

Einer der Kritikpunkte an Linux ist die Verwendung eines monolithischen Kernels anstelle des heute »modernen« Micro-Kernels. Diese Kritik ist jedoch nur teilweise berechtigt. Interessierte Leser seien hier auf eine Diskussion zwischen Linus Torvalds und Andrew S. Tanenbaum, dem Autor von Minix und Betriebssystemspezialist, verwiesen, in der die Vor- und Nachteile der Neuentwicklung von Linux diskutiert wurden. Diese Diskussion aus der Newsgruppe *news:comp.os.minix* finden Sie in der Datei `linux_is_obsolete` im `doc`-Verzeichnis der Linux `ftp`-Server.

Unter einem Micro-Kernel (wie z. B. Mach) versteht man einen minimalen Kernel, der nur die notwendigsten Funktionen (wie Speicher- und Prozessmanagement oder Hardware-Treiber) enthält. Dateisysteme, Personalities (wie ein BSD- oder Unix-Interface) oder andere Funktionen werden später geladen und können im User-Space, also mit Hilfe von normalen Programmen, implementiert werden. Dadurch erhält man größere Flexibilität, aber erkauft sich diese durch höheren Verwaltungsaufwand. Da Linux zunächst auf (aus heutiger Sicht) recht langsamen 386er PCs entwickelt wurde, hat Linus Torvalds auf »moderne« Kon-

zepte wie einen Micro-Kernel verzichtet. Hurd, der Kernel des GNU Systems basiert hingegen auf einem Micro-Kernel (zurzeit wird Mach verwendet). Dokumente zum Design von Hurd finden Sie unter *http://www.gnu.org/software/hurd/hurd.html*.

Im Laufe der Zeit haben sich aber auch einige Nachteile des bisherigen Konzepts gezeigt. So musste für einen neuen Treiber der Kernel beinahe vollständig neu übersetzt werden. Bei der Entwicklung eines Treibers musste nach jeder Änderung der Kernel neu gestartet werden, was mit einem Neustart des Rechners verbunden ist. Dadurch dauerte die Entwicklung von Treibern länger, und Anwender vermissen die Flexibilität anderer Systeme.

Aus dieser Erkenntnis heraus wurde das Konzept der Kernel-Module entwickelt. Dabei werden zur Laufzeit des Kernels (ohne einen Neustart des Rechners) Treiber oder andere Funktionen zum Kernel hinzugefügt (mit dem Programm `insmod`) oder wieder aus diesem entfernt (mit dem Programm `rmmod`). Eine Liste der geladenen Module können Sie mit dem Befehl `lsmod` anzeigen.

Ein Modul ist eine beinahe normale Objektdatei, wie sie vom C-Compiler erzeugt wird. Da ein Modul nicht allein funktionieren kann, sondern der Zugriff auf einige interne Kernel-Strukturen notwendig ist, müssen innerhalb des Kernels diese Symbole exportiert werden. Alle diese Symbole müssen in der Datei `ksyms.c` definiert werden. In aktuellen Kernel-Versionen gibt es verschiedene Dateien, die den Aufruf `EXPORT_SYMBOL` enthalten – je nach Hardware-Architektur müssen andere Symbole exportiert werden.

Fehlt ein Symbol, so kann das Modul nicht geladen werden – es wird eine entsprechende Fehlermeldung ausgegeben. Genauso muss die Kernel-Version entweder exakt zum Modul passen (wenn der Kernel ohne `CONFIG_MODVERSIONS` übersetzt wurde) oder die Modul-Version muss mit der des Kernels verträglich sein. Diese Prüfungen werden anhand von Prüfsummen durch den Befehl `insmod` beim Laden des Moduls durchgeführt.

Im zweiten Schritt wird das Modul initialisiert, ein Treiber kann z. B. nach der zugehörigen Hardware suchen. Ist die Initialisierung nicht erfolgreich, so wird das Modul wieder aus dem Speicher entfernt. Andernfalls läuft das Modul genauso im Kernel-Space, als wäre es in den Kernel einkompiliert. Damit besteht voller Zugriff auf die gesamte Hardware, so dass bei der Programmierung eines neuen Moduls mit derselben Vorsicht (Datensicherungen, defensive Programmierung) wie im eigentlichen Kernel vorgegangen werden sollte.

Nach der Benutzung können Module, soweit sie nicht mehr benötigt werden, mit dem Befehl `rmmod` aus dem Kernel-Speicher entfernt werden. Dabei wird eine Clean-up-Funktion aufgerufen, die eventuell belegte Ressourcen wie Speicher, IO-Ports oder Interrupts wieder freigibt. Module, die noch aktiv sind, können nicht gelöscht werden.

Beim Laden von Modulen können Variablen, wie z. B. zu verwendende IO-Ports oder Interrupts, als Parameter angegeben werden. Diese Variablen werden von `insmod` vor der Initialisierung des Moduls verändert. Damit können langwierige und teilweise gefährliche Auto-Probes verhindert werden. In Listing 4.4 finden Sie ein Beispiel für das Laden und Entfernen eines Moduls.

```
linux:~# insmod sbpcd.o sbpcd=0x300,1
sbpcd.c v3.6-1 Eberhard Moenkeberg <emoenke@gwdg.de>
Scanning 0x300 (LaserMate)...
Drive 0 (ID=0): CR-562 (0.80) at 0x300 (type 0)
data buffer size: 8 frames.
linux:~# lsmod
Module:        pages:  Used by:
minix              5
sbpcd             10
...Benutzung des Moduls...
linux:~# rmmod sbpcd
sbpcd module released.
```

Listing 4.4 Benutzung eines Kernel-Moduls

Welche Parameter ein Modul verwendet, können Sie mit dem Befehl `modinfo` herausfinden. Das Listing 4.5 zeigt eine solche Ausgabe.

```
linux:/lib/modules/2.4.18/kernel/drivers/net/irda$ modinfo
irport.o
filename:    irport.o
description: "Half duplex serial driver for IrDA SIR mode"
author:      "Dag Brattli <dagb@cs.uit.no>"
license:     "GPL"
parm:        io int array (min = 1, max = 4), description "Base
             I/O addresses"
parm:        irq int array (min = 1, max = 4), description "IRQ
             lines"
```

Listing 4.5 Informationen zu einem Kernel-Modul

Die bei der Kernel-Konfiguration angewählten Module werden mit dem Befehl `make modules` übersetzt und mit `make modules_install` in das Verzeichnis `/lib/modules/`*`Kernelversion`* installiert. Die Module können auf verschiedene Arten geladen werden:

- Der Systemadministrator kann Module mit dem Befehl `insmod` laden.
- Mit dem Befehl `depmod` kann eine Art Makefile erstellt werden, das Abhängigkeiten von Modulen enthält. Anschließend kann mit dem Programm `modprobe` ein einzelnes Modul oder ein ganzer Stapel von Modulen geladen werden.
- Der Kernel kann mit Hilfe der `CONFIG_KMOD`-Option ein Modul bei Bedarf automatisch laden.

Die Programme `depmod`, `modprobe` und `kmodd` lesen die Konfigurationsdatei `/etc/modules.conf`. In dieser Datei können Aliasnamen für Module oder Parameter für das Laden der Module festgelegt werden. Damit ist es möglich, nur die notwendigsten Treiber und Funktionen in den Kernel einzukompilieren und den Rest dem `kmod` zu überlassen. Ein Beispiel für die Datei `modules.conf` finden Sie in Listing 4.6.

```
# Das Dummy-Device für das Netz
alias dummy0 dummy
options dummy -o dummy0

# Aliasnamen für ISO-Dateisystem (CD)
alias iso9660fs isofs
alias cdfs isofs

# Aliasnamen für den Soundblaster-CD-ROM-Treiber
alias block-major-25 sbpcd
options sbpcd sbpcd=0x300,1

# Aliasnamen für a.out
alias binfmt-204 binfmt_aout
alias binfmt-267 binfmt_aout

# ftape-Treiber
alias char-major-27 ftape
```

Listing 4.6 Die Datei `conf.modules`

Kommentare werden durch eine Raute (`#`) eingeleitet und reichen bis zum Zeilenende. Mit dem Schlüsselwort `alias` kann ein Aliasname zu einem Modul festgelegt werden. So kann z. B. beim Zugriff auf ein zeichenorientiertes Gerät mit der Major-Nummer 27 (Aliasname) das `ftape`-Modul (echter Name) geladen werden. Wird als echter Name `off` angegeben, so wird das Modul nicht von `modprobe` oder `kerneld` geladen.

Mit `options` können Optionen angegeben werden, die beim Laden des Moduls verwendet werden. Damit kann der interne Name eines Moduls verändert (Option `-o`) oder eine Variable gesetzt werden. Die Schlüsselwörter `path` und `depfile` geben an, in welchem Pfad Module bzw. die Datei mit den Abhängigkeiten gespeichert werden. In der Regel müssen Sie diese Einstellungen nicht verändern.

Beim Systemstart erstellt man mit `depmod -a` die Datei `modules.dep`, die die Abhängigkeitsstruktur der Module enthält. Anschließend kann ein Modul mit `modprobe` *`Modul`* geladen werden, wobei eventuell notwendige Module automatisch vorher geladen werden.

Wesentlich bequemer ist die Verwendung von `kmod`, der diese Funktion automatisch und für den Benutzer vollkommen transparent durchführt. `kmod` benötigt die Datei `modules.dep` und die Konfigurationsdatei `/etc/modules.conf`. Die meisten Kernel enthalten heute den Kernel-Thread `kmod`, der die Verwaltung der Module übernimmt.

Aufgrund der Implementierung der Modul-Schnittstelle ist es möglich, dass Hardware-Anbieter Treiber für ihre Hardware nur als Binärcode freigeben. Die Hoffnung der Linux-Entwickler ist jedoch, dass aufgrund von Inkompatibilitäten bei zukünftigen Kernel-Versionen und der einfacheren Wartung von Treibern im Standard-Kernel dennoch der Quellcode freigegeben wird.

Die Kernel-Entwickler haben für die Verwendung von Kernel-Modulen einige Einschränkungen. Zunächst ist es nicht möglich, einen neuen Systemcall in einem Modul zu laden. Auf der einen Seite wäre es technisch relativ schwer, diese stabil zu implementieren, auf der anderen Seite erzwingt man damit, dass derartige Erweiterungen des Kernels unter der GPL verfügbar sind.

Ab Kernel-Release 2.4 werden in Modulen nicht nur Informationen zum Autor und eine kurze Modul-Beschreibung gespeichert, sondern auch die Lizenz des Moduls. Beim Laden eines Binär-Moduls bzw. eines Moduls ohne diese Kennzeichnung wird der Kernel entsprechend markiert. Bei einem Kernel-Oops oder einer Kernel-Panik wird dieses Flag mit ausgegeben. Fehlermeldungen, an denen Binär-Module beteiligt sind, können nur vom Hersteller dieser Module bearbeitet werden. Die Kernel-Entwickler erkennen das sofort und verweisen entsprechend weiter.

Kernel-Symbole wurden bisher an alle Kernel-Module exportiert, egal, unter welcher Lizenz diese standen. Auch mit Kernel-Release 2.4 wurde hier eine Erweiterung implementiert: Kernel-Symbole können in Zukunft nur für GPL-Module bereitgestellt werden. Hintergrund dafür ist, dass für einige Erweiterungen der Zugriff auf mehr »interne« Kernel-Symbole möglich sein muss, auf der anderen Seite die Verwendung dieser Symbole in Binärmodulen die GPL des Kernels unterwandern könnten. Linus hat verkündet, dass alle bisher exportierten Symbole weiterhin für alle zugänglich bleiben werden, diese Einschränkung also höchstens für neu aufgenommene Symbole zutreffen kann.

Viele Distributionen enthalten nur noch eine Boot-Diskette für alle Systeme. Damit entfällt die Auswahl einer (hoffentlich) passenden Boot-Diskette. Auf dieser sind dann alle Hardware-Treiber als Modul enthalten. Nach der Installation wird ein ebenfalls modularer Kernel mit den zugehörigen Modulen auf die Festplatte kopiert.

Was passiert aber, wenn das Root-Dateisystem auf einem Medium liegt, das nur mit Hilfe eines Kernel-Moduls angesprochen werden kann? Hier wird die Funktion `initrd` eingesetzt. Es wird eine initiale Ram-Disk erstellt, die alle Treiber

enthält. Beim Einhängen des Root-Dateisystems werden alle notwendigen Module geladen und anschließend wird die RAM-Disk wieder freigegeben. Wie Sie eine initiale Ram-Disk einbinden, das erfahren Sie in der Dokumentation Ihres Boot-Laders.

4.4 Systemkonfiguration

Eine Reihe von Einstellungen können nur vom Systemadministrator vorgenommen werden, beispielsweise das Verwalten von Benutzerkennungen, Dateisystemen, Netzwerkanschlüssen und andere Einstellungen. Hier werden die Bereiche vorgestellt, die nicht in einem eigenen Kapitel besprochen werden (wie z. B. die Netzkonfiguration, siehe Kapitel 14, »Linux in einer vernetzten Umgebung«).

4.4.1 Dateisysteme und Datenträger

Unter Unix kennt man keine Laufwerksbuchstaben. Alle Daten sind in einen Verzeichnisbaum integriert und dort ansprechbar. Für Benutzer ist es vollkommen unerheblich, auf welcher physikalischen Festplatte oder welchem File-Server die Daten gespeichert sind. Wenn die Pfade zu den Dateien auf allen Rechnern im Netz gleich sind, dann merkt der Benutzer keinen Unterschied, egal ob er auf dem File-Server angemeldet ist oder an einem beliebigen Client.

Das Einhängen von Dateisystemen zum Aufbau der Verzeichnishierarchie ist normalerweise Aufgabe des Systemadministrators. Dennoch ist es üblich, dass auch nicht privilegierte Benutzer auf mitgebrachte Datenträger wie Disketten, Magnetbänder oder CDs zugreifen können. Je nach Datenträger und Dateisystem bieten sich verschiedene Möglichkeiten an.

Unter Linux verfügbare Dateisysteme

Unter Linux haben Sie die Qual der Wahl: Welches Dateisystem verwenden Sie zum Speichern der Daten? Bis vor einiger Zeit wurde praktisch immer das Extended-2-Dateisystem verwendet. Es ist sehr stabil, hinreichend schnell und sehr häufig im Einsatz. Der größte Nachteil war und ist, dass bei größeren Dateisystemen der Filesystemcheck (nach einem Absturz oder turnusmäßig) sehr lange dauern kann.

Abhilfe gegen einen lange dauernden Filesystemcheck nach einem Absturz sind so genannte Journaled Filesystems. In dem Journal werden Änderungen am Dateisystem gespeichert, solange diese noch nicht vollständig durchgeführt sind. Nach einem Absturz muss dann nur noch das Journal abgearbeitet werden und das Dateisystem befindet sich wieder in einem konsistenten Zustand. Heute können Sie aus eine Reihe von Möglichkeiten auswählen:

- Das ext3-Dateisystem ist eine Weiterentwicklung des ext2-Dateisystems. Auf der Platte sind beide Dateisysteme identisch, so dass Sie einfach migrieren (und im Fehlerfall wieder zurückgehen) können. Mit dem Befehl tune2fs -j *Device* können Sie auf einem bestehenden ext2-Dateisystem ein Journal anlegen und dieses dann als ext3 verwenden.
- Eine Neuentwicklung ist das reiserfs von Hans Reiser. Neben einem Journal werden ausgefeilte Datenstrukturen zur Geschwindigkeitssteigerung verwendet. Leider kann man nur durch Neuanlage von ext2 zu reiserfs wechseln. In der SuSE-Distribution war dieses Dateisystem schon lange enthalten, bevor es Bestandteil des Standard-Kernels wurde.
- SGI hat vor einiger Zeit das XFS auf Linux portiert und unter der GPL freigegeben. Es kann als Patch von *http://oss.sgi.com/projects/xfs/* bezogen werden. XFS hat unter dem SGI-Unix IRIX einen sehr guten Ruf.
- IBM hat das JFS von OS/2 auf Linux portiert, Sie erhalten es unter *http://oss.software.ibm.com/developerworks/opensource/jfs/*.

Wenn Sie in die Verlegenheit kommen, ein Dateisystem manuell anzulegen, dann verwenden Sie das Programm mkfs mit der Option -t *fstyp*. Zur Überprüfung der Struktur des Dateisystems verwenden Sie fsck, ebenfalls mit der Option -t *fstyp*.

Das Paket mtools für DOS-Dateisysteme

Der Zugriff auf DOS-Disketten oder Festplattenpartitionen kann mit Hilfe der mtools erfolgen. Dazu werden in der Datei /etc/mtools.conf die vorhandenen Laufwerke bzw. Partitionen eingetragen, soweit sie verfügbar gemacht werden sollen. Alternativ kann die Datei ~/.mtoolsrc verwendet werden bzw. die Datei, deren Name in der Umgebungsvariablen MTOOLSRC gespeichert ist.

Zur Verwendung der mtools muss der Benutzer Zugriffsrechte auf die entsprechenden Geräte haben. Das kann z. B. durch die Zuordnung zu einer speziellen Gruppe in der Datei /etc/group erfolgen, die dann auch Lese- und Schreibrechte auf die entsprechenden Gerätedateien hat (Listing 4.7). Alle Benutzer, die die Diskettenlaufwerke benutzen dürfen, werden zusätzlich in die Gruppe floppy aufgenommen.

```
(linux):~# chgrp floppy /dev/fd[01]*
(linux):~# chmod 660 /dev/fd[01]*
```

Listing 4.7 Setzen der Berechtigungen zum Zugriff auf Diskettenlaufwerke

Die Befehle für den Zugriff auf DOS-Datenträger sind die von DOS bekannten, wobei jedoch zur Unterscheidung von anderen Befehlen jedem Befehlsnamen ein »m« vorangestellt wird (mcopy, mformat, mtype, mcd usw.). Bei allen Programmen der mtools außer mformat kann das entsprechende Device mit auto-

matischer Erkennung der Datenträgergröße verwendet werden. Da das Formatierprogramm die Größe nicht selbst ermitteln kann, muss hier das passende Gerät verwendet werden. Ein Beispiel für die Datei `/etc/mtools-conf` finden Sie in Listing 4.8. Die entsprechenden Dateisysteme dürfen grundsätzlich nicht gleichzeitig eingehängt sein, da sonst die Dateisysteme beschädigt werden können.

```
drive a: file="/dev/fd0" exclusive
drive b: file="/dev/fd1" exclusive

# First SCSI hard disk partition
drive c: file="/dev/sda1"

# dosemu floppy image
drive m: file="/var/lib/dosemu/diskimage"

# dosemu hdimage
drive n: file="/var/lib/dosemu/hdimage" offset=3840
```

Listing 4.8 Auszug aus der Datei `/etc/mtools.conf`

Auf den meisten unterstützten Plattformen können die `mtools` ohne weitere Konfiguration verwendet werden. Im Listing 4.8 werden zusätzlich noch die DOS-Partition und die virtuelle Diskette und Festplatte für den DOSemu verfügbar gemacht. Weitere Informationen zur Konfiguration der `mtools` finden Sie in der ausführlichen Dokumentation.

Die aktuelle Version der `mtools` unterstützt das VFAT-Dateisystem von Windows 95. Damit ist es möglich, lange Dateinamen auch auf Disketten zu speichern, und es ist wirklich problemlos möglich, derartige Dateien zu transportieren. Im Gegensatz zu mit `tar` oder `cpio` erstellten Disketten sind diese dann auch an einem DOS- oder Windows-PC lesbar.

Das Programm mount

Eine weitere Möglichkeit des Zugriffs auf Datenträger, ist das Einhängen in den Linux-Verzeichnisbaum. Die meisten Partitionen werden beim Systemstart automatisch eingehängt und können, abhängig von den dort vergebenen Berechtigungen (siehe Kapitel 5, »Benutzer, Gruppen und Berechtigungen«) von allen Benutzern verwendet werden. Dazu wird in den Start-Skripten das Programm `mount` aufgerufen, das die Datei `/etc/fstab` auswertet. Die Zuordnung von Dateisystemen und deren Mount-Points kann von einem Unix-Benutzer nicht verändert werden, nur der Systemverwalter `root` kann Dateisysteme an beliebigen Stellen einhängen.

Der Befehl `mount` ist jedoch ein privilegierter Befehl, der nur vom Systemadministrator ausgeführt werden darf. Damit können normalerweise nicht privile-

gierte Benutzer keine zusätzlichen Dateisysteme einhängen. Das Programm `mount` ist `setuid-root`-installiert und führt eine Reihe von zusätzlichen Prüfungen durch. Es können von Benutzern nur Dateisysteme angehängt werden, die in der Datei `/etc/fstab` mit der Option `user` eingetragen sind. Allerdings lassen sich keine zusätzlichen Optionen angeben.

Dabei ist zu beachten, dass diese Dateisysteme mit einer Reihe von zusätzlichen Optionen versehen werden, um Sicherheitsprobleme zu vermeiden. Besonders wichtig sind dabei die Optionen `nosuid` und `nodev`. Wenn die entsprechenden Dateisysteme nicht beim Systemstart angehängt werden sollen, so ist die Option `noauto` anzugeben. Beachten Sie, dass Sie mit der Option `noexec` nur verhindern können, dass ein Programm direkt von diesem Dateisystem gestartet werden kann – ein Anwender kann das Programm kopieren oder mit Hilfe von `ld-linux.so` ausführen.

```
# Gerät    Verzeichnis  Typ     Optionen              Freq Pass
/dev/hda3  /            ext2    defaults              1       1
/dev/hda5  /usr         ext2    defaults              1       2
/dev/sbpcd /mount/cdrom iso9660 user,nodev,ro,noauto
/dev/fd0   /mount/fd0   msdos   user,noauto
/proc      /proc        proc    defaults
/dev/hda2  none         swap    sw
```

Listing 4.9 Auszug aus der Datei `/etc/fstab`

Wenn ein Dateisystem in der Datei `/etc/fstab` eingetragen ist, so kann beim manuellen Anhängen in die Linux-Verzeichnishierarchie auf beinahe alle Optionen verzichtet werden (Listing 4.9 enthält ein Beispiel für diese Datei). Die in Listing 4.10 gezeigten Befehle sind bei der in Listing 4.9 angegebenen `/etc/fstab` äquivalent.

```
(linux):~# mount -t msdos /dev/fd0 /mount/fd0
(linux):~$ mount /dev/fd0
(linux):~$ mount /mount/fd0
```

Listing 4.10 `mount`*-Befehle unter Verwendung der Datei* `/etc/fstab`

In der Datei `/etc/fstab` können eine Reihe von weiteren Optionen angegeben werden, wie z. B. die `uid` und `gid`, unter der DOS-Dateisysteme angehängt werden (DOS kennt keine Benutzerkennungen). Das fünfte Feld gibt die Frequenz an, mit der das Dateisystem mittels `dump` gesichert wird. Das sechste Feld gibt an, in welcher Reihenfolge die Dateisysteme beim Booten mittels `fsck` geprüft werden sollen. Oft wird hier das Root-Dateisystem mit einer 1 versehen und alle übrigen Dateisysteme mit einer 2. Aus Gründen der Geschwindigkeit sollte zu einem Zeitpunkt nur ein `fsck` je physikalischer Platte laufen. Das unter Linux verwendete `fsck`-Programm stellt dies auch ohne entsprechende Einträge in der Datei `/etc/fstab` sicher.

Das Programm `mount` versteht eine Reihe von Optionen, die zum Teil für die Systemsicherheit unbedingt notwendig sind. Andere Optionen sind wichtig für die bequeme und sichere Arbeit nicht privilegierter Benutzer. Im Folgenden werden einige wichtige Optionen näher erläutert. Diese können entweder in die Datei `/etc/fstab` eingetragen oder beim Aufruf des Programms `mount` nach der Option `-o` angegeben werden.

- `nodev` verhindert, dass auf einem Datenträger eine Gerätedatei verwendet werden kann, die aufgrund der zugeordneten Berechtigungen die Systemsicherheit gefährden könnte. Sämtliche Festplattengeräte (`/dev/hd*` und `/dev/sd*`) sollten nur für den Systemadministrator (und gegebenenfalls eine entsprechende Gruppe) lesbar (und schreibbar) sein. Andernfalls könnte ein Benutzer alle Daten von dieser Festplatte lesen oder gar manipulieren. Daher sollte bei allen entfernbaren Medien (auch bei CD-ROMs) diese Option angegeben werden. Das Gegenteil dieser Option ist `dev`.
- Die Ausführung von Programmen von diesem Gerät kann mit der Option `noexec` verboten werden. Die Option `exec` erlaubt die Ausführung von Programmen. Beachten Sie, dass Anwender dennoch Programme starten können, entweder durch eine Kopie in ein Dateisystem, das mit `exec` eingehängt wurde, oder mit Hilfe des dynamischen Laders `/lib/ld-linux.so`.
- Grundsätzlich kann die Ausführung von Programmen gestattet werden, aber speziell die Verwendung von `suid`-Programmen kann mit der Option `nosuid` verboten werden. Die Option `suid` erlaubt die Ausführung von `suid`-Programmen.
- Einige Partitionen werden beim Systemstart automatisch angehängt, andere nicht. Dies wird mit den Schlüsselwörtern `auto` und `noauto`, die nur in der Datei `/etc/fstab` sinnvoll sind, erreicht.
- Dateisysteme können sowohl im Schreib/Lese-Zugriff (`rw`) als auch nur im Lese-Zugriff (`ro`) angehängt werden.
- Die `root`-Partition kann nicht abgehängt werden. Daher wird sie beim Start des Systems zunächst `readonly` angehängt. Nach der Dateisystemprüfung wird die Zugriffsart mit der Option `remount` auf Schreibzugriffe geändert. Beim Herunterfahren des Systems wird mit der Option `remount` wieder in den `readonly`-Betrieb gewechselt.
- Partitionen können mit der Option `sync` angehängt werden, damit Änderungen nicht zunächst im Puffercache gepuffert werden. Der Standardwert ist `async`, mit dem eine Pufferung erlaubt wird. Für Disketten kann es sinnvoll sein, alle Daten ungepuffert zu schreiben, bevor ein Benutzer die Diskette aus dem Laufwerk entfernt.
- Nicht privilegierte Benutzer sollten auch Disketten oder CD-ROMs anhängen dürfen. Dazu wird in der Datei `/etc/fstab` die Option `user` eingetragen. Das System verwendet dann die Optionen `noexec`, `nosuid` und `nodev` als

Voreinstellung. Der Systemadministrator kann diese Optionen in der Datei `/etc/fstab` überschreiben. Sie sollten die Option `user` nur zusammen mit dem wirklichen Gerätenamen (z. B. `/dev/scd0`) verwenden und nicht mit einem symbolischen Link wie `/dev/cdrom`, da manche `mount`-Versionen dieses Dateisystem nicht wieder aushängen wollen.

- Bei jedem Zugriff auf eine Datei wird die `atime` (access time) dieser Datei verändert und auf der Festplatte gespeichert. Für Dateisysteme, bei denen dieses Verhalten nicht erwünscht ist, kann die Option `noatime` verwendet werden. Interessant ist das auf Laptops und für Verzeichnisse wie `/var/spool/news`.
- Die Option `defaults` impliziert die oben erläuterten Optionen `rw`, `suid`, `dev`, `exec`, `auto`, `nouser` und `async`.

Für einzelne Dateisysteme existieren weitere Optionen. So kennt das DOS-FAT-Dateisystem, genauso wie HPFS, keine Benutzerkennungen. Daher kann beim Anhängen dem gesamten Dateisystem eine Benutzer- und eine Gruppenkennung zugeordnet werden. Dies geschieht mit den Optionen `uid=` und `gid=`, denen jeweils die numerische Identifikation folgt. Auch die Zuordnung von Berechtigungen kann nur für das gesamte Dateisystem erfolgen. Dazu wird beim Anhängen des Dateisystems die `umask=` entsprechend angegeben. Weitere Optionen werden in der Manpage zu `mount(8)` ausführlich beschrieben.

Automatisches Mounten bei Bedarf

Das statische Mounten von Dateisystemen beim Systemstart ist für lokale Dateisysteme praktisch und notwendig. Anders sieht es bei beweglichen Datenträgern oder NFS-Volumes aus. Für Anwender ist es unangenehm, vor dem Zugriff auf den eingelegten Datenträger noch (technische) Befehle eingeben zu müssen. Für Diskettenzugriffe verwende ich daher die `mtools`.

CD-ROMs müssen unter Linux vor dem Zugriff in den Verzeichnisbaum eingehängt werden. Andere Systeme wie Solaris verfügen über das Programm `vold`, das einen eingelegten Datenträger für den Benutzer verfügbar macht. Unter Linux kann hierfür der Automounter `amd` oder das `autofs` verwendet werden. Diese Programme überwachen Verzeichnisse und mounten das entsprechende Dateisystem bei Bedarf. Wird es eine Zeit lang nicht mehr benutzt, so wird das Dateisystem wieder aus dem Verzeichnisbaum entfernt.

Seine volle Leistungsfähigkeit entfaltet der Automounter erst bei der Verwendung im Netzwerk, daher finden Sie weitere Informationen dazu in Abschnitt 20.6, »Der Automounter `amd`« und Abschnitt 20.6.3, »Das Kernel-`autofs` unter Linux«.

Eine weitere Möglichkeit, Dateisysteme automatisch einzubinden, ist der Supermount-Patch. Dieser ist umstritten und wird vermutlich nicht Bestandteil des Standard-Kernels werden. Er kann in vielen Fällen durch `amd` oder `autofs` ersetzt werden.

4.4.2 Der Logical Volume Manager

Eine der wichtigsten Neuerungen im Kernel 2.4 ist die Implementation des Logical Volume Manager (LVM, *http://www.sistina.com/lvm*). Bisher war ein Dateisystem auf eine Partition (unter BSD- bzw. Unix-Systemen heißt ein ähnliches Konzept Slice) begrenzt[1]. Der Logical Volume Manager arbeitet mit einem ganz anderen Konzept, hier spricht man von Physical Volumes (PV) und Logical Volumes (LV).

Neben einem aktuellen Kernel (oder dem passenden Patch für eine alte Version) benötigen Sie die Programme zur Verwaltung von Physical und Logical Volumes. Aktuelle Distributionen bringen die entsprechenden Programme bereits mit.

Physical Volumes

Ein Physical Volume, kurz PV genannt, ist eine normale Partition auf einer Festplatte, die vom Logical Volume Manager verwendet werden soll. Die Partition hat den Typ `0x8e`. Dieser Partitionstyp ist unter Linux für Physical Volumes reserviert. Anschließend wird die Partition dem Logical Volume Manager mit dem Befehl `pvcreate` als Physical Volume bekanntgemacht. Ein Beispiel dafür finden Sie in Listing 4.11.

```
linux:~# fdisk -l /dev/sda
Festplatte /dev/sda: 255 Köpfe, 63 Sektoren, 522 Zylinder
Einheiten: Zylinder mit 16065 * 512 Bytes

Gerät     boot.  Anfang      Ende    Blöcke   Id  Typ
/dev/sda2    *         1        26    208813+  83  Linux
/dev/sda3             27       522   3984120   8e  Unbekannt
linux:~# pvcreate /dev/sda3
```

Listing 4.11 Erzeugen eines Physical Volume

Weitere Programme zur Verwaltung von Physical Volumes sind `pvchange` (Attribute eines PV ändern), `pvdata` (als Fehlersuchhilfe), `pvdisplay` (zur Anzeige von Daten eines Physical Volumes), `pvmove` (Verschieben von Daten auf andere PVs) und `pvscan` (Suche nach PVs). Für genauere Informationen und die möglichen Optionen schauen Sie bitte in die Manpages.

Volume Groups

Der Logical Volume Manager verwaltet eine Gruppe von Physical Volumes in einer so genannten Volume Group (VG). Das Programm `vgcreate` erzeugt eine

1. Na ja, nicht ganz. Spätestens seit der Einführung des `md`-Treibers kann sich ein Dateisystem über mehrere Partitionen erstrecken.

Volume Group, diese kann mit dem Programm `vgextend` erweitert werden. Ein Beispiel dafür finden Sie in Listing 4.12. Der Befehl `vgcreate` erstellt die Volume Group `rootvg`, die zunächst nur das Physical Volume `/dev/sda3` enthält. Das Programm `vgextend` nimmt das Physical Volume `/dev/sdb1` zusätzlich in die Volume Group auf.

```
linux:~# vgcreate rootvg /dev/sda3
linux:~# vgextend rootvg /dev/sdb1
```

Listing 4.12 Erzeugen einer Volume Group mit `vgcreate`

Wenn Sie bei dem Programm `vgcreate` keine zusätzlichen Parameter angeben, dann ist die Größe der Volume Group auf 256 Gbyte beschränkt. Mit der Option `-s` können Sie einen anderen Wert als 4 Mbyte für die Größe eines physikalischen Extent[2] angeben. Beachten Sie außerdem, dass die Anzahl der Physical Volumes und Logical Volumes vom Kernel auf 256 beschränkt sind. Diese Größen werden derzeit nur von wirklich großen Servern erreicht – und selbst da wird es langsam unhandlich.

Das Gegenstück zu `vgextend`, das ein Physical Volume in eine Volume Group hinzufügt, ist `vgreduce`. Ein Physical Volume kann nur dann mittels `vgreduce` aus der Volume Group entfernt werden, wenn auf diesem Physical Volume keine Daten mehr gespeichert sind. Mit dem Befehl `pvdisplay` stellen Sie fest, ob das Physical Volume belegt ist oder nicht, die Option `-v` sorgt dabei für eine ausführliche Anzeige, so dass Sie genau wissen, welche Logical Volumes sich dort noch tummeln. Mit dem Befehl `pvmove` können Sie die Daten auf ein anderes Volume verschieben, ohne den Systembetrieb zu stören[3].

Weitere nützliche Befehle zu Volume Groups sind `vgchange` (Ändern von Attributen der Volume Group), `vgdisplay` (Anzeige von Daten zur Volume Group), `vgrename` (Umbenennen einer inaktiven Volume Group) und `vgscan` (Suche nach Volume Groups und Erzeugen der Dateien `/etc/lvmtab` und `/etc/lvmtab.d`).

Wenn Sie die Daten einer Volume Group auf einem anderen System benötigen, so müssen Sie die Volume Group zunächst aus dem laufenden System entfernen. Die Dateisysteme dürfen nicht mehr aktiv sein, dann kann man die Volume Group mit `vgchange` deaktivieren und mit `vgexport` aus dem System exportieren. Nachdem die Festplatten in das neue System eingebaut sind, kann die Volume Group mit `vgimport` importiert werden. Anschließend lässt sich diese verwenden, als wäre sie auf diesem System angelegt worden.

2. Aus den »Physical Extents« werden die einzelnen Logical Volumes zusammengesetzt – und die Anzahl der Physical Extents ist beschränkt.
3. Naja, nicht ganz: Die Systemlast beim Kopieren der Daten ist durchaus spürbar, aber das ist wesentlich besser als ein kompletter Ausfall des Systems.

Der Logical Volume Manager verwaltet in der Regel viele wichtige Daten. Daher ist es notwendig, nicht nur die Daten in den Dateisystemen, sondern auch die Metainformationen, die den Aufbau der Volume Group beschreiben, regelmäßig zu sichern. Standardmäßig geschieht das in Dateien im Verzeichnis `/etc/lvmconf`, wenn z. B. die Volume Group erweitert oder verkleinert wird. Zusätzlich stehen die Kommandos `vgcfgbackup` und `vgcfgrestore` zum Sichern bzw. Wiederherstellen der Metainformationen zur Verfügung.

Im Verzeichnis `/dev` wird für jede Volume Group ein Unterverzeichnis angelegt, das den Namen der Volume Group trägt. In diesem Verzeichnis finden sich ein Device-Inode für die Volume Group selbst und später für jedes dort enthaltene Logical Volume.

Logical Volumes

Bisher haben wir uns nur mit der Verwaltung beschäftigt – wo aber liegt denn nun der große Vorteil des Logical Volume Managers? Ganz einfach, in der flexiblen Gestaltung der Logical Volumes, in denen später Dateisysteme angelegt werden können.

Das Programm `lvcreate` erzeugt ein neues Logical Volume. Das Listing 4.13 zeigt ein Beispiel für ein 128 Mbyte großes Volume. Hier wird auch gleich das passende Dateisystem neu erzeugt und eingehängt.

```
linux:~# lvcreate -n tmplv -L 128M rootvg
lvcreate -- doing automatic backup of "rootvg"
lvcreate -- logical volume "/dev/rootvg/tmplv"
            successfully created
linux:~# mke2fs /dev/rootvg/tmplv
mke2fs 1.17, 26-Oct-1999 for EXT2 FS 0.5b, 95/08/09
...
linux:~# mount /dev/rootvg/tmplv /mnt
...Verwenden des Dateisystems...
linux:~# umount /mnt
linux:~# lvremove /dev/rootvg/tmplv
lvremove -- do you really want to remove "/dev/rootvg/tmplv"?
[y/n]: y
lvremove -- doing automatic backup of volume group "rootvg"
lvremove -- logical volume "/dev/rootvg/tmplv"
            successfully removed
```

Listing 4.13 Erzeugen eines Logical Volume mit `lvcreate`

Das ist alles noch nichts Neues – nur jede Menge Arbeit. Zur Verdeutlichung finden Sie in Abbildung 4.1 eine kleine Volume Group mit drei Volumes. Der schraffiert dargestellte Bereich ist ein Logical Volume, das sich (wie hier gezeigt) über mehrere Physical Volumes erstrecken kann. Die Zuordnung von freiem Platz in der Volume Group erfolgt in so genannten Physical Extents, deren Grö-

ße bei der Anlage der Volume Group angegeben werden kann. Der Standardwert ist 4 Mbyte, es sind Potenzen von zwei zwischen 8 Kbyte und 512 Mbyte möglich.

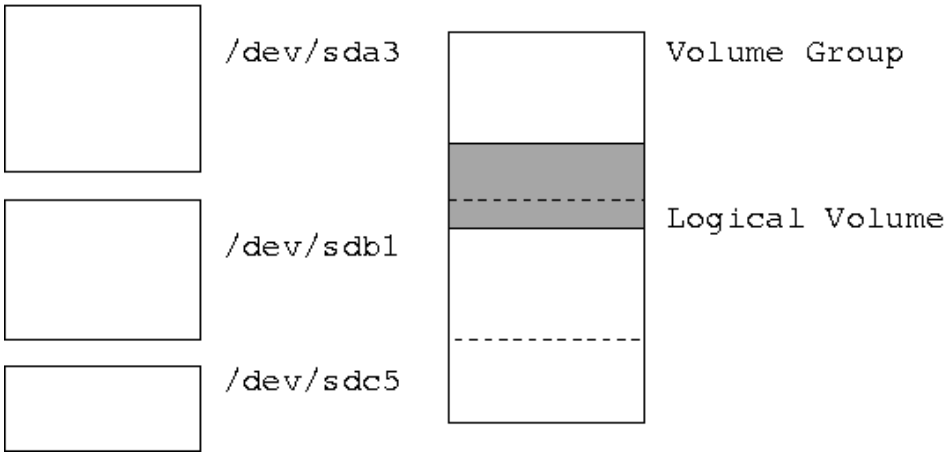


Abbildung 4.1 Eine kleine Volume Group

Der entscheidende Punkt ist, dass man ein Logical Volume so lange vergrößern kann, wie man noch freien Platz in der Volume Group hat. Und selbst wenn diese voll ist, kann man immer noch ein neues Physical Volume definieren und in die Volume Group aufnehmen. Das Vergrößern von Logical Volumes erfolgt mit dem Programm `lvextend`, das Verkleinern mit `lvreduce`. Das Listing 4.14, zeigt ein Beispiel. Sie können die neue Größe des Logical Volume entweder direkt angeben oder relativ zur bisherigen Größe.

```
linux:~# lvextend -L+256 /dev/rootvg/tmplv
lvextend -- extending logical volume "/dev/rootvg/tmplv"
            to 384 MB
lvextend -- doing automatic backup of volume group "rootvg"
lvextend -- logical volume "/dev/rootvg/tmplv"
            successfully extended
linux:~# lvreduce -L-128 /dev/rootvg/tmplv
lvreduce -- WARNING: reducing active logical volume to 256 MB
lvreduce -- THIS MAY DESTROY YOUR DATA
lvreduce -- do you really want to reduce "/dev/rootvg/tmplv"?
[y/n]: y
lvreduce -- doing automatic backup of volume group "rootvg"
lvreduce -- logical volume "/dev/rootvg/tmplv"
            successfully reduced
```

Listing 4.14 Vergrößern bzw. Verkleinern eines Logical Volume

Das ist bisher aber nur die halbe Wahrheit. Jetzt ist in dem Device, das das Dateisystem enthält, wieder Platz, aber das ext2-Dateisystem wird diesen niemals benutzen. Hier sind Programme gefragt, die das Dateisystem entsprechend anpassen können. Dabei gibt es zwei Varianten, Online- und Offline-Programme. Als freie Variante eines Offline-Programms ist das `ext2resize` verfügbar, es erhält als Parameter erst das Device und dann die neue Größe des Dateisystems in Blöcken (Listing 4.15).

```
linux:~# ext2resize /dev/rootvg/tmplv 393216
ext2_resize_fs
...
```

Listing 4.15 Vergrößern eines Dateisystems

Das Programm `e2fsadm` ist ein Frontend zu `lvextend` und `ext2resize`, so dass nur noch ein Befehl ausgeführt werden muss. Außerdem startet das Programm vor der ersten Änderung eine Dateisystemprüfung, so dass nur korrekte Dateisysteme bearbeitet werden.

Im Paket ist außerdem das Programm `ext2online` enthalten, mit dem auch ein aktuell eingehängtes Dateisystem bearbeitet werden kann. Dafür ist derzeit noch ein Kernel-Patch erforderlich (der ebenfalls im Paket enthalten ist), außerdem kann nur bis zu einer gewissen Größe erweitert werden (bis zur nächsten 256 Mbyte-Grenze bei 1K-Blöcken, 2 Gbyte-Grenze bei 2K-Blöcken und zur nächsten 16 Gbyte-Grenze bei 4K-Blöcken). Wenn Sie ein Dateisystem verkleinern wollen, dann nehmen Sie `ext2resize` oder besser `e2fsadm`, dann erhält auch das Logical Volume gleich die passende Größe.

Das Programm `ext2prepare` bearbeitet die Struktur eines inaktiven Dateisystems so, dass dieses später mit `ext2online` auch über diese Grenze erweitert werden kann. Als Parameter erwartet es das Device und die maximal geplante Größe. Beachten Sie, dass alle diese Programme noch recht neu sind – eine Datensicherung sollten Sie haben.

Mit der Einführung des Logical Volume Managers verliert man (zunächst) die Kontrolle darüber, welches Dateisystem sich auf welcher physikalischen Festplatte befindet. Das ist eigentlich auch gut so, denn in der Regel braucht man das auch nicht zu wissen. Dennoch hat der Systemverwalter mit dem Befehl `lvdisplay -v` *`LV`* die Möglichkeit, sich die Verteilung eines Logical Volume über die verschiedenen Partitionen anzeigen zu lassen. Mit dem Befehl `pvmove` können Physical Extents von einem Physical Volume zu einem anderen verschoben werden. Das kann z. B. dann notwendig sein, wenn eine Platte Fehler hat und ausgetauscht werden muss. Mit etwas Glück können die Daten verschoben und dann das Physical Volume aus der Volume Group entfernt werden.

Weitere nützliche Ergänzungen

Die Funktionen des Logical Volume Managers sind besonders für größere Systeme nützlich, dennoch ist der Einsatz auch für Heimanwender sinnvoll. Wenn nämlich mal ein Dateisystem voll wird, so blieb bisher nur das Sichern der Daten (z. B. auf Band oder CD), dann das Löschen und Neuanlegen der Partitionen und am Ende das Einspielen der Daten. Wenn man Glück hat, dann kann man die Daten auf eine freie Partition umkopieren, aber lästig und aufwändig ist das allemal.

Mit dem LVM gibt es praktisch keine vollen Dateisysteme mehr, wenn der Systemverwalter diese rechtzeitig vergrößert. Insbesondere der Einsatz von `ext2online` kann die Verfügbarkeit von Servern deutlich verbessern.

In diesem Zusammenhang fehlte Linux lange ein so genanntes Journaled File System, das besonders bei großen Dateisystemen dafür sorgt, dass ein eventuell notwendiger Filesystem-Check nur noch wenige Sekunden dauert. Hier sind derzeit mehrere Entwicklungen praktisch abgeschlossen: eine Erweiterung des ext2-Dateisystems bzw. des Reiser-fs, der Port des XFS von SGI und der Port den JFS von IBMs OS/2.

Neben Partitionen kann LVM auch auf RAID-Devices aufsetzen. Außerdem kann man ein Logical Volume über mehrere Physical Volumes verteilen (stripen), um eine höhere Schreib- und Lese-Geschwindigkeit zu erreichen. Zur Datensicherheit gilt in etwa das, was auch im Abschnitt 8.2, »Plattenfehler überstehen mit RAID« erwähnt wird. Wenn eine Platte aus der Volume Group ausfällt, sind *alle* Daten in der Volume Group gefährdet – mindestens jedoch die dort gespeicherten Daten sind verloren.

Die Zukunft des LVM

Heute zeichnen sich mehrere Entwicklungen ab:

- Distributionen können bereits bei der Installation Logical Volumes anlegen und dort installiert werden. Damit ist es für Anwender sehr einfach, den Logical Volume Manager einzusetzen – in Zukunft wird man insbesondere auf Servern nicht mehr auf die Vorteile verzichten wollen.
- Neben der Weiterentwicklung des LVM gibt es das Enterprise Volume Management System (EVMS, *http://evms.sourceforge.net/*), das neben Plugins für den traditionellen Linux LVM auch Plugins für die LVMs von AIX und OS/2 enthält. Außerdem ist eine grafische Benutzeroberfläche in der Entwicklung, so dass sich der Einsatz eines Volume Managers weiter vereinfachen wird.
- Derzeit werden LVM2 und EVMS von verschiedenen Gruppen und mit unterschiedlichen Zielen entwickelt. Was sich am Ende durchsetzen wird, ist noch unklar – die Anwender haben die Wahl.

4.4.3 Drucken unter Linux

Unter Unix-Systemen funktioniert das Drucken deutlich anders, als man das beispielsweise von Windows her kennt. Unter Windows installiert man im Betriebssystem einen Druckertreiber, der von allen Programmen zur Druckaufbereitung verwendet wird. Für Programmierer ist dies recht einfach zu programmieren, allerdings sehen die Druckausgaben auf den verschiedenen Druckern immer wieder anders aus. Unter Unix ist zunächst jedes Programm selbst für die Aufbereitung zuständig, daher wird in vielen Fällen PostScript erzeugt, das dann gedruckt wird. Der Vorteil ist hier, dass die Druckaufbereitung nicht vom Druckertyp abhängt. Der Nachteil ist, dass das Drucken für den Programmierer unter Unix kompliziert ist. Auch das X-Window-System hat hier keine wesentliche Verbesserung gebracht, allerdings sind in die Desktops KDE und GNOME entsprechende Bibliotheken integriert. Viele nützliche Informationen zum Drucken unter Linux finden Sie auf der Webseite *http://www.linuxprinting.org/*.

Der einfachste Weg des Druckens ist in der Regel, den Ausdruck innerhalb der Applikation zu einem PostScript-Drucker zu schicken und diese Daten dann bei Bedarf in das Druckerformat umzuwandeln. Wie das genau geht, werden wir auf den folgenden Seiten sehen.

Zunächst muss zum Drucken unter Linux der entsprechende Druckerport vom Kernel erkannt und verwaltet werden. Dazu kann entweder ein paralleler oder serieller Anschluss verwendet werden. Hier wird im Weiteren vom Anschluss des Druckers am Parallelport ausgegangen. Der Treiber für diesen Port kann entweder bei der Kernel-Konfiguration in den Kernel einkompiliert oder später als Modul geladen werden (siehe auch Abschnitt 4.2.1, »Konfigurieren des Kernels« und Abschnitt 4.3, »Kernel-Module«). Wenn der Treiber in den Kernel einkompiliert wurde, sollte nach dem Neustart des Rechners eine Meldung ähnlich der in Listing 4.16 ausgegeben werden.

```
lp1 at 0x0378, using polling driver
```

Listing 4.16 Initialisierung des Parallelports durch den Kernel

Damit hat der Kernel die entsprechende Schnittstelle erkannt und initialisiert. Einige Parameter der Schnittstelle können mit dem Programm `tunelp` verändert werden. Zunächst wird der Treiber im so genannten »Polling-Mode« betrieben, in dem keine Interrupts zur Ansteuerung der parallelen Schnittstelle verwendet werden. Das System wird jedoch insgesamt weniger belastet, wenn der Druckerport im Interrupt-Betrieb benutzt wird. Dazu ist allerdings ein freier Interrupt erforderlich und die verwendete Schnittstellenkarte muss diese Funktion unterstützen.

Eine weitere nützliche Option von `tunelp` ist `-C`, die besonders vorsichtige Prüfungen beim Drucken verlangt. Dadurch kann der Drucker zunächst ausgeschaltet bleiben. Die Daten werden so lange zwischengespeichert und erst dann ge-

druckt, wenn der Drucker eingeschaltet wird. Ausführliche Informationen zu `tunelp` finden Sie in der entsprechenden Manpage. Weitere Informationen zum Drucken unter Linux erhalten Sie auch in der Printing-HowTo.

Zunächst sollte ein Test erfolgen, ob der Drucker tatsächlich von Linux aus angesprochen werden kann. Dies kann als `root` einfach mit dem Befehl `cat /etc/passwd > /dev/lp1` erfolgen. Unmittelbar nach diesem Befehl sollte der Drucker mit einem Ausdruck beginnen. Dabei kann es, abhängig vom Druckertyp, verschiedene Probleme geben.

Ein bei vielen Druckern auftretendes Phänomen ist der so genannte Staircase-Effekt. Dabei wird eine neue Zeile nicht in der ersten Spalte der nächsten Zeile begonnen, sondern hinter dem letzten Zeichen der vorhergehenden Zeile. Die Ursache hierfür ist, dass in Texten unter Linux (und Unix) die einzelnen Zeilen mit dem Zeichen LF (Line Feed) getrennt werden. In Texten unter DOS erfolgt diese Trennung mit der Zeichenkombination CR und LF (Carriage Return und Line Feed). Viele Drucker sind so konfiguriert, dass unbedingt die Zeichen CR und LF notwendig sind, um eine neue Zeile zu beginnen. Dieses Problem lässt sich auf zwei Arten lösen:

- Der Text kann vor der Ausgabe entsprechend umgesetzt werden. Das kann zum Beispiel mit dem Programm `recode` (siehe auch Abschnitt »Zeichensatzkonvertierung« in Kapitel 9.3.4) oder einem kurzen `sed`-Skript (Abschnitt 9.7, »Textdateien bearbeiten mit `sed`«) erfolgen.
- Der Drucker kann entweder mittels einer Steuersequenz oder mit einer veränderten Einstellung der DIP-Schalter des Druckers so konfiguriert werden, dass das Zeichen LF als CR+LF interpretiert wird. Diese Einstellung hat auch bei der parallelen Verwendung unter DOS keine negativen Auswirkungen, da das zweite CR ignoriert wird. Bitte sehen Sie hierfür in Ihr Druckerhandbuch.

Ein weiteres Problem ist, dass unter Linux der ISO-Latin-1-Zeichensatz verwendet wird, unter DOS jedoch meist die Codepage 437 oder 850. Damit werden Umlaute und andere Sonderzeichen unterschiedlich kodiert, so dass Ausdrucke, die Umlaute enthalten, fehlerhaft sind. Hier kann entweder der Drucker durch eine geeignete Steuersequenz oder mittels DIP-Schaltern konfiguriert oder der Text entsprechend umgesetzt werden. Zu diesem Zweck eignet sich wieder das Programm `recode`.

Das direkte Ansprechen der Druckerschnittstelle, das wir zum Testen verwendet haben, hat verschiedene Nachteile:

- Die Schnittstelle muss für alle Benutzer beschreibbar sein.
- Der gleichzeitige Zugriff mehrerer Benutzer wird nicht koordiniert.
- Die Ausgabe muss immer für den Drucker fertig aufbereitet sein.
- Der Drucker muss eingeschaltet sein, wenn eine Druckausgabe erfolgen soll.

Alle diese Probleme können mit dem `lpr`-System gelöst werden. Unter Linux wird das auch von den BSD-Systemen bekannte System eingesetzt, nicht das `lp`-System von System V. Im folgenden Abschnitt wird zunächst der Drucker in das System eingebunden und anschließend die Aufbereitung der Ausgabe besprochen.

Das lpr/lpd-System

Das BSD-Drucksystem besteht aus einer Reihe von Programmen und Dateien, die zur Verwaltung von Druckaufträgen in Warteschlangen benutzt werden. Druckaufträge werden mit dem Programm `lpr` erstellt, sie können später mit dem Programm `lprm` gelöscht werden. Anstehende Druckaufträge werden mit `lpq` angezeigt. Der Systemadministrator kann mit dem Programm `lpc` das gesamte Drucksystem steuern. Die zentrale Konfiguration erfolgt in der Datei `/etc/printcap`. Für den Einsatz des `lpd` kann es notwendig sein, das Netzsystem minimal (für `loopback`) zu konfigurieren. Hinweise dazu finden Sie z. B. im Abschnitt 14.4.3, »Aktivierung der Netzwerkgeräte«.

Die Datei /etc/printcap

Die Datei `/etc/printcap` (Printer Capabilities) ist die zentrale Konfigurationsdatei für das BSD-Drucksystem. Diese Datei hat ein ähnliches Format wie die Datei `/etc/termcap`. Jedes Feld wird durch Doppelpunkte separiert, die Fortsetzung von langen Zeilen erfolgt durch einen Backslash (\) am Zeilenende. Hier wird für jeden Drucker eingetragen, welche Warteschlangen für ihn zuständig sind und wie Druckdaten zu behandeln sind. Ein solcher Eintrag wird auch *Stanza* genannt.

Das Listing 4.17 zeigt einen Ausschnitt aus einer `/etc/printcap`-Datei. Dieser Auszug enthält nur die wichtigsten Einträge, die zunächst erläutert werden, bevor komplexere Einträge erstellt werden.

Ein einzelner Eintrag (Capability) besteht aus einem zweistelligen Kürzel. Alle Kürzel werden in der Manpage zu `printcap(5)` erläutert. Kürzeln, die als Parameter einen String benötigen, folgt ein Gleichheitszeichen (=), numerische Parameter werden mit einer Raute (#) abgetrennt.

```
lp|Standarddrucker:\
        :lp=/dev/lp1:\
        :sd=/var/spool/lpd/lp1:\
        :sh:\
        :mx#0:\
        :lf=/var/log/lpd:\
        :lo=/var/spool/lpd/lp1/lock/lpd.lock
```

Listing 4.17 Auszug aus der Datei `/etc/printcap`

- Das erste Feld gibt den Namen des Druckers an. Mehrere Namen können durch senkrechte Striche (|) getrennt angegeben werden. Der letzte Name ist normalerweise auch der ausführlichste.
- Der Wert `lp` (line printer) gibt die Schnittstelle an, die zum Drucken benutzt werden soll. Hier wird die lokale Schnittstelle `/dev/lp1` verwendet.
- Der Wert `sd` (spool directory) legt das Verzeichnis fest, in dem Druckaufträge zwischengespeichert werden. Je Warteschlange sollte ein eigenes Verzeichnis verwendet werden.
- Der Wert `sh` (suppress header) unterdrückt die Ausgabe der Header-Seite. Dies ist für Single-User-Systeme sinnvoll. Bei Druckern, die von mehreren Benutzern verwendet werden, sollte zumindest eine kurze Header-Zeile ausgegeben werden (`sb` = short banner). Wird ein Drucker aber von vielen Benutzern verwendet, dann ist die Ausgabe einer Trennseite zwischen den Listen sehr sinnvoll.
- Das Drucksystem nimmt standardmäßig nur Aufträge an, die kleiner als 1 Mbyte sind. Mit dem Wert `mx` kann diese Kenngröße verändert werden. Der Wert 0 steht dabei für unendliche Größe. Eine andere Möglichkeit, große Dateien zu drucken, ist die Verwendung des Schalters `-s` (symbolischer Link) beim Druck mit `lpr`. Sie dürfen eine derartige Datei nicht bearbeiten oder löschen, solange der Druck nicht endgültig erfolgt ist.
- Die Datei `/var/log/lpd` wird als Log-Datei verwendet. Der Name wird mit Hilfe der Capability `lf` festgelegt.
- Der Wert `lo` legt den Namen fest, der für die Lock-Datei verwendet werden soll.

Es können viele (verschiedene) Drucker in die Datei `/etc/printcap` aufgenommen werden. Der Standarddrucker des Systems heißt üblicherweise `lp`. Jeder Benutzer kann mit der Umgebungsvariablen `PRINTER` einen anderen Drucker als Standard setzen. Zusätzlich kann beim Aufruf von `lpr` mit dem Schalter `-P` ein anderer Drucker angewählt werden. Weitere Informationen über das Drucksystem und seine Konfiguration finden Sie in den Handbuchseiten zu `lpr(1)`, `lpq(1)`, `lprm(1)`, `lpd(8)`, `lpc(8)` und `printcap(5)` sowie in der Printing-HowTo.

Drucken auf entfernten Rechnern

Das `lpr`-System bietet auch die Möglichkeit, auf entfernten Rechnern zu drucken. Dazu muss natürlich zwischen diesen Rechnern eine Netzwerkverbindung bestehen. Mehr zum Thema Netzwerke finden Sie in Kapitel 14, »Linux in einer vernetzten Umgebung«.

Auf dem Druck-Server (der Rechner, an dem der Drucker physikalisch angeschlossen ist) muss der Zugriff auf den Drucker von anderen Rechnern aus zugelassen werden. Dazu wird der entsprechende Rechnername in die Datei `/etc/`

`hosts.lpd` aufgenommen. Hier muss der Name eingegeben werden, der beim Reverse-Lookup zur IP-Adresse gefunden wird. Wenn Sie DNS verwenden, dann ist dies der Name inklusive Domain.

Auf dem Client, der jetzt auf am Server angeschlossene Drucker zugreifen darf, muss in der Datei `/etc/printcap` der entsprechende entfernte Drucker eingerichtet werden. Das erfolgt mit den Einträgen `rm` (remote machine) und `rp` (remote printer), siehe Listing 4.18. Es ist sinnvoll, die Namen der Druckerwarteschlange (Printqueue) auf dem Server und den Clients identisch zu halten. Leider müssen sich die `/etc/printcap`-Dateien zwischen dem Server und den Clients unterscheiden, so dass diese nicht direkt im Netz verteilt werden können. Möglicherweise kann Ihnen das GNU-Programm `cfengine` hier helfen.

```
drucker|Am Server angeschlossener Drucker:\
        :lp=:\
        :rp=drucker-am-server:\
        :rm=server:\
        :sd=/var/spool/lpd/drucker:
```

Listing 4.18 `printcap` *für einen Druck-Client*

Neben dem BSD-portierten `lpr` gibt es weitere Drucksysteme. Oft wird das PLP-System (Portable Line Printer) verwendet, manchmal auch LPRng (`lpr` next Generation). Beide Systeme sind zunächst analog zu dem hier vorgestellen `lpr` zu konfigurieren, haben aber für den einen oder anderen Zweck möglicherweise Vorteile.

PLP – das Portable Line Printer Spooler System

PLP (Portable Line Printer Spooler System, *http://jmason.org/plp.html*) ist eine Neuimplementierung des BSD-`lpr`-Systems. Dabei wurde Wert auf eine flexiblere Konfiguration, mehr Sicherheit und bessere Interoperabilität mit anderen LPD-Systemen gelegt. Der originale BSD-`lpr` bezieht seine Konfiguration nur aus der Datei `/etc/printcap`, dort können nur Printqueues eingerichtet werden. Im Gegensatz dazu wertet PLP zusätzlich die Datei `plp.conf` aus, in der weitere Konfigurationen durchgeführt werden können.

Ebenfalls vereinfacht wurde die Administration von Druckern im Netzwerk. In der Datei `/etc/printcap` können Drucker in Abhängigkeit vom Rechner definiert werden. Damit kann eine Queue-Definition auf dem Server anders aussehen als auf dem Client, es kann aber auf beiden Rechnern dieselbe Datei verwendet werden. Außerdem können Einträge aus anderen Dateien gelesen werden, so dass der Administrator Queues sehr flexibel einrichten kann. Neben der Verteilung der vollständigen `/etc/printcap` kann man die Queue-Definitionen auch in einer eigenen NIS-Datenbank verwalten. Die PLP-Dokumentation enthält entsprechende Beispiele.

Im einfachsten Fall benötigt ein Client keine Datei `/etc/printcap`. Beim Aufruf von `lpr` kann mit Hilfe der `-P`-Option einfach die gewünschte Queue auf dem gewünschten Server angegeben werden, etwa mit `lpr -P`*`Queue@Server`*.

Ein weiterer Vorteil von PLP gegenüber dem BSD-`lpr` ist die Möglichkeit, einen Input-Filter auch für Remote-Printer einzurichten. Damit kann ein Linux-Rechner zwischen verschiedenen Formaten und Protokollen ohne großen Aufwand konvertieren.

Berechtigungen für den Zugriff auf die Drucker lassen sich global und je Queue in der Datei `/etc/printer_perms` vergeben. Dabei kann man für jeden Rechner und Benutzer abhängig vom Namen der Queue eine maximale Priorität festlegen. Standardpriorität ist X, A ist die höchste mögliche Priorität.

```
#host   user    group queue Op  priority  max Pg  current
*       root    *     *     R   A         0       0
*       *       *     *     R   C         0       0
```

Listing 4.19 Die Datei `printer_perms`

In den Namen steht das Fragezeichen (`?`) für genau ein beliebiges Zeichen, der Stern (`*`) für eine beliebige Anzahl von Zeichen. Ein Ausrufezeichen in der ersten Spalte sorgt dafür, dass der Zugriff verweigert wird, wenn die aktuelle Zeile auf die Operation passt. Wird in der ersten Spalte eine Tilde (`~`) angegeben, so wird der Zugriff verweigert, wenn die Zeile nicht auf die Operation passt. Im Gegensatz zur Markierung mit dem Ausrufezeichen wird die Suche nach weiteren Matches fortgesetzt.

In den ersten vier Feldern werden der Host-Name, der Benutzername, die Unix-Gruppe und der Queue-Name angegeben, so dass man je nach Rechner, Benutzer oder Queue eigene Berechtigungen vergeben kann. Dadurch kann die Datei groß und unübersichtlich werden, andererseits hat man hier wenigstens eine Kontrolle über die Zugriffsrechte. Wenn PLP mit NIS-Support übersetzt wurde, dann können Sie im Feld `host` oder `user` auch Netgroups mit `@`*`netgroup`* angeben.

Das fünfte Feld enthält die für den Anwender erlaubten Operationen, dabei steht `C` für »Control« und erlaubt die Verwendung von `lpc` und `lprm` für die angegebenen Queues. Zusätzlich ist natürlich das Drucken auf diese Drucker erlaubt. Das Zeichen `M` steht für »Move« und gestattet das Verschieben von Jobs mit dem `lpc`-Kommando `move`. Das Drucken und das Entfernen von eigenen Druckjobs erlaubt der Eintrag `R` im Feld `queue OP`.

Das Feld `priority` legt die maximal erlaubte Priorität fest. Erlaubt sind Einträge von `A-Z`, die Standardpriorität ist `X`. Die Priorität kann beim Aufruf von `lpr` mit der Option `-C` angegeben werden.

Die letzten beiden Spalten können dazu verwendet werden, die Anzahl der gedruckten Seiten zu beschränken. Das Feld `max` enthält die maximal erlaubte Anzahl von Seiten, das letzte Feld die aktuell gedruckten Seiten.

Neben den vielen neuen Optionen und Konfigurationsmöglichkeiten bietet PLP eine Reihe von Funktionen zur Fehlersuche. Das wichtigste Programm ist `checkpc`, das die Konfigurationsdateien liest und gefundene Fehler anzeigt. Wenn Sie die Option `-f` angeben, dann versucht `checkpc`, die Fehler zu beheben. Mehr Informationen hierzu finden Sie in der Manpage zu `checkpc(8)`. Beim Aufruf von `lpr` und `lpd` können Sie mit der Option `-D` Ausgaben zur Fehlersuche anfordern. Auch hier finden Sie weitere Details in der Manpage.

LPR next generation

Patrick Powell, der ursprüngliche Autor von PLP, hat die Wartung an Justin Mason abgegeben und mit der Implementation von LPRng (*http://www.lprng.org/*) begonnen. LPRng ist wiederum eine Neuentwicklung, die kompatibel zum ursprünglichen BSD-`lpr`-System ist. Sie enthält viele der Funktionen von PLP, wurde aber basierend auf den bisher gesammelten Erfahrungen neu entworfen. Ein Grund für die Entwicklung von PLP und LPRng waren die Probleme mit [rfc1179]: Der RFC ist die Beschreibung einer Implementation (nämlich der BSD-Implementation). Es wurden keine Erweiterungsmöglichkeiten vorgesehen, die Beschreibung ist unvollständig und in einigen Fällen undeutlich formuliert. Aus diesen Gründen sind viele LPD-Implementationen miteinander inkompatibel, das heißt, sie können nicht in jedem Fall Druckjobs austauschen. Diese Probleme sind für Anwender nur schwer zu durchschauen. Die LPRng-HowTo enthält einige weitere Informationen dazu.

Die Konfiguration von LPRng erfolgt in der Datei `/etc/lpd.conf`. Wie bei PLP ist das Drucken ohne Konfiguration bzw. `/etc/printcap` mit dem Aufruf `lpr -P`*`printer@host`* möglich. Der Vorteil gegenüber PLP ist ein erweitertes Format der `/etc/printcap`, das z. B. die Option `tc=` zum Einbinden einer weiteren Konfiguration kennt. Damit und mit zusätzlichen Optionen für die Generierung des `sd=`-Eintrags ist es einfacher, komplexe `printcap`-Dateien zu erstellen.

Unter PLP kann zum Zugriff auf die `printcap`-Datenbank Hesiod oder NIS verwendet werden. In der Konfigurationsdatei `plp.conf` muss das entsprechend eingerichtet werden und die Programme müssen geeignet kompiliert werden. LPRng kennt das Konzept eines `printcap`-Filters, ein Programm, das auf der Standardeingabe den Queue-Namen liest und den zugehörigen `printcap`-Eintrag auf der Standardausgabe ausspuckt. Damit ist der Anwender viel flexibler; insbesondere benötigt `lpd` keinerlei Kenntnisse über die verwendeten Zugriffsverfahren.

Wie bei PLP wird zur Fehlersuche das Programm `checkpc` mitgeliefert. Mit Hilfe dieses Programms ist es sehr einfach möglich, die verschiedensten Konfigurationsprobleme aufzuspüren und oft automatisch beheben zu lassen. Besonders

hilfreich ist die Überprüfung der Zugriffsberechtigungen auf Programme, Dateien und Verzeichnisse.

LPRng hat wie PLP ebenfalls eine Datei, in der Zugriffsrechte auf die verschiedenen Queues und Funktionen vergeben werden können. Aufgrund der Erfahrungen mit PLP hat diese aber ein anderes Format erhalten. Im Vergleich zur BSD-Implementation wurden auch die Filtermöglichkeiten stark erweitert, LPRng kann aber mit dem `printcap`-Eintrag `:bkpf:` in das alte Format umgeschaltet werden. Insgesamt bietet LPRng viele nützliche Funktionen für Netzwerke mit anspruchsvollen Druckerkonfigurationen. Für einfache Netze ist bereits die BSD-Implementation in vielen Fällen ausreichend.

CUPS – Common Unix Printing System

Die aktuellen `lpr`-Entwicklungen sind Fortschritte, aber die Anforderungen an Print-Systeme steigen immer weiter. Insbesondere die Beschränkungen des `lpd`-Protokolls, auch in Bezug auf die Interoperabilität zwischen verschiedenen `lpd`-Implementationen, sorgten für die Entwicklung des *Internet Printing Protocol* (IPP, *http://www.pwg.org/ipp*).

Wesentliches Ziel bei der Entwicklung des Protokolls war die Standardisierung des Zugriffs auf den Print-Server. Dabei geht es um das Starten, Überwachen und Löschen von Printjobs, aber auch um die Möglichkeit, einfach Auskunft über die Funktionen des Druckers zu erhalten. Das Protokoll ist in den RFCs [rfc2910] und [rfc2911] definiert.

Eine interessante Entscheidung ist, dass IPP nicht direkt auf TCP/IP aufsetzt, sondern zum Transport HTTP verwendet. Damit lassen sich viele Ideen aus bestehenden Programmen übernehmen und die meisten Netzwerkdrucker haben heute bereits einen HTTP-Server integriert. Mit SSL (Secure Socket Layer) ist auch die Möglichkeit gegeben, Druckdaten zu verschlüsseln. [rfc2568] beschreibt einige Hintergründe dieser Entscheidung.

Das Internet Printing Protocol verwendet als Standard den Port 631. Nach der Installation und dem Start des `cupsd` Servers können Sie mit einem Web-Browser auf die Administrationsoberfläche unter *http://localhost:631/* zugreifen. In der Datei `/etc/cups/cups.conf` können Sie den Zugriff auch für andere Rechner im lokalen Netz freigeben. Innerhalb der Administrationsoberfläche haben Sie unter anderem Zugriff auf die gesamte Dokumentation.

Neben der grafischen Oberfläche steht für die Konsole das Programm `lpadmin` zur Verfügung, die Online-Dokumentation enthält Beispiele für das Anlegen eines Druckers sowohl mit der Web-Oberfläche als auch mit der Shell. `cups` kann sowohl das von BSD bekannte `lpr`-Frontend als auch das `lp`-Kommando von System V emulieren. Damit ist für Unix-Anwender der Umstellungsaufwand sehr gering.

Seine Stärken spielt das Common UNIX Printing System (CUPS) im Netzwerk aus. Sie können mit Hilfe des IPP auf entfernte Rechner zugreifen, ohne dass am Client konfiguriert werden muss. Aktuelle Windows-Versionen können ebenfalls direkt auf IPP-Server drucken – ohne Konfiguration oder Treiberinstallation.

Druckerfilter

Unter Unix hat sich die Verwendung von PostScript als Standardformat für Druckaufträge eingebürgert. Der Vorteil dieses Formats ist, dass Ausdrucke auf allen Druckern praktisch identisch sind und man sich PostScript-Dateien am Bildschirm ansehen kann.

An viele PC-Systeme ist jedoch nur ein Nadel- oder Tintenstrahldrucker angeschlossen, der nicht PostScript-fähig ist. Für viele gängige Druckertypen enthält das GNU-Programm GhostScript-Treiber, so dass PostScript-Dateien auf diesen Druckern ausgegeben werden können. Dazu kann die Ausgabe des Programms `gs` mit `lpr` auf den Drucker ausgegeben werden. Diese Lösung ist auf Dauer nicht befriedigend, da stets ein Eingreifen des Benutzers notwendig ist. Außerdem werden die Kommandozeilen recht schnell sehr lang und unübersichtlich.

Zur Lösung dieses Problems kann ein so genannter Druckerfilter verwendet werden, der die Druckdaten umsetzen kann. Dazu wird ein Input-Filter verwendet, der mit dem Befehl `file` versucht, den Typ der Datei festzustellen und dann die Ausgabe entsprechend umzuwandeln. Ein Input-Filter wird für jeden Druckjob neu gestartet. Ein Output-Filter, der zunächst genauso geeignet scheint, wird jedoch nur einmal gestartet und weitere Druckjobs werden durch denselben Prozess bearbeitet.

Bei der Erstellung von Druckerfiltern ist unbedingt auf die Systemsicherheit zu achten, da diese Programme unter der Kennung von `root`, `daemon` oder `lp` laufen, abhängig vom verwendeten Printsystem. Insbesondere die Verwendung von GhostScript (`gs`) kann zu Sicherheitsproblemen führen, da PostScript eine vollständige Programmiersprache und der Zugriff auf beliebige Dateien möglich ist. Daher sollte unbedingt der Schalter `-dSAFER` angegeben werden. Damit werden die Funktionen zum Löschen und Umbenennen von Dateien deaktiviert und Dateien können nur im Lesezugriff geöffnet werden.

Ein einfach zu installierender Druckerfilter ist das Paket `magic-filter` von H. Peter Anvin. Die Übersetzung und Installation erfolgt konform zum GNU-Standard. Das Programm `./configure` untersucht das System nach den zur Verfügung stehenden Programmen und passt die Filter entsprechend an. Mit `make` werden die Programme übersetzt, `make install` übernimmt die Installation. Der Befehl `make install_filters` installiert alle Filter im Verzeichnis `/usr/local/bin`. Oft genügen jedoch ein oder zwei Filter für die angeschlossenen Drucker, so dass es sinnvoll ist, genau diese Filter von Hand zu kopieren. Diese Filter sind Konfigurationsdateien für das Programm `magicfilter`, das anhand

des Beginns der Druckdaten entscheidet, wie die Daten umzuwandeln sind, um dann gedruckt zu werden.

Ein weiterer Druckerfilter ist `apsfilter` von Andreas Klemm. Das Grundprinzip dieses Filters ist ähnlich wie das von `magic-filter`, die Realisierung sieht jedoch etwas anders aus. Im Wesentlichen basiert hier die Unterscheidung der verschiedenen Dateiformate auf dem Programm `file`, das seine Daten aus der Datei `/etc/magic` liest. Damit ist `apsfilter` abhängig von der Vollständigkeit der Datei `/etc/magic`.

`apsfilter` erstellt für jeden Drucker eine Reihe von Einträgen in der Datei `/etc/printcap`. Mit diesen Einträgen kann zum einen mit dem automatischen Filter gedruckt werden, es ist aber auch möglich, eine spezielle Bearbeitung vorzugeben, indem der passende Druckereintrag aus der `/etc/printcap` angegeben wird.

Drucken auf andere Netzwerkdrucker

In vielen Netzen sind nicht alle Drucker über TCP/IP ansprechbar, oft können diese nur mit dem NetWare-Protokoll oder dem LAN-Manager-Protokoll (SMB, Windows-Netzwerk) angesprochen werden. Da Linux in beide Netzwerke integriert werden kann, ist es möglich, auch diese Drucker für Unix-Queues zu verwenden.

Ist das IPX-System installiert und konfiguriert (siehe Abschnitt 21.2.1, »Linux als NetWare-Client«), dann kann mit dem Befehl `nprint` auf NetWare-Drucker gedruckt werden. Wenn Sie den Aufruf von `nprint` in einen Input-Filter des `lpr`-Systems aufnehmen, dann können Sie einen Linux-Rechner als Gateway zwischen den verschiedenen Systemen verwenden. In diesem Fall ist der Input-Filter ein einfaches Shell-Skript (Listing 4.20), das eine Ausgabe ausführt, an die die `lpr`-Entwickler nicht gedacht hatten.

```
#!/bin/sh
nprint -U nwuser -S nwserver -n
```

Listing 4.20 Ein einfacher Input-Filter zum Ansprechen von Netware-Queues

Im Listing 4.20, wird ein Netware-Benutzer ohne Passwort verwendet. Alternativ kann eine Benutzerkennung verwendet werden, die mit einem Passwort geschützt ist. In diesem Fall muss im Home-Verzeichnis von `root`, `daemon` oder `lp` (das ist von der Konfiguration des Drucksystems abhängig) die Datei `~/.nwclient`, wie in Abschnitt 21.2.1, »Linux als NetWare-Client« beschrieben, eingerichtet werden.

Auf Drucker eines Windows-Netzwerks können Sie mit `smbclient` zugreifen. Dieses Programm ist Bestandteil des `samba`-Pakets. Mehr Informationen hierzu finden Sie im Abschnitt 21.4, »Linux als SMB-Client«.

Hinweise zu Filtern

Druckerfilter sind sehr einfache und nützliche Hilfsmittel bei der Lösung komplexer Druckprobleme. Da man als Filter alles – vom einfachen Shell-Skript bis zu komplexen Programmen in beliebigen Programmiersprachen – verwenden kann und dabei keinen direkten Zugriff auf die Ausgaben des Skripts hat ist die Fehlersuche nicht immer einfach.

Einige einfache Tipps: Sie haben die Möglichkeit, den Filter zunächst von einem Shell-Prompt aus zu testen, bevor Sie ihn als Filter in das Drucksystem einbinden. Der Filter muss für den Benutzer, unter dem das Spool-System läuft, ausführbar sein, bei einem Shell-Skript muss dieses auch lesbar sein. Um den Ablauf des Filters im Spool-System verfolgen zu können, müssen Sie Ausgaben entweder auf die Standardfehlerausgabe umlenken (die Standardausgabe ist für die Druckdaten reserviert) oder mittels Ausgabeumlenkung direkt in eine Datei schreiben. Alternativ können Sie auch das Programm `logger` zum Schreiben von `syslog`-Einträgen verwenden. Das Listing 4.21 zeigt verschiedene Beispiele für die Verfahren.

```
#!/bin/sh
echo "Start des Filters" >&2
...
echo "Mittendrin" >> /var/log/printer.log
...
logger -p daemon.debug -t lpr-filter "Ende des Filters"
```

Listing 4.21 Fehlersuche in Druckerfiltern

Die Standardfehlerausgabe eines Input-Filters wird auf die Konsole ausgegeben. Mit dem `printcap`-Eintrag `lf=`*`Datei`* kann die Ausgabe in eine Datei geschrieben werden. Diese Datei muss vor dem ersten Aufruf des Filters existieren, ansonsten werden entsprechende Syslog-Einträge in die Facility `lpr` geschrieben. Auf vielen Systemen werden diese Meldungen nicht in Dateien mitprotokolliert, bei der Fehlersuche sollten Sie diese Funktion unbedingt aktivieren.

4.4.4 System-Logs

Jedes Unix-System und die darauf laufenden Programme schreiben zur Kontrolle des Systemverhaltens und zur Fehlersuche eine Reihe von Log-Dateien. Im Prinzip kann jedes Programm selbst diese Dateien öffnen und mit Logs füllen. In vielen Fällen (z. B. beim Web-Server `apache`) geschieht das auch. Nachteil dieses Verfahrens ist, dass jedes Programm einzeln auf die entsprechenden Verzeichnisse zu konfigurieren ist und dafür möglicherweise sogar neu kompiliert werden muss.

Eine allgemein zugängliche und einheitliche Schnittstelle, mit deren Hilfe Programme Log-Dateien anlegen[4] und schreiben können, ist der `syslogd`-Dämon. Dieser Dämon wird mit Einträgen in der Datei `/etc/syslog.conf` konfiguriert. Dabei können Nachrichten getrennt nach Herkunft oder Wichtigkeit in verschiedene Dateien umgeleitet oder angezeigt werden.

In vernetzten Umgebungen ist es möglich, die Log-Dateien von einem speziellen Rechner führen zu lassen, dem so genannten Log-Host. Bei einem Systemcrash eines Rechners hat der Log-Host diese Daten auf seiner lokalen Festplatte gespeichert, sie gehen also normalerweise nicht verloren. Ein weiterer Vorteil ist, dass ein Angreifer zwar in einen Rechner einbrechen kann, die Log-Dateien auf dem Log-Host sind aber zunächst sicher vor ihm. Außerdem ist es praktisch, wenn die Log-Dateien von verschiedenen Rechnern an zentraler Stelle gesammelt sind: Man kann viel einfacher darin suchen und Verknüpfungen finden.

Das `syslog`-Protokoll ist in [rfc3164] dokumentiert. Dies ist kein Internetstandard, sondern dient nur zur Beschreibung des implementierten Protokolls.

Der Standard-syslogd

Die Log-Einträge können abhängig von der Wichtigkeit und der Herkunft in unterschiedlicher Weise behandelt werden. In Tabelle 4.1 werden die verschiedenen Herkunftsarten (Facilities) beschrieben. Die meisten hier vorgestellten Herkunftsarten sind in ihrer Bedeutung festgelegt. Wenn Sie die Herkunftsarten `local0` bis `local7` verwenden, achten Sie auf eine entsprechende Dokumentation, mindestens in der Datei `/etc/syslog.conf`.

Herkunftsart	Beschreibung
`auth`	Für Sicherheits- bzw. Autorisierungsmeldungen
`authpriv`	Private Sicherheits- bzw. Autorisierungsmeldungen
`cron`	Meldungen von `cron` und `at`
`daemon`	Andere Dämonen des Systems
`kern`	Ausgaben des Kernels
`lpr`	Für Logs des `lpr`-Systems
`mail`	Nachrichten des Mail-Systems (`sendmail` u. a.)
`mark`	Aufnehmen von Marken in die Log-Dateien
`news`	Ausgaben des News-Systems
`syslog`	Nachrichten des `syslogd`-Dämons

4. Bei vielen `syslogd`-Versionen müssen die Dateien vor dem Start z. B. mittels `touch` angelegt werden.

Herkunftsart	Beschreibung
`user`	Allgemeine Nachrichten (Standardwert)
`uucp`	Logs des UUCP-Systems
`local0-7`	Lokal verwendbare Herkunftsarten

Tabelle 4.1 Herkunftsarten für `syslogd`*-Nachrichten*

In jeder Herkunftsart kann eine Nachricht unterschiedliche Wichtigkeit (Priorität) haben. Anhand der Priorität kann `syslogd` Nachrichten selektieren und unterschiedlich behandeln. Tabelle 4.2 enthält die verschiedenen Prioritäten, nach absteigender Wichtigkeit geordnet.

Wichtigkeit	Beschreibung
`emerg`	Es wird ein Notfall signalisiert
`alert`	»Gelber Alarm, Schilde hoch!«
`crit`	Eine kritische Situation ist aufgetreten
`err`	Eine Fehlersituation wurde erkannt
`warning`	Eine Warnung
`notice`	Ein Hinweis
`info`	Informative Meldungen
`debug`	Ausgaben zur Fehlersuche
`none`	Ignorieren dieser Herkunftsart

Tabelle 4.2 Wichtigkeiten für `syslogd`*-Nachrichten*

Zur Erläuterung finden Sie in Listing 4.22 ein Beispiel für die Konfiguration des `syslogd`-Dämons. Beachten Sie, dass bei vielen `syslog`-Versionen die Herkunft der Nachrichten und das Ziel durch Tabulatoren voneinander getrennt werden müssen. Leerzeichen vor oder nach dem Ziel sind bei vielen `syslogd`-Implementationen nicht erlaubt.

```
# Facility.Level            Destination
kern.*                      /var/log/kernel
kern.emerg                  *
*.err                       /var/log/errorlog
*.emerg                     /dev/console
*.=debug                    /var/log/debug
*.=crit                     root,jochen
*.crit                      @loghost
```

Listing 4.22 Auszug aus der Datei `/etc/syslog.conf`

Alle Meldungen, die der Unix-Kernel (`kern.*`) ausgibt, werden in die Datei `/var/log/kernel` geschrieben. Der `syslogd`-Dämon erkennt an dem Schrägstrich (`/`) als erstem Zeichen, dass dieses Ziel eine Datei oder ein Gerät (Device)

ist. Steht vor dem Dateinamen ein Minus-Zeichen (-), dann werden die Nachrichten in die Log-Datei geschrieben, es erfolgt aber keine Synchronisation der Puffer. Normalerweise versucht `syslogd`, die Daten möglichst direkt auf der Festplatte abzulegen. Das kann für Systeme mit sehr aktiven Log-Dateien eine erhebliche Systembelastung verursachen. Für weniger wichtige Log-Dateien kann der Systemverwalter hier möglicherweise die Systembelastung deutlich verringern.

Meldungen aus dem Kernel, die einen Notfall signalisieren, werden an alle angemeldeten Benutzer (`*`) geschickt. Alle Meldungen aller Systeme, die `err` oder wichtiger sind, werden in der Datei `/var/log/errorlog` protokolliert. Meldungen, die einen Notfall signalisieren, werden zusätzlich auf der Konsole protokolliert.

Will man vermeiden, dass auch wichtigere Nachrichten in ein Protokoll aufgenommen werden, so gibt man die Wichtigkeit mit einem Gleichheitszeichen (`=`) ein. Alle Nachrichten zur Fehlersuche (und keine anderen) werden in der Datei `/var/log/debug` protokolliert. Mit einem Ausrufezeichen (`!`) kann die Bedingung negiert werden. Mit dem Eintrag `kern.!=info` werden beispielsweise alle Nachrichten selektiert, die nicht die Priorität `info` haben. Die Verwendung der Zeichen `=` und `!` stellt eine Erweiterung des unter Linux verwendeten `sysklogd` dar.

Die letzten beiden Zeilen in Listing 4.22, demonstrieren die Benachrichtigung der Benutzer `root` und `jochen` sowie die Übertragung von Nachrichten an einen anderen Rechner (`loghost`) zur Protokollierung. Als Übertragungsprotokoll wird UDP und der Port 514 verwendet.

Wenn Sie Nachrichten auf einem Log-Host speichern wollen, dann starten Sie dort den `syslogd`-Prozess mit der Kommandozeilenoption `-r`. Andernfalls werden keine Nachrichten von fremden Rechnern angenommen, um zu verhindern, dass jeder Benutzer im Netz die Festplatte mit Hilfe des `logger`-Programms füllen kann. Weitere nützliche Optionen sind `-l`, gefolgt von einer Liste von Rechnern, deren Domainnamen nicht gespeichert werden sollen, und `-s`, gefolgt von einer Liste von Domainnamen, die ebenfalls nicht im Log auftauchen sollen. Mit Hilfe von Firewall-Regeln können unter Linux einfach die Rechner bestimmt werden, die einen entfernten `syslog` verwenden dürfen.

Auf einem privaten System kann es sinnvoll sein, alle `syslog`-Nachrichten auf eine unbenutzte virtuelle Konsole, z. B. `/dev/tty12`, auszugeben. Dann kann man mit einem Tastendruck die letzten Log-Einträge sehen und eventuell notwendige Maßnahmen ergreifen. Den dazu notwendigen Eintrag in die Datei `/etc/syslog.conf` finden Sie im Listing 4.23. Beachten Sie, dass nicht jedes Unix einen `*.*`-Eintrag unterstützt. Für spezielle Bearbeitungen der Log-Dateien kann zusätzlich eine Named-Pipe (FIFO) mit `mkfifo` eingerichtet werden. `sysklogd` unterstützt FIFOs durch das Pipe-Zeichen (`|`) vor dem Dateinamen.

```
*.*                                     /dev/tty12
```

Listing 4.23 Alle Meldungen auf die virtuelle Konsole `/dev/tty12`

Die Ausgabedateien müssen beim Start des `syslogd`-Dämons existieren. Wenn eine Datei noch nicht angelegt ist, so gibt der Dämon eine Fehlermeldung aus. Die entsprechende Datei kann dann mit dem Befehl `touch` *`Dateiname`* angelegt werden.

Nach einer Veränderung der Konfigurationsdatei `/etc/syslog.conf` muss der Dämon neu initialisiert werden. Dazu sendet man das Signal `HUP` an den laufenden `syslogd`, z. B. mit dem Befehl `kill -HUP $(cat /var/run/syslog.pid)`. Zur Überprüfung, ob die gewünschte Nachricht auch korrekt protokolliert wird, kann der Befehl `logger -p`*`Herkunft.Wichtigkeit "Text"`* verwendet werden. Damit ist es allerdings auch möglich, verwirrende Meldungen abzusetzen und die Log-Dateien stark zu vergrößern.

Das Verhalten von `syslogd` kann beim Start mit einigen Optionen bestimmt werden. Zur Laufzeit ist die Steuerung von wichtigen Funktionen mit Hilfe von Signalen möglich.

`-d`
: Versetzt den Dämon in den Fehlersuchmodus. Dabei läuft `syslogd` nicht im Hintergrund und gibt viele Nachrichten zur Fehlersuche aus. Wurde `syslogd` mit dieser Option gestartet, dann kann mit dem Signal `SIGUSR1` die Ausgabe der Debug-Meldungen ein- bzw. ausgeschaltet werden.

`-h`
: Diese Option sorgt dafür, dass über das Netzwerk empfangene Nachrichten auch an andere Rechner weitergeleitet werden. In manchen Fällen mag das sinnvoll sein, Sie können damit aber auch sehr schnell eine Endlosschleife generieren. Benutzen Sie diese Option mit Vorsicht.

`-m` *`Intervall`*
: In Log-Dateien, in denen nur sehr selten etwas passiert, kann mit der Option `-m` alle *`Intervall`* Minuten der Eintrag `-- MARK --` eingefügt werden.

`-n`
: Normalerweise startet `syslogd` im Hintergrund. Auf Rechnern, auf denen es unbedingt erforderlich ist, dass zu jedem Zeitpunkt ein `syslogd`-Programm läuft, kann man dieses durch `init` starten lassen und dafür sorgen, dass bei einem Abbruch das Programm neu gestartet wird. In diesem Fall ist die Option `-n` notwendig, damit `init` den Prozess überwachen kann.

`-r`
: Nur mit dieser Option nimmt `syslogd` Nachrichten von anderen Systemen an. Die einfachste Möglichkeit, die erlaubten Systeme zu selektieren, ist die Verwendung des Linux-internen Firewalling. Andernfalls kann jeder Benut-

zer im Netz bei Kenntnis der `syslog`-Konfiguration die Log-Verzeichnisse bis zum Rand füllen.

`-l` *`Host`*`[:`*`Host`*`...]`
Wenn Nachrichten von fremden Rechnern angenommen werden, dann wird normalerweise der vollständige Name (FQDN, inklusive Domain) mitgeloggt. Dadurch werden die Zeilen in den Log-Dateien lang und unübersichtlich. Mit der Option `-l` bestimmt der Systemverwalter die Rechnernamen, die nur in Kurzform (ohne Domain) gespeichert werden sollen.

`-s` *`Domain`*`[:`*`Domain`*`...]`
Analog zur Option `-l` werden die Host-Namen aus den angegebenen Domains nur als kurze Namen gespeichert.

Insgesamt ist `syslogd` ein sehr nützliches Programm, das aber auch seine Schwächen hat. Zunächst kann die durch `syslogd` erzeugte Systemlast sehr hoch werden und jeder Benutzer kann beliebige Meldungen in die Log-Dateien einfügen. Außerdem werden die Log-Dateien immer größer, bis sie letzten Endes sämtlichen zur Verfügung stehenden Plattenplatz belegen.

Für die Zukunft denkbar und nützlich könnten verschiedene Erweiterungen sein. Nachrichten könnten zusätzlich zur Protokollierung in Textdateien auch in einer Datenbank gespeichert werden. Innerhalb der Datenbank ließen sich viele Auswertungen einfacher durchführen als mit den verschiedenen Dateien. In die `syslogd`-Konfiguration könnte das Umwälzen der Log-Dateien integriert werden. Warum sollte der Systemverwalter diese Informationen mehrfach pflegen? Bei der Debian-Distribution läuft täglich ein Cronjob, der die Dateien aus der `/etc/syslog.conf` rotiert.

Zu guter Letzt könnte man beliebige Herkunftsarten zulassen, die dann als Textinformation in der Datenbank gespeichert werden. Mit diesen Änderungen könnte `syslogd` noch leistungsfähiger, nützlicher und einfacher werden. Derzeit ist mir aber kein Projekt bekannt, das an diesen Problemen arbeitet.

Andere Systemlog-Programme

Der traditionelle `syslogd` ist vielen Systemverwaltern bekannt und manchmal verhasst. Zum einen ist der Parser der Konfigurationsdatei relativ eigen (manche `syslogd`-Varianten erwarten, dass die Spalten durch Tabulatoren getrennt sind, und verweigern bei Leerzeichen die Arbeit), zum anderen kann `syslogd` bei weitem nicht das, was man heute erwartet. Es gibt allerdings einige neuere Entwicklungen auf diesem Gebiet.

Der »new generation« Systemlog syslog-ng

Ein wichtiges Ziel bei der Entwicklung von `syslog-ng` (*http://www.balabit.hu/products/syslog-ng/*) war, eine deutliche Verbesserung der Filtermöglichkeiten. Es loggen beispielsweise sehr viele Programme in die `daemon`-Facility, manchmal

möchte man die Nachrichten aber doch voneinander trennen. Die Konfiguration von `syslog-ng` erfolgt in der Datei `/etc/syslog-ng/syslog-ng.conf` und ist in fünf Teile unterteilt:

- Allgemeine Einstellungen werden im `options`-Teil vorgenommen. Normalerweise werden Sie hier nur etwas wie `sync(0);` eintragen, damit alle Nachrichten direkt auf der Platte landen. Weitere Optionen erklärt die HTML-Dokumentation.
- Quellen von Nachrichten können Dateien, UDP- oder TCP-Verbindugen, FIFOs oder Unix-Sockets sein. Eine Quelle bekommt einen Namen und wird mit dem Schlüsselwort `source` beschrieben. Das Listing 4.24 zeigt ein Beispiel für Nachrichtenquellen.

```
# Lesen vom Unix-Socket /dev/log und interne Nachrichten
source src { unix-dgram("/dev/log"); internal(); };
# Wenn der Rechner als Loghost fungieren soll:
# auch Nachrichten via UDP (Standardport 514) empfangen
# source src { unix-dgram("/dev/log"); internal(); udp(); };
```

Listing 4.24 Nachrichtenquellen für `syslog-ng`

- Das Ziel der Nachrichten wird mit dem Schlüsselwort `destination` festgelegt. Es können dieselben Arten von Kommunikationsmethoden verwendet werden wie bei den Nachrichtenquellen, zusätzlich können Nachrichten an angemeldete Benutzer und Programme gesendet werden. Im Listing 4.25 finden Sie ein Beispiel für verschiedene Ziele.

```
# Die Datei /var/log/mail.log
destination mail { file("/var/log/mail.log" owner("root")
group("adm")
                   perm(0640)); };
destination messages { file("/var/log/messages" owner("root")
group("adm")
                   perm(0640)); };
# Virtual console.
destination console_all { file("/dev/tty8"); };
# Für das Programm xconsole
destination xconsole { pipe("/dev/xconsole"); };
# Und den externen Loghost als Ziel
destination loghost { udp("loghost"); };
# An den angemeldeten Benutzer
destination root { usertty("root"); };
```

Listing 4.25 Ziele beim Loggen von Nachrichten

Besonders auffällig ist, dass Sie bei den Dateien zusätzliche Parameter wie Dateieigentümer und Permissions angeben können. Beim normalen `syslogd` bleiben diese so, wie der Systemverwalter die Dateien angelegt hat. Auch die

anderen `destination`-Einträge können vielfach noch genauer eingerichtet werden, die Details finden Sie in der HTML-Dokumentation.

- Nachrichten können mit Hilfe von Filtern selektiert werden. Ein Filter wird mit dem Schlüsselwort `filter` deklariert, das Listing 4.26 zeigt wieder einige Möglichkeiten.

```
filter f_authpriv { facility(auth, authpriv); };
filter f_mail { facility(mail); };
filter f_messages { level(info .. warn)
        and not facility(auth, authpriv, cron, daemon, mail,
news); };
filter f_emergency { level(emerg); };
```

Listing 4.26 Filter für den `syslog-ng`

Sie können nicht nur nach Herkunft und Wichtigkeit, sondern auch nach Programmname (`program(`*`regex`*`)`), Host-Name (`host(`*`regex`*) oder regulären Ausdrücken (`match(`*`regex`*) in der Nachricht filtern. Für komplexere Filter können Sie mit dem Schlüsselwort `filter(`*`filtername`*`)` auf einen weitern Filter verweisen.

- Und im letzten Schritt setzen wir die Einzelteile zusammen. Mit dem `log`-Schlüsselwort werden nun die Nachrichtenquelle, die Filter und das Nachrichtenziel zusammengesetzt. In einem `log`-Eintrag können mehrere Quellen, Filter und Ziele angegeben werden. `syslog-ng` wird die Nachrichten aus den Quellen sammeln, mit Hilfe der Filter selektieren und gegebenenfalls in die Ziele schreiben. Das Listing 4.27 zeigt wieder einige Möglichkeiten.

```
# Allgemeine Nachrichten nach /var/log/messages
log { source(src); filter(f_messages); destination(messages);};
# Logge alles auf die Konsole
log { source(src); destination(console_all); };
# Emergency an den Benutzer root
log { source(src); filter(f_emergency); destination(root); };
```

Listing 4.27 Das eigentliche Logging

Wenn Sie bisher keine Probleme mit dem `syslogd` hatten, dann benötigen Sie `syslog-ng` vermutlich nicht. Wenn Sie aber schon immer etwas flexibler sein wollten, dann könnte dieses Programm etwas für Sie sein.

Der modulare Systemlog-Daemon msyslog

Das Programm `msyslog` (*http://www.corest.com/*) ist sehr modular aufgebaut. Es gibt Module für die verschiedenen Nachrichtenquellen (etwa dieselben wie bei `syslog-ng`) und Ziele. Bei den Zielen sind neben den »üblichen Verdächtigen« auch die Datenbanken MySQL und PostgreSQL möglich. Damit hat man die Möglichkeit, `syslog`-Meldungen in Datenbanken zu speichern und sehr einfach auszuwerten.

Die Syntax der Konfigurationsdatei `/etc/msyslog.conf` ist derjenigen des Standard-`syslogd` sehr ähnlich, damit ist der Aufwand für eine Umstellung zunächst relativ gering. Die Module für Nachrichtenquellen werden mit Hilfe des Kommandozeilenparameters `-i` *`Modul`* angegeben. In der Standardversion werden Sie die Module `unix` und `linux` laden, damit werden Nachrichten aus einem Unix-Domain-Socket und dem Linux-Kernel gelesen. Ist der Rechner ein Loghost für andere Systeme, dann müssen Sie das Modul `udp` laden. Informationen zu diesem Modul und Beispiele finden Sie in der Manpage `im_udp(8)`.

Die Module für die Ausgabe werden in der Datei `/etc/msyslog.conf` eingerichtet. Das Modul `classic` (in der Konfiguration wird vor dem Modulnamen ein Prozent-Zeichen angefügt) kann in Dateien und via Netzwerk loggen und außerdem Nachrichten an angemeldete Benutzer schicken – die Syntax entspricht dem, was man vom Standard-`syslogd` her kennt. Wird in der Datei `/etc/msyslog.conf` kein Modulname angegeben, so wird dieses Modul verwendet. Dokumentation zu diesem Modul finden Sie in der Manpage zu `om_classic(8)`.

Mit dem Modul `regex` können Sie filtern und reguläre Ausdrücke auf Host-Namen (`-h`), die Nachricht (`-m`), das Datum (`-d`) oder die Uhrzeit (`-t`) anwenden. Mit der Option `-v` können Sie wie bei `grep` die Nachrichten auswählen, die nicht zu dem regulären Ausdruck passen.

Als Ausgabe werden außerdem die Datenbanken MySQL und PostgreSQL unterstützt. Wie die Datenbanktabellen definiert werden müssen, das erfahren Sie in der Manpage `om_mysql(8)` bzw. `om_pgsql`. Wenn Sie diese Ausgabemodule verwenden, dann sollten Sie die Berechtigungen der Datei `/etc/msyslog.conf` so einrichten, dass sie nicht global lesbar sind. Im Listing 4.28 finden Sie ein Beispiel für die verschiedenen Konfigurationsoptionen.

```
# Loggen in die Datei /var/log/auth.log
auth,authpriv.*                 %classic /var/log/auth.log
# Meldungen auf ein unbenutztes TTY ausgeben
#*.*;auth,authpriv.none         %classic /dev/tty8
# Nachrichten für xconsole bereitstellen
*.warn                          %classic |/dev/xconsole
# An den Loghost senden
# *.*                           %classic @loghost
# Benutzer 'root' in der Zeit vom 20:00:00 bis 09:59:59
auth.info %regex -v -t '^1' %regex -m 'root' %classic
/var/log/webserver
# PostgreSQL als Datenspeicher
```

```
*.*     %pgsql -s logger.jochen.org \
               -u loguser -p loguserpassword \
               -d syslogDB -t syslogTB
```

Listing 4.28 Die Konfiguration für den modularen Syslog

Lohnt sich ein anderer syslogd?
Wie so häufig ist das Geschmackssache. Wenn Sie das Systemlog in einer Datenbank speichern wollen, dann werden Sie `msyslog` verwenden. Leider muss der Systemverwalter sich erneut selbst um das Löschen alter Einträge kümmern. Ein Vorteil ist, dass die Syntax der Konfigurationsdatei derjenigen des Standard-`syslogd` sehr ähnlich ist. Damit ist ein Wechsel sehr einfach.

`syslog-ng` hat eine vollständig neue Syntax, die mir etwas übersichtlicher scheint, wenn man die Filter, Datenquellen und Ziele mit sprechenden Namen versieht. In diesem Fall kann man mit dem Lesen der `log`-Einträge die Konfiguration nachvollziehen.

Beide `syslog`-Varianten bieten Möglichkeiten zur Filterung der Einträge, so dass auch auf einem größeren Loghost die Logdateien wieder geordnet gespeichert werden können. Für einen einzelnen Rechner sind vermutlich beide überdimensioniert.

Automatisches Kürzen von Log-Dateien

Protokolldateien haben die unangenehme Eigenschaft, dass sie immer länger werden und immer mehr Plattenplatz benötigen. Daher sollten Log-Dateien regelmäßig geleert werden. Sinnvoll ist es hier, entweder die Logs dem Systemadministrator per Mail zuzuschicken oder eine Sicherungskopie anzulegen, so dass die alten Meldungen nicht unmittelbar gelöscht werden. Zum Teil (z. B. für das News-System oder UUCP) erfolgt das bereits durch Skripten, die zu dem entsprechenden System gehören. Für die anderen Log-Dateien kann ein eigenes Skript verwendet oder auf das Paket `prune` zurückgegriffen werden. Leistungsfähig und verbreitet ist das Programm `logrotate`.

Das Programm `logrotate` kann beliebige Dateien bearbeiten, alte Log-Dateien löschen oder komprimieren, eine beliebige Anzahl von alten Versionen aufheben und beliebige Befehle vor oder nach dem Rotieren ausführen. Das Kürzen von Log-Dateien kann täglich, wöchentlich oder monatlich angestoßen oder von der Größe der Log-Datei abhängig gemacht werden. Bis auf die nicht vorhandene Integration mit `syslogd` erfüllt `logrotate` praktisch jede Anforderung, die ein Systemverwalter an ein derartiges Programm stellt.

Bei einem Mehrbenutzer-System ist es notwendig und sinnvoll, dass nicht jeder Benutzer alle Log-Dateien lesen kann. So sollten z. B. Fehlermeldungen bei der Anmeldung, wo in den Meldungen Benutzernamen oder Passwörter auftauchen könnten, nur für den Systemadministrator lesbar sein. Bei der Neuanlage einer

Protokolldatei mit `touch` werden die Berechtigungen gemäß der eingestellten `umask` gesetzt. Dies können die falschen Permissions sein, daher sollten Log-Dateien nicht einfach gelöscht und mit `touch` neu angelegt werden. Besser ist es, die Dateien mit Hilfe der Ausgabe-Umleitung zurückzusetzen, beispielsweise mit dem Befehl `echo -n > Logfile`.

`logrotate` erstellt mit der Option `create` in der Konfiguration für eine Log-Datei eine leere Datei mit den angegebenen Berechtigungen (Eigentümer, Gruppe und Zugriffsmode). Das Listing 4.29 zeigt ein Beispiel für eine einfache Konfiguration. Genauere Informationen finden Sie in der Manpage zu `logrotate(8)`.

```
# Log-Dateien werden wöchentlich bearbeitet
weekly
# Vier Versionen (alte Log-Dateien) aufheben
rotate 4
# Fehlermeldungen gehen per E-Mail an 'root'
errors root
# Neue Log-Dateien werden angelegt
create
# Weitere Konfigurationen in diesem Verzeichnis suchen
include /etc/logrotate.d
```

Listing 4.29 Eine einfache Konfiguration für `logrotate`

Die in Listing 4.29 dargestellte Konfiguration gilt für alle Dateien, die mittels `logrotate` bearbeitet werden sollen. Für einzelne Dateien können diese Einstellungen jedoch gezielt angepasst werden. Das Verzeichnis `/etc/logrotate.d` dient als Sammelstelle für die Konfiguration von `logrotate`, so dass jedes Paket, das ein Log-Dateien schreibendes Programm enthält, hier eine Basiskonfiguration ablegen kann. Beispiele für derartige Pakete sind UUCP, `apache` oder `samba`.

```
/var/log/messages {
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
/var/log/secure {
    monthly
    rotate 6
    create 640 root admin
    postrotate
        /usr/bin/killall -HUP syslogd
    endscript
}
```

Listing 4.30 Ein Beispiel für einzelne Log-Dateien und `logrotate`

Das Listing 4.30 zeigt die Definition einer Log-Datei in `logrotate`. Ein Eintrag beginnt mit dem Namen der zu bearbeitenden Datei, in geschweiften Klammern werden die spezifischen Einstellungen angegeben. Das erste Beispiel in Listing 4.30 enthält nur ein Skript, das nach der Umwälzung der Datei ausgeführt wird. Hier wird nur der `syslogd` angestoßen, so dass er die neu angelegte Datei (die Option `create` in Listing 4.29) verwendet.

Als zweite Datei wird `/var/log/secure` von `logrotate` bearbeitet. Da diese Datei sicherheitsrelevante Daten enthält, wird sie mit speziellen Berechtigungen angelegt. Der Benutzer `root` hat Schreibzugriff, die Gruppe `admin` darf die entsprechende Datei lesen, aber kein anderer Benutzer hat Zugriff auf diese Log-Datei. Außerdem wird diese Datei nur monatlich bearbeitet und sechs Monate lang aufgehoben. Damit können Sie sicherheitsrelevante Log-Dateien länger aufheben und bei Problemen darauf Bezug nehmen.

Wer liest eigentlich die ganzen Logs?

Bei aktiven Rechnern oder gar Rechnerfarmen kommt im Laufe eines Tages eine beachtliche Menge an Logs zusammen. Normalerweise sind es so viele, dass kein Systemverwalter diese noch durchsehen kann – außerdem ist das ziemlich langweilig. Aus diesem Grund wird vermutlich bei AIX in der Standardauslieferung kein einziges Systemlog geschrieben.

Man möchte aber auf die Protokollierung nicht verzichten, um im Falle eines Fehlers oder gar eines Angriffs auf das System dies noch nachvollziehen zu können. Nur, woran erkennt man eine solche Situation? Indem man die Logs liest. Oder lesen lässt.

Wenn Sie bei *http://freshmeat.net/* suchen, dann werden Sie verschiedene Programme finden, wie zum Beispiel `log-analysis`, `logtool`, `logwatch`, `logcheck`. Die Funktionsweise der Programme ist recht ähnlich: Sie geben die zu durchsuchenden Log-Dateien an und reguläre Ausdrücke, die ignoriert werden oder zu einem Alarm führen sollen. Die Auswertung erfolgt dann zum Beispiel in einem Cronjob und das Ergebnis wird per Mail versendet. Sie werden in der ersten Zeit recht viele falsche Treffer haben, so dass Sie die Regeln anpassen müssen.

Eine andere Möglichkeit ist die Überwachung der Systeme mit Tools wie `mon`, »Big Brother« (*http://www.bb4.com/*) oder der freien Reimplementierung »Big Sister« (*http://bigsister.graeff.com/*). Diese Programme überwachen Rechner und Dienste permanent und sind in der Lage, auch zwischendurch eine Mail zu schicken.

4.4.5 Shared Libraries

Die unter Linux verwendeten *Shared Libraries* werden nur einmal in den virtuellen Speicher geladen, selbst wenn mehrere Programme diese Bibliotheken benutzen. Daher kann der zur Verfügung stehende Speicher für andere Dinge, wie weitere Prozesse oder Platten-Cache, verwendet werden. Aus diesem Grund lässt sich auch ein Linux-System mit nur vergleichsweise wenig Hauptspeicher immer noch gut betreiben. Shared Libraries sind aufwärtskompatibel, solange sich die erste Ziffer der Versionsnummer (die Hauptversion oder *major version*) nicht verändert. Daher ist es relativ einfach möglich, eine neue, verbesserte Bibliothek zu benutzen.

Die Verwendung von Shared Libraries hat jedoch auch Nachteile. So müssen die benötigten Bibliotheken stets auf dem System verfügbar sein. Sollte der Systemadministrator diese Bibliotheken gelöscht haben oder sollten sie bei einem Systemabsturz beschädigt worden sein, so kann es notwendig werden, das System mit einer Notfalldiskette zu starten und entsprechende Reparaturen vorzunehmen. Zur Vermeidung solcher Probleme kann eine Reihe von Programmen statisch gelinkt werden, d. h., es werden keine Shared Libraries benötigt. Da diese Programme erheblich mehr Plattenplatz belegen, beschränkt man sich auf eine Reihe von wichtigen Programmen. Das können z. B. eine kleine, aber dennoch leistungsfähige Shell wie `sash` und Utilities wie `ln` sein. Ein statisch gelinktes `ln` findet man oft auch unter dem Namen `sln`.

Das Einbinden von Shared Libraries zur Laufzeit in Programme wird durch den dynamischen Linker /lib/ld-linux.so durchgeführt, das ist ebenfalls eine Shared Library. Bei älteren Versionen der `libc` heißt der dynamische Linker `/lib/ld.so`, unter `ld.so` versteht man auch eine aktuelle Version. Mit aktuellen `libc`-Versionen ist diese Library auch ein ausführbares Programm und kann beispielsweise zum Starten von Programmen von einem Medium dienen, das mit der Option `noexec` eingehängt wurde.

Die von einem Programm benötigten Bibliotheken werden zur Laufzeit gesucht und in den Adressraum des Prozesses geladen. Sie können mit dem Befehl `ldd` *`Programm`* angezeigt werden. Eine Anwendung für diesen Befehl finden Sie in Listing 4.31.

```
(linux):~$ ldd /usr/X11R6/bin/xterm
libXaw.so.6 (DLL Jump 6.0) => /usr/X11R6/lib/libXaw.so.6.0
libXt.so.6 (DLL Jump 6.0) => /usr/X11R6/lib/libXt.so.6.0
libX11.so.6 (DLL Jump 6.0) => /usr/X11R6/lib/libX11.so.6.0
libc.so.4 (DLL Jump 4.5pl26) => /lib/libc.so.4.6.20
```

Listing 4.31 Ausgabe des Befehls `ldd`

In der ersten Spalte wird die Bibliotheksversion ausgegeben, die mit dem Programm gelinkt wurde. In Klammern wird die exakte Versionsnummer der Bi-

bliothek aufgeführt, auf der rechten Seite stehen die nun zur Laufzeit gefundenen Bibliotheken. Im Listing 4.31 wurde das Programm `xterm` mit der Bibliothek gelinkt, zur Laufzeit steht die (neuere) Version zur Verfügung.

Der dynamische Linker durchsucht eine Reihe von Verzeichnissen, um alle benötigten Bibliotheken zu finden. Dabei wird im Einzelnen in folgenden Verzeichnissen gesucht:

- Zunächst wird in den Verzeichnissen gesucht, die in der Umgebungsvariablen `LD_LIBRARY_PATH` aufgeführt sind. Diese Variable wird bei Programmen, die `setuid` oder `getuid` sind, ignoriert. Andernfalls könnte jeder Benutzer beliebige Funktionen unter der Kennung des Dateieigentümers ausführen, indem eine modifizierte Bibliothek verwendet wird.
- In der Datei `ld.so.cache` wird eine Liste von Bibliotheken und Verzeichnissen gespeichert. Dies erfolgt mit dem Programm `ldconfig`.
- Die Datei `/etc/ld.so.conf` enthält eine Liste von Verzeichnissen, die von `ld.so` `ldconfig` nach den passenden Bibliotheken durchsucht werden.
- Zuletzt wird in den Systemverzeichnissen `/usr/lib` und `/lib` nach Bibliotheken gesucht.

Update von Shared Libraries

Ein weiterer Vorteil von Shared Libraries ist, dass eine neue, verbesserte Version einer Bibliothek ohne größere Probleme installiert werden kann und Programme die neue, in der Regel leistungsfähigere und stabilere Version benutzen. Da jedoch die Shared Libraries zum Betrieb des Systems unbedingt notwendig sind, muss der Systemadministrator bei der Installation neuer Bibliotheken mit der gebotenen Vorsicht zu Werke gehen.

Der Zugriff auf die `libc`-Bibliothek erfolgt mit dem Namen (siehe auch Listing 4.31). Dies ist jedoch nicht die Bibliothek selbst, sondern ein symbolischer Link, der auf die aktuelle Bibliothek zeigt. Eine neue Bibliothek wird aktiviert, indem der symbolische Link auf die neue Version umgestellt wird. Der Link darf nicht gelöscht und dann neu angelegt werden, da das Programm `ln` nicht statisch gelinkt ist und die Shared Library ohne diesen Link nicht geladen werden kann. Die Verwendung von `ln` ist auch nicht notwendig; ein Update der Bibliotheken erfolgt sinnvollerweise mit dem (statisch gelinkten) Programm `ldconfig`.

Beim Update der Bibliothek sind weitere Punkte zu beachten. Sie sollten die entsprechenden Release-Notes lesen und die Anweisungen darin verstehen und ausführen. Dort werden dann auch die entsprechenden Headerdateien des C-Compilers und weitere notwendige Bibliotheken installiert, wie z. B. neue Versionen des dynamischen Linkers `ld.so`.

4.4.6 Anmeldung eines Benutzers

Ein Benutzer meldet sich bei einem Linux-System mit seinem Namen und seinem Passwort an. Dies kann z. B. an der Konsole, an einem seriellen Terminal (oder einem seriell angeschlossenen PC mit einer Terminal-Emulation wie Telix), im Netzwerk mit `telnet` oder `rlogin` oder unter X mit `xdm` erfolgen. An der Konsole stehen mehrere virtuelle Bildschirme zur Verfügung, zwischen denen mit Alt+Fn umgeschaltet werden kann. Auf einem Terminal lässt sich eine ähnliche Funktionalität mit dem Programm `screen` erreichen. Dieses Programm finden Sie auf jedem GNU-Mirror.

Im einfachsten Fall werden vom Programm `init`, gesteuert durch die Datei `/etc/inittab`, auf einigen virtuellen Konsolen die `getty`-Programme gestartet. Dieses Programm initialisiert das entsprechende Gerät (`tty`) und gibt wahlweise die Datei `/etc/issue` oder eine andere Meldung aus. Anschließend wird der `login:`-Prompt ausgegeben und der Benutzername gelesen. Erst danach wird die Kontrolle an das Programm `login` übergeben, das die Identität des Benutzers prüft, indem es das eingegebene Passwort mit dem in der Datei `/etc/passwd` eingetragenen vergleicht[5].

So einfach, wie es zunächst beschrieben wurde, ist es in der Praxis nicht immer, da es unter Linux verschiedene `getty`-Programme gibt, die für unterschiedliche Situationen besonders geeignet sind. Im Folgenden werden die verbreiteten `getty`-Versionen vorgestellt und deren Vor- und Nachteile für verschiedene Einsatzbereiche gegenübergestellt.

Bei einer Änderung an `getty`-Einstellungen kann es leicht vorkommen, dass keine Anmeldung mehr möglich ist, daher sollte stets eine Notfalldiskette verfügbar sein. Eine weitere Möglichkeit ist, zumindest eine virtuelle Konsole mit einer alten Version oder einem komplett anderen `getty` zu betreiben. Als einfache Alternative kann hier das Programm `agetty` bzw. das interne `getty` des System-V-`init` verwendet werden.

Die Verwendung von sehr vielen `getty`-Programmen ist nicht zu empfehlen, da zum einen viele Prozesse Hauptspeicher benötigen und zum anderen bei jeder Anmeldung erneut das Passwort eingegeben werden muss. Mit dem Programm `open` lässt sich jedes beliebige Programm auf einer weiteren virtuellen Konsole starten, ohne dass dort eine Anmeldung erfolgen muss oder ein `getty` gestartet wurde.

5. Das gilt für die Standardinstallation. Passwörter können auch mit Hilfe eines NIS-, LDAP- oder Samba-Servers verifiziert werden.

Das Programm agetty

Das Programm `agetty` wurde von W. Z. Venema entwickelt und von Peter Ørbæk nach Linux portiert. Es wird mit der Kollektion `util-linux` von Rik Faith verteilt und ist ein relativ einfaches `getty`. Daher kann einerseits bei der Installation wenig falsch gemacht werden, andererseits ist es in der Anwendung relativ unflexibel. Der Einsatz von `agetty` auf den virtuellen Konsolen ist in der Kombination mit dem `simpleinit` problemlos, beim Anschluss von seriellen Terminals oder Einwählen per Modem eignet sich jedoch oft ein anderes `getty` besser.

Das Programm `agetty` sollte nur im Zusammenspiel mit dem `simpleinit` verwendet werden, da hier bereits `init` das korrekte Einstellen der Umgebungsvariablen `TERM` durchführt. Als Parameter werden die Baudrate des angeschlossenen Geräts und das zu verwendende `tty` erwartet. Die Reihenfolge der Parameter sollte bei diesem `getty` keine Rolle spielen. Das Listing 4.32 zeigt die Verwendung von `agetty` in Zusammenarbeit mit `simpleinit` für eine virtuelle Konsole und ein direkt angeschlossenes `vt100`-Terminal.

```
tty1:linux:/sbin/getty 9600 tty1
ttyS1:vt100:/sbin/getty -L 19200 ttyS1
```

Listing 4.32 Aufruf des `agetty` *aus der* `inittab` *von* `simpleinit`

Vor dem `login:`-Prompt wird die Datei `/etc/issue` angezeigt, wenn nicht die Option `-i` angegeben wurde. Bei der Anzeige werden einige spezielle Zeichenfolgen durch die entsprechenden Werte ersetzt, so dass z. B. stets die aktuelle Kernel-Version angezeigt werden kann. Die vollständige Liste der möglichen Variablen finden Sie in Tabelle 4.3.

Zeichenfolge	Ersetzung durch
`\s`	Name des Betriebssystems (Linux)
`\n`	Name des Rechners (Host-Name)
`\r`	Release des Betriebssystems (z. B. 1.2.3)
`\v`	Weitere Versionsinformationen
`\m`	Rechnerarchitektur (i386)
`\o`	Domainname
`\d`	Aktuelles Datum
`\t`	Aktuelle Uhrzeit
`\b`	Baudrate

Tabelle 4.3 Ersetzungen in der Datei `/etc/issue`

Einige Distributionen erzeugen die Datei `/etc/issue` bei jedem Systemstart im Skript `/etc/rc.local` neu und tragen dort die aktuelle Kernel-Version ein. Bei

der systemspezifischen Anpassung der Datei `/etc/issue` sollte dieses »Feature« unbedingt abgeschaltet werden.

Beim Anschluss eines seriellen Terminals muss die Option `-L` angegeben werden, damit das Programm `agetty` das Terminal auch als lokal angeschlossen erkennt. Darüber hinaus können verschiedene Baudraten durch Kommata getrennt angegeben werden, zwischen denen mittels `Break` umgeschaltet werden kann.

Internes getty des System-V-Init

Zu dem System-V-Init existiert ein Patch, der ein einfaches `getty`, abgeleitet vom `agetty`, in das Programm `init` integriert. Ebenso ist ein einfaches `login` integriert, so dass eine Reihe von Fehlerquellen ausgeschlossen werden können. Dadurch wird zum einen die Konfiguration einfacher, zum anderen kann ein guter Teil des Hauptspeichers für andere Zwecke verwendet werden, insbesondere, wenn viele `getty`-Programme gestartet werden und der Rechner über nur wenig Hauptspeicher verfügt.

Zur Konfiguration wurde das neue Schlüsselwort `igetty` für die `inittab` eingeführt, durch das das interne `getty` verwendet wird. Das Listing 4.33 zeigt die Verwendung in der Datei `inittab`.

```
i1:123456:igetty:tty1
```

Listing 4.33 Starten des internen `getty`

Dieses interne `getty` ist nicht für Terminals oder Modems geeignet, da hier keine Konfigurationsmöglichkeiten für Baudraten und Ähnliches bestehen. Für die virtuellen Konsolen ist es jedoch oft eine sinnvolle Alternative, wenn viele `getty`-Programme gestartet werden sollen.

Das Programm getty_ps

Das Programm `getty_ps` wurde ursprünglich von Paul Sutcliffe entwickelt und wird heute von Kris Gleason gepflegt. Dieses Programm eignet sich sowohl für den Betrieb auf den virtuellen Konsolen als auch für ein- und ausgehende Verbindungen über ein Modem. Aufgrund der größeren Flexibilität ist die Konfiguration etwas komplizierter als beim einfachen `agetty`, sie erfolgt fast vollständig zur Laufzeit.

Zunächst muss das `getty` in die Datei `/etc/inittab` aufgenommen werden. Dabei ist es wichtig, dass zumindest die Reihenfolge der Parameter richtig ist. Hinweise dazu liefert die Manpage. Außerdem kann das Programm auf einer freien Konsole zunächst ausprobiert werden, ohne in die Datei `inittab` eingetragen zu sein, indem einfach der Teil nach dem dritten Doppelpunkt direkt als Befehl eingegeben wird. Das Programm `getty` erwartet als Parameter mindestens das `tty`, auf dem eine Anmeldung erfolgen soll. Optional ist die Angabe

der Geschwindigkeit und des Terminal-Typs, für den ein Eintrag in der Datei `/etc/termcap` existiert. Das Listing 4.34 zeigt je einen Eintrag in der Datei `/etc/inittab` für das System-V-`init` für eine virtuelle Konsole und ein serielles Terminal.

```
1:12345:respawn:/sbin/getty tty1 VC linux-28
S1:2345:respawn:/sbin/getty ttyS1 DT9600 vt220
```

Listing 4.34 `getty`*-Start in der Datei* `/etc/inittab`

Im zweiten Schritt muss möglicherweise die Datei `/etc/gettydefs` angepasst werden. Das Format ist in der Manpage `gettydefs(5)` dokumentiert. Die Quellen des Programms `getty_ps` enthalten ein Beispiel, das zunächst unverändert eingesetzt werden kann. Dort sind die Standardwerte für die Initialisierung der Anschlüsse und der gewählten Geschwindigkeiten konfiguriert. Jede einzelne Zeile hat folgendes Format:

```
speed# init-flags # final-flags #login-string#next-speed
```

Nach jedem Eintrag muss eine Leerzeile folgen. Die erste Zeile ist der Standardwert, falls beim Aufruf von `getty` keine Geschwindigkeit angegeben wird. Kommentare werden durch ein # am Zeilenanfang markiert. Nach Änderungen an dieser Datei sollte mit dem Befehl `getty -c` die Syntax dieser Datei überprüft werden, da bei Syntaxfehlern in dieser Datei keine Anmeldung ans System möglich ist.

`speed` ist die symbolische Angabe der Geschwindigkeit, die beim Aufruf von `getty` als Parameter angegeben wird. Wird bei `next-speed` dieser Wert erneut angegeben, so lässt sich die Geschwindigkeit nicht verändern. Andernfalls kann mit der Taste [Break] zwischen den verschiedenen Geschwindigkeiten umgeschaltet werden.

Die Werte `init-flags` und `final-flags` bestimmen die initialen und endgültigen TERMIO-Einstellungen. Im Normalfall ist hier keine Änderung notwendig, für mein Terminal musste jedoch das XON/XOFF-Protokoll ausgeschaltet werden. Dazu habe ich die Einstellungen `-IXON -IXOFF` zusätzlich in die `final-flags` aufgenommen. Die hier anzugebenden Flags sind genau die Parameter, die auch dem Programm `stty` bekannt sind. Interessierte Benutzer seien hier auf die Manpage zu `stty(1)` verwiesen.

Der `login-string` enthält den Text, der als Login-Prompt ausgegeben wird. Dabei werden, genau wie bei der Anzeige der Datei `/etc/issue`, eine Reihe von speziellen Zeichenfolgen ersetzt. Tabelle 4.4 enthält eine Auflistung dieser Zeichenfolgen und deren Ersetzungen.

Zeichenfolge	Ersetzung durch
`@B`	Baudrate
`@D`	Das aktuelle Datum (MM/DD/YY)
`@L`	Das aktuelle `tty`
`@S`	Name des Rechners (Host-Name)
`@T`	Die aktuelle Uhrzeit (HH:MM:SS)
`@V`	Version des Betriebssystems
`@@`	Ein einzelnes `@`-Zeichen
`\\`	Ein Backslash (`\`)
`\b`	Ein Backspace (Strg-H)
`\c`	Verhindert Zeilenvorschub
`\f`	Ein Formfeed (Strg-L)
`\n`	Ein New-Line (Strg-J)
`\r`	Ein Carriage-Return (Strg-M)
`\s`	Ein einzelnes Leerzeichen
`\t`	Ein Tabulator (Strg-I)
`\`*nnn*	ASCII-Zeichen mit Oktal-Wert *nnn*
`\@`	Ein `@`-Zeichen

Tabelle 4.4 Ersetzungen von `getty_ps`

Im letzten Schritt kann das Programm für jeden Anschluss speziell konfiguriert werden. Das betrifft zum einen die Datei `issue`, aber auch Geschwindigkeiten oder erweiterte Funktionen. Konfigurationen, die für alle Anschlüsse gelten, werden in der Datei `/etc/conf.getty` (bzw. `/etc/defaults/getty`) vorgenommen. Für einen speziellen Anschluss heißt die Datei `/etc/conf.getty.`*`Line`* (bzw. `/etc/defaults/getty.`*`Line`*). Mit der Option `-d` kann eine andere Datei angegeben werden. Dies ist beim Test von Änderungen oft sinnvoll, damit die übrigen `getty`-Programme auf jeden Fall weiterhin lauffähig bleiben.

Die Konfigurationsdateien enthalten Zeilen der Form `NAME=`*`Wert`*. Es existieren eine ganze Reihe von Werten, die so zur Laufzeit verändert werden können. Die wichtigsten werden im Folgenden vorgestellt, eine ausführliche Darstellung aller möglichen Einstellungen finden Sie in der Manpage `getty(1)`.

`SYSTEM=`*`name`*
: Die Zeichenfolge `@S` wird normalerweise durch den Host-Namen ersetzt. Mit dieser Einstellung kann ein anderer Name für das `getty` eingestellt werden.

`VERSION=`*`string`*
: Die Zeichenfolge `@V` wird im Normalfall durch die Version des Betriebssystems ersetzt. Stattdessen kann der hier angegebene Text verwendet werden. Beginnt der Text mit einem Slash (`/`), dann wird der Inhalt der angegebenen Datei angezeigt.

`ISSUE=`*`string`*
: `getty` zeigt beim Start den Inhalt der Datei `/etc/issue` an. Hier kann entweder ein anderer Text angegeben werden oder, wenn der String mit einem Slash (`/`) beginnt, eine andere Datei.

`LOGIN=`*`name`*
: *name* ist der komplette Pfad zu einem Programm, das anstelle des normalen `/bin/login` verwendet werden soll. Als Parameter wird der eingegebene Benutzername übergeben.

`CLEAR=`*`value`*
: `getty` löscht den Bildschirm, bevor die Datei `/etc/issue` ausgegeben wird. Wird hier der Wert `NO` eingetragen, so wird der Bildschirm nicht gelöscht.

Nach den allgemeinen Einstellungen folgen nun eine Reihe von Konfigurationsmöglichkeiten für die Verwendung von Modems bzw. seriellen Terminals in Verbindung mit `getty_ps`.

`INIT=`*`string`*
: Mit diesem Wert kann das Modem beim Start von `getty` initialisiert werden. Dazu kann z. B. der Eintrag `INIT="" ATZ\r` dienen. Die Zeichen werden als »Chat-Skript« interpretiert, beginnend mit einem Text, der vom Modem erwartet wird.

`HANGUP=`*`value`*
: Normalerweise unterbricht `getty` beim Start die Verbindung, so dass nach drei Fehlversuchen oder dem Ausloggen die Leitung getrennt wird. Wird hier der Wert `NO` eingetragen, so wird nicht aufgelegt, sondern einfach ein neuer Login-Prompt angezeigt. Dieselbe Funktion erfüllt die Kommandozeilenoption `-h`.

`WAITCHAR=`*`value`*
: `getty` wartet mit der Ausgabe des Login-Prompts, bis ein Zeichen eingegeben wurde. Dies ist sinnvoll für direkt angeschlossene Terminals oder Modems, die stets `Carrier Detect` anzeigen.

`DELAY=`*`seconds`*
: Zwischen dem ersten eingegebenen Zeichen und der Ausgabe des Login-Prompts wird normalerweise nicht gewartet. Mit dieser Option (oder der Kommandozeilenoption `-r`) lässt sich eine Wartezeit festlegen.

`TIMEOUT=`*`number`*
: `getty` wartet unendlich lange auf einen Benutzernamen. Für Wahlverbindungen kann es sinnvoll sein, hier einen Timeout einzutragen. Allerdings sollte dann auch beim Ende/Neustart von `getty` aufgelegt werden.

`CONNECT=`*`string`*
: Hier kann wieder eine Sequenz von Strings und Modembefehlen angegeben werden, mit der die Verbindung zwischen Modem und `getty` hergestellt

wird. Das automatische Einstellen der Übertragungsgeschwindigkeit hat sich hierbei nicht als zuverlässig erwiesen.

`WAITFOR=`*`string`*
: Mit diesem Parameter wird ähnlich wie bei `WAITCHAR` auf eine Zeichenfolge gewartet. Das Programm `getty` läuft erst weiter, wenn diese Zeichenfolge erkannt wurde. Im folgenden Beispiel wartet `getty` auf einen Anruf, der vom Modem mit dem String `RING` signalisiert wird. Anschließend wird abgenommen (`ATA`) und auf die Zeichenfolge `CONNECT` vom Modem gewartet. Die Zeichenfolge `""` steht dabei für »nicht warten«.

```
WAITFOR=RING
CONNECT="" ATA\r CONNECT
```

Listing 4.35 Chat-Skript für `gettyps`

Für den Modembetrieb ist es möglich, ein Login nur zu bestimmten Zeiten oder nach einer vorher festgelegten Abfolge von Klingeln und erneut anrufen zu gestatten. Diese Einstellungen können mit den folgenden Schlüsselworten gesteuert werden:

`RINGBACK=`*`value`*
: Wenn dieser Wert auf `YES` gesetzt wird, so wird ein Login nur erlaubt, wenn es ein- bis dreimal klingelt, dann aufgelegt wird und anschließend binnen 60 Sekunden erneut angerufen wird. Diese Werte können mit einer Reihe von weiteren Parametern verändert werden. `MINRINGS` und `MAXRINGS` bestimmen die minimale und maximale Anzahl von Klingelzeichen beim ersten Anruf. Die Werte `MINRBTIME` und `MAXRBTIME` geben die minimale bzw. maximale Zeit zwischen dem ersten und zweiten Anruf an.

`SCHED=`*`range1 range2 range3 ...`*
: `getty` erlaubt nur Anmeldungen in den Zeitspannen, die durch *range* festgelegt sind. Außerhalb dieser Zeiten sendet `getty` den `OFF`-String und wartet auf die nächste On-Zeit. Eine erlaubte Zeitspanne wird in der Form `d:hh:mm-d:hh:mm` angegeben. Dabei benennt `d` den Tag der Woche (0 = Sonntag, 1 = Montag etc.), `hh` die Stunde und `mm` die Minute, zu der ein Zeitraum beginnt oder endet.

`OFF=`*`string`*
: Das Format dieser Einstellung ist dasselbe wie für den `INIT`-String. Dieser String wird an das Modem gesendet, wenn der Anschluss keine Logins mehr erlaubt. Damit kann z. B. am Modem die Auto-Answer-Funktion ausgeschaltet werden oder abgehoben werden, damit Anrufer ein Besetztzeichen erhalten.

Das Programm `getty_ps` ist auch in der Lage, mit einem Fido-Mailer zusammenzuarbeiten. Interessierte Anwender finden mehr Informationen dazu in der Manpage zu `getty(1)`.

Das Programm getty_ps ist ein flexibles und leistungsfähiges getty, das aber auch eine Reihe von Fallstricken bei der Konfiguration bereithält. Für den Einsatz von Modems oder seriellen Terminals lohnt sich der erhöhte Aufwand jedoch auf jeden Fall, da praktisch jeder Konfigurationswunsch erfüllt werden kann. Besonders nützlich ist die Fähigkeit, einzelne Anschlüsse unterschiedlich zu konfigurieren und damit unterschiedlich zu behandeln.

Das Programm uugetty

Das Programm uugetty gehört zum Paket getty_ps und wird im Wesentlichen genauso konfiguriert. Es ermöglicht jedoch die Zusammenarbeit mit Programmen wie uucp auf einer seriellen Leitung, da Lock-Dateien angelegt werden, die die mehrfache, gleichzeitige Benutzung verhindern.

Linux stellt zwei Devices bereit, um mit einer seriellen Schnittstelle zu kommunizieren. Das sind zum einen die Geräte /dev/ttyS*, die für eingehende Verbindungen dienen. Die Geräte /dev/cua* hingegen sind für ausgehende Anrufe gedacht. Damit kann der Kernel die Sperrverwaltung selbst durchführen. Dieses System funktioniert jedoch nur, wenn das Modem im Auto-Answer-Modus ist. Damit nimmt das Modem bei jedem Anruf nach einer festgelegten Anzahl von Klingelzeichen ab, auch wenn der Rechner nicht bereit ist oder keine Anmeldung erlaubt ist. Nimmt das getty beim Anruf selbst ab, dann muss für beide Arten der Verbindung das Gerät /dev/ttyS* verwendet werden und die Anwendungen müssen sich selbst um die Sperrverwaltung kümmern.

Der Filesystem-Hierarchie-Standard legt fest, dass Lock-Dateien im Verzeichnis /var/lock abgelegt werden. Nicht alle Programme verwenden dieses Verzeichnis, daher ist hier Vorsicht geboten. Nur wenn alle Programme die Lock-Dateien im richtigen Verzeichnis suchen bzw. anlegen und diese Dateien das richtige Format haben, funktioniert diese Sperrverwaltung zuverlässig. So verwenden einige Distributionen das Gerät /dev/modem zum Zugriff auf die serielle Schnittstelle, an der das Modem angeschlossen ist. Dann müssen alle Programme natürlich auch dieses Gerät verwenden (und sperren). Auf Dauer ist es sinnvoller, das echte Gerät (/dev/ttyS*) anzugeben, um diesen Problemen aus dem Weg zu gehen.

Nach diesen allgemeinen Erläuterungen nun zurück zum uugetty. Wird beim Start eine gültige Lock-Datei gefunden, so wartet uugetty, bis der Prozess diesen Lock aufhebt, und beendet sich dann. Anschließend wird von init ein neues uugetty gestartet, das die Leitung neu initialisieren kann. Wird eine ungültige Lock-Datei gefunden (d. h., der Prozess, der diese Datei erzeugt hat, existiert nicht mehr), so wird diese Datei gelöscht und uugetty arbeitet wie gewohnt weiter.

Zur Laufzeit wird, wenn auf einen Anruf gewartet wird, regelmäßig nach Lock-Dateien gesucht, z. B. von `uucp`. Dann beendet sich `uugetty` und das von `init` neu gestartete Programm kann das Modem neu initialisieren.

Wird keine Lock-Datei gefunden, so wartet `uugetty` auf eingehende Anrufe, ohne eine Lock-Datei zu erzeugen. Damit ist es möglich, das Modem für ausgehende Anrufe zu benutzen. Geht nun ein Anruf ein, so wird die entsprechende Lock-Datei erzeugt. Bei der Verwendung der `RINGBACK`-Funktion werden bereits beim ersten Anruf Lock-Dateien erzeugt, so dass der später zu erwartende zweite Anruf nicht durch ausgehende Calls gestört werden kann.

Das Programm mgetty

Das Programm `mgetty` wurde von Gert Döring entwickelt und wird zusammen mit `sendfax` in einem Paket verteilt. Dieses Paket gestattet sowohl ein- als auch ausgehende Anrufe und kann dabei zwischen Daten- und Faxverbindungen unterscheiden. Ein weiteres Programm (`vgetty`) verwandelt Ihr Modem in einen Anrufbeantworter, sofern Ihr Modem diese Funktion unterstützt.

Zur Installation von `mgetty` müssen Sie die Datei `policy.h-dist` als `policy.h` kopieren und an Ihr System anpassen. Achten Sie hier besonders auf die korrekten Lock-Dateien. Diese Datei ist ausführlich kommentiert und enthält viele Informationen, die für den Betrieb von Fax-Modems nützlich sind. Das ausführliche Handbuch zu `mgetty` enthält weitere Hinweise für spezielle Modemtypen.

Da viele Konfigurationen zur Übersetzungszeit vorgenommen werden müssen, sollten Sie `mgetty` selbst übersetzen und die Quelltexte noch eine Zeit lang verfügbar halten, um weitere Anpassungen vornehmen zu können. Wenn Sie mit Ihrer Konfiguration zufrieden sind, so sichern Sie Ihre Änderungen, indem Sie einen context- oder unified-`diff` zwischen den Dateien `policy.h-dist` und `policy.h` dauerhaft speichern. Bei der Installation einer neuen Version oder notwendig gewordenen Anpassungen können Sie Ihre Änderungen direkt nachvollziehen (entweder von Hand, mit dem Emacs-Mode `ediff` oder mit dem Programm `patch`).

4.4.7 Arbeit an einem Terminal oder über eine Modemverbindung

Wenn Sie an Ihrem Rechner ein serielles Terminal angeschlossen haben, einen PC als Terminal verwenden oder sich mit einem Modem an einem anderen System anmelden, haben Sie nicht mehr den Komfort der virtuellen Konsolen unter Linux oder die Möglichkeit, mit `gpm` Daten zwischen den Konsolen mit Hilfe der Maus zu übertragen. In manchen Fällen kann man sich hier mit Emacs behelfen, aber oft ist das keine gute Idee.

In diesem Fall sollten Sie sich unbedingt das Programm GNU-`screen` ansehen. Dieses emuliert auf praktisch jedem Terminaltyp ein VT100 und verwaltet mehrere Sessions parallel. Sie können mit nur wenigen Tastendrücken zwischen den verschiedenen Sitzungen wechseln und Textdaten zwischen den simulierten Terminals übertragen. Tabelle 4.5 enthält eine Übersicht über die wichtigsten Tastenkombinationen.

Taste	Bedeutung
Strg+A	Escape
Strg+A ?	Hilfe anzeigen
Strg+A a	Ein Strg+A an die Anwendung senden
Strg+A [	Copy
Strg+A]	Paste
Strg+A d	Detach
Strg+A 0-9	Terminal 0-9 anwählen

Tabelle 4.5 Tastenkombinationen für `screen`

Außerdem können Sie sich einmal anmelden, `screen` verwenden und beim Abmelden nur in den Hintergrund schicken (detachen), aber nicht beenden. Diese Sitzung können Sie mit `screen -r` an einem anderen Terminal wieder aufnehmen.

4.4.8 Benutzerinformationen

Nach der erfolgten Anmeldung wird von dem Programm `login` die Datei `/etc/motd` (message of the day) angezeigt, bevor die Shell des Benutzers gestartet wird. In dieser Datei werden oft Meldungen untergebracht, die über die Systemverfügbarkeit oder neue Software informieren.

Wenn Sie auf Ihrem System `xdm` verwenden, dann können Sie das Programm `xmessage` verwenden, um eine Nachricht anzuzeigen. Andere Programme für diesen Zweck sind `xbanner` und `xmotd`.

4.5 Benutzerveränderbare Systemkonfiguration

Im folgenden Abschnitt werden die Möglichkeiten eines Benutzers erläutert, wie er, obwohl dies eigentlich Aufgabe des Systemverwalters ist, auch systemspezifische Dinge für seine Anmeldung verändern kann. Dies ist ein wesentlicher Vorteil eines Unix-Systems, da viele Probleme auch ohne Eingreifen des Systemverwalters gelöst werden können.

Bei vielen Dingen, insbesondere bei der Implementierung, sind eine Reihe von Sicherheitsüberlegungen zu berücksichtigen. Da jedoch der Quellcode des gesamten Systems zur Verfügung steht, ist es für erfahrene Unix-Kenner möglich, Sicherheitslücken durch das Studium der Programmquellen zu finden und dann zu beseitigen. Allgemein hält man Systeme, deren Quellcode verfügbar und damit überprüfbar ist, für sicherer als Systeme, die von außen nicht überprüfbar sind (security by obscurity).

4.6 Terminal-Konfiguration

Die Ansteuerung von Bildschirmgeräten (einem Terminal und der Konsole) erfolgt unter Unix mit Hilfe von Bibliotheken, die die Programme unabhängig von der verwendeten Hardware halten. Diese Bibliotheken lesen die Beschreibung der Terminals mit ihren Fähigkeiten und Befehlssequenzen aus so genannten Terminal-Datenbanken.

Zunächst wurden Unix-Systeme mit einem Fernschreiber (*Typewriter*, TTY) bedient. Mit dem Aufkommen der ersten Terminals war es notwendig, auch diese Geräte zu unterstützen. Da es sehr viele verschiedene und natürlich inkompatible Terminals gab und gibt, wurde eine Methode gesucht, mit der Programme unabhängig vom verwendeten Terminal bleiben. Zunächst wurde im Rahmen des BSD-Systems die `termcap`-Bibliothek entwickelt. Auf den System-V-Systemen wurde die `curses`-Bibliothek entwickelt. Heutige Unix-Systeme unterstützen in der Regel beide Bibliotheken.

4.6.1 Die termcap-Datenbank

Eine verbreitete Datenbank ist `/etc/termcap` (*Terminal Capabilities*). Dies ist eine Textdatei, in die der Systemverwalter mittels eines Editors neue Einträge aufnehmen oder bestehende verändern kann. Da es jedoch unüberschaubar viele verschiedene Terminals gibt, ist diese Datenbank auf vielen Systemen nicht besonders aktuell oder gut gepflegt. Bei Problemen mit dieser Datenbank kann man versuchen, einen verbreiteten Standard zu verwenden (z. B. indem die Umgebungsvariable `TERM` auf den Wert `vt100` gesetzt wird), sofern dieses Terminal emuliert wird. Damit verliert man jedoch den Zugriff auf eventuell verfügbare Erweiterungen oder Verbesserungen des Geräts gegenüber dem Standard.

Aufgrund dieser Problematik ist es oft sinnvoll, sei es probeweise oder als dauerhafte Einrichtung, wenn einzelne Benutzer eine andere Terminal-Datenbank benutzen. Ein Benutzer hat verschiedene Möglichkeiten, um die Auswahl der Datenbank zu beeinflussen:

- Man kann die Systemdatenbank `/etc/termcap` verwenden, indem nur die Umgebungsvariable `TERM` auf den richtigen Terminaltyp gesetzt wird.
- Ist die Systemdatenbank nicht geeignet, so kann ein Benutzer eine eigene Datenbank in seinem Home-Verzeichnis anlegen und dann aktivieren (Listing 4.36).
- Der Eintrag aus der Termcap-Datenbank kann in die Umgebungsvariable `TERMCAP` kopiert und dann modifiziert werden.

```
(linux):~> cp /etc/termcap .termcap
(linux):~> setenv TERMCAP $HOME/.termcap
```

Listing 4.36 Anlegen einer benutzereigenen Termcap-Datenbank

Im Listing 4.37 finden Sie ein Beispiel für einen Termcap-Eintrag. Das Format ist dasselbe wie in der Datei `/etc/printcap`. Die Bedeutung der einzelnen Einträge erfahren Sie in der Manpage zu `termcap(5)` oder der entsprechenden Info-Dokumentation. Wenn Sie nicht ganze Abende in den Untiefen der Terminal-Konfiguration verbringen wollen, dann lassen Sie am besten die Finger davon.

```
d0|vt100|vt100-am|vt100am|dec vt100:\
        :do=^J:co#80:li#24:cl=\E[;H\E[2J:sf=\ED:\
        :le=^H:bs:am:cm=\E[%i%d;%dH:nd=\E[C:up=\E[A:\
        :ce=\E[K:cd=\E[J:so=\E[7m:se=\E[m:us=\E[4m:\
        :ue=\E[m:md=\E[1m:mr=\E[7m:mb=\E[5m:\
        :rs=\E>\E[?3l\E[?4l\E[?5l\E[?7h\E[?8h:\
        :ks=\E[?1h\E=:\
        :ku=\EOA:kd=\EOB:kr=\EOC:kl=\EOD:kb=^H:\
        :ho=\E[H:k1=\EOP:k2=\EOQ:k3=\EOR:k4=\EOS:sr=\EM:\
        :ke=\E[?1l\E>:vt#3:xn:me=\E[m:is=\E[1;24r\E[24;1H:\
        :sc=\E7:rc=\E8:cs=\E[%i%d;%dr:pt:
```

Listing 4.37 Ein Beispiel für einen `termcap`-Eintrag

4.6.2 Die terminfo-Datenbank

Eine weitere Datenbank zur Ansteuerung von Terminals, die auf praktisch jedem Unix-System vorhanden ist, ist die `terminfo`-Datenbank. Diese Datenbank wird von der `ncurses`-Bibliothek verwendet. Die systemweit verwendete `terminfo`-Datenbank befindet sich im Verzeichnis `/usr/share/terminfo`.

Zum schnelleren Zugriff auf die einzelnen Einträge in der Datenbank existiert für jeden Buchstaben ein Unterverzeichnis, in dem dann die mit diesem Buchstaben beginnenden Terminal-Beschreibungen angelegt werden. Diese Dateien werden aus dem zugehörigen Quelltext der Terminal-Beschreibung mit dem Programm `tic` (TermInfo-Compiler) erzeugt. Ist diese Datenbank unvollständig oder fehlerhaft, so kann jeder Benutzer in seinem Home-Bereich eine eigene Da-

tenbank anlegen und mit der Umgebungsvariablen TERMINFO darauf verweisen (Listing 4.38).

```
(linux):~> cp -dR /usr/lib/terminfo terminfo
(linux):~> setenv TERMINFO $HOME/terminfo
```

Listing 4.38 Anlegen einer benutzereigenen Terminfo-Datenbank

Der Standardpfad für eine benutzereigene terminfo-Datenbank ist das Verzeichnis ~/.terminfo. Dieser Pfad wird z. B. von tic verwendet, wenn Sie als nicht privilegierter Benutzer einen Terminfo-Eintrag kompilieren wollen. Kompilierte Einträge können Sie mit dem Befehl infocmp wieder in ein lesbares Format überführen.

4.7 Dokumentationen

Eine wichtige Quelle für Dokumentationen ist unter Unix das man-Programm. Dieses Programm liest die vorhandene Dokumentation ein, formatiert sie und zeigt sie am Bildschirm an. Dabei kann der Suchpfad nach den Manpages vom Systemadministrator in der Datei man.config oder manpath.config im Verzeichnis /etc festgelegt werden. Jeder Benutzer kann jedoch einen eigenen Pfad vorgeben, indem er die Umgebungsvariable MANPATH setzt.

```
(linux):~$ export MANPATH=$HOME/man:$MANPATH
(linux):~> setenv MANPATH $HOME/man:$MANPATH
```

Listing 4.39 Setzen der Umgebungsvariablen MANPATH

Die man-Seiten findet man normalerweise in den Verzeichnissen /usr/man, /usr/local/man und /usr/X11R6/man. Da die Formatierung besonders bei großen Manpages relativ lange dauert, können die verbreiteten man-Programme die formatierten Seiten als cat-pages abspeichern und dann direkt von dort lesen.

Manpages lassen sich mit groff auch in PostScript oder DVI-Dateien umwandeln. Damit ist ein optisch ansprechender Ausdruck möglich. Voraussetzung dafür ist jedoch, dass die nroff-Quellen der Manpages installiert sind. Der Befehl in Listing 4.40 formatiert eine Manpage und speichert sie in einer PostScript-Datei ab.

```
(linux):~> groff -Tps -mandoc /usr/man/man1/ls.1 > ls.ps
```

Listing 4.40 Erzeugen einer PostScript-Ausgabe einer Manpage

Manpages dienen zur schnellen Information über die Optionen und die Bedienung des Programms. Oft findet man dort auch einen Abschnitt mit Beispielen, aber selten ein Tutorial.

Wenn Sie nicht wissen, was ein Befehl tut, dann lesen Sie sich die Manpage durch, bevor Sie ihn ausprobieren. Wenn Sie einen Befehl für eine bestimmte Funktion suchen, so versuchen Sie es mit dem Befehl `apropos`, gefolgt vom Suchbegriff: `apropos` *`Stichwort`*. Die stichwortartige Beschreibung zu einem Befehl können Sie mit `whatis` *`Befehl`* anzeigen. Zu praktisch jedem Befehl gibt es unter Linux eine Manpage, die in der Regel die meisten Fragen beantwortet.

Ein anderes System zur Dokumentation wird vom GNU-Projekt verwendet. Dieses System (`texinfo`) ermöglicht es, aus einem Quelltext sowohl ein online-lesbares Dokument, die so genannten Info-Dateien, mit dem Programm `makeinfo` zu erstellen als auch die Dokumentation mit TeX auszudrucken. Dieses Format ist das Standardformat für die Dokumentation der GNU-Programme. Diese Dokumentation ist als ausführliche Benutzerdokumentation mit Beispielen gedacht. Ausgedruckt kann man diese Dokumentation wie ein Buch lesen, auch das Format ist optisch ansprechend.

Die online-lesbaren Infodateien finden Sie im Verzeichnis `/usr/info` oder `/usr/share/info`, der Suchpfad nach diesen Dateien kann jedoch mit der Umgebungsvariablen `INFOPATH` verändert werden. Infodateien können Sie entweder mit dem Emacs-Info-Mode lesen oder mit dem eigenständigen Programm `info` oder `xinfo`.

```
(linux):~$ export INFOPATH=$HOME/info:$INFOPATH
(linux):~> setenv INFOPATH $HOME/info:$INFOPATH
```

Listing 4.41 Die Umgebungsvariable `INFOPATH` *setzen*

Weitere Informationen zu einzelnen Programmen findet man oft auch im Verzeichnis `/usr/doc`. Dort befinden sich oft `README`-Dateien oder ausführliche Anleitungen. Viele Programme geben, wenn sie mit falschen Parametern oder der Option `-?`, `-h` oder `--help` aufgerufen wurden, eine kurze Bedienungsanleitung aus.

4.8 Bibliotheken

Wie oben bereits erläutert, werden unter Linux Shared Libraries verwendet. Diese Bibliotheken werden in den Adressraum eines Prozesses eingeblendet und für alle Prozesse nur einmal geladen. Es ist jedoch möglich, dass diese Bibliothek Fehler enthält, die Programme eines Benutzers zum Absturz bringen. In solchen Fällen ist es sinnvoll, möglicherweise um eine Fehlerkorrektur zu testen, temporär eine eigene Shared-Library zu benutzen. Der Suchpfad nach diesen Bibliotheken kann mit Hilfe der Umgebungsvariablen `LD_LIBRARY_PATH` verändert werden (Listing 4.42).

```
(linux):~$ export LD_LIBRARY_PATH=$HOME/lib:$LD_LIBRARY_PATH
(linux):~> setenv LD_LIBRARY_PATH $HOME/lib:$LD_LIBRARY_PATH
```

Listing 4.42 Eigene dynamische Bibliotheken verwenden

Beim Start von Programmen, die nicht unter der normalen Kennung dieses Benutzers laufen, sondern `setuid` oder `setgid` installiert sind, wird der in der Variablen gesetzte Pfad ignoriert. Andernfalls könnte man beliebige Funktionen unter der `setuid`-Kennung ablaufen lassen.

Darüber hinaus kann der Benutzer mit der Umgebungsvariablen `LD_PRELOAD` eine zusätzliche Bibliothek bestimmen, die vor allen anderen Bibliotheken geladen wird. In dieser Bibliothek können z. B. einzelne Funktionen aus anderen Bibliotheken überschrieben werden, so dass einzelne Funktionen einfach verändert und getestet werden können.

4.9 Benutzerbezogene Konfiguration

Unix-Systeme stellen eine Reihe von Möglichkeiten bereit, mit deren Hilfe jeder Benutzer sich individuell seine Arbeitsumgebung einrichten kann. Aufgabe des Systemadministrators ist es hierbei zunächst, den Benutzern von Beginn an ein vernünftiges Arbeiten zu ermöglichen. Dazu sollte bei der Einrichtung einer neuen Benutzerkennung eine Standardumgebung installiert werden. Die hierzu notwendigen Dateien sollten im Verzeichnis `/etc/skel` abgelegt werden und können entweder von Hand oder durch ein Skript (siehe auch das Programm `useradd`) in das Home-Verzeichnis des neuen Benutzers kopiert werden. Dabei sollte natürlich dieser Benutzer auch Eigentümer der Dateien werden, damit er diese Dateien später nach eigenen Wünschen anpassen kann.

Moderne Shells verwenden das Zeichen Tilde (~), um das Home-Verzeichnis eines Benutzers zu referenzieren. Portabel (und damit z. B. in `sh`-Skripten verwendbar) ist nur der Zugriff über die Umgebungsvariable `HOME`. Der Befehl `cd` (bzw. `chdir`, change directory) ohne Parameter wechselt immer zurück in das Home-Verzeichnis, so dass zur Anzeige des aktuellen Verzeichnisses der Befehl `pwd` (print working directory) verwendet werden muss. Dieser Befehl ist oft ein interner Befehl der Shell. Aus Kompatibilitätsgründen mit dem POSIX.2-Standard existiert auch ein Programm `/bin/pwd`.

Die Namen der meisten Konfigurationsdateien beginnen mit einem Punkt (`.`), so dass diese Dateien bei einem `ls` nicht angezeigt werden. Zur Anzeige aller Dateien muss dann zusätzlich die Option `-a` (bzw. `--all` beim GNU `ls`) angegeben werden.

4.9.1 Auswahl einer interaktiven Shell

Bei der Anmeldung an einem Text-Terminal oder der Konsole wird die in der Datei `/etc/passwd` eingetragene Shell gestartet. Das wird normalerweise ein Programm wie die `bash` oder die `tcsh` sein. Diese Shell wird auch von dem Programm `xterm` gestartet, wenn kein Programmname als Parameter angegeben wird. Die Shell ist die erste Benutzerschnittstelle, die einem Anwender auf einem Unix-System begegnet. Daher ist die Verwendung einer geeigneten Shell wichtig, um effektiv arbeiten zu können.

Für interaktives Arbeiten wird gewöhnlich eine Shell verwendet, die das Editieren der Kommandozeile erlaubt und eine History-Funktion hat. Weiterhin ist die Vervollständigung der Kommando- und Dateinamen eine sehr nützliche Funktion. Häufig verwendet man auch Aliasnamen, um lange Befehle abzukürzen oder spezielle Optionen stets anzugeben.

Die bash

Die `bash` ist die Shell des GNU-Projekts und auf vielen Linux-Systemen die Standard-Shell. Auch diese Shell verfügt über History-Funktionen und vervollständigt Programm- und Dateinamen. Aktuelle Versionen verfügen wie die `tcsh` über eine programmierbare Erweiterung (unter der URL *http://www.caliban.org/bash/index.shtml#completion* finden Sie eine sehr interessante Sammlung).

Die csh oder tcsh

Die `csh` (oder `tcsh`) wird oft als `login`-Shell verwendet. Sie ist für interaktives Arbeiten gut geeignet und verfügt über praktisch alle Funktionen, die man von einer Shell erwartet. Dennoch ist sie nicht die »optimale« Shell, da es Aufgaben gibt, die diese Shell nicht erfüllen kann. Eine ausführliche Beschreibung der Nachteile der `csh` findet man im Text »Why csh is considered harmful« von Tom Christiansen. Dieser Text wird regelmäßig, z. B. in der Newsgruppe *news:comp.unix.shell*, gepostet und befindet sich im FAQ-Archiv auf `rtfm.mit.edu`.

Eine nützliche Funktion der `tcsh` ist die programmierbare Erweiterungsfunktion. Dort können z. B. Compilerflags oder andere Parameter einprogrammiert werden, die nur bei diesen Befehlen zur Verfügung stehen. Ein ausführliches Beispiel ist in den Quellen der `tcsh` enthalten.

Die leistungsfähigste Shell – zsh

Eine weitere verbreitete Shell ist die `zsh`. Sie ist relativ kompatibel zur Bourne-Shell, hat aber viele Erweiterungen der `csh` und `tcsh` eingebaut. Wenn Sie eine `sh`-kompatible Shell mit programmierbarer Erweiterung suchen, dann ist die `zsh` genau das Richtige für Sie.

Andere Shells

Es existieren noch eine Reihe weiterer (auch frei verfügbarer) Shells, wie z. B. die `pdksh`. Die Auswahl einer speziellen Shell wird immer von der Erfahrung und den Erwartungen des Benutzers abhängen.

Manche Systeme bieten keine Möglichkeit, die Shell zu ändern, da der Befehl `chsh` nicht existiert oder nicht alle Shells in der Datei `/etc/shells` eingetragen sind. Man kann sich dann damit behelfen, die neue Shell in einem Initialisierungsskript der `login`-Shell zu starten. Das sollte mit dem Befehl exec erledigt werden, so dass beim Beenden der Shell auch gleichzeitig die Sitzung beendet und nicht unnötig Speicher für eine zweite Shell belegt wird. Der Befehl `exec` ersetzt die gerade laufende Shell durch das als Parameter angegebene Programm. Sofern man zumindest noch eine interaktive Shell gestartet hat, kann mit diesem Befehl ein Programm gestartet werden, auch wenn die Meldung `no more processes` ausgegeben wird. Die Shell wird durch das Programm ersetzt, wählen Sie das Programm also mit Bedacht.

4.9.2 Nicht interaktive Shells

Zur Programmierung von Shell-Skripten wird normalerweise die Bourne-Shell `sh` benutzt. Diese Shell findet man auf praktisch jedem Unix-System. Unter Linux ist die Shell `/bin/sh` häufig ein symbolischer Link auf die `/bin/bash`. Dies ist dann hinderlich, wenn man Shell-Skripten entwickeln möchte, die auch auf anderen Unix-Systemen laufen sollen. Die `bash` stellt, auch wenn sie mit dem Namen `sh` aufgerufen wird, eine Reihe von nützlichen Erweiterungen zur Verfügung. In diesen Fällen bietet sich die Verwendung der `ash` an, die man allerdings im interaktiven Betrieb selten verwenden wird. Hier ist die `bash` dann angenehmer zu bedienen, da sie über History- und Erweiterungsfunktionen verfügt. Auf einem durchschnittlichen Linux-System sind jedoch etliche Skripten (auch für den Systemstart notwendige) unter Benutzung von speziellen Features der `bash` geschrieben worden. Sie rufen allerdings in der ersten Zeile die `/bin/sh` auf.

Die `csh` sollte zur Programmierung von Skripten besser nicht eingesetzt werden, da man hier ebenfalls an die Grenzen der Shell stößt. Mehr Informationen zu diesem Thema finden Sie wiederum im Text »Why csh is considered harmful«.

4.9.3 Shell-Initialisierung

Je nach Shell werden bei der Initialisierung verschiedene Dateien eingelesen. Für die weit verbreiteten Shells `bash` und `tcsh` werden hier die Konfigurationsdateien vorgestellt.

Der Systemadministrator sollte in den System-Profiles möglichst keine Aliasnamen für Programme definieren. Dies ist eine sehr benutzerspezifische Konfiguration, die demzufolge jeder Anwender für sich selbst erledigen sollte. Dennoch ist es manchmal sinnvoll, dass der Systemadministrator Aliasnamen (wie z. B. `dir`) anlegt, um z. B. beliebte Fehler auszuschließen oder das System insgesamt benutzerfreundlicher zu machen.

Initialisierung der tcsh

Zunächst wird von der `tcsh` die Datei `/etc/csh.cshrc` eingelesen. Wenn eine `login`-Shell gestartet wird, dann wird zusätzlich die Datei `/etc/csh.login` eingelesen. Damit ist die Initialisierung aufgrund der systemweit vorgegebenen Einstellungen abgeschlossen. Jeder Benutzer kann in seinem Home-Verzeichnis eine Reihe von Dateien anlegen, in denen er für sich selbst weitere Einstellungen vornehmen kann. Zunächst wird die Datei `~/.tcshrc` eingelesen. Falls diese nicht vorhanden ist, liest die Shell die Datei `~/.cshrc` ein. Ist diese Shell eine `login`-Shell, so wird zusätzlich die Datei `~/.login` abgearbeitet.

Beim Verlassen einer `login`-Shell werden die Dateien `/etc/csh.logout` und `~/.logout` eingelesen und die darin enthaltenen Befehle ausgeführt.

Die Initialisierung der bash

Auch die `bash` kennt eine Reihe von Initialisierungsdateien. Dabei werden je nach Art der Shell und abhängig von angegebenen Kommandozeilenparametern unterschiedliche Dateien ausgeführt.

Bei einem Start der `bash` als Login-Shell wird zunächst die Datei `/etc/profile` abgearbeitet. Existiert die Datei `~/.bash_profile`, so wird diese Datei eingelesen. Andernfalls wird die Datei `~/.bash_login` abgearbeitet, sofern diese existiert. Existiert keine dieser Dateien im Home-Verzeichnis, wird die Datei `~/.profile` eingelesen, wenn diese existiert.

Bei der Abmeldung vom System wird die Datei `~/.bash_logout` abgearbeitet. Dort können beispielsweise Arbeitsdateien gelöscht oder eine Sicherung der bearbeiteten Daten auf Diskette durchgeführt werden.

Die `sh` liest nur die Dateien `/etc/profile` und `~/.profile` ein, daher sollten Funktionen, die nur für die `bash` gelten sollen oder nur dort möglich sind, in den oben vorgestellten anderen Dateien aufgerufen werden. Andernfalls sind die entsprechenden Skripten nicht mehr in einer heterogenen Umgebung verwendbar.

Wird die `bash` als interaktive Shell gestartet, so wird die Datei `~/.bashrc` eingelesen. Wenn auch in Shells, die unter X gestartet werden, Aliasnamen gesetzt werden sollen, so sollte man hier einen entsprechenden Aufruf einfügen.

Schließlich kann die bash auch als nicht interaktive Shell mit der Option -posix gestartet werden. Dann werden die Initialisierungen durchgeführt, die in den Dateien stehen, die in der Umgebungsvariablen ENV abgelegt sind.

Initialisierung der zsh

Der Ablauf bei der Initialisierung ist bei der zsh ähnlich, aber doch anders. Zunächst wird die Datei /etc/zshenv gelesen. Dort kann die Option RCS gesetzt werden, die das Lesen weiterer Konfigurationsdateien verhindert.

Nach der Datei /etc/zshenv wird die Datei ~/.zshenv gelesen, die entweder im Verzeichnis ZDOTDIR oder HOME gesucht wird. Wenn die zsh mit der Option -l gestartet wird oder das erste Zeichen des Programmnamens ($0) ein Minuszeichen ist, dann werden die Dateien /etc/zprofile und ~/.zprofile gelesen.

Ist die zsh interaktiv, dann werden die Dateien /etc/zshrc und ~/.zshrc abgearbeitet. Wenn die zsh als Login-Shell gestartet wird, dann werden die Dateien /etc/zlogin und ~/.zlogin gelesen.

Beim Abmelden werden die Dateien /etc/zlogout und ~/.zlogout durchlaufen, sofern diese existieren. Wie man sieht, hat man als Benutzer der zsh viele verschiedene Konfigurationsmöglichkeiten.

4.9.4 Konfiguration von Editoren

Da auch die Benutzung von Editoren sehr vom persönlichen Geschmack abhängt, sollte jeder Benutzer für sich einen Standardeditor bestimmen, der dann von (fast) allen Programmen (wie z. B. tin oder crontab) verwendet wird. Ist keine der Umgebungsvariablen EDITOR oder VISUAL gesetzt, so wird der Standardeditor vi verwendet. Viele Benutzer verwenden lieber einen anderen Editor wie joe oder Emacs (siehe Listing 4.43).

```
export EDITOR=emacs
export VISUAL=emacs
```

Listing 4.43 Anpassen des Standardeditors in der bash

Schließlich kann praktisch jeder Editor an die speziellen Wünsche und Anforderungen des Benutzers angepasst werden. Im folgenden werden nur die Konfigurationsdateien vorgestellt, ohne näher auf die Konfiguration selbst einzugehen.

Analog wird für die Anzeige von Manpages das Programm more verwendet. Jeder Benutzer kann ein anderes Programm für die Anzeige wählen, indem die Umgebungsvariable PAGER entsprechend gesetzt wird (siehe Listing 4.44). Oft wird hier das Programm less, das wesentlich leistungsfähiger als more ist, eingetragen. Andere Systeme verfügen über ein Programm pg.

```
setenv PAGER less
```

Listing 4.44 Anpassen des Standard-Pagers in der `tcsh`

Der Editor emacs

Konfiguration und Anwendung dieses leistungsfähigen Editors werden genauer im Kapitel 6, »Der Editor Emacs«, beschrieben. Die benutzereigene Konfiguration finden Sie in der Datei `~/.emacs`.

Der Editor joe

Der Editor `joe` ist ein einfach zu bedienender WordStar-Clone, der von Joseph H. Allen programmiert wurde. Dieser Editor wird für viele Umsteiger von DOS zunächst der am einfachsten zu bedienende sein. Der Editor wird vom Systemadministrator durch die Datei `/usr/lib/joerc` konfiguriert. Ein Benutzer kann diese Datei als `~/.joerc` kopieren und dann seinen eigenen Wünschen entsprechend modifizieren.

Der Autor hat weitere Konfigurationen für die bessere Emulation des WordStar- oder Turbo-C-Editors (`jstar`) und eine noch recht rudimentäre Emulation von Emacs (`jmacs`) erstellt. Alle Beispielkonfigurationen sind gut dokumentierte Textdateien, die mit einem beliebigen Editor bearbeitet werden können.

4.9.5 Tastaturbelegungen

Unter Linux lässt sich die Tastaturbelegung sehr flexibel den eigenen Wünschen anpassen. Das gilt sowohl für die Arbeit an der Konsole als auch für die Arbeit unter X. In diesem Abschnitt geht es nur um die Tastaturbelegung der Konsole. Hinweise zur Umbelegung von Tasten unter X finden Sie im Abschnitt 7.5.3, »Tastaturbelegung unter X«.

Die Tastaturbelegung für die Linux-Konsole wird mit dem Programm `loadkeys` durchgeführt. Dabei kann ein Dateiname als Parameter angegeben werden (z. B. `loadkeys de-latin1.map`). Diese Datei wird eingelesen und die Tastaturbelegung wird entsprechend verändert. Das Programm `loadkeys` kann aber auch seine Eingabe von der Standardeingabe lesen. Das ist z. B. nützlich, wenn man die Tastaturbelegungen in komprimierter Form abspeichert (z. B. auf einer Boot- oder Notfalldiskette). Der Aufruf ist in Listing 4.45, dargestellt.

```
zcat /etc/keytables/de-latin1.map.gz | loadkeys
```

Listing 4.45 Tastaturbelegung mit `loadkeys` *ändern*

Im Paket `kdb-0.90.tar.gz`, das es überall dort gibt, wo man auch einen aktuellen Linux-Kernel bekommt, befindet sich eine Reihe von Tastaturtabellen, darun-

ter auch drei für deutsche Tastaturen. Diese Tastaturtabellen sind im Verzeichnis `/usr/lib/kbd/keytables` installiert.

- `de.map`: Hier sind die Umlaut-Tasten mit den folgenden Sonderzeichen belegt:

Taste	Zeichen	Umschalt-Zeichen
ö	[	{
ä	]	}
ü	@	\
ß	\	?

Tabelle 4.6 Tastaturbelegung

- `de-latin1.map`: Die Umlaut-Tasten liefern mit dieser Tastaturbelegung die erwarteten Umlaute. Dabei ist jedoch zu beachten, dass das Programm, mit dem man die Tastaturbelegung testet, 8-bit-clean sein muss. Das trifft beispielsweise für die `bash` nicht direkt zu. Der `vi`-Clone `elvis` eignet sich recht gut zum schnellen Test einer neuen Tastaturbelegung, da das Programm 8-bit-clean ist.
- `de-latin1-nodeadkeys.map`: Die Akzent-Tasten sind hier, im Gegensatz zur Tastaturbelegung `de-latin1.map`, nicht als dead-keys definiert. Damit wirken diese Tasten nicht wie unter `de-latin1.map` als Modifizierer, sondern geben das entsprechende Zeichen direkt aus.

Die Tastaturbelegung wird zunächst beim Systemstart durch das Programm `loadkeys` geladen. Es kann jedoch jeder Benutzer mit dem Programm `loadkeys` die Tastatur nach seinen Wünschen einrichten. Dabei lässt sich auch die Tastatur in einen nicht mehr benutzbaren Zustand versetzen. Als Beispiel für die Veränderung der Tastaturbelegung kann die Datei `de-latin1.map` dienen.

5 Benutzer, Gruppen und Berechtigungen

5.1 Benutzerrechte als Konzept

Im Gegensatz zu Ein-Benutzer-Systemen wie MS-DOS oder MS-Windows verfügt Unix über eine Reihe von Sicherheitsmechanismen, die sowohl die Veränderung von Systemdateien verhindern als auch die Speicherung von privaten Daten ermöglichen. Auch ist die Einrichtung von Benutzergruppen für spezielle Aufgaben oder Projekte möglich. Die Berechtigungsprüfung erfolgt anhand der zugeordneten Gruppe und Benutzerkennung für Dateien, Verzeichnisse und Prozesse innerhalb des Kernels. Nach der Installation sind auf einem Linux- oder Unix-System keine Benutzerkennungen eingerichtet, die von neuen Benutzern direkt verwendet werden können. Darüber hinaus sollte für alle bereits bestehenden Kennungen, wie z. B. `daemon`, `uucp` oder `news`, entweder ein Passwort vergeben oder die Anmeldung grundsätzlich verboten werden. Der nächste Schritt nach der Installation sollte das Anlegen einer nicht privilegierten Benutzerkennung sein. Die neue Kennung muss in die Datei `/etc/passwd` eingetragen werden. Das kann, je nach System, auf verschiedene Weisen erfolgen.

- Eine neue Kennung kann einfach mit einem beliebigen Editor eingefügt werden. Dabei wird jedoch die Datei nicht gegen Änderungen durch andere Benutzer gesperrt, so dass es hier zu Zugriffskonflikten kommen kann, wenn mehrere Systemadministratoren gleichzeitig die Passwortdatei bearbeiten. Für ein Ein-Benutzer-System spielt dies jedoch keine Rolle.
- Es kann eine spezielle Version eines Editors verwendet werden, die dann eine Lock-Datei anlegt, so dass andere Benutzer in der Zeit nicht verändernd auf die Datei `/etc/passwd` zugreifen können. Das Paket `util-linux` enthält das Programm `vipw`, mit dem die Passwortdatei bearbeitet werden kann. Als Standardeditor wird `vi` verwendet, ein anderer Editor kann mit Hilfe der Umgebungsvariablen `EDITOR` konfiguriert werden.
- Die neue Kennung kann mit dem Programm `useradd` angelegt werden. Dabei wird dann automatisch auch das Home-Verzeichnis dieses Benutzers angelegt und eventuell vorhandene Standardkonfigurationen aus `/etc/skel` werden in dieses Verzeichnis kopiert. Gerade bei der Einrichtung von vielen Kennungen bietet diese Möglichkeit den Vorteil, dass alle Benutzer eine einheitliche Umgebung vorfinden.

- Bei vielen Distributionen ist diese Funktion in das mitgelieferte Administrationstool (z. B. COAS unter Caldera, `linuxconf` unter Red Hat oder Yast unter S.u.S.E.) integriert. Mehr zu diesen Tools erfahren Sie in der Dokumentation zu Ihrer Distribution.

Jede Kennung besteht aus genau einer Zeile in der Datei `/etc/passwd`. Jede Zeile ist in eine Reihe von Feldern unterteilt, die jeweils durch Doppelpunkte getrennt sind und eine spezielle Bedeutung haben (Listing 5.1). Die Dokumentation dazu finden Sie auch in der Manpage `passwd(5)`.

Kennung`:`*Passwort*`:`*uid*`:`*gid*`:`*Name*`:`*Verzeichnis*`:`*Shell*

Listing 5.1 Das Format in der Datei `/etc/passwd`

- Der Name der Benutzerkennung muss aus einem Wort bestehen und darf keine Sonderzeichen enthalten, darf also nur aus Buchstaben ohne Umlaute und Ziffern bestehen. Die Verwendung von Umlauten kann die unterschiedlichsten Effekte hervorrufen. Eine wesentliche Einschränkung ist, dass viele Mail-Systeme eine Adresse, die z. B. Umlaute enthält, verstümmeln. Der Name der Benutzerkennung muss systemweit eindeutig sein. Üblich ist die Verwendung von Vor- oder Nachnamen in Kleinbuchstaben sowie Spitznamen. In manchen Organisationen ist es üblich, numerische oder anders strukturierte Kennungen zentral zu vergeben, zum Beispiel ein Kürzel für die Abteilung und dann die Initialen.

  Unix-Systeme würden es prinzipiell erlauben, Benutzerkennungen in Groß- und Kleinbuchstaben gemischt anzugeben. Dabei ist die Groß- bzw. Kleinschreibung signifikant: Die Namen `Jochen` und `jochen` sind unterschiedlich. Einige Protokolle bzw. Programme beachten diese Unterscheidung jedoch nicht, so dass man Benutzerkennungen grundsätzlich in Kleinbuchstaben anlegen sollte.
- Bei der Einrichtung einer neuen Benutzerkennung sollte das Feld, das das verschlüsselte Passwort enthält, zunächst leer bleiben. Das Passwort kann als `root` erstmalig mit dem Befehl `passwd` *Kennung* gesetzt werden. Wenn eine Benutzerkennung eingerichtet werden soll, unter der keine Anmeldung erlaubt sein soll, so kann anstelle des verschlüsselten Passworts ein Stern (`*`) eingetragen werden. In Passwörtern sollten keine Umlaute verwendet werden.

  Bei der Verwendung von Shadow-Passwörtern wird das (verschlüsselte) Passwort nicht in der Datei `/etc/passwd` gespeichert, sondern in der Datei `/etc/shadow`. Zur Vermeidung von Sicherheitsproblemen sollte jedoch unbedingt ein Stern (`*`) eingetragen werden. Das Passwort wird mit der `libc`-Funktion `crypt` verschlüsselt, deren Quellcode natürlich auch frei verfügbar ist.
- Die `uid`-Nummer (User-Identifikation) ist die eindeutige Kennung eines Benutzers, die auch vom Kernel intern zur Berechtigungsprüfung verwendet wird. Diese Nummer sollte für alle Benutzer des Systems eindeutig sein. Auf

manchen Systemen werden mehrere Benutzerkennungen mit der `uid`-Nummer 0 angelegt und als `root`-Äquivalent verwendet. Es ist in diesem Fall besser, ein Programm wie `sudo` zu verwenden, das nebenbei auch noch ein Protokoll der eingegebenen Befehle schreibt.

- Die `gid`-Nummer (Group-Identifikation) gibt die Zuordnung eines Benutzers zu einer Benutzergruppe an. Damit ist es möglich, dass eine Gruppe von Benutzern Dateien gemeinsam bearbeiten oder einzelne Benutzer bestimmte Programme verwenden dürfen.
- Der Name im vierten Feld in der Datei `/etc/passwd` sollte der volle (reale) Name des Benutzers sein. Dies ist sinnvoll für Programme wie `finger`, aber auch bei der Verwendung von Mail und News. Der hier eingetragene Name kann mit dem Programm `chfn` geändert werden. Dieses Feld, das auch als `gecos`-Feld bekannt ist, kann auch die Nummer des Büros sowie die dienstlichen und privaten Telefonnummern aufnehmen.
- Jedem Benutzer ist ein Home-Verzeichnis zugeordnet, in das gewechselt wird, wenn der Benutzer sich am System anmeldet. Dieses Verzeichnis sollte für alle anderen Benutzer ausführbar sein, so dass z. B. der `finger`-Dämon in dieses Verzeichnis wechseln kann. Nur der Eigentümer sollte Schreibrechte besitzen. Will der Benutzer Daten vor dem Zugriff durch andere schützen, so sollten die Dateien in Unterverzeichnissen, die die entsprechenden Berechtigungen gesetzt haben, unterhalb des Home-Verzeichnisses gespeichert werden. Existiert das Home-Verzeichnis nicht, so bekommt der Benutzer je nach verwendetem `login` das Verzeichnis `/` als Home-Verzeichnis (und es wird eine entsprechende Warnung ausgegeben) oder die Anmeldung wird nicht durchgeführt.
- Nach der erfolgreichen Anmeldung beim System wird die im letzten Feld angegebene Shell gestartet. Die Shell kann von jedem Benutzer mit dem Programm `chsh` geändert werden. Dabei sind jedoch nur die Programme erlaubt, die in der Datei `/etc/shells` aufgeführt sind. Diese Shell sollte existieren, da sonst keine Anmeldung mehr möglich ist. Besonders unangenehm ist dies, wenn der Systemadministrator `root` keine gültige Shell hat. Eine gültige Shell ist auch für Zugriffe mittels `ftp` erforderlich (siehe auch Abschnitt 19.3, »Das File-Transfer-Protokoll (`ftp`)«).

```
root:4IIeEz9nkoE:0:0:System-Administrator:/root:/bin/bash
uucp:*:5:5::/usr/uucp:
jochen:EgIrRB.hrBU:540:6:Jochen Hein:/home/jochen:/bin/tcsh
gast::41:14:Gast-Account:/home/gast:/bin/tcsh
nobody:*:65534:65534:Nobody:/:/bin/false
```

Listing 5.2 Beispiel aus einer Passwortdatei

In Listing 5.2 sehen Sie einen Auszug aus einer Passwortdatei. Dem Systemadministrator `root` ist ein Passwort zugeordnet. Er hat stets die Benutzer- und

Gruppen-Nummer 0. Diese Nummer dient im Kernel zur Berechtigungsprüfung und der Wert 0 als User-Identifikation bedeutet Superuser-Privilegien. Damit ist unter Unix keine weitere Unterteilung der Administratorrechte möglich. Der nicht verabschiedete POSIX.1e-Standard sollte u. a. hierfür einen Standard definieren. Mehr zu diesem Thema finden Sie im Abschnitt 5.10, »Feiner unterteilte Berechtigungen oder der allmächtige Benutzer `root`«.

Als Home-Verzeichnis wird `/root` und als Shell die `bash` verwendet. Das Home-Verzeichnis und die Shell sollten in jedem Fall auf der `root`-Partition liegen, damit der Benutzer `root` sich auch im Single-User-Modus anmelden und arbeiten kann. Wenn Sie die Shell des Benutzers `root` ändern, stellen Sie sicher, dass Sie sich mit der neuen Shell anmelden können, bevor Sie sich vollständig abmelden.

In der zweiten Zeile ist eine Benutzerkennung für `uucp` eingerichtet. Für diesen Benutzer ist ein Stern (`*`) als Passwort eingetragen, eine Anmeldung beim System ist mit dieser Kennung nicht möglich. Eine solche Kennung dient zur Verwaltung einer UUCP-Installation. Der Systemadministrator wechselt mit dem Befehl `suÂ -Â uucp` auf diese Benutzerkennung, bevor er mit der Änderung an Konfigurationsdateien des UUCP-Systems beginnt.

Der Benutzer `jochen` hat eine von Null verschiedene, eindeutige Benutzernummer und ist der Gruppe 6 zugeordnet. In der Datei `/etc/group`, die Sie in Listing 5.6 finden, ist dieser Gruppennummer die Bezeichnung `users` zugeordnet. Diese Zuordnung ist nicht standardisiert. In einem Netzwerk, in dem Sie NFS verwenden, sollten die Dateien `/etc/passwd` und `/etc/group` weitgehend zwischen den verschiedenen Rechnern übereinstimmen. Diese Übereinstimmung können Sie am einfachsten mit Hilfe von *NIS*, dem *Network Information Service*, erreichen. Mehr zu diesem Thema erfahren Sie im Kapitel 23, »Network Information Service«.

In der letzten Zeile finden Sie einen Gastzugang, der nicht mit einem Passwort geschützt ist. Derartige Zugänge sind nur in Sonderfällen sinnvoll, da zumindest Lesezugriff auf zahlreiche Systemdateien besteht. Viele (bekannte) Sicherheitslücken setzen voraus, dass wenigstens ein lokaler Zugang zum System besteht, was bei einer Benutzerkennung ohne Passwort sofort der Fall ist.

Wenn es nicht bereits durch Skripten oder Programme wie `adduser` erledigt wurde, so muss jetzt das Home-Verzeichnis dieses Benutzers angelegt werden. Dieses Verzeichnis muss diesem Benutzer gehören und für diesen schreibbar sein. Für andere Benutzer und Dämonen sollte dieses Verzeichnis ausführbar sein, da z. B. der `finger`-Dämon in dieses Verzeichnis wechselt und die Dateien `.plan` und `.project` aus dem Home-Verzeichnis des Benutzers liest. Ein weiterer Vorteil ist, dass Benutzer in der Lage sind, untereinander Konfigurationsdateien zu lesen und so die eigene Umgebung besser an ihre Anforderungen anpassen können.

Konfigurationsdateien können in das Home-Verzeichnis kopiert werden, so dass neue Anwender eine vorkonfigurierte Umgebung vorfinden. Das Home-Verzeichnis eines Benutzers darf nur für diesen schreibbar sein. Andernfalls kann ein anderer Benutzer dort Dateien anlegen und damit die Berechtigungen dieses Benutzers erlangen. Dies gilt auch und besonders für Pseudo-Benutzer wie `uucp`, `mail` oder `news`.

Die Benutzernummern von 0 bis 99 sind auf LSB-konformen Systemen statisch durch den Distributor festgelegt. Der Bereich von 100 bis 499 ist für automatisch angelegte Benutzer (Skripten des Systemverwalters oder Installationsskripten) reserviert. Der Systemverwalter kann die anderen Nummern frei vergeben.

5.2 Shadow-Passwörter

In der Standard-Unix-Variante werden die verschlüsselten Passwörter in der Datei `/etc/passwd` gespeichert. Da der zur Verschlüsselung verwendete Algorithmus bekannt ist (interessierte Leser finden die Quellen zur Funktion in der `libc`) und als sicher angesehen wird, ist es nur möglich, ein Passwort zu »erraten«. Dazu wählt man Wörter aus einem Wörterbuch, verschlüsselt diese und vergleicht das Ergebnis mit dem in der Datei `/etc/passwd` gespeicherten Passwort. Sind beide Zeichenketten gleich, so hat man das Passwort erraten.

Es gibt Programme (z. B. `crack`, »John The Ripper«), die genau dies durchführen. Dabei können etliche Regeln angewandt werden, die aus Wörtern des Wörterbuchs durch unterschiedliche Groß-/Kleinschreibung oder das Anhängen von Ziffern mögliche Passwörter erzeugen. Alle diese Regeln sind einfach zu konfigurieren. Damit sind prinzipiell alle Passwörter, die in einem auch fremdsprachlichen Wörterbuch stehen oder stehen könnten, unsicher. Das Programm `npasswd` versucht, die Prüfungen, die Crack aufwändig mit der `crypt`-Funktion durchführt, bereits bei der Eingabe des Passworts anzuwenden. Dieses Verfahren benötigt weit weniger Systemressourcen als `crack`, da die aufwändige Verschlüsselung vor dem Vergleich entfällt.

Sichere Passwörter können auf verschiedene Art und Weise generiert werden. Für die meisten Benutzer ist es sinnvoll, einen Spruch oder Satz aus einem Buch zu wählen und die Anfangs- oder Endbuchstaben der Wörter dieses Satzes zu verwenden. Dabei kann man unvermittelt zwischen Groß- und Kleinschreibung wechseln bzw. Ziffern oder Sonderzeichen (wie z. B. `,` oder `;`) einfügen. Auch eine veränderte Reihenfolge oder vertauschte Buchstaben bieten Schutz gegen das einfache Erraten des Passworts.

Man sollte darauf achten, dass manchmal noch die deutsche Tastaturbelegung fehlt und einige Sonderzeichen nicht über die entsprechend beschrifteten Tasten abrufbar sind. Aus diesem Grund empfehlen sich keine Umlaute im Passwort,

außerdem ist bei manchen Systemen die Länge auf acht Zeichen beschränkt. Für sicherheitsrelevante Passwörter empfiehlt sich das Auswürfeln des Passworts mit Würfeln oder einem Zufallszahlengenerator. Eine entsprechende Tabelle finden Sie in Anhang B, »Passwörter generieren«.

Achten Sie darauf, dass Sie Ihr Passwort nicht aufschreiben oder auf andere Weise anderen Personen zugänglich machen. Anders sieht es beim Passwort für den Systemadministrator aus, das auch in dessen Urlaub verfügbar sein sollte, z. B. im Firmensafe hinterlegt. Oft reicht zur Kompromittierung des Passworts auch ein Blick über die Schulter des Benutzers, der sich gerade anmeldet. Höfliche Leute wenden sich in solchen Fällen diskret ab.

Die Datei `/etc/passwd` muss für alle Benutzer lesbar sein, damit z. B. beim Befehl `ls -l` die Namen der Benutzer anstelle der numerischen Benutzeridentifikation angezeigt werden können. Wenn man die Passwörter in einer weiteren, nur für den Systemadministrator lesbaren Datei speichert, können weiterhin alle Programme die in der Datei `/etc/passwd` gespeicherten Daten benutzen. Sie haben jedoch keinen Zugriff auf die verschlüsselten Passwörter. Alle Programme, die die Anmeldung durchführen, laufen unter der Kennung `root`, so dass auch das verschlüsselte Passwort gelesen werden kann. Andere Anwender haben keinen Zugriff auf die Passwörter und können damit auch keinen Wörterbuchangriff ausführen.

Diese Funktion der Shadow-Passwörter ist in der Shadow-Password-Suite implementiert, die Passwörter werden in der Datei `/etc/shadow` gespeichert. Weitere Funktionen des Shadow-Systems sind Passwörter, die nach einer gewissen Zeit verfallen, so dass ein neues Passwort gewählt werden muss. Die Gültigkeit von Passwörtern sollte nicht zu kurz gewählt werden. Insbesondere wenn die Gültigkeit etwa einen Monat beträgt, neigen Benutzer dazu, ein Passwort zu wählen und nur die Nummer des aktuellen Monats anzuhängen. Dies dient nicht der Systemsicherheit. In diesen Fällen ist es oft sinnvoller, wenn Passwörter etwa ein halbes Jahr lang gültig sind.

Mit Shadow-Passwörtern wird ein Unix-System sicherer, es können jedoch unbeabsichtigt neue Sicherheitsprobleme geschaffen werden. Jedes Programm, das sich mit der Anmeldung von Benutzern an das System beschäftigt, muss die Shadow-Routinen verwenden. Daher sollte, obwohl es für Shadow-Programme nicht notwendig ist, ein Stern (`*`) in das Passwort-Feld in der Datei `/etc/passwd` eingetragen werden. Fehlt das Passwort und benutzt ein Programm nicht die Shadow-Routinen, sondern die reguläre Passwortdatei, kann die Anmeldung ohne Passwort erfolgen.

5.3 Network Information Service – NIS

Eine weitere Möglichkeit, Benutzerkennungen zu verwalten, ist das *Network Information Service NIS* oder NIS+, früher bekannt als YP (Yellow Pages). Dabei werden die Kennungen auf einem NIS-Server eingerichtet und dann über das Netzwerk von den NIS-Clients abgerufen. Damit müssen Benutzerkennungen nur einmal für das gesamte Netzwerk eingerichtet und gepflegt werden. Nähere Informationen zur Einrichtung und zur Verwendung von NIS bzw. NIS+ finden Sie in Kapitel 23, »Network Information Service«.

5.4 Pluggable Authentication Modules (PAM)

Auf einem Unix-System müssen viele verschiedene Programme eine Benutzerprüfung durchführen. Das beginnt mit `login`, dem `xdm` und diversen anderen Programmen für Anmeldungen unter X und endet noch lange nicht mit beispielsweise dem `ftpd`-Dämon. Wenn Sie Ihr System auf die Verwendung von Shadow-Passwörtern umstellen wollen, so müssen Sie alle diese Programme prüfen und neu übersetzen oder linken.

Die Änderung oder Erweiterung der Benutzerprüfung ist sehr aufwändig, aber oft genug notwendig. Als Ausweg hierzu wurde von Sun das Verfahren der *Pluggable Authentication Modules* (PAM) entwickelt. Hier werden die Konsumenten, also die Programme, die eine Benutzerprüfung durchführen wollen, von der eigentlichen Überprüfung entkoppelt. Damit kann dynamisch eine zusätzliche Prüfung eingebaut oder entfernt werden, ohne dass die einzelnen Programme davon direkt betroffen sind. Abbildung 5.1 zeigt die Struktur aus der Sicht des Programmierers.

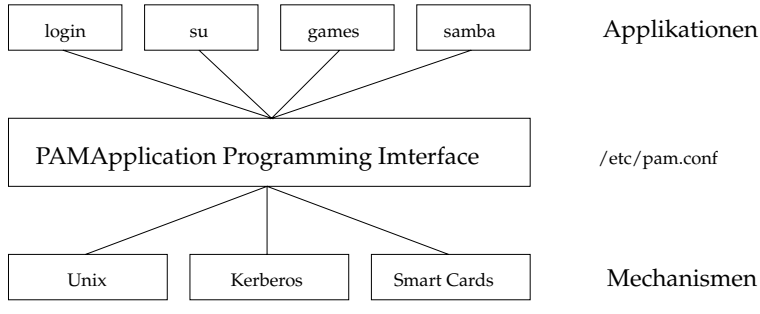


Abbildung 5.1 Struktur der Benutzerprüfung mit Pluggable Authentication Modules

Alle Programme, die sich mit Authentifizierung befassen, benutzen das einheitliche PAM-API (Pluggable Authentication Modules Application Programming Interface), das vollkommen unabhängig von der verwendeten Authentifizierungs-

methode ist. Mit Hilfe der Datei `/etc/pam.conf` bzw. den Dateien im Verzeichnis `/etc/pam.d` kann der Systemadministrator die verwendeten Verfahren bestimmen. Diese Verfahren gestatten eine große Flexibilität bei der Auswahl und Konfiguration.

Bevor wir aber zur Konfiguration kommen, noch einige Worte zu den Möglichkeiten von PAM. PAM beschäftigt sich neben der Authentifizierung mit dem Passwort- bzw. User- und Session-Management. Die einzelnen Module sind austauschbar, können »gestapelt« werden, erstellen Log-Einträge und ermöglichen es, ein »Single-Logon« zu betreiben. Die Ideen hinter PAM und die Definition der Schnittstellen sind im `rfc86.0.txt` der Open Software Foundation erläutert.

Die eigenständige Implementation von PAM unter Linux ist in den Dateien `pam.txt`, `pam_modules.txt` und `pam_appl.txt` dokumentiert. Noch befindet sich das Projekt in der Entwicklung, so dass hier noch weitere Änderungen zu erwarten sind. Dieses Verfahren hat sich jedoch aufgrund der Flexibilität unter Linux schnell durchgesetzt.

Die Konfiguration von PAM kann für jede Applikation einzeln angepasst werden. Man hat die Möglichkeit, entweder die zentrale Datei `/etc/pam.conf` zu pflegen oder für jede Applikation eine Datei mit dem entsprechenden Namen im Verzeichnis `/etc/pam.d` anzulegen. Beide Lösungen haben ihre Vor- und Nachteile, ich tendiere zur Abspeicherung im Verzeichnis `/etc/pam.d`. Wenn Sie hier einen Fehler machen, dann wird höchstens ein Dienst gestört, und die Dateien sind kleiner und übersichtlicher. Das Listing 5.3 zeigt den Eintrag für den `ftp`-Dienst in der Datei `/etc/pam.conf`.

```
# ftp authorization
# Dienst Typ   Steuerung  Modul
ftp   auth     required   /lib/security/pam_listfile.so \
      item=user sense=deny file=/etc/ftpusers onerr=succeed
ftp   auth     required   /lib/security/pam_unix_auth.so
ftp   auth     required   /lib/security/pam_shells.so
ftp   account  required   /lib/security/pam_unix_acct.so
ftp   session  required   /lib/security/pam_unix_session.so
```

Listing 5.3 Ein Ausschnitt aus der Datei `/etc/pam.conf`

In der ersten Spalte steht der Name des Dienstes. Wenn Sie das Verzeichnis `/etc/pam.d` verwenden, dann wird die Datei `/etc/pam.conf` ignoriert und dort die Datei `ftp` gelesen. Das Format dieser Datei ist bis auf das Fehlen der ersten Spalte identisch mit dem der Datei `/etc/pam.conf`. Existiert kein Eintrag für den gewünschten Dienst, dann wird der Eintrag `OTHER` bzw. die Datei `/etc/pam.d/other` gesucht und die dort vorgenommenen Einstellungen werden angewendet.

In der zweiten Spalte im Listing 5.3 wird festgehalten, für welchen Teil von PAM die Einträge gelten sollen. Möglich sind hier `auth`, `account`, `session` und `password`.

`auth`
: Die Applikation liest den Benutzernamen und das Passwort ein, die entsprechenden Module von PAM prüfen die »Echtheit« des Benutzers. Im einfachsten Fall kann keinerlei Prüfung erfolgen, was natürlich nicht besonders empfehlenswert ist. Es existieren Module, die das Verhalten von Unix nachbilden oder auch darüber hinausgehen können, z. B. durch Abfrage einer Smart-Card.

  Eine weitere Funktion dieser Module ist das »Credential Granting«. Das bedeutet, dass für den Benutzer das Recht auf die Benutzung gewisser Dienste erteilt wird.

`account`
: Mit dem Account-Management können z. B. Restriktionen bei der Anmeldung eingerichtet werden. Das können Einschränkungen der Zeiten sein, wann dieser Dienst in Anspruch genommen werden kann, oder die Einschränkung auf bestimmte Terminals. Ebenfalls denkbar ist hier die Überwachung von Systemressourcen, damit sich ein Benutzer beispielsweise nicht mehrfach anmelden kann.

  Für viele private Systeme sind diese Funktionen nicht notwendig, aber besonders bei viel benutzten, öffentlich zugänglichen Systemen kann der Einsatz derartiger Verfahren sinnvoll sein.

`session`
: Die hier aufgeführten Module sind für den Aufbau und das Ende der Sitzung zuständig. Hier können Umgebungsvariablen gesetzt oder andere Einstellungen, wie die maximale Anzahl der Prozesse, vorgenommen werden. Außerdem kann es hier sinnvoll sein, den Beginn und das Ende der Sitzung zu protokollieren.

`password`
: Dient zum Ändern und Verwalten des Passworts. Mit Hilfe dieser Module kann z. B. ein Single-Login simuliert werden, indem alle Passwörter zentral verwaltet werden.

Die dritte Spalte in der Datei `/etc/pam.conf` beschreibt, wann und wie ein Modul verwendet wird. Hier kann das »Stapeln« von Modulen eingestellt werden, indem für eine Applikation mehrere Einträge vorgenommen werden. Einige Beispiele hierfür werden wir im Folgenden kennen lernen.

`required`
: Dieser Eintrag bedeutet, dass der erfolgreiche Durchlauf dieses Moduls erforderlich ist, damit die Prüfung dieses Typs erfolgreich ist. Der Benutzer kann

nicht erkennen, in welchem Modul seine Anmeldung fehlgeschlagen ist, da erst alle weiteren Module durchlaufen werden.

`requisite`
: Der erfolgreiche Durchlauf dieses Moduls ist wie bei `required` erforderlich. Bei einem Fehlschlag in diesem Modul wird allerdings die weitere Verarbeitung abgebrochen. Dieses Verfahren kann sinnvoll sein, wenn man keine vollständige Benutzerprüfung mehr durchführen will. Es ist allerdings möglich, dass ein Angreifer aufgrund des anderen Verhaltens Aufschlüsse über das System und darauf eingerichtete Benutzerkennungen erlangen kann.

`sufficient`
: Der Erfolg dieses Moduls beendet das gesamte Verfahren mit Erfolg. Ein Fehlschlag ist nicht endgültig, so dass in diesem Fall weitere Module aufgerufen werden können.

`optional`
: Dieses Modul ist nicht wichtig für den Erfolg oder Misserfolg der Anmeldung. Ein solches Modul kann aber die Antwort der Applikation auf den Anmeldeversuch beeinflussen.

Die Dateinamen in der vierten Spalte der Datei `/etc/pam.conf` legen fest, welches Modul die angeforderte Funktion ausführt. Linux-PAM enthält eine Reihe von Modulen, mit denen das Verhalten von PAM analog zu den bekannten Unix-Verfahren eingerichtet werden kann. Zusätzlich sind aber noch Zeiteinschränkungen und viele andere Dinge möglich. Eine vollständigere und aktuellere Dokumentation finden Sie in der Datei `pam.txt`.

Die Zeile kann mit einem Backslash `\` fortgesetzt werden. Das erhöht die Übersichtlichkeit, wenn noch weitere Parameter angegeben werden sollen. Ein Beispiel dafür finden Sie in der ersten Zeile von Listing 5.3. Das Listing 5.4 zeigt zum Vergleich die Datei `/etc/pam.d/ftp`, die äquivalent zu den gezeigten Einträgen in der Datei `/etc/pam.conf` ist.

```
auth     required   /lib/security/pam_listfile.so \
    item=user sense=deny file=/etc/ftpusers onerr=succeed
auth     required   /lib/security/pam_pwdb.so shadow nullok
auth     required   /lib/security/pam_shells.so
account  required   /lib/security/pam_pwdb.so
session  required   /lib/security/pam_pwdb.so
```

Listing 5.4 Die Datei `/etc/pam.d/ftp`

Das Listing 5.4 zeigt eine Möglichkeit, wie mehrere Module für einen Dienst verwendet und damit gestapelt werden können. Die Dokumentation zu allen verfügbaren Modulen finden Sie in der Datei `pam.txt`, wenn Sie das PAM-System installiert haben.

In der ersten Zeile werden alle Benutzerkennungen aussortiert, deren Namen in der Datei `/etc/ftpusers` aufgeführt sind. In der Regel sind dort Kennungen wie `root` und verschiedene andere Systemkennungen eingetragen. Es wird das Modul `pam_listfile` verwendet, das den Namen der Datei und einige andere Parameter übergeben bekommt. Entscheidend ist hier, dass mit dem Parameter `onerr=succeed` die Bedingung so verändert wird, dass alle Benutzer, die nicht in dieser Datei aufgeführt sind, sich prinzipiell anmelden können.

Die zweite Zeile prüft anhand der Passwortdatei (bzw. der Shadow-Datei) die Benutzerkennung auf Gültigkeit. Die dritte Prüfung, die beim Anmelden mittels `ftp` durchlaufen werden muss, betrifft die Validität der Shell (geprüft gegen die Datei `/etc/shells`). Damit kann mit drei relativ einfachen Modulen das Verhalten der normalen `ftp`-Anmeldung realisiert werden.

Listing 5.5 zeigt die Konfiguration für das `rlogin`-Programm (siehe auch Abschnitt 19.4, »Die `r`-Tools«). Es wird geprüft, ob die Anmeldung von einem erlaubten Terminal aus erfolgt. In der zweiten Zeile wird durch das Modul `pam_rhosts_auth` geprüft, ob aufgrund der Datei `/etc/hosts.equiv` bzw. `~/.rhosts` die Anmeldung ohne Passwort gestattet ist. Wenn dies der Fall ist, dann ist an dieser Stelle die Benutzerprüfung zu Ende.

```
auth        required     /lib/security/pam_securetty.so
auth        sufficient   /lib/security/pam_rhosts_auth.so
auth        required     /lib/security/pam_pwdb.so \
            shadow nullok
auth        required     /lib/security/pam_nologin.so
account     required     /lib/security/pam_pwdb.so
password    required     /lib/security/pam_cracklib.so
password    required     /lib/security/pam_pwdb.so \
            shadow nullok use_authtok
session     required     /lib/security/pam_pwdb.so
```

Listing 5.5 Die Datei `/etc/pam.d/rlogin` als zweites Beispiel

Ist die Anmeldung ohne Passwort nicht möglich, so wird die normale Passwortroutine des Unix-Systems verwendet. Wieder stellt man fest, dass man mit einigen wenigen und vergleichsweise einfachen Modulen das Verhalten traditioneller Unix-Systeme emulieren kann. Es ist zu erwarten, dass immer mehr Linux-Distributionen und andere Unix-Systeme diese Bibliothek enthalten werden. Der manuelle Umstieg ohne die Hilfe einer Distribution ist aus meiner Sicht nicht empfehlenswert.

5.5 Benutzergruppen

Eine Benutzerkennung wird bereits in der Datei `/etc/passwd` einer Benutzergruppe zugeordnet. Welche Gruppe dies ist, kann mit dem Befehl `id` abgefragt oder in der Passwortdatei nachgelesen werden. Diese Gruppe ist die Standardgruppe dieses Benutzers und wird z. B. verwendet, wenn eine Datei neu angelegt wird. Die Gruppenzuordnung einer bestehenden Datei kann mit dem Befehl `chgrp` *`Gruppe Dateiname`* verändert werden.

Der Eigentümer einer Datei kann (nur vom Benutzer `root`) mit dem Befehl `chown` verändert werden. Damit ist es nicht möglich, Dateien zu »verschenken« und damit Sicherheitslücken zu schaffen oder Quotas zu umgehen.

Eine Gruppe wird in der Datei `/etc/group` definiert. Sie können eine Gruppe mit dem Befehl `groupadd` anlegen oder dazu einen beliebigen Editor verwenden. Auf größeren Systemen sollten Sie in jedem Fall `vigr` verwenden, damit die Datei während der Änderung gesperrt ist. Andernfalls würden Änderungen des einen Administrators durch parallel durchgeführte Anpassungen eines Kollegen überschrieben. Gerade auf aktiven Systemen mit mehreren Systemverwaltern passiert das schneller, als man erwartet.

Ein Benutzer kann mehreren Gruppen zugeordnet werden. Der entsprechende Benutzername wird nach der Gruppe in der Datei `/etc/group` eingetragen, in der er zusätzlich zur in der Datei `/etc/passwd` festgelegten Gruppe mit aufgenommen werden soll. Das Format der Gruppendatei (Listing 5.6) ähnelt dem der Passwortdatei, die Bedeutung der einzelnen Felder wird im Folgenden genauer erläutert. Ein Benutzer kann mit dem Befehl `newgrp` eine andere Gruppe für die aktuelle Sitzung zu seiner Standardgruppe machen.

```
Gruppenname:Passwort:gid:Benutzerkennungen
```

Listing 5.6 Aufbau einer Zeile in der Datei `/etc/passwd`

- Der Gruppenname ist wie bei der Benutzerkennung ein Wort, das keine Sonderzeichen enthält und die Gruppe eindeutig identifiziert.
- Wenn kein Passwort angegeben ist, dann ist das Wechseln zu dieser Gruppe ohne Passwort möglich, sofern der Benutzer bereits in dieser Gruppe eingetragen ist. Ist ein Passwort vorhanden, so wird dieses von dem Programm `newgrp` abgefragt, wenn ein Benutzer in diese Gruppe wechseln will, der nicht in dieser Gruppe eingetragen ist. Dieses Passwort ist mit demselben Algorithmus wie in der Passwortdatei verschlüsselt und kann mit dem Befehl `gpasswd` geändert werden.
- Die Gruppennummer entspricht der eindeutigen Identifikation einer Gruppe. Diese Nummer wird vom Kernel zur Berechtigungsprüfung verwendet.

- Im letzten Feld kann eine Liste von Benutzerkennungen angegeben werden, die dieser Gruppe bereits bei der Anmeldung zugeordnet werden. Ein Benutzer kann in bis zu 32 dieser »supplemental Groups« Mitglied sein und erhält damit möglicherweise mehr Zugriffsrechte auf einige Dateien, die einer dieser Gruppen zugeordnet sind. Die Namen der einzelnen Gruppen werden durch Kommata getrennt.

Alle zugeordneten Gruppen des angemeldeten Benutzers erhält man mit dem Befehl `id` oder `groups` (siehe Listing 5.7). Wenn Sie bei den Befehlen einen Benutzernamen als Parameter angeben, dann erhalten Sie die Gruppenzuordnung dieses Benutzers angezeigt.

```
(linux):~$ id
uid=540(jochen) gid=6(users) groups=6(users),7(projekt)
(linux):~$ groups
users floppy
```

Listing 5.7 Die Befehle `id` *und* `groups`

Im Listing 5.7 wird durch den Befehl `id` an erster Stelle die Benutzer-ID und als zweites die primäre Gruppe ausgegeben. Diese Gruppe bestimmt in der Regel die Gruppe, zu der eine neu angelegte Datei gehört. Sie können diese Gruppe mit Hilfe des Befehls `newgrp` wechseln. Das Listing 5.8, zeigt Ihnen den Befehl.

Die weiteren Gruppen in der Ausgabe von `id` im Listing 5.7 sind die »supplemental Groups«. Ein Benutzer hat auf Dateien über die Gruppenrechte jeweils den entsprechenden Zugriff, egal, ob er diese Gruppe als primäre oder supplementäre Gruppe hat.

```
(linux):~$ id
uid=40(jochen) gid=6(users) groups=6(users),7(projekt)
(linux):~$ newgrp projekt
(linux):~$ id
uid=40(jochen) gid=6(projekt) groups=6(users),7(projekt)
```

Listing 5.8 Wechseln der primären Gruppe mit `newgrp`

Der manuelle Wechsel zwischen den verschiedenen Gruppen ist jedoch unkomfortabel und das Verfahren ist relativ fehleranfällig, wenn mehrere Benutzer in verschiedenen Gruppen oder Projekten zusammenarbeiten. Wenn ein Benutzer nicht auf die korrekte Verwendung der Gruppe achtet, dann haben die anderen Gruppenmitglieder keinen Zugriff auf die angelegten Dateien. Wie das so ist, fällt das natürlich erst auf, wenn der Benutzer gerade nicht greifbar ist, und man fängt an, den Systemverwalter zu suchen. Wir werden später in diesem Kapitel eine viel komfortablere Methode kennen lernen.

5.6 Berechtigungen für Dateien

Die Unterscheidung der Benutzer in verschiedene Gruppen kann für verschiedene Funktionen benutzt werden. Beispielhaft betrachten wir die in Listing 5.9 und Listing 5.10 dargestellte Passwort- und Gruppendatei.

```
root:4IIeEz9nkoE:0:0:System-Administrator:/root:/bin/bash
jochen:EgIrRB.hrBU:40:6:Jochen Hein:/home/jochen:/bin/tcsh
karl::41:6:Karl Eichwalder:/home/karl:/bin/tcsh
michael::42:6:Michael Kohl:/home/michael:/bin/tcsh
```

Listing 5.9 Auszug aus einer Passwortdatei (`/etc/passwd`)

```
root::0:root
admin::3:root,bin,sys,adm,jochen
users::6:
projekt::7:jochen,karl
```

Listing 5.10 Auszug aus einer Gruppendatei (`/etc/group`)

Die Benutzer `karl` und `jochen` arbeiten an einem gemeinsamen Projekt und sind deshalb der Gruppe `projekt` zugeordnet. Die gemeinsam zu bearbeitenden Dateien werden im Verzeichnis `/home/projekt` gespeichert. Die Berechtigungen sind so vergeben, dass andere Benutzer (wie z. B. `michael`) diese Dateien nicht löschen oder verändern dürfen. Die Berechtigungen können mit dem Programm `chmod` gesetzt werden. Listing 5.11 zeigt ein Beispiel für die Vergabe von Berechtigungen bei Verzeichnissen.

```
(linux):~$ ls -ldF jochen projekt
drwxr-x--x  16 jochen  users     2048 Apr 04 21:44 jochen/
drwxrws--t   2 jochen  projekt   1024 Mär 23 08:22 projekt/
```

Listing 5.11 Beispiel für die Berechtigungsvergabe

Im ersten Feld (z. B. `drwxrws--t`) wird der Typ der Datei in der ersten Spalte angezeigt. Eine Liste aller Typen enthält Tabelle 5.1. Im Listing 5.11 sind beide Verzeichniseinträge wieder Unterverzeichnisse.

Dateityp	Bedeutung
`-`	Normale Datei
`d`	Ein Unterverzeichnis (directory)
`p`	Named Pipe
`s`	Socket
`l`	Symbolischer Link
`c`	Character Device
`b`	Block Device

Tabelle 5.1 Arten von Dateien im Dateisystem

Die anderen neun Zeichen geben, in Dreiergruppen unterteilt, die Zugriffsrechte des Eigentümers, der Gruppe und aller anderen Anwender an. Dabei ist zwischen Berechtigungen für den Zugriff auf Verzeichnisse und auf Dateien zu unterscheiden. Tabelle 5.2, listet die möglichen Rechte für Dateien mit Erklärungen dazu auf.

Zugriffsmode	numerisch	Bedeutung
`r`	4	Leseberechtigung
`w`	2	Schreibberechtigung
`x`	1	Programmausführung erlaubt
`s`	4	Start unter anderer Kennung (`s`-Bit, `setuid`)
`s`	2	Start unter anderer Gruppe (`s`-Bit, `setgid`)
`t`	1	Swap anstelle von demand loading

Tabelle 5.2 Berechtigungen für Dateien

Die `s`-Bits können beim Eigentümer oder bei der Gruppe gesetzt werden. Das Programm wird dann nicht unter der Kennung des angemeldeten Benutzers, sondern des Eigentümers bzw. der Gruppe der Datei gestartet (`setuid`). Dies kann, insbesondere wenn der Benutzer `root` Eigentümer der Datei ist, ein Sicherheitsproblem sein. Daher ist zumindest die Ausführung von Skripten unter anderen Kennungen nicht implementiert, da dies immer ein Sicherheitsrisiko darstellt (weitere Informationen zu diesem Komplex finden Sie in der Unix-FAQ und in der Security-FAQ).

Oft ist es möglich, die fehlenden Berechtigungen über eine zusätzliche Gruppe zu erlangen. So können z. B. die Geräte für Disketten (`/dev/fd*`) einer Gruppe `floppy` zugeordnet werden, die auch Schreibberechtigung für diese Geräte erhält. Anschließend kann man die entsprechenden Benutzer zusätzlich in diese Gruppe aufnehmen. Das `t`-Bit bei Programmen bewirkt bei manchen Unix-Systemen[1], dass das Programm nicht vom Dateisystem neu geladen wird, wenn der reale Speicher knapp wird, sondern vom Swapspace aus. Dies könnte für den Betrieb von großen Programmen von langsamen CD-ROM-Laufwerken aus einen bedeutenden Geschwindigkeitsvorteil ausmachen.

1. Unter Linux ist diese Funktion nicht implementiert, da man aufgrund der heute verwendeten schnellen Festplatten keinen Geschwindigkeitsvorteil erwartet.

Shell-Skripten müssen im Gegensatz zu Binärprogrammen sowohl ausführbar als auch lesbar sein, da sie zur Ausführung von der Shell gelesen und interpretiert werden müssen. Bei Shell-Skripten wird vom Kernel das `setuid`- bzw. `setgid`-Bit aus Sicherheitsgründen ignoriert. Schreiben Sie entweder einen kleinen C-Wrapper oder implementieren Sie das Skript mit Hilfe von Perl, das den `setuid`-Mechanismus intern bereitstellt und viele Sicherheitsprüfungen durchführt.

Programme, die mit `setuid` oder `setgid` gestartet werden, sollten nach Möglichkeit für Benutzer nicht lesbar sein, um eventuellen Eindringlingen das Leben nicht zu leicht zu machen. Andernfalls lässt sich z. B. mit `strings` feststellen, welche Dateien gelesen und geschrieben werden, oder das Programm disassemblieren[2].

In Tabelle 5.3 sind die möglichen Berechtigungen für Verzeichnisse mit den zugehörigen Erklärungen aufgeführt. Das `s`-Bit für die Gruppe sorgt dafür, dass alle neu in diesem Verzeichnis angelegten Dateien automatisch dieselbe Gruppe bekommen wie das Verzeichnis. Daher ist dieses Bit im Listing 5.11 aktiviert worden.

Mode	numerisch	Bedeutung
`r`	4	Leseberechtigung
`w`	2	Schreibberechtigung
`x`	1	Wechsel in das Verzeichnis erlaubt
`s`	2	Dateien gehören automatisch zur selben Gruppe
`t`	1	Nur Eigentümer darf löschen (`t`-Bit, sticky-Bit)

Tabelle 5.3 Berechtigungen für Verzeichnisse

Eine Datei kann gelöscht werden, wenn der Benutzer Schreibzugriff auf das entsprechende Verzeichnis hat. Dazu muss er die Datei nicht lesen oder verändern dürfen. Bei den global schreibbaren Verzeichnissen wie `/tmp` wird daher das `t`-Bit gesetzt. Damit kann nur noch der Eigentümer der Datei bzw. der Eigentümer des Verzeichnisses eine Datei löschen.

Im Listing 5.11 wurden die Rechte mit dem Programm `chmod` vergeben. Die genauen Parameter für die Rechte sind in Tabelle 5.4 sowohl in Ziffern als auch in Buchstaben aufgelistet. Je nach Erfahrung und Einsatzzweck wird man mal die numerische, mal die verbale Beschreibung bevorzugen. Mehr Informationen zur Vergabe von Berechtigungen finden Sie in der Manpage zu `chmod(1)`.

2. Da von den meisten Linux-Programmen der Quellcode verfügbar ist, wird man dies sowieso nicht machen :-)

Verzeichnis	numerischer Modus	textuelle Angabe
jochen	**751**	**u=rwx,g=rx,o=x**
projekt	**3771**	**+t,u=rwx,g=wrxs,o+x**

Tabelle 5.4 Setzen von Rechten

Nach diesem Ausflug in das Unix-Berechtigungskonzept kehren wir nun zurück zum Listing 5.11. In der zweiten Spalte wird der Link-Count der Datei oder des Verzeichnisses angegeben. Die meisten Dateien haben den Link-Count 1. Für jeden Hard-Link, der auf diese Datei verweist, wird dieser Wert um 1 erhöht. Verzeichnisse enthalten stets den Eintrag `.`, dies ist ein Hard-Link auf sich selbst. Daher hat jedes Verzeichnis mindestens den Link-Count 2. Ist ein weiteres Unterverzeichnis enthalten, so wird der Link-Count um 1 erhöht, da jedes Unterverzeichnis einen Eintrag `..` enthält, der auf das übergeordnete Verzeichnis verweist. Unter Unix werden keine Dateien gelöscht. Die Befehle `rm` und `unlink` vermindern nur den Link-Count. Solange der Link-Count nicht Null ist oder die Datei noch von einem Programm verwendet wird, werden die Daten nicht von der Platte gelöscht. Das kann dazu führen, dass nach dem Löschen einer großen Log-Datei, die von `syslog` erzeugt wurde, der Platz immer noch belegt ist, solange der `syslog`-Dämon nicht neu gestartet wird (bis dahin ist die Datei noch in Benutzung, auch wenn kein benannter Verzeichniseintrag mehr dafür existiert).

Der Eigentümer der Datei wird im dritten Feld angezeigt. Die Gruppenzugehörigkeit findet man im vierten Feld. Das fünfte Feld gibt die Größe der Datei oder des Verzeichnisses an. Anschließend erfolgt die Ausgabe des Datums der letzten Änderung dieser Datei. Liegt die Änderung nur kurze Zeit zurück (weniger als ein halbes Jahr), so wird die Uhrzeit der Änderung mit angegeben, andernfalls das Jahr. Das letzte Feld enthält den Datei- oder Verzeichnisnamen.

Neben der Zeit der letzten Änderung verwaltet Unix noch zwei weitere Zeitstempel: die Zeit des letzten Dateizugriffs (Access-Time, `atime`) und die Change-Time (`ctime`) des zugehörigen i-Node. Die Change-Time wird beim Erzeugen der Datei, beim Umbenennen oder Verändern von Berechtigungen gesetzt; manchmal wird diese auch fälschlicherweise als Create-Time bezeichnet. Diese drei Zeiten kann man zur Suche nach Dateien verwenden und sich beispielsweise mit dem Programm `stat` anzeigen lassen.

5.7 Standardwerte für Berechtigungen

Neue Dateien gehören immer dem anlegenden Benutzer. Als Gruppe wird die Standardgruppe verwendet, die dem Benutzer in der Datei `/etc/passwd` zugeordnet ist. Die Gruppe, die zur Anlage von Dateien verwendet wird, kann mit

dem Befehl `newgrp` gewechselt werden. Dabei werden Textdateien grundsätzlich mit der `rw`-Berechtigung (`666` bzw. `a=rw`), Programme, die der Compiler oder Linker erzeugt, mit der `rwx`-Berechtigung (`777` bzw. `a=rwx`) versehen. Von diesen Werten werden die Bits entfernt, die in der `umask` gesetzt sind. Bei einer `umask` von `027` wäre die endgültige Berechtigung `640` bzw. `u=rw,g=r,o=` für Textdateien und `750` bzw. `u=rwx,g=rx,o=` für Programme.

Die bestehende Einstellung der `umask` lässt sich mit dem Befehl `umask` anzeigen und verändern. Dabei kann, wenn die `bash` als Shell verwendet wird, auch die symbolische Schreibweise anstelle der umgekehrten Bit-Schreibweise verwendet werden. Bei der Ausgabe der `umask` ist die Option `-S` anzugeben. Bei der Eingabe wird die symbolische Schreibweise automatisch erkannt. Damit entfällt die Berechnung der Bitmask, die zum Ausblenden der nicht gewünschten Rechte benutzt wird. Die im Listing 5.12 angegebenen Schreibweisen sind äquivalent.

```
(linux):~$ umask 027
(linux):~$ umask u=rwx,g=rx,o=
```

Listing 5.12 Beispiele für die Einstellung der `umask`

Auf vielen Systemen sind die Standardberechtigungen so gesetzt, dass andere Benutzer grundsätzlich keinen Zugriff auf neu angelegte Dateien haben. Das hat den Nachteil, dass man eine möglicherweise sinnvolle Kooperation der Benutzer auf seinem System erschwert, da jeder Benutzer selbst auf die Vergabe entsprechender Berechtigungen achten muss. Hier ist es angenehmer, wenn neu angelegte Dateien grundsätzlich für alle anderen Benutzer lesbar sind. Wenn bestimmte Dateien nicht zugreifbar sein sollen, dann können diese in einem extra gesicherten Unterverzeichnis abgelegt werden.

Manchmal hat auch der Benutzer `root` eine `umask`, die dafür sorgt, dass normale Benutzer keine Dateien lesen dürfen, die von `root` bearbeitet wurden. Das mag in einigen Fällen der Sicherheit dienlich sein. Wenn aber beispielsweise die Datei `/etc/passwd` nicht mehr für alle Benutzer lesbar ist, dann zeigt der Befehl `ls -l` keine Benutzernamen mehr an. Auch hier muss man zwischen einer hohen Sicherheit und der einfachen Bedienbarkeit abwägen.

5.8 Berechtigungen und Gruppen im Einsatz

Man kann eine Gruppe auch als eine Rolle betrachten, in der ein Benutzer gerade auftritt. Jedes Mal, wenn man die Rolle wechselt, also z. B. für ein anderes Projekt arbeitet, dann wechselt man auch die (primäre) Gruppe. Das kann z. B. mit dem Befehl `newgrp` erfolgen, aber man muss daran denken, auch die `umask` entsprechend einzurichten. Das Listing 5.13 zeigt ein Beispiel, wie es auf vielen Unix-Systemen üblich ist.

```
(linux):~$ id
uid=40(jochen) gid=6(users) groups=6(users),7(projekt)
(linux):~$ umask
22
(linux):~$ newgrp projekt
(linux):~$ id
uid=40(jochen) gid=6(projekt) groups=6(users),7(projekt)
(linux):~$ umask 002
(linux):~$ cd /home/projekt
... hier am Projekt arbeiten
(linux):~$ exit
```

Listing 5.13 Verwendung von supplementären Gruppen unter Unix

Die Red Hat-Distribution verwendet das Konzept der »User Private Groups«. Jedem Benutzer wird als primäre Gruppe eine eigene, am besten mit demselben Namen wie seine Benutzerkennung benannte, Gruppe zugeordnet. Diese symbolisiert, dass der Benutzer zunächst in der Rolle des privaten Benutzers auftritt.

Als Standard-`umask` wird `002` verwendet. Alle Dateien sind also vom Benutzer und seiner eigenen Gruppe les- und schreibbar. Da jeder Benutzer zunächst eine private Gruppe hat, sind hier also die Rechte an neu angelegten Dateien für andere Benutzer nicht weiter gesetzt als mit anderen Schemata ohne private Gruppe und `umask 022`. Mit diesem Trick kann also die `umask` immer auf einen Wert eingestellt werden, bei dem alle Dateien und Verzeichnisse gruppenschreibbar sind. Seien Sie vorsichtig bei Programmen wie `ssh`, die prüfen, dass sicherheitsrelevante Konfigurationsdateien nur vom Eigentümer verändert werden können. Diesen Programmen fällt nicht auf, dass eine Datei oder ein Verzeichnis, die für eine private Gruppe schreibbar sind, kein Problem für die Sicherheit darstellen. `ssh` mit der Option `-v` gibt eine passende Meldung aus, so dass in diesem Fall das Problem einfach zu lokalisieren ist.

In der Regel wird man Dateien, die von einer Gruppe von Personen in ihrer jeweiligen Rolle gemeinsam bearbeitet werden sollen, in speziellen Verzeichnissen speichern. Wenn man jetzt dafür sorgt, dass dieses Verzeichnis der Gruppe gehört und für diese beschreibbar ist, dann können alle Mitglieder der Gruppe dort Dateien ablegen. Der zweite Trick ist, diesem Verzeichnis das `s`-Bit auf die Gruppe zu geben, so dass alle neu angelegten Dateien automatisch derselben Gruppe wie das Verzeichnis gehören. Das Listing 5.14 zeigt, wie einfach es sein kann, mit »user private groups« zu arbeiten.

```
(linux):~$ id
uid=40(jochen) gid=6(users) groups=6(users),7(projekt)
(linux):~$ umask
2
(linux):~$ cd /home/projekt
```

```
... hier am Projekt arbeiten
(linux):~$ cd
```

Listing 5.14 Verwendung von supplementären Gruppen unter Unix

Mit dieser Methode kann man dafür sorgen, dass jeder Benutzer weiterhin private Dateien ablegen kann. Aber sobald er in einer neuen Rolle auftritt, werden alle Daten in diesem Verzeichnis mit den passenden Berechtigungen gespeichert. Für die normale Arbeit am Shell-Prompt sind diese Einstellungen ausreichend, aber möglicherweise sollten Sie die `umask` des `ftpd`-Programms in der `/etc/inetd.conf` anpassen. Wenn Sie Gruppenverzeichnisse mittels Samba exportieren, dann sollten Sie auch hier die entsprechende `umask` in der Datei `/etc/smb.conf` eintragen.

Ein weiteres Problem kann auftreten, wenn Sie Dateien zunächst in Ihrem Home-Verzeichnis erstellen und erst später in das Gruppenverzeichnis kopieren. Ein Kopieren ohne zusätzliche Parameter funktioniert problemlos, wenn Sie aber die Option `-p` für das Behalten der Änderungszeit und der Berechtigungen angeben, dann wird die Datei der falschen Gruppe zugeordnet und die anderen Mitarbeiter am Projekt haben keinen Schreibzugriff auf diese Datei.

5.9 Weitere Dateirechte

Die bisher vorgestellten Berechtigungen für Dateien erfüllen nicht alle Anforderungen, die an ein System zu stellen sind. Daher wurden im extended2-Dateisystem weitere Zugriffsrechte in Access Control Lists (ACL) implementiert. In diesen Listen können für jede einzelne Datei weitere Optionen zum Dateizugriff festgelegt werden. Tabelle 5.5 enthält eine vollständige Liste der möglichen Attribute in einer ACL. Die Änderung dieser Attribute erfolgt mit dem Programm `chattr`, die Anzeige mit `lsattr`.

Attribut	Wirkung
`a`	Nur Anfügen ist möglich (`APPEND`)
`c`	Datei komprimieren (nicht implementiert)
`d`	Die Datei wird nicht von `dump` gesichert
`i`	Die Datei ist immutable (nicht veränderbar): kein Löschen, Umbenennen, Schreiben und Linken erlaubt
`s`	Sichere Löschung der Datei
`S`	Synchrones Schreiben von Änderungen
`u`	Undelete ermöglichen (nicht implementiert)

Tabelle 5.5 Attribute für Access Control Lists

Mit diesen Mitteln ist es möglich, einzelne Dateien besser und flexibler zu schützen oder gezielter zu bearbeiten. So können Log-Dateien, die vom syslogd-Dämon erzeugt werden, das Attribut a erhalten, um sie vor der Änderung mit einem Editor zu schützen. Dieses Attribut verhindert allerdings die Bearbeitung von Log-Dateien mittels prune oder ähnlichen Programmen. Auch das Attribut S für synchrones Schreiben ist sinnvoll, damit möglichst alle Meldungen sofort auf der Festplatte gespeichert werden. Bei einem Absturz findet man hoffentlich auch die letzte Meldung, die Aufschluss über die Fehlerursache geben könnte.

Andere Unix-Systeme haben zusätzlich zu den Rechten für den Eigentümer, die Gruppe und andere Benutzer so genannte Access Control Lists für einzelne Benutzer oder mehrere Gruppen. In einzelnen Fällen kann es durchaus sein, dass man diese Funktionen vermisst. Derzeit ist mir aber kein Projekt bekannt, das diese Funktionen unter Linux bereitstellen will. Wenn Sie allerdings nur eine Gruppe von Benutzern aus der Verwendung bestimmter Dateien ausschließen wollen, dann können Sie das mit Hilfe der Unix-Berechtigungen erreichen. Ordnen Sie die Datei der entsprechenden Gruppe mit chgrp zu und entfernen Sie die Rechte für die Gruppe, erlauben Sie aber die Zugriffe für alle anderen Benutzer, etwa mit chmod g=,o=rx *Programm*.

5.10 Feiner unterteilte Berechtigungen oder der allmächtige Benutzer root

Unter Unix ist es üblich, dass der einzige, besonders ausgezeichnete Benutzer der Systemverwalter ist. Das ist der Benutzer root, identifiziert durch die Benutzernummer 0. In der Praxis erweist sich dies als viel zu unflexibel, da häufig andere Benutzer durchaus in der Lage sein sollen, einzelne Prozesse (geordnet) zu beenden, Drucker zu steuern oder das System zu starten.

5.10.1 Mehr Rechte mit sudo

Wenn man an der grundsätzlichen Einteilung unter Unix in Systemverwalter und »normale« Benutzer nicht rütteln kann oder will, dann kann man das Programm sudo verwenden. Dieses Programm wird mit Hilfe der Konfigurationsdatei sudoers konfiguriert, in der festgelegt wird, welche Benutzer unter welchen Bedingungen welches Programm ausführen dürfen.

Dieses System hat, bei allen Vorteilen auch einige gewichtige Nachteile. Zunächst einmal die wichtigsten Vorteile:

- Gezielte Rechtevergabe an Benutzer, gesteuert über die Programmnamen
- Protokollierung aller aufgerufenen Befehle mit Hilfe des syslogd
- Keine Passwortabfrage für bestimmte Benutzer oder Benutzergruppen

- Zusätzliche Authentifizierung durch die erneute Eingabe des Passworts des aufrufenden Users
- Caching der Authentifizierung für begrenzte Zeit
- Deutliche Erleichterung des »Schnell-mal-als-`root`-Anmeldens«

Dabei darf man die Nachteile nicht aus den Augen verlieren. Unix und seine Programme sind nicht mit dem Ziel der Rechteverteilung entworfen worden. Daraus ergeben sich auch die gewichtigsten Nachteile:

- Die Berechtigungsprüfung erfolgt in einem Anwendungsprogramm und nicht im Kernel, wo sie eigentlich hingehört.
- Man verlässt sich auf die Sicherheit der erlaubten Programme. Da dies kein Designziel bei der Implementation der Programme war, muss man mit extremen Sicherheitsproblemen rechnen.
- Aus dem letzten Punkt folgt, dass `sudo` nur in einer relativ überschaubaren Umgebung eingesetzt werden kann, ohne zu großen Sicherheitslücken zu führen.
- Es ist nicht sichergestellt, dass die Berechtigungen auf die vergebenen Programme beschränkt bleiben. Es ist also denkbar, dass ein Benutzer sich mit Hilfe eines erlaubten Programms unerlaubt die Rechte zu einem weiteren Programm verschafft.

Die Anwendung von `sudo` ist grundsätzlich sehr einfach. Wenn Sie für einen Befehl die Rechte des Systemverwalters benötigen, dann starten Sie das Programm einfach mit `sudo` *`Befehl`*. Der entsprechende Befehl wird dann mit der Kennung des Systemverwalters ausgeführt. Welcher Benutzer welchen Befehl starten darf, wird in der Datei `/etc/sudoers` (siehe Listing 5.15) festgelegt. Der Systemverwalter kann diese Datei mit dem Befehl `visudo` ändern. Damit wird die Datei gegen konkurrierende Änderungen gesperrt und außerdem einer Syntaxüberprüfung unterzogen.

```
# Host alias specification
Host~Alias      SERVER=server
Host~Alias      CLIENTS=client1,client2

# User alias specification
User~Alias      OPERATOR=michael

# Cmnd alias specification
Cmnd~Alias      LPR=/usr/bin/lprm,/usr/sbin/lpc

# Festlegung der Privilegien je Benutzer
#Wer            Wo=(als~Wer) Was
OPERATOR        SERVER=LPR
%wheel          ALL=(ALL)
```

```
+clientadm      CLIENTS=(ALL)
jochen          ALL=(root,uucp) NOPASSWD: ALL
```

Listing 5.15 Die Datei sudoers

Im ersten Teil der Datei sudoers können Aliasnamen für Rechner und Netze, Benutzer und Kommandos definiert werden, mit denen sich die weitere Konfiguration vereinfachen lässt. Die eigentliche Konfiguration in Listing 5.15 befindet sich in den letzten Zeilen. Die in dem Alias OPERATOR definierten Benutzer dürfen auf den durch den Alias SERVER festgelegten Rechnern die im Alias LPR angegebenen Befehle ausführen. In unserem Beispiel ist das der Benutzer michael, der die Befehle lpc und lprm auf dem Rechner server ausführen darf. Mehrere Befehle, Benutzernamen oder Host-Namen werden durch Kommata voneinander getrennt.

Im nächsten Eintrag wird festgelegt, dass die Mitglieder der Unix-Gruppe wheel alle Befehle auf allen Rechnern verwenden dürfen. Der Name ALL ist ein interner Alias von sudo. In vielen Fällen sind in der Gruppe wheel die Systemverwalter zusammengefasst, da auf manchen Unix-Systemen nur die Mitglieder dieser Gruppe den Befehl su aufrufen dürfen. Der Eintrag +clientadm verweist auf die Netgroup clientadm, in der in diesem Beispiel die Administratoren der Clients zusammengefasst sein könnten. Mehr zu Netgroups finden Sie im Abschnitt 23.6, »Weitere NIS-Anwendungen«.

Der letzte Eintrag in Listing 5.15 erlaubt dem Benutzer jochen die Ausführung aller Befehle als Benutzer root oder uucp, ohne dass ein Passwort angegeben werden muss. Wenn die Systemverwalter diszipliniert sind, dann kann dieses Verfahren eine echte Erleichterung bei der Administration sein.

Wird sudo ohne weitere Parameter aufgerufen, dann wird auf die Benutzerkennung root umgeschaltet. Mit der Option -u kann, sofern erlaubt, auch ein anderer Benutzer angegeben werden.

Beim ersten Start wird sudo normalerweise nach einem Passwort fragen. Einzugeben ist nicht das root-Passwort, sondern das eigene Passwort. Schließlich soll sudo oft nur Teile der root-Rechte ermöglichen, so dass die entsprechenden Anwender nicht das Passwort des Systemverwalters kennen sollten. In der Standardkonfiguration merkt sich sudo für fünf Minuten, dass die Eingabe des Passworts erfolgt ist, so dass nicht bei jedem Aufruf das Passwort erneut eingegeben werden muss. Mit dem Befehl sudo -k wird dieser Merker gelöscht, so dass beim nächsten Aufruf wieder die Eingabe eines Passworts erforderlich ist.

Der Befehl sudo -l zeigt die erlaubten Befehle an. sudo schreibt mittels syslog ein Protokoll der eingegebenen Befehle und Parameter. Als Standard wird die Facility local2 verwendet, bei der Übersetzung kann in der Datei options.h eine andere Kategorie eingestellt werden. Die weiteren Parameter und möglichen

Einstellungen in der Datei `sudoers` entnehmen Sie bitte den Manpages `sudo(8)` und `sudoers(5)`.

Am Ende des Kapitels finden Sie in der Info-Box die URL dieses Programms. Ich setze es selbst in einer überschaubaren Umgebung ein, halte aber die dadurch aufgeworfenen Sicherheitsprobleme für nicht vernachlässigbar. Was man also braucht, ist ein überdachtes und verbessertes, auf die Anforderungen aus der Praxis angepasstes Berechtigungskonzept.

5.10.2 Capabilities unter Linux

Andere Systeme, wie Windows NT oder VMS, kennen eine ganze Reihe von verschiedenen Berechtigungen, die einzeln vergeben und wieder entzogen werden können. Unter VMS kann ein Benutzer sogar das Recht haben, sich bestimmte Berechtigungen zu verschaffen. Diese sind aber bei einer normalen Anmeldung nicht aktiv. Dadurch ist das System wesentlich besser kontrollierbar.

Im Rahmen der POSIX-Standardisierung ist ein spezieller Standard für die Unix-Sicherheit in der Entwicklung. Dieser Standard ist als POSIX.1e (früher POSIX.6) bekannt und auch nach vielen Jahren immer noch nicht verabschiedet. Schlimmer noch, die Arbeit daran wurde eingestellt.

Der Standard definiert eine Reihe von Security-Schnittstellen, wie z. B. Auditing, Access Control Lists oder die Aufteilung der `root`-Privilegien. Unter Linux existiert ein Projekt (Linux-Privs), das diese Features im Kernel und den entsprechenden User-Programmen implementiert. Noch sind diese Funktionen in der Entwicklung, ich hoffe aber, dass sie baldmöglichst Einzug in die Distributionen halten.

Im Kernel wird an vielen zentralen Stellen nicht mehr geprüft, ob der Benutzer der Superuser `root` ist, sondern abgefragt, ob der Benutzer eine bestimmte Berechtigung (capability) besitzt. Das ist, solange der Rest des Systems noch nicht entsprechend angepasst ist, für den Systemverwalter der Fall. Im Wesentlichen scheint die Arbeit an Capabilities im Linux-Kernel abgeschlossen zu sein.

Im nächsten Schritt ist das gesamte System, d. h. die Anwendungen, entsprechend umzustellen. Alte Anwendungen, die nicht für Capabilities angepasst wurden, sollen ohne Probleme weiter funktionieren. Neue Anwendungen sollten diese Funktionen dann direkt nutzen. Unter Linux sind nicht nur die POSIX-Capabilities verfügbar, es wurden außerdem viele Linux-spezifische Berechtigungen eingeführt.

Capabilities sind einmal als Zusatz zu einer Programmdatei gespeichert (die genaue Form der Speicherung ist noch umstritten). Wenn ein Programm neu ausgeführt wird (mit dem `exec`-Systemaufruf), dann werden die Capabilities dieser Datei wie folgt berücksichtigt:

- *inheritable capabilities* geben an, welche Capabilities überhaupt vererbt werden können. Für die meisten Programme wird man hier fast alle Berechtigungen ausschließen, so dass auch ein Sicherheitsproblem eines anderen Programms nicht jedes Sicherheitskonzept aushebeln kann.
- *permitted capabilities* werden beim Ausführen des Programms automatisch aktiviert.
- *effective capabilities* sind entweder vollständig gesetzt oder leer und werden für ältere Programme aktiviert.

Außerdem sind Capabilities ein Merkmal des aktuellen Prozesses. Die tatsächlich aktiven Capabilities ergeben sich durch die Verknüpfung der Dateirechte mit denen der Task.

Auf der einen Seite kann man die Rechte viel feiner verteilen, was sicher ein großer Vorteil für größere Installationen oder sehr sicherheitsbewusste Anwender ist. Auf der anderen Seite greifen diese Anpassungen tief in das System ein und die Rechteverwaltung wird wesentlich komplexer. Ob sich diese Anpassungen auf breiter Front durchsetzen können, ist fraglich, es hängt sehr viel von der Oberfläche der Programme und der Dokumentation ab.

5.11 Vergleich von Linux mit anderen Systemen

Noch sind die Linux-Implementationen von ACLs und Capabilitites nicht im Standard enthalten. Andere Unix-Systeme und Windows NT sind da eine ganze Ecke weiter. Oder auch nicht, denn in der Praxis verwendet man auch bei kommerziellen Unix-Systemen fast ausschließlich die normalen Dateirechte. ACLs sind auch dort eher unhandlich und man muss die beteiligten Tools, wie zum Beispiel Backup-Systeme, gezielt überprüfen.

In den meisten Fällen werden Sie auch ohne ACLs auskommen, für exponierte Server ist allerdings das Capability-Konzept eine interessante Sache.

6 Der Editor Emacs

Emacs Makes All Computing Simple
Eine Auflösung des Akronyms Emacs

6.1 Allgemeines zum Emacs

In der Dokumentation zum Emacs wird dieser als erweiterbarer, anpassbarer, benutzerfreundlicher, selbst dokumentierender Editor bezeichnet. Das sind große Worte, die aber, verglichen mit anderen Editoren, nicht zu viel versprechen. Das folgende Kapitel wird einige Funktionen des Emacs vorstellen, die ihn zu etwas Besonderem machen.

Der Editor Emacs wurde ursprünglich von Richard M. Stallman, dem Gründer der Free Software Foundation (FSF) und Begründer der GNU-Idee geschrieben. Die Anfänge dieses Editors reichen bis auf Vor-Unix-Zeiten zurück: Der erste Emacs (Editor MACroS) war ein Makro-Paket für TECO (Text Editor and COrrector) und lief auf der legendären PDP-11 von DEC.

Aus diesem Editor hat sich eine ganze Familie von Emacs-artigen Editoren entwickelt, die sowohl kommerziell als auch frei vertrieben werden. Das bekannteste Mitglied dieser Familie ist der GNU-Emacs, der immer noch von seinem Autor, Richard M. Stallman, weiterentwickelt wird. Außerdem gehören z. B. XEmacs, der aus einer Vorabversion des GNU-Emacs 19 entwickelt wurde, `jed`, MicroEmacs oder `jove` dazu.

Emacs ist recht groß und auf manchen Rechnern (insbesondere solchen mit wenig Speicher und CPU-Leistung) relativ langsam. Auf solchen Rechnern hat der Einsatz eines Emacs-Clone durchaus seinen Sinn. Allerdings unterscheidet sich die Bedienung in vielen Details vom Original, so dass ein ständiger Wechsel zwischen zwei Emacs-artigen Editoren recht anstrengend ist. Außerdem sind viele Lisp-Pakete einfach nicht einsetzbar, wenn man nicht (X)Emacs benutzt.

Viele Betriebssystem- und Computerhersteller liefern Emacs als zusätzliches Software-Paket mit. Andernfalls kann der Emacs auf praktisch jedem verbreiteten Unix-System selbst übersetzt werden, da er wie alle GNU-Programme mit `./configure` automatisch konfiguriert wird. Emacs verlangt viel vom Betriebssystem, sowohl bei der Übersetzung als auch bei der Ausführung – dennoch ist er recht portabel.

Der Editor Emacs ist sehr mächtig; tatsächlich ist er ein Lisp-Interpreter, der es versteht, mit Texten umzugehen. Der Lisp-Dialekt von Emacs ist ein (vollwertiges) Lisp, das Emacs-Lisp (kurz elisp). Bei der Ausführung von Makros wird der

unbedarfte Benutzer deshalb z. B. auch von Meldungen wie »Garbage collecting...« überrascht.

Diese Meldung teilt dem Benutzer mit, dass eine interne Funktion des Lisp-Interpreters abläuft, die nicht mehr benutzten Speicherplatz wieder freigibt. Da Lisp keine explizite Anforderung und Freigabe von Speicher kennt, wird diese Funktion regelmäßig (bzw. bei Speicherbedarf) aufgerufen.

Es gibt unzählige Anpassungen, Erweiterungen und Makros für den Emacs, die in elisp geschrieben sind. Diese Makros werden in Dateien gespeichert, die auf `*.el` enden. `*.elc`-Dateien sind entsprechende (Byte-)Kompilate, die vom Lisp-Interpreter schneller ausgeführt werden können. Diese Lisp-Programme, zusammen mit der wirklich ausführlichen und gut verständlichen Dokumentation, sorgen für den nicht gerade kleinen Platzbedarf des Emacs.

6.2 Welche Emacs-Version soll ich nehmen?

Die Familie der Emacs-Editoren besteht aus beinahe unüberschaubar vielen Varianten und mehr oder weniger kompatiblen Clones. Diese Vielfalt macht die Auswahl einer Version relativ schwierig, aber es gibt stets gute Gründe für die eine oder andere Wahl.

Als wohl am weitesten verbreitete Version findet man den GNU-Emacs, der direkt von der FSF unter Federführung von Richard Stallman entwickelt wird. Diese Version enthält eine recht gute Auswahl an zusätzlichen Lisp-Paketen und läuft sowohl unter X als auch auf einem Terminal. Unter X können verschiedene Fonts verwendet werden, Farben gibt es jetzt auch auf einem Terminal. Texte können, wenn dies gewünscht wird, im Rich-Text-Format mit Attributen abgespeichert werden. Aktuelle Versionen können mit Fonts mit variabler Laufweite umgehen und Emacs soll sich in Zukunft immer mehr in Richtung einer Textverarbeitung weiterentwickeln.

Die härteste Konkurrenz ist der XEmacs. Diese Version wurde zunächst für ein besseres X-Interface entwickelt, kann jetzt jedoch auch an einem Terminal verwendet werden. Spezielle Funktionen sind hier die Verwendung von Farben auf Terminals und unter X sowie die Möglichkeit, einen XEmacs-Prozess sowohl von einem Terminal aus als auch unter X anzusprechen. Ein weiterer Vorteil ist, dass mehr Lisp-Pakete als zum Emacs mitgeliefert werden. Diese Pakete sind außerdem besser strukturiert gespeichert, so dass es einfacher ist, im Quellcode eines Pakets zu navigieren, das aus mehreren Dateien besteht.

Ein Nachteil des XEmacs ist, dass der Lisp-Interpreter und einige Funktionen nicht kompatibel zum GNU-Emacs sind – XEmacs ist halt nicht der »Standard-Emacs«. Daher laufen manche Lisp-Erweiterungen nur mit dem GNU-Emacs

oder einer älteren Version des XEmacs. XEmacs benötigt deutlich mehr Speicher und mehr CPU-Leistung als der GNU-Emacs. Die größere Leistungsfähigkeit fordert hier ihren Tribut.

Ich persönlich bevorzuge den XEmacs aufgrund der besseren Ausstattung mit elisp-Paketen und der bereits in älteren Versionen enthaltenen Fähigkeit, verschiedene Fonts (unter X) und verschiedene Farben auf der Konsole darstellen zu können – heute hat GNU Emacs aufgeholt. Das ist aber eine persönliche Vorliebe und man muss die zusätzlichen Funktionen gegen die größeren Hardware-Anforderungen abwägen.

Wenn im Folgenden vom Emacs gesprochen wird, dann sind damit sowohl der GNU-Emacs als auch der XEmacs gemeint. In den meisten Fällen verhalten sich diese beiden sehr ähnlich oder identisch, so dass eine Unterscheidung nicht notwendig ist. An Stellen, wo sich diese beiden Versionen für den Benutzer oder die Konfiguration unterscheiden, wird darauf aber eingegangen.

Außerdem existiert eine ganze Reihe von (einfachen) Emacs-Clones. Diese Programme haben in der Regel den Vorteil, dass sie wesentlich weniger Platz auf der Festplatte oder im Hauptspeicher benötigen. Dafür vermisst man einige Features des »echten« Emacs:

- Manche Clones haben keine integrierte Programmiersprache, andere verwenden eine eigene Sprache und wieder andere eine abgespeckte und inkompatible Lisp-Version.
- Die Konfiguration erfolgt mit Hilfe anderer Dateien und einer anderen Syntax.
- Es gibt bei weitem nicht so viele Erweiterungen wie für den GNU-Emacs.

Immer wenn auf einem Rechner der Plattenplatz oder die verfügbare CPU-Leistung begrenzt ist, dann kann es sinnvoll sein, zumindest einen Emacs-Clone einzusetzen. Für den unbedarften Benutzer sind diese Programme wesentlich einfacher zu bedienen als z. B. der `vi`. Gerade ein Rechner, der von mehreren Benutzern gemeinsam verwendet wird, gerät bei der Verwendung des Emacs durch den doch recht großen Speicherbedarf leicht ins Swapping und wird damit unerträglich langsam.

6.3 Kompilieren des GNU-Emacs

Obwohl der GNU-Emacs natürlich im Quellcode vorliegt (er wird unter der GPL vertrieben), gibt es nur wenige Menschen, die ihn tatsächlich unter Linux selbst übersetzen. Das liegt zum einen am Umfang des Quellcodes, zum anderen existieren fertig kompilierte Binärversionen für viele Systeme.

Emacs ist nur zu einem relativ kleinen Teil in C geschrieben. Das sind der Lisp-Interpreter und die wichtigsten Operationen mit Texten, die so genannten Primitiven. Dadurch sind die zentralen Funktionen zur Textmanipulation vergleichsweise schnell.

Alle weiteren Funktionen werden in Emacs-Lisp implementiert. Durch diese Hochsprache ist es relativ einfach möglich, den Emacs an die eigenen Bedürfnisse anzupassen oder eigene Lisp-Erweiterungen zu entwickeln. Die direkte Ausführung der Lisp-Programme durch den Interpreter ist relativ langsam, daher werden fast alle Lisp-Dateien mit dem »Byte-Compiler« bearbeitet. Der dadurch erzeugte Byte-Code ist für den Computer einfacher (und damit schneller) zu interpretieren.

Da Emacs-Lisp eine interpretierte Sprache ist und Lisp-Funktionen zur Laufzeit definiert und geändert werden können, ist es mit etwas Übung sehr leicht, eigene Anpassungen vorzunehmen oder Erweiterungen zu entwickeln. In diesem Kapitel werden wir einige Beispiele für Anpassungen sehen.

Emacs wird wie die anderen GNU-Programme mit dem Befehl `./configure` an das System angepasst und kann dann mit `make` übersetzt und mit `make install` installiert werden. Viele Optionen können mit `configure` eingestellt werden. Einige wichtige möchte ich hier vorstellen, die vollständige Liste finden Sie in der Datei `INSTALL`:

- `--prefix` bestimmt das Installationspräfix. Die Binärversionen in den Distributionen wurden meist mit `--prefix=/usr` übersetzt. Der Standardwert ist `/usr/local`.
- `--with-x-toolkit` bindet die X-Oberfläche, sofern auf dem Rechner vorhanden, ein, so dass unter X ein komfortableres Arbeiten (z. B. mit mausbedienbaren Menüs, weiteren Mausfunktionen und Syntax-Highlighting) möglich ist. Sinnvoll ist die Verwendung von `--with-x-toolkit=lucid`. Es werden hiermit bedienbare und gut aussehende Menüs eingebunden. Ohne den Zusatz `=lucid` würde das X-Toolkit verwendet werden, wo die Menüs nicht besonders gut aussehen.
- `--with-jpeg` und `--with-png` zur Unterstützung von Grafiken im JPEG- und PNG-Format.

Sollten Sie Probleme bei der Übersetzung von (X)Emacs haben, so werfen Sie einen Blick in die Dateien `./PROBLEMS` und `./etc/MACHINES`. Dort sind viele Probleme und deren Lösung für verschiedene Rechnertypen und Betriebssysteme dokumentiert. Einige ältere C-Compiler hatten echte Probleme, Emacs zu übersetzen. Teilweise ließen diese Probleme sich beheben, indem man einzelne Module mit oder ohne Optimierung kompilierte, bei anderen waren Änderungen im Quelltext erforderlich. In der Regel sind die Editoren der Emacs-Familie aber sehr einfach zu installieren.

Zunächst wird der C-Kern übersetzt. Anschließend wird ein minimaler Emacs (`temacs`) gestartet, der die wichtigsten Lisp-Erweiterungen lädt. Dieses Programm wird dann aus dem Speicher auf Platte »gedumpt«. Dadurch ist das Laden von Emacs schneller, als wenn bei jedem Start zunächst der Interpreter und dann die wichtigsten Lisp-Pakete geladen würden. Zusätzliche Lisp-Pakete können geladen werden, wenn diese in der Datei `./lisp/site-init.el` angegeben sind.

Nach der Installation in das entsprechende Verzeichnis ist der Emacs einsatzbereit. Können oder wollen Sie Emacs nicht in ein Systemverzeichnis installieren, so können Sie beim `configure` die Option `--run-in-place` angeben und Emacs direkt aus dem Übersetzungsverzeichnis heraus ausführen. Dies kann sinnvoll sein, wenn Sie eine neue Version testen möchten oder diese nicht systemweit installieren können oder wollen.

Alle Konfigurationsoptionen für die Übersetzung von Emacs sind in der Datei `INSTALL` dokumentiert. Zusätzlich dazu finden Sie für viele Rechner- und Betriebssystemvarianten weitere Informationen, Tipps und Bugfixes in der Datei `./etc/MACHINES`. Darüber hinaus enthält das `./etc`-Verzeichnis noch eine Reihe weiterer interessanter Dateien, wie die Emacs-FAQ, weitere Dokumentationen zu Emacs und zum GNU-Projekt und einige nicht ganz ernst gemeinte Manpages.

6.4 Allgemeines zur Arbeit mit Emacs

Emacs ist ein »selbst erklärender« Editor. Auch ohne ein gedrucktes Handbuch und einen Guru in Ihrer Nähe können Sie mit Emacs sinnvoll arbeiten. Emacs zeigt Ihnen beim Start (Abbildung 6.1), wie Sie weitere Hilfe erhalten können. Sie können die Dokumentation direkt im Emacs lesen und in den Puffern viele Emacs-Kommandos (z. B. zum Suchen) verwenden. Die Dokumentation ist zwar in englischer Sprache, aber gut lesbar geschrieben und sehr ausführlich.

Sie können außerdem die im `texinfo`-Format geschriebene Dokumentation selbst ausdrucken oder bei der FSF bestellen. Texte in diesem Format können am Bildschirm angezeigt werden (z. B. mit dem `info-mode`), als Buch gedruckt oder auch nach HTML konvertiert werden. Es muss von den Autoren nur ein Dokument gepflegt werden, die Inhalte der verschiedenen Medien sind konsistent und die gedruckte Version ist ordentlich gesetzt und gut lesbar.

Mit Strg-h, t starten Sie das Tutorial, das Sie in die wichtigsten Begriffe und Funktionen des Emacs einführt. Das Durcharbeiten des Tutorials sollte nicht länger als eine halbe Stunde dauern, danach können Sie Emacs bequem für einfache Editiertätigkeiten verwenden. Eine deutschsprachige Version des Tutorials ist bei GNU-Emacs und XEmacs enthalten.

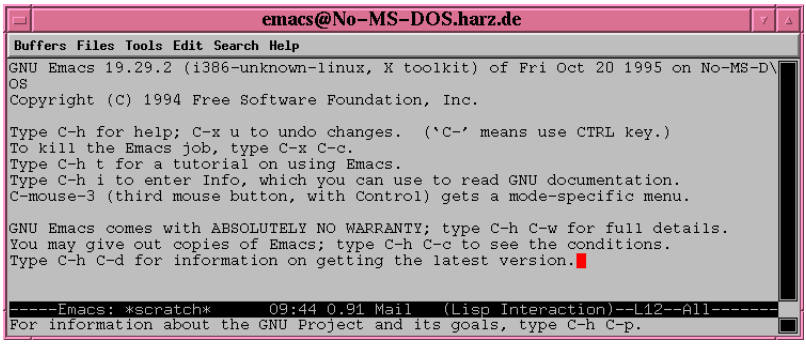


Abbildung 6.1 Der Begrüßungsbildschirm des Emacs

Die Info-Dokumentation zu Emacs und praktisch allen GNU-Programmen können Sie mit Strg-h, i ansehen. Diese Darstellung wurde aus denselben Dateien erzeugt wie das gedruckte Handbuch. Bei der Anzeige werden die Querverweise zur Navigation in der Dokumentation benutzt. Wenn Sie Emacs unter X benutzen, so können Sie mit der mittleren Maustaste einzelnen Verweisen im Hypertext folgen.

6.5 Konzepte und Begriffe

Emacs verwaltet eine geladene Datei in einem *Puffer*. Puffer werden aber auch intern verwendet, um z. B. eine Auswahlliste anzuzeigen. Ein Puffer wird in einem *Window* angezeigt. Ein Window ist ein Ausschnitt aus einem Emacs-Fenster (unter X) oder ein Teil des Bildschirms, der einen Puffer darstellt. Unter X kann Emacs mehrere Fenster (*Frame* genannt) öffnen. In Abbildung 6.2 sehen Sie zwei Emacs-Frames, wobei der vordere Frame zwei Windows enthält.

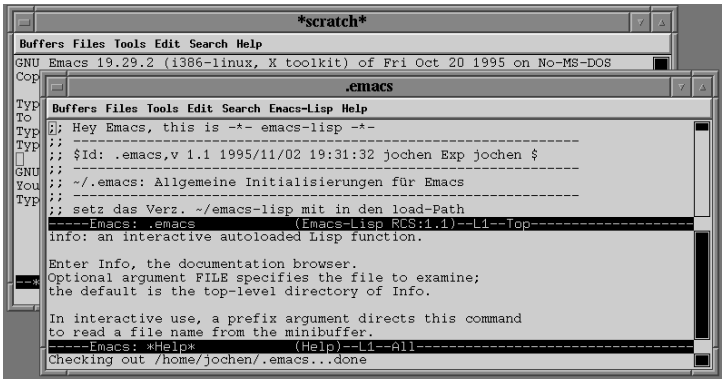


Abbildung 6.2 Emacs-Frames und -Windows

Innerhalb eines Puffers gibt es zwei wichtige Positionen: die Position des *Point* (zwischen dem Zeichen, auf dem der Cursor steht, und dem folgenden Zeichen) und die Position der *Mark* (einer Marke). Der Bereich zwischen Point und Mark heißt *Region*. Auf der Region (einem markierten Bereich) können viele Befehle ausgeführt werden.

Standardmäßig zeigt der GNU-Emacs die gesetzte Region nicht an. Wenn Sie möchten, dass die Region optisch dargestellt wird, so können Sie den Transient Mark Mode aktivieren (Meta-x `transient-mark-mode`). Der XEmacs hingegen zeigt die Region an, im Gegensatz zu GNU-Emacs auch auf der Textkonsole. Sie können die Darstellung der aktiven Region unter GNU-Emacs außerdem durch die in Listing 6.1 angegebenen Variablen Ihren Wünschen anpassen.

```
(transient-mark-mode t)
(set-face-background 'region "red")
(set-face-foreground 'region "yellow")
```

Listing 6.1 Verändern der Darstellung einer Emacs-Region

Zunächst konnte man Emacs nur mit Hilfe von Initialisierungsdateien in Emacs-Lisp konfigurieren. Das war für Hacker, die wie Richard Stallman aus der KI-Forschung kamen und meistens Lisp beherrschten, eine gute Idee. Für unbedarfte Benutzer wirkt Lisp auf den ersten Blick ungewohnt und häufig sogar abschreckend.

Die aktuellen Emacs-Versionen können neben der Lisp-Programmierung mit Hilfe der `customize`-Funktion angepasst werden. Die entsprechenden Einstellungen werden dann in der Datei `~/.emacs` (bzw. `~/.xemacs/custom.el` für XEmacs) abgespeichert und beim nächsten Start von Emacs eingelesen. Ich persönlich bevorzuge die Konfiguration mit Emacs-Lisp, da ich die entsprechenden Einstellungen kommentieren, je nach Modus in verschiedenen Dateien speichern und die Dateien mittels einer Versionsverwaltung organisieren kann.

Unter jedem Window finden Sie die *Modeline*, in der Informationen zur aktuellen Datei und zu den eingestellten Modi angezeigt werden. Dort finden Sie den Namen der geladenen Datei oder den Namen des Puffers, die verwendeten Modes und viele andere Informationen. So stehen z. B. am linken Rand die Zeichen `**` für einen modifizierten Puffer, die Zeichen `%%` für einen schreibgeschützten. Der Inhalt der Modeline kann (natürlich) auch den Bedürfnissen des Benutzers angepasst werden, indem die Variable `modeline-format` verändert wird. Die Beschreibung dieser Variable erhalten Sie mit Strg-h v `modeline-format`.

In der letzten Zeile des Frames findet die Kommunikation mit dem Benutzer über den *Minibuffer* statt. Hier werden Meldungen ausgegeben (wie in Abbildung 6.2 dargestellt) oder Eingaben vom Benutzer verlangt. Wenn eine Funktion mit Hilfe der Maus aus einem Menü heraus aufgerufen wird, dann wird unter X

eine Dialogbox geöffnet. Damit ist der Emacs einerseits durch Mausbedienung relativ komfortabel, andererseits können erfahrene Benutzer den Editor nur mit Hilfe der Tastatur bedienen und sehr effizient arbeiten.

Unter X kann Emacs zusätzlich zu den Tastenkombinationen auch über die in Abbildung 6.1, dargestellten Menüs bedient werden. Neuere Versionen des GNU-Emacs verfügen auch auf einem Text-Terminal über ein Menü, das Sie mit der F10-Taste (`tmm-menubar` oder Meta-`) aktivieren können. Leider ist die Bedienung dieser Menüs vollkommen anders, als man es von anderen Systemen gewöhnt ist. Wenn Sie das Menü ein- oder ausschalten möchten, rufen Sie die Funktion `menu-bar-mode` auf. XEmacs stellt auf einem Text-Terminal keine Menüs dar und ist dort nur mit der Tastatur zu bedienen.

Für viele Dateiformate hat Emacs eine spezielle Unterstützung mit einem Emacs-*Mode*, der zum Beispiel zusätzliche Tastenbelegungen und Funktionen bereitstellt. Modes existieren für praktisch alle verbreiteten Programmiersprachen, aber auch zur Durchführung der unterschiedlichsten Aufgaben. Jeder Puffer befindet sich in genau einem *Major-Mode*, wie z. B. `text` oder `emacs-lisp`.

Außerdem gibt es den *Minor-Mode*, der das Verhalten des Puffers weiter beeinflussen kann. Häufig verwendete Minor-Modes sind `auto-fill` (für automatischen Zeilenumbruch) oder `font-lock` (zur farblichen Hervorhebung von Kommentaren, Strings oder Schlüsselwörtern). In einem Puffer können mehrere verschiedene Minor-Modes aktiv sein. Für mich sind die vielen in der Regel sehr guten Modes *das* Argument für die Verwendung des Emacs: Man lernt den Umgang mit einem Editor und kann diesen für alle Aufgaben einsetzen. Das ginge nicht, wenn jedes Programm seinen eigenen Editor mitbrächte.

Auf der anderen Seite hat man aber die Vorteile eines speziellen Editors durch die Modes. Damit ist Emacs für viele Anwender zu einem Werkzeug für alle Fälle geworden.

6.6 Aufruf und Kommandozeilenoptionen

Wie jedem Editor kann dem Emacs beim Aufruf der Name einer oder mehrerer Dateien als Argument zum Editieren übergeben werden. Diese Dateien werden geladen und in verschiedenen Windows angezeigt. Wenn der Emacs ohne Argumente und Optionen aufgerufen wird, startet er standardmäßig mit der in Abbildung 6.1 dargestellten Meldung, die den Benutzer über das Tutorial, die Hilfe und die Lizenzbedingungen informiert.

Emacs kann sowohl auf einem Text-Terminal als auch unter X verwendet werden. Wird Emacs unter X gestartet, so wird standardmäßig das X-Interface verwendet. Möchte man dennoch das Text-Interface verwenden, so muss man Emacs mit der Option `-nw` starten. Soll ein anderes X-Display verwendet werden

als durch die DISPLAY-Variable vorgegeben, so kann dieses nach der Option --display (bzw. -d) angegeben werden.

Wenn der Emacs mit einem Verzeichnis als Argument gestartet wird, geht er automatisch in den dired-Modus (der Puffer enthält so etwas wie die Ausgabe von ls -la). In diesem Modus können Dateien geladen, gelöscht oder verschoben werden. Mehr zum dired-Mode finden Sie im Abschnitt 6.13.3, »Der dired-mode«.

Es können als Argumente auch beim Starten auszuführende Funktionen, z. B. via -f ispell-change-dictionary, übergeben werden. Durch entsprechende Parametrisierung kann Emacs auch im Batch-Modus, d. h. ohne Benutzerinteraktion, betrieben werden, z. B. um aus einer Textdatei und einer Datei mit aufgezeichneten Benutzeraktionen (einzelnen Tastendrücken) eine neue Datei zu erzeugen. Dies kann z. B. sinnvoll sein, wenn eine Editiersitzung abgestürzt ist.

Der Emacs kann auch eine Zeilennummer als Argument erhalten (z. B. +10) und würde in der entsprechenden Datei zur angegebenen Zeile springen. Dies wird z. B. von Programmen wie elm oder tin benutzt, um den Cursor auf den Beginn des Nachrichtentextes zu positionieren. Wenn man programmiert und einen Syntaxfehler in Zeile 1437 hat, kann man mit dieser Option schnell in diese Zeile springen. Innerhalb von Emacs geht das mit der Funktion goto-line, die unter XEmacs auch mit Meta-g erreichbar ist. Kompiliert man im Emacs mit der Funktion Meta-x compile, so kann man mit der Tastenkombination Strg-x ` zum nächsten Fehler springen.

Mit der Option -q kann das Laden der eigenen Konfigurationsdatei (~/.emacs) verhindert werden, wenn dort z. B. ein Fehler enthalten ist, der zunächst berichtigt werden muss. Auch für einen Bug-Report ist es sinnvoll, keine eigene Konfiguration zu verwenden, um den Fehler mit den Originaleinstellungen reproduzieren zu können.

6.7 Bedienung: Escape, Meta, Alt, Control und Shift

Eine weitere Auflösung der Abkürzung Emacs ist *Escape, Meta, Alt, Control und Shift*. Dies spielt auf die Bedienung des Editors mittels Tastenkombinationen an. Neben vielen Tastenkombinationen können praktisch alle Funktionen auch mit den ausführlichen Namen der zugehörigen Emacs-Lisp-Funktion aufgerufen werden. Wenn Sie Emacs unter X oder den GNU-Emacs verwenden, dann können Sie einige Funktionen auch aus Menüs auswählen. Einsteiger werden häufig das Menü verwenden. Erfahrene Benutzer erlernen im Laufe der Zeit viele Befehle und Tastenkombinationen und steigern damit ihre Arbeitsgeschwindigkeit. Die zusammen mit dem Emacs gelieferte Referenzkarte enthält die wichtigsten Befehle – sie ist auch in deutscher Sprache erhältlich.

Wenn im Folgenden Tastenkombinationen, d. h. das Drücken von zwei Tasten gleichzeitig oder kurz nacheinander, dargestellt werden, geschieht dies in der Form Strg-x oder Meta-<. Strg-x bedeutet: Drücke die Strg-Taste und die x-Taste gleichzeitig, d. h. Steuerungstaste drücken und x-Taste kurz antippen. Die Schreibweise Meta-< steht für das Drücken der Meta-Taste und der <-Taste. Die *Meta-Taste* kann auch mit Alt oder âç beschriftet sein.

Ist keine Meta-Taste vorhanden (oder funktioniert sie nicht, weil sie nicht entsprechend konfiguriert ist) kann die Esc-Taste benutzt werden. Bei der Esc-Taste ist jedoch zu beachten, dass die Taste nur kurz gedrückt werden darf (wie zur Erzeugung eines einzelnen Buchstabens) und danach die zweite Taste der Tastenkombination folgen muss.

Die Tabelle 6.1 enthält eine kurze Übersicht über die wichtigsten Tastenkombinationen. Weiterhin existiert eine Referenzkarte, die Sie in der Datei `/usr/share/emacs/`*`Version`*`/etc/refcard.tex` finden. Die deutsche Übersetzung der Referenzkarte finden Sie in der Datei `/usr/share/emacs/`*`Version`*`/etc/de-refcard.tex`

Sparte	Tasten	Beschreibung
Exit	Strg-x Strg-c	Emacs verlassen
Sichern	Strg-x Strg-s	Pufferinhalt speichern
Ausführen	Meta-x	Lisp-Funktionen ausführen
Cursor bewegen	Strg-n, Strg-p	Nächste bzw. vorherige Zeile
	Strg-f, Strg-b	Nächste bzw. vorherige Spalte
	Strg-a, Strg-e	Anfang bzw. Ende der Zeile
	Meta-a, Meta-e	Satzweise vor bzw. zurück
Muffe-Taste	Strg-g	Letzten bzw. laufenden Befehl abbrechen
Undo	Strg-x u	Letzten Befehl rückgängig machen
Hilfe	Strg-h	
Quoted-Insert	Strg-q	Spezielle Zeichen einfügen
Präfix	Strg-u	Nächsten Befehl ohne Parameter starten
Fenster-Befehle	Strg-x 2	Aktuelles Fenster horizontal teilen
	Strg-x 1	Andere Fenster löschen
	Strg-x 0	Aktuelles Fenster löschen
	Strg-x o	Zum nächsten Fenster wechseln
Suchen	Strg-s	Inkrementelle Vorwärtssuche
	Strg-r	Inkrementelle Rückwärtssuche
	Meta-%	Suchen und Ersetzen
Makros	Strg-x [	Aufzeichnung beginnen
	Strg-x]	Aufzeichnung beenden
	Strg-x e	Letztes Makro ausführen

Sparte	Tasten	Beschreibung
Kill und Yank	Strg-k	Bis Zeilenende löschen
	Strg-w	Region löschen
	Strg-y	Zuletzt Gelöschtes einfügen

Tabelle 6.1 Wichtige Emacs-Tastaturkürzel

Neben diesen Tastenkombinationen können Sie viele Befehle auch direkt mit der Tastenkombination Meta-x (`execute-extended-command`) aufrufen. Anschließend können Sie im Minibuffer den entsprechenden Befehl eingeben. Dabei steht, wie z. B. beim Laden von Dateien auch, eine Erweiterungs- und History-Funktion (Tab / Space bzw. Meta-p und Meta-n oder die Cursor-Tasten) zur Verfügung.

Im Minibuffer komplettiert Space (in der Emacs-Dokumentation als SPC geschrieben) die Eingabe bis zum nächsten Bindestrich. Die Tab-Taste komplettiert bis zur nächsten Mehrdeutigkeit, die Enter-Taste (RET) komplettiert wie Space, führt jedoch einen vollständigen Befehl sofort aus.

Die Funktions- und Variablennamen innerhalb von Emacs sind ausführlich und sprechend, was aber die Benutzung des Emacs nicht beeinträchtigt. Vielmehr ist die Dokumentation sehr gut, so dass man mittels `apropos` die gesuchten Funktionen finden kann, und durch die Vervollständigungsfunktionen ist kein großer Tipp-Aufwand nötig. Im Folgenden werden zusätzlich zu den Tastenkombinationen stets die entsprechenden Befehle in Klammern angegeben. Für jede Tastenkombination können Sie sich mit Strg-h k (`describe-key`) den auszuführenden Befehl und eine kurze Dokumentation ansehen.

Sollte wider Erwarten das System oder der Editor Emacs einmal abstürzen, so sind die gerade bearbeiteten Daten nicht verloren. Emacs speichert Ihre Daten regelmäßig in einer Auto-Save-Datei (`#`*`Dateiname`*`#`), die nach einem Systemabsturz zur Wiederherstellung verwendet werden kann. Beim Einlesen einer Datei, zu der eine Auto-Save-Datei existiert, wird der Benutzer darauf hingewiesen, dass er die Änderungen mit dem Befehl Meta-x `recover-file` wiederherstellen kann. Haben Sie viele Dateien editiert, so können Sie die gesamte Sitzung mit Meta-x `recover-session` wiederherstellen. Emacs präsentiert eine Liste der vorangegangenen Sitzungen, aus denen Sie die passende auswählen können. Bewegen Sie den Cursor auf die entsprechende Zeile und drücken Sie Strg-c Strg-c (`recover-session-finish`).

Das Erstellen von Sicherheitskopien kann mit einer Reihe von Variablen konfiguriert werden. Ist die Variable `auto-save-default` nicht `nil`, so wird die Auto-Save-Funktion für jeden neuen Puffer aktiviert. Der Wert der Variablen `auto-save-interval` bestimmt, nach wie vielen eingegebenen Zeichen eine

Sicherheitskopie erstellt wird. Mit der Variable `auto-save-timeout` wird bestimmt, nach wie vielen Sekunden ohne Benutzereingabe eine Auto-Save-Datei geschrieben wird.

Normalerweise ist die Auto-Save-Funktion beim Start von Emacs aktiv und die Variable `auto-save-intervall` hat den Wert 300. Wenn in einem Puffer viele Zeichen gelöscht wurden, so deaktiviert Emacs die Auto-Save-Funktion, da diese dann unverhältnismäßig aufwändig ist. Sie können jedoch den Puffer mit Strg-x k (`kill-buffer`) löschen und die Datei neu laden, so dass Emacs wieder Auto-Save-Kopien erstellt.

6.8 Tutorial, Hilfe und Info-Mode

Bevor Sie eine Entscheidung über den Einsatz von Emacs als Editor treffen, arbeiten Sie unbedingt das Tutorial (Strg-h t; Meta-x `help-with-tutorial`) durch. Dort werden auch noch andere Möglichkeiten des Emacs, wie z. B. die Verwendung von Makros, vorgestellt. Nach dem Tutorial sollten Sie die wichtigsten Funktionen, wie Navigation im Puffer, Wechseln von Fenstern und Suchen/Ersetzen, beherrschen.

Im weiteren Einsatz erweist sich Emacs als ausführlich dokumentiert. Zunächst ist das gesamte Handbuch im Info-Format verfügbar, so dass Sie es mit dem in Emacs integrierten Info-Browser (Strg-h i; Meta-x `info`) lesen können. Sie können die Dokumentation aber auch mit TeX ausdrucken oder ein entsprechendes Buch erwerben.

Diese Dokumentation ist aber nur ein Teil der gesamten Dokumentation. Zu den meisten Variablen können Sie genauere Informationen mit dem Befehl `describe-variable` erhalten. Um diese Funktion auszuführen, verwenden Sie entweder die Tastenkombination Strg-h v oder Meta-x `describe-variable`. Den Namen der Funktion bzw. den Namen der Variablen müssen Sie nicht vollständig eingeben. Mit den Tasten TAB bzw. SPC wird der Name soweit wie möglich vervollständigt.

Die Dokumentation zu Lisp-Funktionen erhalten Sie mit der Tastenkombination Strg-h f (`describe-function`). Wenn Sie den Namen der Variablen oder Funktion nicht wissen, können Sie mit Strg-h a (`apropos`) nach Funktionen oder Variablen suchen, die eine bestimmte Zeichenfolge enthalten. Die Suchergebnisse werden in einem Puffer angezeigt, in dem Sie mit normalen Emacs-Funktionen suchen können. Wenn Sie mit dem Cursor über einem Funktions- oder Variablennamen stehen, dann können Sie sich die zugehörige Dokumentation mit RET anzeigen lassen.

Die Tastenbelegungen eines bestimmten Mode erhalten Sie mit der Tastenkombination Strg-h m (`describe-mode`). Die Funktion, die mit einer bestimmten

Taste aufgerufen wird, erfahren Sie mit Strg-h k (describe-key). Die Tastenkombination Strg-h Strg-k springt zur Beschreibung der folgenden Tastenkombination in der Emacs-Info-Datei. Mit diesen Hilfsmitteln und den Quellen der Emacs-Lisp-Pakete, die häufig Informationen zur Konfiguration enthalten, sollten Sie den Emacs komplett an Ihre Bedürfnisse anpassen können. Einige beispielhafte Anpassungen werden im weiteren Verlauf dieses Kapitels vorgestellt.

Durch die Vergabe von sprechenden Namen für Variablen und Funktionen kann recht einfach mit dem Befehl `apropos` gesucht werden. Die Namen von Variablen und Funktionen folgen in der Regel gewissen Konventionen, so dass die Suche sehr häufig von Erfolg gekrönt ist. Dies ist einer der Gründe dafür, dass Emacs als »selbst dokumentierender« Editor bezeichnet wird.

6.9 Konfiguration

Emacs ist durch die weitgehende Programmierung in Emacs-Lisp sehr flexibel und an praktisch jede Anforderung anpassbar. Zu vielen Aufgaben gibt es bereits fertige Pakete, die im Emacs-Lisp-Archiv gesammelt werden. Die Adresse des Archivs finden Sie in der Info-Box am Ende des Kapitels. Fast alle Pakete kennen eigene Variablen zur Konfiguration, die jeder Benutzer für sich einstellen kann.

6.9.1 Die customize-Funktion

Emacs ist sehr flexibel an die Anforderungen der Benutzer anzupassen. Für größere Änderungen kann es notwendig sein, einen eigenen Mode mit verschiedenen Lisp-Funktionen zu implementieren. Viele kleine Anpassungen erfordern oft nur das Zuweisen eines Wertes an eine Variable. Da viele Anwender Lisp nicht beherrschen und auch nicht erlernen wollen, haben die Entwickler die `customize`-Funktion implementiert. Diese Funktion kann über das Menü, die Funktion `customize` oder im XEmacs mit der Tastenkombination Strg-h C aufgerufen werden.

Der Benutzer kann sich dann durch einen Baum an Konfigurationsmöglichkeiten hindurchbewegen und jede vorgesehene Einstellung ändern. Dabei ist es möglich, diese Änderungen nur für die aktuelle Sitzung oder permanent zu speichern. Diese Einstellungen werden unter XEmacs in der Datei `~/.xemacs/custom.el` gespeichert, GNU Emacs verwendet die Datei `~/.emacs`. Wenn Sie wollen, können Sie die Customize-Einstellungen später per Editor ändern oder weiter bei Customize bleiben.

Je nach einzustellendem Wert kann entweder aus einer Liste ausgewählt, eine Liste von Werten erfasst oder ein einzelner Wert eingegeben werden. Damit ist diese Funktion sehr flexibel und durch die Verweise auf die normale Emacs-Dokumentation auch recht gut beschrieben. Abbildung 6.3 zeigt die Einstellungen zu grundlegenden Editierfunktionen.

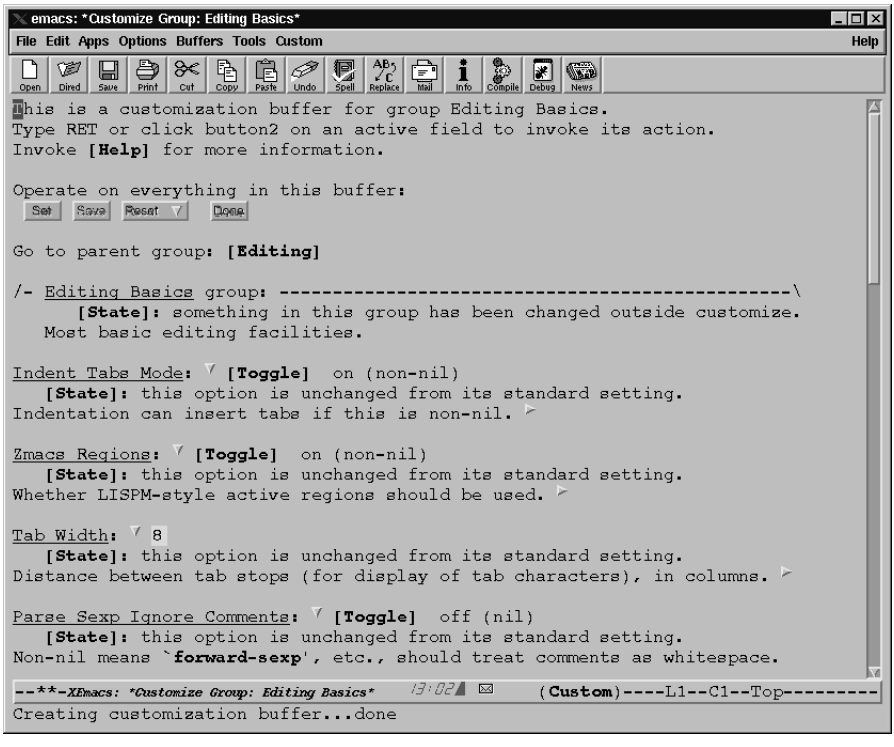


Abbildung 6.3 Anpassen des Emacs mit der `customize`*-Funktion*

Die `customize`-Funktion ist nützlich, wenn man ohne Studium der Dokumentation oder der Emacs-Lisp-Programme auskommen will oder um einfach nur eine Übersicht über die möglichen, in `customize` vorgesehenen Optionen zu bekommen. Ein Nachteil ist, dass man in den gespeicherten Einstellungen keine Kommentare hinterlegen kann, also Bemerkungen, warum man diese oder jene Einstellung vorgenommen hat, oder einfach, was man später noch ausprobieren will.

6.9.2 Konfigurationsdateien

Beim Laden des GNU Emacs wird zunächst die Datei `~/.emacs` gelesen. Diese Datei enthält Emacs-Lisp-Befehle, mit denen das Verhalten oder Aussehen des Emacs verändert werden kann. In dieser Datei kann jeder Benutzer den Editor

gemäß seinen Präferenzen konfigurieren. Anschließend wird, wenn die Variable `inhibit-default-init` nicht `nil` ist, die Datei `default.el` aus dem Suchpfad für Lisp-Dateien geladen. XEmacs hat früher ebenfalls die Datei `~/.emacs` gelesen, heute wird die Datei `~/.xemacs/init.el` verwendet. Damit können Sie recht einfach verschiedene Konfigurationen verwenden, wenn Sie das wollen. Ich persönlich mag es lieber, wenn beide Editoren eine sehr ähnliche Konfiguration verwenden.

Zusätzlich dazu kann es eine weitere systemweite Konfigurationsdatei (`site-init.el`) geben, die noch vor der Datei `~/.emacs` geladen wird. Das Laden dieser Datei lässt sich mit der Kommandozeilenoption `-no-site-file` verhindern. Dies sollte aber nur in seltenen Fällen notwendig sein, z. B. zur Fehlersuche. Wenn im Folgenden von der Datei `~/.emacs` die Rede ist, so ist eine der hier genannten Konfigurationsdateien gemeint.

Jede dieser Dateien enthält Lisp-Code. Sie müssen Lisp in aller Regel nicht beherrschen, da für viele Anwendungsfälle bereits entsprechende Vorschläge zur Konfiguration in der (X)Emacs-FAQ, im Info-System oder in den Lisp-Dateien des entsprechenden Modus zu finden sind. Alles, was Sie sonst noch über Lisp wissen müssen, werden Sie in den folgenden Abschnitten lernen.

Der Name Lisp ist eine Abkürzung von »List Programming«. Listen (und Programme oder Lisp-Funktionen sind nichts anderes als Listen) werden in Lisp in Klammern eingeschlossen. Daher kommt es häufiger vor, dass mehrere Klammern am Ende einer Zeile oder Funktion auftauchen. Beim Einfügen einer neuen Klammer wird im `lisp-mode` die zugehörige öffnende Klammer markiert. Dies stellt eine große Hilfe für den Programmierer oder Anwender dar. Die Emacs-FAQ bietet zu diesem Thema einige weitere Tipps.

Kommentare in Lisp beginnen mit einem Semikolon (`;`) und reichen bis zum Zeilenende. Der Lisp-Mode des Emacs (nicht der Lisp-Interpreter) unterscheidet dabei zwischen Kommentaren, die mit ein, zwei oder drei Semikola beginnen. Diese markieren Kommentare, die unterschiedlich eingerückt werden sollen. Ein Semikolon bedeutet Einrückung bis zur Spalte 40, zwei stehen für die Einrückung wie der aktuelle Code und Kommentare, die mit drei Semikola beginnen, werden nicht verschoben. Eine Zeile kann mit der Taste Tab (`lisp-indent-line`) entsprechend formatiert werden.

Emacs stellt einige Funktionen bereit, die die Lisp-Programmierung beinahe zu einem Vergnügen machen. Die oben bereits erwähnte Anzeige der zugehörigen öffnenden Klammer ist nur ein Beispiel. Sie können eine Konfigurationsdatei auswerten, ohne Emacs zu verlassen, indem Sie den Befehl `eval-buffer` verwenden. Eine Region können Sie mit `eval-region` an den Lisp-Interpreter übergeben. Diese Funktionen sind im `elisp-mode` und im `*scratch*`-Puffer verfügbar. Im `*scratch*`-Puffer wird der `Lisp Interaction`-Mode verwendet, in dem Befehle eingegeben und ausgeführt werden können.

Weitere nützliche Funktionen sind `eval-defun` (Meta-Strg-x, Evaluieren der Funktionsdefinition, in der sich der Cursor befindet) und `eval-last-sexp` (Strg-x Strg-e, den letzten Lisp-Ausdruck ausführen). Dadurch ist es in vielen Fällen möglich, iterativ zu programmieren oder neue Konfigurationseinstellungen zu testen.

```
;; vergrößert den Puffer am Ende nicht
(setq next-line-add-newlines nil)
;; aktuelle Zeilennummer in der Modeline anzeigen
(setq line-number-mode 't)
;; Highlight der Suchergebnisse
(setq search-highlight 't)
;; Standard-Mode ist "text-mode"
(setq default-major-mode 'text-mode)
;; Zeilenumbruch automatisch
(add-hook 'text-mode-hook 'turn-on-auto-fill)
```

Listing 6.2 Einfache Befehle für die Datei `~/.emacs`

Das Listing 6.2 enthält einige Befehle, die Sie möglicherweise in Ihre `~/.emacs`-Datei aufnehmen möchten. Die einzelnen Befehle werden im Folgenden näher erklärt, bevor wir auch kompliziertere Konfigurationen durchführen. Lisp besteht aus Ausdrücken, die in Klammern eingeschlossen sind. Diese Ausdrücke können auch als Listen interpretiert werden. Direkt nach der Klammer steht die auszuführende Funktion, danach folgen die Parameter der Funktion. Funktionen geben teilweise einen Wert zurück, der dann als Parameter für einen anderen Funktionsaufruf dienen kann.

Variablen werden mit der Funktion `set` bzw. `setq` verändert. Die in Listing 6.3 gezeigten Funktionsaufrufe sind äquivalent (das 'q' in `setq` steht für »Quote«, also das einfache Anführungszeichen `'`). Mit einem Quote wird die Auswertung des folgenden Ausdrucks verhindert. Im obigen Beispiel tauchen die Werte `t` und `nil` mehrfach auf. Diese Werte haben in Lisp eine spezielle Bedeutung und stehen für TRUE und FALSE (bzw. für die leere Liste, »not in list«).

```
(setq next-line-add-newlines nil)
(set 'next-line-add-newlines nil)
```

Listing 6.3 Vergleich von `set` *und* `setq`

Zurück zu den Beispielen in Listing 6.2. Emacs vergrößert den Puffer automatisch, wenn man mit dem Cursor über das Dateiende hinausblättert. Oft ist dies störend (der Puffer wird modifiziert, obwohl keine »echten« Änderungen durchgeführt wurden) und nicht erwünscht. Der erste Lisp-Ausdruck stellt dieses Verhalten ab.

Der zweite Ausdruck in Listing 6.2 sorgt dafür, dass in der Modeline stets die aktuelle Zeilennummer angezeigt wird. Mit der Variablen `mode-line-format`

bzw. `modeline-format` unter XEmacs können Sie das Format der angezeigten Modeline genauer bestimmen. Wenn Sie die Nummer der aktuellen Spalte ebenfalls in der Modeline angezeigt bekommen möchten, dann können Sie die Funktion `column-number-mode` aufrufen.

Die dritte Einstellung sorgt dafür, dass Suchergebnisse durch eine Hervorhebung deutlich angezeigt werden. Ansonsten kann es passieren, dass man nach einer Suche erst eine Weile nach dem Cursor Ausschau halten muss, um dort den gefundenen Begriff zu sehen.

Die letzten beiden Ausdrücke steuern das Verhalten von Emacs, wenn kein spezieller Mode für einen Puffer vorhanden ist. Normalerweise befindet sich ein solcher Puffer im `fundamental-mode`. Oft enthält dieser lediglich natürlichsprachlichen Text, so dass der `text-mode` angemessen wäre. Mit dem vorletzten Ausdruck wird dieser Mode als Standard verwendet, wenn keine speziellen Anpassungen verfügbar sind.

Bei vielen Anwendungsfällen ist es sinnvoll, wenn ein Text automatisch an Wortgrenzen umbrochen wird. Dieses Verhalten wird von einem Minor-Mode (`auto-fill-mode`) implementiert. Der letzte Lisp-Ausdruck aus Listing 6.2 aktiviert diesen Minor-Mode immer dann, wenn ein Puffer in den `text-mode` versetzt wird.

Sie sollten Ihre Anpassungen ausführlich dokumentieren, so dass andere von Ihren Erfahrungen profitieren können und Sie selbst den Überblick behalten. Meine Emacs-Konfigurationsdateien verändern sich relativ regelmäßig, deshalb werden sie mit CVS verwaltet. Das erfordert zwar einige Disziplin, macht aber Änderungen nachvollziehbar und sehr einfach zwischen verschiedenen Rechnern synchronisierbar.

Es reicht nicht immer aus, die Werte einzelner Variablen anzupassen. Emacs stellt an vielen Stellen so genannte *Hooks* zur Verfügung, die Benutzer verwenden können, um spezielle Funktionen auszuführen. So wird bei jedem Öffnen einer Datei der Hook `find-file-hook` aufgerufen und die darin enthaltenen Befehle werden ausgeführt. Viele andere Lisp-Pakete ermöglichen die flexible Erweiterung ihrer Funktionalität durch eine Reihe von Hooks. Die Namen der Hooks und deren Bedeutung finden Sie in der Dokumentation zu dem entsprechenden Modus.

```
;; Direktes Einfügen in den Hook
(add-hook 'text-mode-hook 'turn-on-auto-fill)

;; Unbenannte Funktion in den Hook aufnehmen
(add-hook 'text-mode-hook
          (lambda ()
             (turn-on-auto-fill)))
```

```
;; Benannte Funktion in den Hook aufnehmen
(defun my-text-mode-hook ()
       (turn-on-auto-fill))
(add-hook 'text-mode-hook 'my-text-mode-hook)

;; Hook löschen; remove-hook löscht einzelne Aufrufe
(setq text-mode-hook nil)

;; Hook auf definierten Wert setzen
(setq text-mode-hook 'turn-on-auto-fill)
```

Listing 6.4 Erweitern und Setzen von Hooks

In der Regel werden Benutzer zusätzliche Funktionen in den Hook aufnehmen. Die zusätzlichen Funktionen können mit der Funktion `add-hook` in den Hook aufgenommen werden. Soll nicht nur eine Funktion ausgeführt werden, so muss eine (eventuell unbenannte) Funktion erzeugt werden, die in den Hook aufgenommen wird. Beispiele für die Verwendung von Hooks finden Sie in Listing 6.4. Das direkte Setzen eines Hooks, der nichts anderes als eine Variable ist, mit `setq` sollten Sie nur in wirklichen Sonderfällen verwenden, weil Sie möglicherweise andere Einstellungen als gewollt verändern.

Einige weitere Dinge können mit X-Ressourcen (`~/.Xresources`) konfiguriert werden. Eine komplette Liste der veränderbaren Ressourcen finden Sie in der Manpage, beim GNU-Emacs sind das im Wesentlichen die Größe des Fensters, der verwendete Font und die Farben. XEmacs nutzt viel mehr Features des X-Window-Systems, daher kann man dort deutlich mehr Anpassungen vornehmen. Für den XEmacs sind das alle Farben, die verwendeten Fonts und diverse Einstellungen für die Maus. Sie finden ein Beispiel im XEmacs-Quellcode.

Zusammen mit dem XEmacs werden die Beispieldateien `sample.emacs` und `sample.Xdefaults` installiert. In diesen finden Sie bereits viele Tipps und Anregungen für die Konfiguration des XEmacs. Viele einfache Konfigurationstips finden Sie außerdem in der FAQ zu Emacs (bzw. XEmacs).

6.9.3 Große Konfigurationsdateien

Wenn Sie Emacs für viele Tätigkeiten einsetzen, so werden Sie für viele Modes eigene Konfigurationen erstellen. Die meisten Modes kann man zwar ohne vorherige Konfiguration verwenden, aber manchmal möchte man ein weiteres Zusatzpaket integrieren oder einfach spezielle Funktionen nutzen. In jedem dieser Fälle werden Sie einige Einträge in die Datei `~/.emacs` aufnehmen müssen.

Einfache Editoren bieten diesen Komfort und diese Flexibilität nicht. Sie sollten aber zumindest am Anfang nur wenige Einstellungen vornehmen und sich langsam die Funktionen des Emacs erschließen. Es ist nicht besonders sinnvoll, einfach die Konfiguration eines anderen Emacs-Benutzers zu verwenden, da sich

Emacs dann in einigen Funktionen nicht wie im Handbuch beschrieben verhält. Da das unbedarfte oder unerfahrene Anwender verwirren könnte, sollten Sie mit der unbesehenen Weitergabe Ihrer Konfiguration vorsichtig sein.

Bei vielen verschiedenen Anpassungen wird die Datei `~/.emacs` relativ schnell groß und unübersichtlich und das Laden und Ausführen der Datei dauert relativ lange. In diesem Fall bietet es sich an, die Konfigurationen in eine andere Datei als `~/.emacs` auszulagern und diese mit der Funktion `byte-compile-file` zu kompilieren.

In der Praxis erweist es sich als nützlich, für jedes größere Lisp-Paket eine eigene, relativ kleine Konfigurationsdatei anzulegen und diese nötigenfalls automatisch beim Start von Emacs kompilieren zu lassen. Dadurch wird die Konfiguration übersichtlicher und ist leichter zu pflegen. Zusammengehörige Einstellungen sind jeweils in einer relativ kleinen Datei gespeichert. Außerdem hat dies den Vorteil, dass die Konfigurationen zu einem Paket nicht über die ganze `~/.emacs`-Datei verteilt sind.

Wenn Sie jede der Dateien unter einer Versionsverwaltung halten, können Sie immer Ihre Änderungen nachvollziehen. Gerade wenn man zwischen GNU-Emacs und XEmacs sowie verschiedenen Rechnern wechselt, kann das sehr hilfreich sein. In Listing 6.5 finden Sie ein Beispiel für die Datei `~/.emacs`, in der die meisten Konfigurationen ausgelagert sind.

```
;; das Verzeichnis ~/emacs-lisp in den load-Path aufnehmen
(setq load-path
      (append (list (expand-file-name "~/emacs-lisp/"))
              load-path))

;; Trick, um die .el-Datei neu zu kompilieren,
;; wenn sie neuer als die .elc-Datei ist
(defun byte-compile-if-newer-and-load (file)
  "Byte compile file.el if newer than file.elc"
  (if (file-newer-than-file-p (concat file ".el")
                              (concat file ".elc"))
      (byte-compile-file (concat file ".el")))
  (load file))

;; Die init-Datei eventuell kompilieren und dann laden
(byte-compile-if-newer-and-load " /emacs.init/emacs.init")
```

Listing 6.5 Eine minimale `.emacs`*-Datei*

Die Datei `~/.emacs` enthält dann eine Reihe von einfachen, global gültigen Einstellungen, wie z. B. die in Listing 6.2 gezeigten, und für jedes größere Paket, das mit einer Reihe von Variablen und Hooks konfiguriert wird, einen Aufruf der Funktion `byte-compile-if-newer-and-load`. Dann ist auch das Byte-Kom-

pilieren der modifizierten Konfigurationsdateien recht schnell, da jede Datei selbst nur sehr klein ist. Die Datei `emacs.init.el` könnte dann aus einer Reihe von Zeilen bestehen, die denen in Listing 6.6 ähneln.

```
;; Gnus: Der beste Newsreader überhaupt
(byte-compile-if-newer-and-load " /emacs.init/gnus.conf")
;; Einstellungen für die Rechtschreibprüfung
(byte-compile-if-newer-and-load " /emacs.init/ispell.conf")
```

Listing 6.6 Die Datei `emacs.init.el`

Durch die Zusammenfassung aller Anpassungen zu einem Modus oder Paket können Sie diese auf einen Blick erfassen. Dies ist zur Fehlersuche recht nützlich und bietet die Möglichkeit, Ihre Einstellungen ohne großen Aufwand gezielt an andere Benutzer weiterzugeben. Kopieren Sie nicht einfach die Dateien, da dort häufig sehr persönliche Einstellungen vorgenommen werden, sondern extrahieren Sie die Einstellungen für den Modus, um den es gerade geht.

6.9.4 Dateilokale Konfiguration

Neben der Konfiguration, die in den Initialisierungsdateien festgelegt wird, kennt Emacs noch die puffer- oder dateilokale Konfiguration. Dies sind oft Variablen, die für jeden Puffer einen anderen Wert annehmen können. Das sind z. B. Tastaturbelegungen, Einstellungen zur Auto-Save-Funktion und vieles andere mehr. Sie können bei vielen Dateitypen diese (und andere) Variablen permanent in der Datei als lokale Variablen speichern.

Lokale Variablen werden am Ende der Datei gespeichert, in der Regel in einem Kommentar der verwendeten Programmiersprache. Eingeleitet wird der Block der lokalen Variablen durch den Text `Local variables:`, das Ende des Blocks ist mit dem Text `End:` erreicht. Dazwischen finden Sie den Namen einer (pufferlokalen) Variablen, gefolgt von einem Doppelpunkt und dem neuen Wert dieser Variablen. Damit kann z. B. ein spezieller Einrückstil für bestimmte Dateien vorgegeben werden.

```
% Local Variables:
% mode: latex
% TeX-master: "master"
% End:
```

Listing 6.7 Ein Beispiel für lokale Variablen in einem LaTeX-Quelltext

In Listing 6.7 finden Sie ein Beispiel für lokale Variablen in einem LaTeX-Quelltext. Diese Variablen werden vom `auctex`-Modus ausgewertet, der leider nicht zum Standardumfang von GNU-Emacs gehört; XEmacs enthält dieses Paket.

Dieser Modus ist für TeX- und LaTeX-Dateien verwendbar, die Variable `mode` wählt für diese Datei stets den LaTeX-Modus an. Die Datei ist eine von mehreren dieses Dokuments, die von LaTeX zu bearbeitende Hauptdatei heißt `master.tex`.

Ist die Variable `enable-local-variables` `t`, so werden wie in Listing 6.7 definierte lokale Variablen in jedem Fall entsprechend verändert. Ist der Wert `nil`, so werden diese Variablen ignoriert. Jeder andere Wert für die Variable `enable-local-variables` führt dazu, dass der entsprechende Block angezeigt und der Benutzer gefragt wird, ob die dort enthaltenen Variablen gesetzt werden sollen.

Mit Hilfe dieses Mechanismus können auch im Emacs Makro-Viren implementiert werden, da viele Funktionen des Emacs durch Variablen beeinflusst werden und außerdem Hooks nichts anderes als Variablen mit einem speziellen Inhalt sind. Man sollte also lokale Variablen bei fremden Dokumenten nicht blind übernehmen, sondern prüfen und gegebenenfalls die Nachfrage verneinen.

Mit der Variable `ignored-local-variables` können Sie außerdem bestimmte Variablen von der Verwendung als dateilokale Variable gezielt ausschließen. Das Listing 6.8 enthält als Beispiel die Standardeinstellung des XEmacs. Mit dieser Einstellung kann verhindert werden, dass bei einer Datei zunächst nachgefragt wird und für alle weiteren Dateien diese Nachfrage ausgeschaltet werden kann. Zusätzliche Variablen können Sie in die Liste aufnehmen.

```
;;; Folgende Variablen können nicht pufferlokal sein.
(setq ignored-local-variables '(enable-local-eval))
```

Listing 6.8 Die Variable `ignored-local-variables`

6.10 Emacs als Server – einer für alles

Verschiedene Programme, z. B. `elm`, `tin` oder `crontab`, können einen beliebigen Editor zum Editieren einer Mail bzw. eines Textes aufrufen. In der Regel benutzen diese Programme die Umgebungsvariable `EDITOR` bzw. `VISUAL`, in der sich der Name des Editors befindet. Enthalten diese Variablen keinen Wert, so wird `vi` als Standardeditor verwendet. Dieses Verfahren hat den Vorteil, dass der Anwender nur einen Editor lernen muss, da nicht jedes Programm eine eigene Editorfunktion implementiert.

Wenn in einer der Variablen `EDITOR` bzw. `VISUAL` einfach `emacs` steht, wird bei jedem Aufruf ein neuer Emacs-Prozess gestartet. Dies ist unpraktisch, da es Zeit und Speicher kostet. Außerdem können die Emacs-Prozesse nicht gegenseitig auf ihre Puffer zugreifen.

Echte Emacs-Benutzer erledigen daher fast alles innerhalb des Emacs, den sie einmal beim Einloggen starten und dann bis zum Ausloggen nicht mehr beenden. Das ist möglich, da der Emacs für viele Funktionen eigene Modi enthält. So kann ein Anwender seine Mail mit RMAIL, VM oder Gnus lesen, ohne Emacs zu verlassen. Analog kann als Newsreader statt `tin`, `slrn` oder `knews` Gnus verwendet werden.

Es ist auch möglich, einen bereits laufenden Emacs als Editor für einen neuen Puffer, z. B. von `elm` aus, zu benutzen, indem Emacs als Server konfiguriert wird. Dazu muss innerhalb von GNU-Emacs die Funktion `server-start` aufgerufen werden. Das lässt sich durch einen entsprechenden Eintrag in der Datei `~/.emacs` automatisieren. Die Umgebungsvariable `EDITOR` muss dann auf den Wert `emacsclient` gesetzt werden.

Das hat zur Folge, dass, immer wenn eine Anwendung `emacsclient` aufruft, eine Nachricht an den GNU-Emacs-Server geschickt wird, in der dieser aufgefordert wird, eine bestimmte Datei in einen neuen Puffer zu laden. Der Client gibt die Meldung `Waiting for Emacs` aus und wartet auf das Ende der entsprechenden Server-Session. Arbeiten Sie auf den virtuellen Konsolen, so müssen Sie auf die Konsole mit Emacs wechseln (dazu können Sie ein Programm wie `chvt` oder `switchto` verwenden). Unter X aktivieren Sie das entsprechende Fenster.

Wenn der Text im Server fertig bearbeitet ist, wird mit [Strg]-[x] [#] der Puffer verlassen und gegebenenfalls der vorher editierte Puffer wieder im Fenster gezeigt. Danach wird eine Nachricht an das Programm `emacsclient` geschickt, in der dieses aufgefordert wird, sich selbst zu beenden. Die Programme, die `EDITOR` bzw. `VISUAL` benutzen, warten darauf, dass der dahinterstehende `emacsclient` beendet wird. Für diese Programme ergibt sich also kein Unterschied zum direkten Aufruf eines Editors.

Das Verfahren mit Emacs-Client und -Server funktioniert auch auf Textkonsolen. Hier ist es sinnvoll, Emacs auf einer virtuellen Konsole zu starten und das Mailprogramm auf einer anderen, da es schneller geht, die Konsole zu wechseln, als den Emacs komplett zu laden. Besonders beim XEmacs dauert das Laden der Konfiguration sehr lange, so dass sich dieses Verfahren lohnt.

6.11 XEmacs als Server – noch mehr Features

Die Funktionen, die GNU-Emacs als Server bereitstellt, sind nicht immer ausreichend. So ist es nicht möglich, einzelne Lisp-Ausdrücke durch den bereits laufenden Prozess auswerten zu lassen. Außerdem ist es nicht möglich, den Emacs-Prozess »mal eben« auf einen anderen Bildschirm, z. B. am Arbeitsplatz eines Kollegen oder über eine Modemverbindung anzusprechen. In der aktuellen Version vom XEmacs ist dies alles implementiert.

Damit sich GNU-Emacs und XEmacs nicht in die Quere kommen, verwendet XEmacs andere Lisp-Funktionen und Programme. Sie sollten Ihre Emacs-Lisp-Dateien allerdings so verfassen, dass Sie diese mit beiden Editoren verwenden können. Das ist hilfreich, falls auf einem neuen Rechner (noch) kein XEmacs existiert oder Sie häufig zwischen beiden Umgebungen wechseln müssen.

Der Server wird beim XEmacs mit der Funktion `gnuserv-start` gestartet. Das Listing 6.9 zeigt ein Beispiel, wie man zwischen GNU-Emacs und XEmacs innerhalb von Lisp-Funktionen unterscheiden kann.

```
;; Läuft XEmacs oder GNU-Emacs?
(defvar running-xemacs
        (string-match "XEmacs\\|Lucid" emacs-version))

(cond
 ((not running-xemacs)
  ;; GNU-Emacs-spezifischer Code
  (server-start))
 (t
  ;; XEmacs-spezifischer Code
  (gnuserv-start)))
```

Listing 6.9 Automatisches Starten des Emacs-Servers für GNU-Emacs und XEmacs

Anschließend kann mit dem Programm `gnuclient` genauso gearbeitet werden wie mit `emacsclient`. Der einzige Unterschied ist, dass die Bearbeitung der Puffer auch von einem anderen Rechner aus erfolgen kann. Dieses Verfahren kann dazu führen, dass Ihr Editor von beliebigen Benutzern aus dem Netzwerk gesteuert werden kann. Daher sind einige Sicherheitsmechanismen implementiert.

Sie können entweder eine host-basierte Authentifizierung oder besser eine Art Passwortabfrage durchführen. Eine host-basierte Authentifizierung sollten Sie nach Möglichkeit nicht verwenden, da andernfalls jeder Benutzer dieses Rechners auf Ihren Editor zugreifen kann (dieses Verfahren ist mit dem `xhosts`-Programm für X-Sitzungen vergleichbar, siehe auch Abschnitt 7.8.1, »Rechnerweise Zugriff erlauben mit `xhost`«). Die Variable `GNU_SECURE` enthält den Namen einer Datei, in der die Namen der erlaubten Hosts gespeichert sind.

Besser ist es, wenn Sie analog zu `xauth` die Authentifizierung mit »magischen Cookies« verwenden. Wenn Emacs korrekt installiert wurde, dann kann dazu das Programm `xauth` (siehe auch den Abschnitt 7.8.2, »Benutzerbezogenen Zugriff erlauben mit `xauth`«) verwendet werden, als Display-Nummer wird 999 verwendet. Ein entsprechender `MAGIC-COOKIE` wird von `gnuserv` automatisch erzeugt. Auf welchen Host Sie zugreifen möchten, müssen Sie beim Aufruf von `gnuclient` entweder mit der Option `-h` oder der Umgebungsvariablen `GNU_HOST` angeben. Welche Ports verwendet werden, sollten Sie in der Manpage zu `gnuclient(1)` nachlesen.

Besonders interessante Erweiterungen gegenüber GNU-Emacs sind die Programme `gnuattach` und `gnudoit`. Mit `gnuattach` kann ein bereits laufender XEmacs-Prozess dazu veranlasst werden, zusätzlich Eingaben auf dem aktuellen TTY anzunehmen. Dies ist nützlich, wenn man mit einem Modem eingeloggt ist oder am Arbeitsplatz eines Kollegen sitzt und keinen neuen XEmacs starten kann oder will. Es kann allerdings vorkommen, dass die Tastaturbelegung und die Farben nicht zu dem aktuellen Terminal passen, da diese Einstellungen oft nur beim Start des XEmacs durchgeführt werden.

Das Programm `gnudoit` wird dazu verwendet, Lisp-Ausdrücke durch XEmacs auswerten zu lassen. Beispiele für den Aufruf finden Sie z. B. in der Manpage zu `gnudoit`. Mit dieser Methode können Sie einen laufenden XEmacs von außen steuern – die Sicherheitsimplikationen sind dieselben wie bei `gnuattach`.

6.12 Tastaturbelegung

Neben dem Aufruf von Funktionen über Menüs, vordefinierten Tastaturkürzeln und dem Aufruf durch Meta-x *Kommando* können praktisch alle Tasten mit Emacs-Funktionen belegt werden. Emacs unterscheidet dabei zunächst zwischen einer globalen, über alle Modi gültigen Belegung und einer möglicherweise für einzelne Modi angepassten lokalen Belegung (lokale Keyboard-Maps).

Beachten Sie, dass viele Modi einzelne Tasten in einer lokalen Tabelle neu definieren, so dass sich die Belegung zwischen Puffern ändern kann und eine globale Anpassung in einem einzelnen Modus möglicherweise nicht wirksam ist. In Listing 6.10 finden Sie Beispiele für die Anpassung der globalen und einer lokalen Tastaturtabelle. Beachten Sie, dass die Darstellung vieler Tasten in GNU-Emacs und XEmacs unterschiedlich ist, das wird sich vermutlich auch in absehbarer Zeit nicht ändern. Die Unterschiede in der Programmierung der Tastaturbelegung sind in Listing 6.11 und Listing 6.12 dargestellt.

```
;; Globale Anpassungen
(global-set-key [home]           'beginning-of-line)
(global-set-key [end]            'end-of-line)
(global-set-key \C-cg'goto-line)

;; Anpassung der LaTeX-Tastaturbelegung
(defun my-latex-mode-hook ()
   (define-key LaTeX-mode-map '[delete] 'delete-character))
(add-hook 'latex-mode-hook 'my-latex-mode-hook)
```

Listing 6.10 Anpassung der Tastaturbelegung

Zunächst werden die Tasten [Pos1] und [Ende] so belegt, dass der Cursor an den Zeilenanfang bzw. das Zeilenende gestellt wird. Die Standardeinstellung von Emacs ist hier, dass zum Anfang bzw. Ende des Puffers gesprungen wird. Hier werden symbolische Namen für die Tasten (`[home]` und `[end]`) verwendet, die Emacs intern auf die entsprechenden Tasten des Terminals legt. Dadurch ist es möglich, mit relativ einfachen Mitteln auf verschiedenen Terminals und Tastaturen eine einheitliche Tastaturbelegung festzulegen.

Beachten Sie, dass einige Tastenkombinationen wie [Umschalt]-[F1] nur unter X eindeutig definiert sind. Außerdem sind etliche Tasten auf den verschiedenen Terminals oder Terminal-Emulationen nicht verfügbar, so dass sich die Tastaturbelegung zwischen verschiedenen Betriebssystemen oder Terminals deutlich unterscheiden kann. Im Emacs sind aber Konfigurationen für die gängigen Terminals enthalten, außerdem wird die `terminfo`- bzw. `termcap`-Datenbank ausgelesen.

In der dritten Zeile von Listing 6.10, wird der Befehl `goto-line` auf die Tastenkombination [Strg]-[c] [g] gelegt (im XEmacs ist diese Funktion mit [Meta]-[g] erreichbar). Alle bisher erläuterten Belegungen beziehen sich auf die globale Tastaturtabelle. Zum Festlegen dieser Belegungen wird die elisp-Funktion `global-set-key` verwendet, die Sie auch interaktiv aufrufen können.

Viele Modi haben zusätzlich zu den Standardtasten weitere Tasten belegt. Diese Änderungen sind in einer moduslokalen Tastaturtabelle hinterlegt. Alle dort nicht veränderten Tasten behalten ihre globale Belegung. Einzelne Modi, wie z. B. der `dired`-Modus, verändern die Funktionen sehr vieler Tasten.

Im zweiten Teil von Listing 6.10 wird die [Entf]-Taste im LaTeX-Modus so belegt, wie man das von anderen Editoren gewöhnt ist. In der aktuellen Version von `auctex` ist diese Einstellung nicht mehr notwendig. Möglicherweise kann Ihnen aber dieses Beispiel bei anderen Modi helfen. Die Belegung der Backspace- und Delete-Tasten ist unter Unix leider nicht konsistent und heftig umstritten.

Besonders bei der Verwendung von Sondertasten (wie Funktionstasten, Umschalt- oder Steuerungstaste) unterscheidet sich die Syntax bei der Tastenbelegung zwischen XEmacs und GNU-Emacs. Diese Unterschiede entstehen durch die Differenzen bei der Implementierung der beiden Versionen und werden vermutlich auch in absehbarer Zeit nicht verschwinden. Ein Beispiel für die verschiedenen Tastaturbelegungen finden Sie in Abschnitt 6.13.2, »Der `text-mode`« zum `text-mode`.

Wenn Sie die Belegung einer Taste wissen möchten, so erfahren Sie diese mit [Strg]-[h] [k] (`describe-key`). Die Dokumentation zu einer Funktion erhalten Sie mit der Tastenkombination [Strg]-[h] [f] (`describe-function`). Wenn Sie die Taste suchen, die eine bestimmte Funktion aufruft, so verwenden Sie [Strg]-[h] [w] (`where-is`). Sie werden nach einem Funktionsnamen gefragt und bekommen eine Liste der mit dieser Funktion belegten Tasten angezeigt.

6.13 Emacs-Erweiterungen (Modi)

Die besondere Leistungsfähigkeit erhält Emacs erst durch die vielen zur Verfügung stehenden Modi. Mit diesen Modi wird Emacs für verschiedene Dateitypen oder Aufgaben speziell angepasst, so dass ein Emacs-Benutzer viele Vorteile durch zusätzliche Funktionen oder Tastenkombinationen hat. Oft kann Emacs zur Syntax der Dateien Hilfestellung geben, so dass sich beliebte Fehler, wie eine fehlende schließende Klammer, vermeiden lassen.

Basierend auf der Syntax der Programmiersprache kann Emacs auch eine farbliche Hervorhebung von Schlüsselwörtern, Kommentaren und Literalen durchführen. Dadurch ist es oft viel einfacher, sich in fremden Programmen oder Texten zurechtzufinden.

Alle verfügbaren Modi zu beschreiben, würde den Rahmen dieses Buches sprengen, daher wird im Folgenden nur eine kleine Auswahl vorgestellt. Viele Modi gehören bereits zum Standardumfang von Emacs und es werden von Version zu Version mehr. Außerdem können Sie zusätzliche Modi aus dem Emacs-Lisp-Archiv bekommen. Die Bezugsquelle finden Sie am Ende dieses Kapitels. Erfahrene Emacs-Benutzer können Modi auch selbst entwickeln. Mehr Informationen dazu finden Sie im Emacs-Lisp-Handbuch.

Emacs erkennt oft, z. B. anhand des Dateinamens oder der Erweiterung, welcher Modus zu verwenden ist. Sie können Emacs aber auch eine Hilfestellung geben, indem Sie in der ersten Zeile der Datei den Text `-*- Mode -*-` einfügen. In den Quelltexten zu diesem Buch verwende ich folgenden Text: `<!-- Hey Emacs, this is -*- xml -*- -->`.

Emacs erkennt anhand der Zeichenkette `-*-`, dass danach der Name des zu verwendenden Modus folgt. Der Name wird wieder mit der Zeichenfolge `-*-` abgeschlossen, der einleitende Text ist nur ein Kommentar. Im obigen Beispiel wird das `psgml`-Paket angewiesen, diese Datei als XML-Quelltext zu betrachten. Oft erkennt Emacs das aufgrund des Anfangs der Datei auch selbständig, aber das funktioniert nicht immer perfekt.

6.13.1 Der fundamental-mode

Wird Emacs mit einer Datei gestartet, für die kein spezieller Modus zur Verfügung steht, so wird in den `fundamental-mode` gewechselt. Dieser Modus stellt praktisch keine speziellen Funktionen bereit. Man kann daher alle anderen Modi als Erweiterungen und Anpassungen des `fundamental-mode` betrachten.

Viele Benutzer verwenden als Standardmodus den `text-mode`. Ein Beispiel für diese Einstellung finden Sie in Listing 6.2.

6.13.2 Der text-mode

Emacs kennt bestimmte Charakteristika eines Textes. Zunächst kennt er Wörter und kann auf Wörtern verschiedene Kommandos ausführen: Wörter löschen (`kill-word`, Meta-d), überspringen (`forward-word`, Meta-f oder Strg-rechts und `backward-word`, Meta-b oder Strg-links) oder deren Rechtschreibung überprüfen (`ispell-word`, Meta-$). Dies unterscheidet den Emacs nicht von anderen Editoren.

Darüber hinaus kennt der Emacs aber auch höhere Konstrukte eines (natürlichsprachigen) Textes, nämlich Sätze und Absätze. Ein Satz wird mit einem Interpunktionszeichen, gefolgt von zwei Leerzeichen, beendet, dies ist eine amerikanische Satzkonvention. Damit ist es möglich, Kommandos (wie z. B. `kill-sentence`, Meta-k) auf Sätze anzuwenden oder satzweise durch den Text (`backward-sentence`, Meta-a und `forward-sentence`, Meta-e) zu springen. Diese Funktionen können natürlich auf geeignete Tastenkombinationen (z. B. Meta-Cursortasten) gelegt werden.

Mit den Tasten Meta-} (`forward-paragraph`) und Meta-{ (`backward-paragraph`) kann man sich absatzweise bewegen. Diese Funktionen sind zunächst nur für natürlichsprachige Texte interessant, werden aber analog in verschiedenen Modi für Programmiersprachen verwendet. Das ist einer der großen Vorteile von Emacs: Man muss viele Funktionen nur einmal erlernen und kann diese dann in den verschiedenen Modi verwenden. So muss man nicht für jedes Programm oder jede Programmiersprache den Umgang mit einem neuen Editor erlernen, der weniger oder andere Funktionen hat und außerdem anders zu bedienen ist.

Das Listing 6.11 zeigt ein Beispiel für die entsprechende Tastaturbelegung im GNU-Emacs. Hierbei ist zu beachten, dass diese Tastaturkombinationen nur unter X in einem für den Emacs auswertbaren Format geliefert werden. Daher wird diese Belegung so nicht auf der Konsole funktionieren.

```
(global-set-key [C-up]   'backward-sentence)  ; M-a
(global-set-key [C-down] 'forward-sentence)   ; M-e
(global-set-key [M-up]   'backward-paragraph) ; M-{
(global-set-key [M-down] 'forward-paragraph)  ; M-}
```

Listing 6.11 Tastaturbelegung von GNU-Emacs

Das Listing 6.12 zeigt die Anpassungen für XEmacs. Diese Syntax ist übersichtlicher als die des GNU-Emacs, es ist jedoch nicht zu erwarten, dass hier die beiden Entwicklungslinien wieder zusammenfließen werden.

```
(global-set-key [(control prior)]  'beginning-of-buffer)
(global-set-key [(control next)]   'end-of-buffer)
```

```
(global-set-key [(meta up)]         'backward-paragraph)
(global-set-key [(meta down)]       'forward-paragraph)
```

Listing 6.12 Tastaturbelegung für den XEmacs

Aktuelle Emacs-Versionen enthalten die Funktion `kbd`, die eine Zeichenkette, wie sie in der Emacs-Dokumentation verwendet wird, in das jeweilige interne Format umwandelt. Damit ist es wieder einfacher geworden, Modes oder Anpassungen für beide Emacs-Varianten zu entwickeln. Ein Beispiel für den Einsatz dieser Funktion finden Sie in Listing 6.13.

```
(define-key message-mode-map (kbd "C-c s")
'message-change-subject)
```

Listing 6.13 Portable Tastaturbelegung

Eine Erweiterung des `text-mode` gegenüber dem `fundamental-mode` ist die Lisp-Funktion `ispell-complete-word` (Meta-TAB), mit der ein Wort automatisch ergänzt wird, wenn es im `ispell`-Wörterbuch enthalten ist. Das verwendete Wörterbuch wird mit der Funktion `ispell-change-dictionary` eingestellt. Die aktuelle Version von `ispell` kann im Zusammenspiel mit einem deutschen Wörterbuch auch für deutsche Texte verwendet werden.

Der `text-mode` stellt bereits einfache Funktionen zur Textformatierung bereit:

- TAB, um zum nächsten Tabulator-Stop zu springen,
- Meta-s, um eine Zeile zu zentrieren, und
- Meta-S zum Zentrieren des ganzen Absatzes.

Auf meinem System ist der `auto-fill-mode` aktiviert (siehe Listing 6.4). Für die Bearbeitung normaler Texte ist dies sinnvoll, bei der Bearbeitung von Konfigurationsdateien oder anderen speziell formatierten Texten, wie z. B. Mail-Headern, ist diese Funktion eher hinderlich.

6.13.3 Der dired-mode

Der `dired`-Modus (directory-editor) stellt ein Verzeichnis in einem Emacs-Puffer dar, in dem dieses Verzeichnis »editiert« werden kann. Dateien können angezeigt, umbenannt, editiert oder kopiert werden. Darüber hinaus stehen noch etliche weitere Funktionen wie `chmod` zur Verfügung. Der `dired`-Mode kann auf verschiedene Arten erreicht werden:

- durch den Aufruf des Emacs mit einem Verzeichnis als Argument, z. B. mit dem Befehl `emacs .` im einfachsten Fall,
- durch das Aufrufen der Funktion `dired` (mit Meta-x `dired`),

- durch Strg-x d (nicht zu verwechseln mit Strg-x Strg-d, was zu einem kurzen Directory-Listing führt) oder
- durch den Aufruf des entsprechenden Menüs.

Der Puffer zeigt im `dired`-Mode in etwa das, was auf einer Textkonsole durch den Befehl `ls -la` ausgegeben wird. Das Format der Ausgabe kann mit der Variablen `dired-listing-switches` verändert werden. Das Ändern des Textes in diesem Puffer ist nicht sinnvoll, so dass viele Tasten mit speziellen Kommandos belegt werden. Eine Übersicht über die wichtigsten Kommandos finden Sie in Tabelle 6.2. In diesem Puffer kann wie in einem normalen Textpuffer nach Strings (z. B. Dateinamen) gesucht werden.

Der `dired`-Modus arbeitet mit Markierungen der Dateien. Ist keine Datei markiert, so wird die aktuelle Datei verwendet. Die aktuelle Datei ist durch die Cursor-Position bestimmt. Marken werden durch die Taste m gesetzt. Eine (unvollständige) Übersicht über mögliche Tastenkombinationen bietet Tabelle 6.2 die komplette Dokumentation finden Sie im Info-System.

Taste	Funktion
g	Neuaufbau des Verzeichnisses
k	Löschen der Datei aus dem Listing
M-k	Löschen aller markierten Dateien aus dem Listing
m	Markieren der Datei oder des Verzeichnisses
u	Aufheben der Markierung
M-DEL	Aufheben aller Markierungen
c	Datei kopieren
r	Datei umbenennen
d	Datei zum Löschen markieren
x	Löschen ausführen
X	Löschen der mit * markierten Dateien
#	Löschmarkierung der Auto-Save-Dateien
~	Löschmarkierung der Backup-Dateien
!	Shell-Kommando ausführen
C / U	Komprimieren/Dekomprimieren der Datei
M	Zugriffsrechte der Datei ändern
O	Eigentümer der Datei ändern
G	Gruppenzugehörigkeit ändern
f	Editieren der Datei (oder e)
v	Anzeige der Datei

Tabelle 6.2 Die wichtigsten `dired`-Kommandos

Darüber hinaus kennt `dired` weitere Kommandos zur Bearbeitung von Verzeichnissen (Erzeugen, Einblenden in das Listing). Mit Hilfe des `dired`-Modus kann also die komplette Dateiverwaltung erledigt werden; der Einsatz eines anderen Tools ist nicht notwendig. Man bleibt also in einer gewohnten Umgebung und die Dateiverwaltung ist im Vergleich zu vielen X-Filemanagern sehr schnell. Weitere Informationen über die Verwendung von `dired` und die Konfigurationsmöglichkeiten finden Sie im Info-System.

Viele moderne Dateimanager sind unter X implementiert – für die Bearbeitung von lokalen Dateien sicher keine schlechte Idee. Wenn Sie aber mittels `ssh` auf einem entfernten Rechner angemeldet sind, dann ist ein Dateimanager im Textmodus manchmal wirklich nützlich. Und in Zusammenarbeit mit `ange-ftp` können Sie auch Dateien mittels `ftp` verwalten. Der Dateimanager `mc` (Midnight Commander) ist in der Oberfläche dem bekannten Norten Commander nachempfunden und hat eine ähnliche Funktionalität.

6.13.4 Dateien auf anderen Rechnern mit ange-ftp bearbeiten

Der Editor Emacs ist nicht nur in der Lage, lokal gespeicherte Dateien zu bearbeiten. Er stellt sowohl eine Oberfläche für das `ftp`-Programm bereit als auch die Möglichkeit, Dateien auf entfernten Rechnern zu bearbeiten. Dazu wird eine `ftp`-Verbindung zum betreffenden Rechner aufgebaut und das dortige Verzeichnis in einem `dired`-Puffer angezeigt. Damit kann auch auf entfernten Rechnern komfortabel mit Dateien gearbeitet werden.

Im einfachsten Fall sind keine weiteren Konfigurationen notwendig. Der Name der zu bearbeitenden Datei wird dabei in der Form */Benutzer@Rechner:Datei* angegeben. `ange-ftp` fragt nach dem Passwort des Benutzers und überträgt die Datei auf den lokalen Rechner. Anschließend kann die Datei wie gewohnt bearbeitet werden und wird beim Speichern wieder mittels `ftp` auf den entfernten Rechner übertragen.

Der Mode arbeitet mit dem `dired`-Modus zusammen, so dass es auch möglich ist, auf entfernten Rechnern Dateiverwaltung zu betreiben – eine Datei mit `ftp` zu übertragen funktioniert genauso wie das lokale Kopieren. Wenn man als Benutzernamen `ftp` oder `anonymous` angibt, kann man sich natürlich auch auf öffentlichen `ftp`-Servern umsehen.

Sollte die Verbindung aufgrund eines Time-Outs unterbrochen werden, so baut `ange-ftp` diese bei Bedarf automatisch wieder auf. Für den Benutzer stellt sich die entfernte Datei wie ein ganz normaler Puffer dar. Es können also alle Kommandos verwendet werden, die auf Emacs-Puffern arbeiten. Der einzige Unterschied ist die langsamere Übertragung der Dateien über das Netzwerk.

Falls Sie entfernte Rechner nur über ein Gateway erreichen, so können Sie trotzdem `ange-ftp` verwenden. Dazu müssen Sie den Namen des Gateways in die Variable `ange-ftp-gateway-host` eintragen. Mehr zur Konfiguration eines Gateways finden Sie in der Info-Dokumentation.

Auch wenn Sie keinen Internetanschluss haben, kann die Verwendung von `ange-ftp` sinnvoll sein. Viele Benutzer arbeiten in der Regel nur unter X und verwenden (dies ist der Standardwert unter Linux) `xauth` zur Autorisierung der Clients. Damit kann aus einem `xterm` nach dem Befehl `su` zum Erlangen der Superuser-Privilegien kein X-Client gestartet werden. Entweder kann man den Magic-Cookie (siehe auch Abschnitt 7.8.2, »Benutzerbezogenen Zugriff erlauben mit `xauth`«) mittels `xauth` übertragen oder die Zugriffe aller lokalen Clients mit `xhost` erlauben.

In einem größeren Netz ist die Verwendung von `xhost` nicht empfehlenswert und die Übertragung der X-Authority nicht besonders bequem. In diesem Fall können Sie die Anmeldung des Benutzers `root` mittels `ftp` gestatten und `ange-ftp` verwenden. Die meisten Rechner verbieten jedoch aus Sicherheitsgründen den `ftp`-Zugriff als `root`. Mein Tipp in diesen Fällen ist die Verwendung der `ssh`, mehr dazu finden Sie im Kapitel 18, »Die Secure Shell ssh«.

6.13.5 Der outline-mode

Bei der Bearbeitung von größeren Dateien verliert man leicht den Überblick und blättert dann nur noch wild hin und her. Ein weiteres Problem ist, dass die inhaltliche Struktur sich in der Datei kaum widerspiegelt. Abhilfe kann hier der `outline-mode` schaffen. Dieser Modus kann sowohl als Major- als auch als Minor-Modus (`outline-minor-mode`) verwendet werden, so dass auch innerhalb von Lisp-, LaTeX- oder C-Dateien die Funktionen des `outline-mode` benutzt werden können.

Die Emacs-Funktion zur Anzeige der NEWS (Strg-h n, `view-emacs-news`, also der Änderungen in der aktuellen Version) verwendet den `outline-mode`. In diesem Buffer können Sie nur die Überschriften anzeigen lassen und nur von einzelnen Abschnitten die verdeckten Knoten aufklappen und lesen. Die Funktionen lassen sich auch mit Hilfe des Menüs aufrufen; Tabelle 6.3 enthält eine Übersicht über die wichtigsten Tastenkürzel.

Die Entscheidung, was eine Überschrift ist und in welcher Reihenfolge Überschriften ineinander geschachtelt sind, wird mit der Variablen `outline-regexp` beeinflusst. Zeilen, die am Zeilenanfang auf den regulären Ausdruck passen, sind Überschriften. Dabei sind längere Treffer (Matches) untergeordnete Überschriften.

Tasten	Funktion	Bedeutung
Strg-c C-n	outline-next-visible-heading	Zur nächsten sichtbaren Überschrift
Strg-c C-p	outline-previous-visible-heading	Zur vorherigen sichtbaren Überschrift
Strg-c Strg-f	outline-forward-same-level	Analog in der aktuellen Ebene
Strg-c Strg-b	outline-backward-same-level	
Strg-c Strg-u	outline-up-heading	In der Hierarchie aufsteigen
Strg-c Strg-t	hide-body	Verstecken aller Texte
	show-all	Alles im Buffer anzeigen
Strg-c Strg-d	hide-subtree	Text und Unterknoten unsichtbar machen
Strg-c Strg-s	show-subtree	Text und Unterknoten anzeigen
Strg-c tab	show-children	Nur direkte Subtitel anzeigen
Strg-c Strg-c	hide-entry	Den folgenden Text unsichtbar machen
Strg-c Strg-e	show-entry	Den folgenden Text ausklappen
Strg-c Strg-l	hide-leaves	Text verstecken, Subtitel bleiben sichtbar
Strg-c Strg-k	show-branches	Alle Titel anzeigen, aber keinen Text

Tabelle 6.3 Tastaturbelegung des `outline-mode`

Für spezielle Anpassungen stellt der `outline-mode` den Hook `outline-mode-hook` zur Verfügung. Dieser Hook wird, sofern definiert, nach dem `text-mode-hook` aufgerufen.

6.13.6 Weitere Emacs-Modi

Neben den bisher vorgestellten Modi existieren noch viele weitere, die bereits im Standardumfang von Emacs enthalten sind. Dazu zählt z. B. ein Modus zum Anzeigen von Manpages (`man`), der `shell-mode` oder ein Kalender (`calendar`). Jeder dieser Modi ist im Lisp-Quellcode verfügbar und flexibel konfigurierbar. Beispiele für Modi, die ich auf meinem Rechner einsetze, sind:

- `Gnus` als Mail- und Newsreader; besondere Features sind das Splitten der Mail in verschiedene Gruppen (z. B. Mailing-Listen), das automatische Löschen von Mail aus Mailing-Listen, das Lesen von entfernten News-Servern und ausgefeiltes Scoring.
- `bbdb` (Big Brothers Database) als kleine Adressverwaltung, die in `Gnus` integriert ist und dort Mail-Adressen und Signaturen verwalten kann.
- `psgml` zum Editieren von SGML- und XML-Dokumenten, z. B. von HTML-Seiten (ist beim XEmacs im Standardumfang enthalten). Es können aber beliebige DTDs verarbeitet werden, für dieses Buch habe ich die Docbook DTD (*http://docbook.sf.net/*) verwendet.
- `calendar/diary` als Terminkalender mit Warnfunktion.
- `zenirc` als IRC-Client.
- Modi für verschiedene Programmiersprachen wie C, Lisp, Java Perl, AWK oder COBOL.
- `shell-mode` als Ersatz für die History-Funktion in anderen Programmen.
- `ispell` zur Rechtschreibprüfung, erweitert um `flyspell` zur Rechtschreibprüfung während des Schreibens.

Eine ausführliche Dokumentation zu diesen Modi finden Sie im Info-System und in den Lisp-Quellen der Modi. Selbst wenn Sie Lisp nicht beherrschen, finden Sie in den Kommentaren viele Hinweise auf Konfigurationsmöglichkeiten und neue Ideen.

6.14 Nützliche Minor-Modi

Zusätzlich zu dem Major-Modus kann jeder Puffer noch in eine Reihe von Minor-Modi versetzt werden, die spezielle Funktionen bereitstellen können, die in mehr als einem Modus nützlich sein können. Im folgenden Abschnitt möchte ich einige dieser Minor-Modi vorstellen, die das Leben eines Unix-Anwenders deutlich erleichtern können.

6.14.1 Der auto-fill-mode

Viele Benutzer möchten, dass ein eingegebener Text automatisch am Ende der Zeile umbrochen wird. Im `auto-fill-mode` wird beim Drücken der Taste [Space] oder [Return] umbrochen, wenn der Cursor hinter der Position `fill-column` steht. Soll trotz `auto-fill-mode` nicht automatisch umbrochen werden, so muss vor der Leer- oder Enter-Taste [Strg]-[q] (`quoted-insert`) gedrückt werden. Damit wird die besondere Bedeutung der Tasten aufgehoben.

Den rechten Rand können Sie betimmen, indem Sie mit dem Cursor auf die passende Spalte gehen und dann Strg-x f (set-fill-column) drücken. Wenn Sie eine bestimmte Spalte angeben, so können Sie diesen Parameter mit Hilfe eines Präfixarguments bestimmen. Mit Strg-u leiten Sie das Präfix ein, geben die gewünschte Zahl ein und drücken dann die gewünschte Tastenkombination. Die Angabe eines Präfixes ist bei vielen Funktionen möglich, z. B. bei der Cursorbewegung.

Wenn Sie einen linken Rand haben möchten, so können Sie ein Fill-Präfix definieren. Schreiben Sie einmal das Präfix an den Anfang einer Zeile, stellen Sie den Cursor an das Ende des Präfixes und drücken Sie Strg-x . (set-fill-prefix). Dieses Präfix wird bei allen neu eingefügten Zeilen automatisch aufgenommen. Auch der Befehl Meta-q (fill-paragraph) beachtet diese Einstellung.

Eine weitere nützliche Funktion ist Meta-s (center-line) zum Zentrieren einer Zeile. Zusätzlich gibt es noch die Funktionen center-paragraph und center-region, die einen Absatz bzw. den markierten Bereich (die Region) zentriert darstellen.

Manchmal kann es sinnvoll sein, einen Text im Blocksatz abzuspeichern. In der Regel verwendet man für diese Zwecke zwar eine Textverarbeitung, aber auch Emacs hat die entsprechenden Funktionen. Mit set-justification-*Option* können die Absätze in der Region formatiert werden. Gültige Werte für *Option* sind left, right, center, full oder none.

6.14.2 Der abbrev-mode

Ein weiterer nützlicher Modus ist der abbrev-mode, ein Minor-Modus, der selbst definierte Abkürzungen expandiert und damit die Tipparbeit erheblich vermindern kann. So kann z. B. die Eingabe **mfg** Meta-Space zu »mit freundlichen Grüßen« expandiert werden. Dabei können sowohl global (d. h. für alle Modi) gültige als auch modusspezifische Abkürzungen definiert werden.

Eine globale Abkürzung kann mit Strg-x a g (add-global-abbrev) definiert werden. Diese Funktion liest das unmittelbar vor dem Point stehende Wort ein und fragt im Minibuffer eine Abkürzung dafür ab. Wollen Sie mehrere Wörter (z. B. drei) abkürzen, so müssen Sie die Anzahl mit einem Präfix angeben: Strg-u 3 Strg-x a g. Modusspezifische Abkürzungen können Sie mit der Tastenkombination Strg-x a l (add-mode-abbrev) definieren.

Abkürzungen werden in der Datei ~/.abbrev_defs gespeichert. Den Namen der Datei können Sie verändern, indem Sie den gewünschten Dateinamen in der Variablen abbrev-file-name speichern. Mit dem Emacs-Kommando write-abbrev-file speichern Sie die erfassten Abkürzungen für die nächste Sitzung.

Befindet sich der Puffer nicht im abbrev-mode, so werden Abkürzungen nur mit der Tastenkombination Meta-Space expandiert. Befindet sich der Puffer im abbrev-mode, so wird die Abkürzung expandiert, wenn unmittelbar danach ein Leerschritt, ein Punkt oder ein anderes Interpunktionszeichen eingegeben wird. Dies ist nicht immer erwünscht, so dass die Expandierung mit der Tastenkombination Strg-q (quoted-insert) verhindert werden kann. Wurde eine Abkürzung unbeabsichtigt expandiert, so können Sie das mit der Tastenkombination Strg-x u (advertised-undo) oder dem Befehl Meta-x unexpand-abbrev rückgängig machen.

Um einen Überblick über die definierten Abkürzungen zu bekommen, verwenden Sie die Funktionen list-abbrevs und edit-abbrevs. Weitere Informationen zur Verwendung von Abkürzungen finden Sie im Info-System.

Die bisher vorgestellten Abkürzungen werden zwar vom Benutzer erzeugt, sind aber relativ statisch. Emacs kennt auch so genannte »dynamische Abkürzungen«. In diesem Fall wird im Puffer ein Wort gesucht, das mit derselben Buchstabenfolge beginnt, wie der zu expandierende Begriff. Die Erweiterung erfolgt aber nicht dynamisch, sondern muss mit Meta-/ (dabbrev-expand) manuell ausgelöst werden.

6.14.3 Rechtschreibprüfung »on the fly«

Emacs wird nicht nur zum Programmieren benutzt, sondern auch zum Schreiben von Dokumentationen, Mails, Texten oder Büchern. In Zusammenarbeit mit dem externen Programm ispell kann der Text innerhalb von Emacs auf korrekte Rechtschreibung geprüft werden. In der Standardversion wird Emacs ein englisches Wörterbuch verwenden. Mit den in Listing 6.14 dargestellten Befehlen kann auf ein anderes Wörterbuch umgeschaltet werden.

```
; Wörterbuch 'ngerman8' verwenden (mit ISO-Umlauten)
(setq ispell-dictionary "german8")
```

Listing 6.14 Verwendung eines deutschen Wörterbuchs

ispell verwendet ein einfaches Wörterbuch sowie eine Affix-Datei, in der beschrieben wird, wie konjugiert und dekliniert wird. Damit ist ispell in der Lage, viele Texte auf korrekte Rechtschreibung zu prüfen. Es erkennt jedoch keine Grammatikfehler oder die Verwendung eines falschen Falls oder der falschen Zeit. Dafür wäre eine wesentlich ausgefeiltere Methode der Texterkennung notwendig, für die derzeit keine freien Programme zur Verfügung stehen.

Vollständige Texte kann man mit ispell-buffer prüfen. Das ist, gerade bei größeren Texten, aufwändig und fehlerträchtig und außerdem ziemlich langweilig. Bei Mail- oder News-Artikeln kann man sich behelfen, indem man die Rechtschreibprüfung im message-send-hook automatisch aufruft. Man braucht sich

dann nicht mehr dazu zu zwingen, Artikel von `ispell` prüfen zu lassen, es kann aber dennoch aufwändig sein. Bei anderen Modi gewinnt man mit diesem Verfahren gar nichts.

Besser ist es, wenn Emacs den eingegebenen Text bei der Eingabe sofort (»on the fly«) überprüft und möglicherweise falsch geschriebene Wörter hervorhebt. Diese Funktion kennt man von neueren Word-Versionen. Wenn man sich erst einmal daran gewöhnt hat, dann ist `flyspell` eine echte Hilfe. `flyspell` ist nicht im Emacs enthalten, sondern nur als Zusatzpaket unter *http://kaolin.unice.fr/~serrano* zu erhalten. Eine Zusatzfunktion prüft auf Wiederholungen von Wörtern und markiert beispielsweise »der der«.

`flyspell` arbeitet mit dem externen Programm `ispell` zusammen, daher muss auch hier wieder das passende Wörterbuch ausgewählt werden. Anschließend muss der `flyspell-mode` in den gewünschten Modi aktiviert werden. Das erreichen Sie durch den manuellen Aufruf von `fly-spell-mode` oder mit dem entsprechenden Hook. Mit der Funktion `global-flyspell-mode` kann der `flyspell-mode` für alle Modi eingeschaltet werden, die in der Variablen `flyspell-global-modes` gespeichert sind. In Listing 6.15 finden Sie ein Beispiel.

```
; Default-Dictionary für flyspell (neue deutsche Rechtschreibung)
(setq flyspell-default-dictionary "ndeutsch8")
; Flyspell für den Message-Mode, SGML-Mode und LaTeX-Mode
(global-flyspell-mode)
(setq flyspell-global-modes
      '(message-mode sgml-mode latex-mode))
```

Listing 6.15 Aktivieren der Rechtschreibprüfung in einigen Modi

So angenehm wie der `flyspell-mode` ist, er hat auch seine Nachteile. Zunächst ist nicht jeder Text für eine Rechtschreibprüfung geeignet, insbesondere wenn zu viele Fremdwörter, Abkürzungen oder Namen nicht im Wörterbuch enthalten sind. Die Erstellung eines privaten Wörterbuchs hilft hier etwas weiter, ist aber nicht der Weisheit letzter Schluss. Ich verwende häufig Datei-lokale Wörterbücher, die `ispell-mode` am Ende der Datei einfügt. Ein weiteres Problem ist, dass der Aufruf von `ispell` nach praktisch jedem Wort auf langsamen Rechnern störend wirkt.

Der wesentliche Vorteil von `flyspell` ist die unauffällige Prüfung des Textes schon während der Erfassung, so dass ein lästiger zusätzlicher Schritt entfällt. Selbst auf meinem langsamen Laptop ist dieser Modus aktiviert, da die Vorteile die Nachteile bei weitem überwiegen.

6.15 Sonstige Erweiterungen

Die Vorstellung einiger mit Emacs mitgelieferter Modi sollte nun einen Vorgeschmack darauf geliefert haben, was mit Emacs alles möglich ist. In den folgenden Kapiteln wird immer dann, wenn für einen bestimmten Zweck ein Emacs-Modus verfügbar ist, dieser zumindest erwähnt.

Da es weltweit viele Emacs-Benutzer gibt, wurden auch sehr viele Emacs-Lisp-Erweiterungen geschrieben. Um die lokal entstandenen Funktionen und Modi auch anderen zugänglich zu machen, können diese im Emacs-Lisp-Archiv (*ftp://ftp.elisparchive.net/pub/elisp/*, *ftp://ftp.cis.ohio-state.edu/pub/emacs-lisp/* und *news:gnu.emacs.sources*) abgelegt werden. Dort sollten Sie beinahe alles finden, was Sie jemals zu Emacs an Erweiterungen suchen. Auch zur Suche in diesem Archiv gibt es einen speziellen Emacs-Modus.

7 Das X-Window-System

Irgendwann einmal, dann wird vielleicht auch X11 unter Linux laufen
Running Gag unter Linux-Entwicklern, bevor X11 portiert war

7.1 Das Konzept von X

Das *X-Window-System*[1] ist *das* plattformunabhängige und netzwerktransparente Grafiksystem. X ist plattformunabhängig, weil es Portierungen für alle verbreiteten Plattformen gibt. Ursprünglich unter Unix und VMS eingesetzt, gibt es Versionen für Windows jeder Ausprägung und für OS/2. X ist netzwerktransparent, d. h., Programme können auf einem anderen Rechner ablaufen als dort, wo die Anzeige stattfindet. Welche Prozessoren oder Betriebssysteme eingesetzt werden, ist vollkommen unerheblich.

X basiert auf einem Client/Server-Konzept. Der X-Server ist ein Programm, das die Verwaltung des Bildschirms und der Eingabegeräte wie Tastatur und Maus übernimmt und diese als Dienste den X-Clients anbietet. Ein X-Client ist ein Programm, das die Dienste des X-Servers in Anspruch nimmt, indem es Fenster auf dem Bildschirm darstellen lässt oder auf Eingaben wartet. Für viele Anwender verhält sich dies genau andersherum als gewohnt, wo der Arbeitsplatz als Client und der andere Rechner als Server bezeichnet wird.

Die Kommunikation zwischen dem X-Server und den X-Clients findet mit Hilfe des X-Protokolls statt. Dieses kann lokal auf einem Rechner verwendet werden oder mittels TCP/IP im Netz übertragen werden. Andere Protokolle sind möglich, spielen aber heute praktisch keine Rolle mehr. Unter X versteht man zwei Dinge, einmal das Protokoll, mit dem Anwendungen und X-Server miteinander kommunizieren, und die Beispielimplementation, die bei der Opengroup erhältlich ist.

7.2 »Look and Feel« unter X

X erzwingt kein spezielles Look&Feel der Oberfläche. Fenster sind Flächen auf dem Bildschirm, das Aussehen und die Art der Bedienung wird von den X-Clients selbst bestimmt. Die Rahmen um die Fenster werden weder vom Client noch vom X-Server verwaltet. Hierfür ist ein spezieller Client, der *Window-Manager* verantwortlich. X stellt die Technologie bereit und schreibt keine Policy vor. Dadurch ist X sehr flexibel, das hat aber auch Nachteile. In der Konsequenz sehen die Clients (Anwendungsprogramme) unter X nicht so einheitlich aus, wie man das z. B. vom Macintosh oder von MS-Windows her kennt.

1. Oft falsch als X-Windows bezeichnet.

Es gibt im Standard-X keinen Style-Guide. Die frei verfügbare X-Implementation und die zur Programmierung mitgelieferten Bibliotheken sind oft nur Beispielimplementationen. Insbesondere ist die Programmierung mit der Xlib oder dem X-Toolkit aufwändiger, als man das von anderen Grafiksystemen her kennt.

Als Beispiel für ein Toolkit zur Oberflächenprogrammierung wurden die »Athena-Widgets« mitgeliefert. Diese sind vom Look her heute nur als veraltet zu bezeichnen, das Feel ist gewöhnungsbedürftig. Obwohl manche Ideen, wie z. B. die Bedienung der Scroll-Balken, erheblich von dem heute üblichen Schema abweichen, ist eine schnelle und präzise Verwendung möglich. Aus den verschiedensten Gründen hat sich dieses Beispiel nicht als Standard durchsetzen können, aber viele freie X-Programme verwenden diese Benutzeroberfläche. Es gibt zwei spezielle Anpassungen, einmal für das Look&Feel von Windows 95 und einen 3D-Look. Die gängigen Distributionen enthalten diese Bibliotheken als Option, heute bevorzugen Programmierer aber oft andere Toolkits.

Eine (noch) verbreitete Oberfläche ist OpenLook, das von Sun entwickelt wurde und heute frei (auch für Linux) verfügbar ist. Sun entwickelt diese Oberfläche und die zugehörigen Schnittstellen nicht weiter, so dass praktisch keine neuen Anwendungen für OpenLook entwickelt werden.

Die meisten kommerziellen Systeme verwenden heute *OSF/Motif* als Benutzeroberfläche, das aber nicht frei verfügbar ist. Für freie Unix-Systeme kann kostenfrei OpenMotif verwendet werden, dies erfüllt aber nicht die in den Debian Free Software Guidelines festgelegten Regeln. Es existiert jedoch mit `lesstif` eine freie Implementierung des Motif-Standards (Version 1.2), die unter praktisch allen Plattformen lauffähig ist. In der freien Software hat Motif sich nicht durchsetzen können.

Das *Common Desktop Environment* (CDECDE) ist der Desktop-Standard bei kommerziellen Unix-Systemen. Diese Benutzeroberfläche basiert auf Motif, enthält jedoch einige zusätzliche Features und Programme. Auch dieses Produkt ist kommerziell, es ist jedoch möglich, eine Linux-Portierung zu erwerben. Unter Linux und den *BSD-Systemen hat CDE praktisch keine Bedeutung, da es zusätzlich zu erwerben ist und andere Toolkits zur Verfügung stehen. Vorteilhaft ist der Einsatz von CDE nur, wenn man auf Kompatibilität zu kommerziellen Systemen angewiesen ist.

Neben diesen Toolkits gibt es noch eine ganze Reihe von anderen Bibliotheken, die oft ein eigenes Look&Feel implementieren. Das hat zur Folge, dass viele verschiedene Bibliotheken auf dem System installiert sein müssen, Platz im Hauptspeicher belegen und Anwender durch das unterschiedliche Aussehen und die verschiedenartige Bedienung verwirrt werden. Eine Konzentration der freien Unix-Systeme auf CDE ist durch die Einschränkungen der kommerziellen Lizenz nicht zu erwarten. Der Trend geht eher in die andere Richtung, Sun wird GNOME als Desktop für Solaris ausliefern.

Die Probleme sind vielen Entwicklern bekannt und behindern diese bei der Implementation von portabler und benutzerfreundlicher Software. Das KDE-Projekt (The K Desktop Environment, *http://www.kde.org/*) hat einen kompletten Desktop implementiert. Hierbei steht im Vordergrund, dass die Programme sich einheitlich bedienen lassen, einfach mit Hilfe von Menüs zu konfigurieren sind und dem Anwender den ganzen Komfort eines modernen GUI bieten. Im Rahmen des KDE-Projekts wurde nicht nur ein Desktop mit vielen kleinen Tools implementiert und in verschiedene Sprachen übersetzt, sondern auch eine Reihe von Bibliotheken zur Vereinfachung der Anwendungsentwicklung und mit der Implementation von größeren Programmen begonnen. Insbesondere in das Office-Paket K-Office setzt man große Erwartungen.

Dabei soll aber nicht vergessen oder darüber hinweggetäuscht werden, dass unterhalb von KDE ein Unix steckt. Der Anwender soll die Vorteile diese Betriebssystems natürlich weiter nutzen dürfen. KDE ist nicht nur unter Linux lauffähig, obwohl es in erster Linie dort entwickelt wird. Damit steht zum erstenmal eine freie, stabile, konsistente und moderne Benutzeroberfläche als freie Software zur Verfügung.

Einige Anwender waren mit den Lizenzbedingungen, die die Weitergabe einer modifizierten `qt`-Version untersagten, nicht zufrieden und haben ein weiteres Desktop-Projekt entwickelt: GNOME (GNU Network Object Model Environment, *http://www.gnome.org/*). Dieses Projekt basiert auf dem `gtk`-Toolkit, das im Rahmen von The GIMP entstanden ist. Heute sind beide Desktop-Umgebungen frei im Sinne der DFSG, so dass hier eine lebhafte Konkurrenz existiert.

Zu beiden Desktop-Projekten gehört ein Style-Guide, der vorschreibt, wie Applikationen aussehen und wie sie zu bedienen sind. Das ist einer der ersten Schritte zu einem konsistenten Desktop, der auch für Einsteiger einfach zu bedienen ist. Für Anwender, denen die Desktops KDE und GNOME zu groß sind, ist vielleicht XFce (*http://www.xfce.org/*) eine gute Alternative.

Es ist zu hoffen, dass Programme, die für das eine System entwickelt wurden, auch weiterhin unter der anderen Oberfläche lauffähig bleiben und dass die Entwickler sich nicht verzetteln. Der Anwender hat heute die Wahl zwischen zwei ausgereiften und leistungsfähigen Desktop-Oberflächen. Welche Auswirkungen das in der Zukunft auf die kommerziellen Systeme haben wird, bleibt abzuwarten.

7.3 Die Entwicklung von X

Das X-Window-System wurde ab 1984 am Massachusetts Institute of Technology (MIT) im Rahmen des Projekts »Athena« in Zusammenarbeit mit der Digital Equipment Corporation entwickelt. Anfang 1986 gab es die Vorläuferversion

von X11, das X10; die erste Version des X11-Systems, X11R1, wurde im September 1987 veröffentlicht. Die aktuelle Version ist X11R6.4. Das *X-Konsortium*, das die Entwicklung von X übernommen hatte, hat diese nun eingestellt und mit neuen Projekten auf der Basis von X begonnen. Derzeit wird X von »The Open Group« (*http://www.opengroup.org*) weiterentwickelt. Diese Organisation wird von vielen Herstellern mitgetragen, die bereits am X-Konsortium beteiligt waren.

Das Ziel der X11-Entwicklung ist die Schaffung von Standardschnittstellen auf allen Ebenen des X-Window-Systems im Rahmen einer firmenübergreifenden Kooperation. In der »Open Group«, die die Entwicklung von X steuert und koordiniert, sind praktisch alle im Unix-Umfeld aktiven Firmen vertreten, so dass hier eine einheitliche Entwicklung sichergestellt ist. So arbeiten Clients und Server der verschiedenen Hersteller auch unter verschiedenen Versionen von X gut zusammen.

X ist zunächst nur die Spezifikation eines Protokolls zur netzwerktransparenten Darstellung und Übertragung von Grafiken und Events. Dieses Protokoll ist in der `Xlib` implementiert und für alle Interessierten öffentlich zugänglich dokumentiert. Das vom X-Konsortium vertriebene Release ist eine Beispielimplementierung des Protokolls und einiger X-Server. Zunächst entwickelten die beteiligten Firmen nur Beispiel-Server für ihre eigene Hardware; die Zahl der Server wuchs jedoch mit jedem neuen Release.

1988 übernahm das X-Konsortium mit dem Release R2 die Federführung bei der X11-Entwicklung vom MIT. Am X-Konsortium sind viele namhafte Firmen beteiligt, neben DEC auch IBM, Sun und Hewlett Packard. Die Liste der Mitglieder enthält fast alles, was in der Computerwelt Rang und Namen hat. Die meisten Aktivitäten des X11-Konsortiums werden per E-Mail abgewickelt. Die vorgeschlagenen Spezifikationen werden einem öffentlichen »review« unterzogen, das an das RFC-Verfahren im Internet erinnert. Interessenten erhalten unter *http://www.x.org/ftp/pub/DOCS/XConsortium* nähere Informationen.

In dieser Zeit war X praktisch ausschließlich für (größere) Unix-Workstations verfügbar. X stellte (für die damalige Zeit) hohe Anforderungen an die Hardware und das Betriebssystem. Im Mai 1990 publizierte Thomas Röll als Teil des X386-Projekts einen frei verfügbaren X11R4-Server für PC-Unix-Systeme. Auf diesem Server basierend wurde von der Firma SGCS (Snitily Graphics Consulting Services), für die Thomas Röll anschließend arbeitete, eine Folgeversion erstellt, die jedoch recht instabil war.

Daraus entwickelte sich das »The XFree86 Project«, das die Version 1.2 des X386-Servers verbessern wollte. Heute ist XFree86 für viele unterschiedliche Betriebssysteme frei verfügbar. Einige kommerzielle Unix-Hersteller liefern statt einer eigenen Version von X das XFree86-System aus oder verweisen bei Problemen mit der eigenen Implementation auf möglicherweise neuere X-Server aus

dem XFree86-Projekt. Um an der Weiterentwicklung von X teilhaben zu können, wurde die non-profit-Organisation »The XFree86 Project, Inc.« gegründet, die Mitglied im X-Konsortium wurde. Das XFree86-Projekt finanziert sich ausschließlich durch Spenden. Mehr Informationen zu XFree86 finden Sie im WWW unter *http://www.xfree86.org*.

X-Server gibt es aber nicht nur für Unix. Auch für OS/2 und Windows existieren Portierungen, die durchaus benutzbar sind. Wenn Sie jedoch meistens mit Unix-Systemen arbeiten und Ihren PC nur als X-Terminal verwenden, dann ist XFree86 sicher die bessere Wahl im Vergleich zu einem Windows mit einem entsprechenden Aufsatz.

Die X-Software unterliegt nicht der GPL, sie ist auch nicht Public Domain – das Copyright liegt bei den verschiedenen, an der Entwicklung beteiligten Organisationen. Trotzdem ist die X-Distribution frei verfügbar und kann ohne Lizenzgebühren weitergegeben werden. Hierin bestehen deutliche Parallelen zu den Regelungen der GPL. Abweichend von der GPL können diese Programme auch in proprietären Systemen verwendet werden, ohne dass der Quellcode verfügbar sein muss. Einzelne Anwendungs- und Zusatzprogramme, insbesondere aus dem `/Contrib`-Bereich, sind aber durchaus Public-Domain-Software, unterliegen der GPL oder anderen Lizenzbestimmungen.

Aufgrund dieser Lizenzbestimmungen und der Leistungsfähigkeit dieses Konzepts wurde X11 zum Standardgrafiksystem für Unix-Workstations. Praktisch jeder Hersteller liefert zu seinem System ein (eventuell) angepasstes X11 mit. Die Quellen der Beispielimplementation sind jedoch frei verfügbar (z. B. bei `ftp.x.org`), so dass einzelne Systemadministratoren das Window-System auch selbst kompilieren und installieren können. Dies ist nicht schwer, da das Verfahren gut dokumentiert ist; es wird allerdings relativ viel Plattenplatz benötigt.

Neben dem X-System selbst vertreibt das X-Konsortium auch einige Programme als »Contributed«-Software. Diese Programme werden nicht vom X-Konsortium gepflegt, sondern nur über diesen Weg vertrieben. Vor jedem neuen Release wird eine Sammlung an Programmen und Tools zusammengestellt, die bereits an die neue Version angepasst sind. Diese werden dann zusammen mit den X-Quellen auf Magnetband bzw. CD-ROM vertrieben. Anwender mit einer guten Internetanbindung können die Programme auch mittels `ftp` von `ftp.x.org` erhalten.

7.4 Konfiguration der XFree86-Server

In früheren Versionen war die Konfiguration von XFree86-Servern eine Wissenschaft für sich. Die X-Server können, wenn man die Konfigurationsdatei `/etc/X11/XF86Config` manuell anpasst, die Hardware (Grafikkarte und Monitor) bis

an die Belastungsgrenzen (und darüber hinaus) ausnutzen. Das setzt jedoch eine genaue Kenntnis der verwendeten Hardware und der Verfahren voraus, denn möglicherweise kann man mit ungültigen Einstellungen seinen Monitor beschädigen. Kurz gesagt, für den normalen Anwender ist dies nicht zu empfehlen.

Distributionen fragen bei der Installation des X-Servers die wichtigsten Parameter der Grafikkarte und des Monitors ab und erstellen in der Regel eine brauchbare Konfiguration. Alternativ können Sie das Skript `xf86config` verwenden. In der SuSE-Distribution ist das Tool `sax` enthalten, das die Konfiguration des X-Servers nochmals vereinfacht.

Wenn Sie hier Werte eingeben, die über die Leistungsfähigkeit Ihrer Grafikkarte bzw. Ihres Monitors hinausgehen, so können Sie diese Geräte beschädigen. Wenn Sie sich jedoch genauer für die Konfiguration interessieren, so können Sie die Datei `XF86Config` mit einem Editor bearbeiten. Das Format und die möglichen Einstellungen sind in der Manpage `XF86Config(5)` dokumentiert.

Eine wichtige Tastenkombination, die Ihnen bei der Konfiguration (und später auch) behilflich sein kann, ist z. B. Strg+Alt+Backspace zum Abbrechen des X-Server-Prozesses. Wenn Ihr Monitor das Bild nicht synchronisieren kann, dann sollten Sie den X-Server unbedingt beenden, um Schäden an Ihrem Monitor zu vermeiden. Moderne Monitore erkennen, dass sie das Bild nicht darstellen können, und schalten sich ab, ältere Bildschirme fangen an zu flimmern oder zu piepsen. In diesem Fall sollten Sie den Server sofort beenden. Die Tastenkombination Strg+Alt+Backspace können Sie mit dem Eintrag `DontZap` im Abschnitt `ServerFlags` in der Datei `/etc/XF86Config` unwirksam machen.

Bei der Konfiguration können Sie (für eine Farbtiefe) verschiedene Auflösungen einrichten. Zwischen diesen Auflösungen können Sie mit Strg+Alt++ bzw. Strg+Alt+- (jeweils im Ziffernblock) umschalten. Zum Wechsel zwischen den verschiedenen Farbtiefen müssen Sie leider den Server beenden und neu starten (z. B. mit `startx -- -bpp 16` für 16 Bit Farbtiefe).

Sie können auch mehrere X-Server auf einem Rechner starten, indem Sie das zu verwendende Display als Parameter angeben. Der erste X-Server wird (automatisch) das Display `:0` erhalten, den zweiten Server können Sie mit `startx -- :1` starten. Dabei können Sie selbstverständlich für jeden Server eine andere Farbtiefe verwenden. Mit dem Programm `Xnest` können Sie einen X-Server starten, der in einem anderen X-Server dargestellt wird. Dies ist nützlich, wenn man zum Beispiel eine gesamte X-Session und nicht nur ein einzelnes Programm mit Hilfe von `ssh` tunneln will. Starten Sie in der `ssh`-Sitzung den Befehl `Xnest :1 -query localhost`, um einen Anmeldebildschirm zu erhalten (das muss der dortige Display-Manager erlauben) und diesen auf einem anderen X-Server darzustellen.

7.5 Software-Konfiguration

Das X-Window-System definiert nur die Schnittstelle zwischen X-Clients und dem X-Server. Über das Aussehen oder die Benutzung (Look&Feel) der Anwendungen oder des Systems werden praktisch keine Annahmen gemacht. Dadurch ist zum einen das System sehr flexibel, zum anderen aber nicht einfach zu konfigurieren.

Die X-Oberfläche kann für zwei Benutzer auf einem System vollkommen unterschiedlich aussehen und auch in der Benutzung deutliche Unterschiede aufweisen. Glücklicherweise ist dies unter Unix problemlos möglich. Für die Betreuung von Anwendern ist es jedoch sinnvoll, wenn der Systemverwalter eine Umgebung vorgibt, die für die meisten Anwender benutzbar ist, damit nicht zu viele Programme und Umgebungen unterstützt werden müssen. Auf kommerziellen Unix-Systemen ist diese Umgebung häufig CDE, unter Linux und anderen freien Systemen werden KDE und GNOME die Favoriten sein. Wenn Sie möchten, dann können Sie das kommerzielle CDE für Linux erwerben oder auch auf anderen Systemen durch KDE oder GNOME ersetzen.

Prinzipiell gibt es zwei Möglichkeiten, das X-Window-System zu starten. Beide Möglichkeiten haben ihre Berechtigung und werden unterschiedlich konfiguriert. Dennoch können auf einem System beide Varianten parallel benutzt werden.

7.5.1 Start des X-Window-Systems mit startx

Jeder Benutzer kann, wenn er im System angemeldet ist, das X-Window-System mit dem Befehl `startx` starten. Dabei wird durch die Datei `~/.xinitrc`, die vom Programm `xinit` gelesen wird, bestimmt, welche Clients (Anwendungen) zu starten sind. Das sind z. B. ein XTerm und der Window-Manager.

Hat der Benutzer diese Datei nicht in seinem Homeverzeichnis, so wird die Systemdatei `/etc/X11/xinit/xinitrc` verwendet. Dieses Verfahren ist dann sinnvoll, wenn nicht ständig ein X-Server gestartet sein soll, z. B. weil der Rechner über relativ wenig RAM verfügt. Damit dieses Verfahren funktioniert, muss der X-Server auf die Hardware zugreifen können, d. h. `root`-Rechte haben. Dazu wird in aktuellen X-Releases ein Wrapper verwendet, früher musste der X-Server setuid-`root` laufen.

Beim Start mittels `startx` können dem Server weitere Optionen übergeben werden. So kann z. B. mit `startx -- -bpp 16 :1` ein zweiter Server mit einer höheren Anzahl an Farben gestartet werden. Alle Optionen beim Aufruf eines X-Servers finden Sie in der `Xserver(1)`-Manpage dokumentiert. Ebenfalls ein guter Einstieg ist die allgemeine Einführung in der Manpage `X(1)`. Welcher Server von `xinit` gestartet wird, lässt sich durch eine Reihe von Möglichkeiten bestimmen:

- Als Standard wird das Programm `X` gestartet. Dies kann ein Link auf den korrekten Server sein.
- Existiert die Datei `/etc/X11/xinit/xserverrc`, so wird diese ausgeführt. Der verwendete Dateiname kann mit der Umgebungsvariablen `XSERVERRC` überschrieben werden.
- Jeder Benutzer kann in seinem Home-Verzeichnis die Datei `~/.xserverrc` anlegen, die zum Start des X-Servers verwendet wird.

Bei dieser Art, das X-Window-System zu starten, kann der Anwender zusätzliche Parameter an den X-Server übergeben, um zum Beispiel eine andere Farbtiefe einzustellen. Das Listing 7.1, zeigt ein Beispiel für die Datei `~/.xserverrc`, in der der X-Server mit einer Farbtiefe von 16 Bit gestartet wird, unabhängig davon, was in der Konfigurationsdatei `XF86Config` eingetragen ist.

```
#!/bin/bash
# Starten des X-Servers mit Optionen, siehe Xserver(1)
exec Xwrapper -bpp 16 "$@"
```

Listing 7.1 Ein Beispiel für die Datei `~/.xserverrc`

7.5.2 Anmeldung unter X mit dem X-Display-Manager

Alternativ zur Anmeldung an der Konsole kann mit Hilfe des Programms `xdm` (X-Display-Manager) die Anmeldung direkt unter X durchgeführt werden. Dabei wird als Konfigurationsdatei zum Start der Clients `~/.xsession` verwendet. Ist diese Datei nicht vorhanden, so wird ebenfalls der Systemstandard in der Datei `/etc/X11/xdm/Xsession` verwendet.

Ist die Anmeldung, z. B. aufgrund von Fehlern in der Datei `~/.xsession`, nicht möglich, so kann eine Notfallsitzung gestartet werden, wenn das Passwort nicht mit Enter, sondern mit Alt+F1 bestätigt wird. Mögliche Fehlermeldungen findet man bei dieser Form der Anmeldung in der Datei `~/.xsession-errors`. Auch wenn Sie normalerweise keine Besonderheiten feststellen können, sollten Sie ab und zu einen Blick in diese Datei werfen.

Die Anmeldung unter X ist nur bei Rechnern mit hinreichend viel Hauptspeicher und einem guten Monitor sinnvoll. Dann allerdings erreicht man eine große Ähnlichkeit mit kommerziellen Betriebssystemen auf Workstations, genauer gesagt: Die Systeme sind praktisch nicht voneinander zu unterscheiden. Kommerzielle Unix-Systeme verwenden als Display-Manager das CDE-Programm `dtlogin`, das KDE-Projekt hat `kdm` entwickelt. Bei diesen Programmen kann der Anwender mit Hilfe eines Menüs den zu startenden Window-Manager auswählen oder eine Notfallsitzung starten. Neben dem Start der Sitzung auf dem lokalen

Rechner kann Linux auch als *X-Terminal* verwendet werden. Auf einem X-Terminal läuft praktisch nur ein X-Server, der die Hardware verwaltet. Die Anmeldung erfolgt dabei auf einem (fremden) Unix-Rechner irgendwo im Netz.

Die Auswahl des Rechners, auf dem die Anmeldung erfolgen soll, wird durch den `chooser` vorgenommen. Auf dem entfernten Rechner muss der X-Display-Manager `xdm` oder die Äquivalente der Desktops gestartet und die Anmeldung erlaubt sein. Das ist standardmäßig nicht der Fall und wird in der Datei `/etc/X11/xdm/Xaccess` konfiguriert.

Es gibt, je nach Anforderung, verschiedene Möglichkeiten, den X-Server auf dem X-Terminal zu starten. Im Einzelnen sind das:

- `X -broadcast`, wobei im lokalen Subnetz nach Rechnern mit einem laufenden `xdm` gesucht wird. Läuft auf dem lokalen Rechner auch ein `xdm`, so ist auch eine lokale Anmeldung möglich.
- `X -query` *`Host`*, um das Anmeldebild eines bestimmten Rechners zu erhalten.
- `X -indirect` *`Host`* holt die Liste der möglichen Systeme vom Rechner *`Host`*.

Es gibt kommerzielle X-Terminals, die auf Linux basieren. Es ist außerdem möglich, einen Linux-Kernel über das Netz zu booten und das Root-Dateisystem mittels NFS anzuhängen. Damit lassen sich plattenlose Systeme implementieren, die auch als X-Terminal dienen können. Alternativ kann man das Linux-System von der Festplatte laden, aber die Dateisysteme nur im Lesezugriff einhängen. Für `/var` kann man eine kleine RAM-Disk verwenden. Ein derartig eingerichteter Linux-Rechner kann wie ein X-Terminal im laufenden Betrieb ausgeschaltet werden.

Mit der Verfügbarkeit des Java-Development-Kits und der direkten Unterstützung von Java im Kernel ist Linux als Netzwerkcomputer geeignet. Gegenüber anderen Lösungen hat man Vorteile durch die Verwendung von normaler PC-Hardware und die Möglichkeit, dass sich der Rechner zu einer »echten« Linux-Workstation auswächst.

7.5.3 Tastaturbelegung unter X

Auch die Belegung der Tastatur kann unter X weitgehend verändert werden. Dazu dient das Programm `xmodmap`. In der Regel werden Sie nur wenige Änderungen an der Tastaturbelegung vornehmen, da XFree86 die Tastaturbelegung der Konsole übernimmt. Sollten Sie dennoch umfangreichere Änderungen planen, so empfiehlt sich die Benutzung von `xkeycaps`, das Sie im Contrib-Verzeichnis auf `ftp.x.org` finden. Weiterhin hilfreich ist das Programm `xev`, das alle Events (Ereignisse, Tastendrücke, Mausbewegungen), die es erhält, im »Klartext« ausgibt.

Mit dem Programm `xmodmap` können Sie auch die Maus als »Linkshänder-Maus« umkonfigurieren. Da alle Standardskripten beim Start einer X-Session die Datei `~/.Xmodmap` auswerten, tragen Sie dort die Zeile aus Listing 7.2 ein. Desktop-Umgebungen haben hierfür die entsprechenden Konfigurationsdialoge, Sie benötigen die Konfiguration zum Beispiel beim Einsatz eines Window-Managers.

```
pointer = 3 2 1
```

Listing 7.2 Vertauschen der Maus-Buttons für Linkshänder

7.5.4 X-Ressourcen

X ist nicht nur im Look&Feel weitgehend an die Bedürfnisse des Benutzers anpassbar, auch die einzelnen Clients sind in der Regel sehr flexibel zu konfigurieren. Dazu dient das Konzept der Ressourcen. In den Ressourcen werden Werte gespeichert, die systemweit festgelegt sind oder vom Benutzer überschrieben werden können. Die systemweit gültigen Ressourcen werden im Verzeichnis `/etc/X11/app-defaults` gespeichert. Dort finden Sie für praktisch jeden X-Client eine Datei mit Ressourcen. Die Desktop-Umgebungen verwenden nicht das Ressource-System, sondern eigene Techniken.

Welche Ressourcen ein Anwendungsprogramm liest und welche zu dessen Konfiguration verwendet werden können, entnehmen Sie der entsprechenden Manpage im Abschnitt `RESOURCES`. Einige dieser Einstellungen können dem Client auch beim Start als Kommandozeilenoption mitgegeben werden. Jeder Benutzer kann aber für sich selbst eigene Einstellungen festlegen. Dazu kann die Datei `~/.Xresources`[2] angelegt werden. Diese Datei wird beim Start von X (bzw. nach der Anmeldung) automatisch in den Skripten `~/.xinitrc` und `~/.xsession` mit dem Kommando `xrdb` geladen.

Änderungen an dieser Datei werden in einer bestehenden Sitzung nicht automatisch aktiv, sondern müssen erst mittels eines Kommandos, z. B. `xrdb -merge .Xresources`, in das X11-System eingelesen werden. Technisch werden diese Daten im X-Server gespeichert, der den Applikationen die entsprechenden Werte liefert.

Ressourcen können in einer Reihe von verschiedenen Formen angegeben werden. Dabei werden die spezielleren gegenüber den weniger speziellen bevorzugt. Die Angabe von `rxvt.background: lightblue` veranlasst speziell das Programm `rxvt`, die Farbe `lightblue` als Hintergrundfarbe zu verwenden. Bei dem Eintrag `*background: white` wird die Hintergrundfarbe für alle Clients auf die Farbe `white` gesetzt. Sind beide Einträge in den Ressourcen vorhanden, dann verwendet `rxvt` die dafür vorgesehene Farbe, während alle anderen Clients eine andere Hintergrundfarbe verwenden.

2. Auf manchen Systemen wird die Datei `~/.Xdefaults` verwendet.

Wenn Sie Programme nicht in den Systemverzeichnissen installieren können, dann können Sie in der Umgebungsvariablen `XAPPLRESDIR` die zu durchsuchenden Verzeichnisse angeben. Insgesamt macht diese Konfigurationsmethode einen etwas hausgemachten Eindruck und kann vom Bedienungskomfort her nicht mit modernen Systemen mithalten. Andererseits ist das System sehr flexibel und auf jedem X-Server verfügbar.

7.5.5 Konfiguration von Desktops

Von einer modernen grafischen Benutzeroberfläche erwartet man, dass auch die Konfiguration mit Hilfe von entsprechenden Tools durchgeführt werden kann und man nicht darauf angewiesen ist, Textdateien mit einem Editor zu bearbeiten. Die Beispielimplementation des X-Window-Systems erfüllt diese Anforderung nicht – das schaffen erst darauf aufsetzende Benutzeroberflächen.

CDE, GNOME und KDE enthalten alle entsprechende Programme, mit denen die Farben, der Bildschirmhintergrund und vieles mehr eingestellt werden kann. Erfahrene Benutzer können die erstellten Dateien wieder mit einem Editor bearbeiten, notwendig ist das in der Regel nicht mehr. Insbesondere die Programme der freien Desktops GNOME und KDE sind sehr flexibel und doch einfach mit den entsprechenden Menüs einzurichten.

7.5.6 Andere Konfigurationsmöglichkeiten

Ein weiteres Programm zur Anpassung von X ist `xset`. Sie können den Tastaturklick ein- oder ausschalten und in der Lautstärke verändern. Wenn Sie die Beschleunigung der Maus ändern wollen, dann verwenden Sie ebenfalls `xset`. Außerdem können Sie mit diesem Programm dynamisch die Suchpfade nach Fonts verändern. Dabei stehen Ihnen folgende Optionen zur Verfügung:

`fp=`*`Pfad`*`[,...]`
: Verwendet genau die angegebenen Verzeichnisse als Suchpfad nach Fonts. Normalerweise werden hier Verzeichnisse und/oder Font-Server angegeben.

`fp default`
: Setzt den Suchpfad nach Fonts auf den Standardwert des Servers zurück. Wenn Sie eine Einstellung getestet haben und wieder auf die Originaleinstellungen wechseln möchten, dann können Sie diese Option verwenden.

`fp rehash`
: Initialisiert den Font-Pfad mit der aktuellen Einstellung. Dabei werden die Verzeichnisse nach neuen Fonts durchsucht. Um einen neuen Font zu installieren, können Sie diesen in ein Systemverzeichnis kopieren und dort `mkfontdir` aufrufen. Erst der anschließende Aufruf von `xset fp rehash` macht die Daten im X-Server verfügbar, wenn Sie diesen nicht neu starten wollen.

`-fp` *`Pfad`*`[,...]` oder `fp-` *`Pfad`*`[,...]`
Entfernt die angegebenen Verzeichnisse aus dem Suchpfad.

`+fp` *`Pfad`*`[,...]` oder `fp+` *`Pfad`*`[,...]`
Fügt die angegebenen Verzeichnisse oder Font-Server an den Anfang oder das Ende des bestehenden Suchpfads an.

7.5.7 Benutzung eines Font-Servers

Bei der Installation des X-Window-Systems werden bereits eine Reihe von Fonts mitgeliefert. Diese benötigen allerdings relativ viel Platz auf der Festplatte. Wenn Sie in einem Netz mehrere Unix-Systeme haben, dann können Sie alle benötigten Fonts auf einem Rechner installieren und dort den Font-Server `xfs` starten. Auf den anderen Rechnern tragen Sie in der Datei `/etc/XF86Config` oder mit `xset` diesen Rechner als Font-Server im Format `tcp/`*`hostname`*`:`*`Port`* ein. Der Standardport für `xfs` ist 7100.

Dieses Verfahren hat Vorteile für relativ alte Rechner ohne mathematischen Koprozessor, da hier der X-Server die Berechnung der Fonts durchführt und dies recht lange dauern kann. Der X-Server scheint während dieser Zeit zu hängen. Der `xfs` läuft asynchron zum X-Server, so dass es sich gerade bei alten Rechnern lohnen kann, diesen lokal zu starten.

Die verbreiteten X-Server waren nicht in der Lage, TrueType-Fonts darzustellen. Es gibt jedoch für diesen Fall einen speziellen Font-Server (`xfstt`). Damit steht jedem X-Programm die gesamte Auswahl der auch für Windows verfügbaren TrueType-Schriften zur Verfügung.

Beim Start von `xfstt` geben Sie mit der Option `--dir` an, in welchem Verzeichnis Ihre True-Type-Fonts gespeichert sind. Ist auf Ihrem Rechner bereits ein normaler Font-Server gestartet, so müssen Sie mit der Option `--port` einen anderen, noch nicht belegten Port verwenden und diesen auch in der Konfiguration des X-Servers angeben.

7.6 Window-Manager

Das Aussehen (die »Dekoration«) und das Verhalten der Fenster prägt entscheidend das Look&Feel eines Window-Systems. Unter X wird dieser Teil durch den so genannten *Window-Manager* implementiert. Der Window-Manager unter X ist kein Teil des X-Window-Systems selbst, sondern ein Client (Anwendungsprogramm) wie jeder andere auch.

Das hat den Vorteil, dass X selbst keine Vorschriften zum Aussehen und zur Bedienung macht (machen kann) und dass auch bei einem »beschäftigten« X-Client das System immer noch gut reagiert. Auch bei Windows NT kann eine Anwen-

dung das Verschieben oder Verkleinern des eigenen Fensters verhindern, so dass das System für den Benutzer zunächst »steht«.

Window-Manager bestimmen das Layout der Titelzeile der Fenster, der Fensterrahmen, die Art und Weise der Icon-Darstellung und gegebenenfalls die Anzahl und Anordnung von mehreren virtuellen Desktops. Der Window-Manager ist außerdem verantwortlich für den Keyboard-Fokus, also dafür, welche Anwendung Tastatureingaben erhält. Je nach Vorliebe kann man einen Klick in das Fenster oder auf die Titelzeile verlangen, um den Keyboard-Fokus zu übertragen. Ich persönlich bevorzuge die Möglichkeit, dass stets das Fenster, über dem der Maus-Cursor steht, den Fokus hat. Trotz aller Flexibilität existiert der Modus »Focus-follows-brain« noch nicht ...

Es gibt eine ganze Reihe von Window-Managern, die recht unterschiedliche Eigenschaften haben, neuere befolgen meist das »Inter-Client Communication Conventions Manual« (ICCCM). Das ICCCM ist eines der offiziellen Dokumente des X-Konsortiums, die das X-Window-System bzw. die X-Umgebung definieren. Das ICCCM legt die Konventionen fest, denen Applikationen folgen müssen, um mit anderen Clients (auf demselben Server) zusammenzuarbeiten. Die im ICCCM festgelegten Spezifikationen können auch als Protokoll angesehen werden. Insbesondere die Interaktion zwischen dem Window-Manager und den Applikationen sowie der Selektionsmechanismus (Cut&Paste) sind hierbei kritische Punkte.

Jeder der verfügbaren Window-Manager hat spezielle Funktionen und damit Vor- und Nachteile. Ein nützliches Feature ist die Darstellung von virtuellen Screens oder Desktops. Damit können auch auf kleinen Monitoren viele Fenster übersichtlich untergebracht werden. In Tabelle 7.1 finden Sie eine Aufstellung über eine Reihe von verbreiteten Window-Managern.

Programm	Beschreibung
`twm`	»Tab Window Manager«: Standard-Window-Manager im X11
`olwm`	»Open Look Window Manager«: Sun's Window-Manager
`mwm`	»Motif Window Manager«
`fvwm`	»? Virtual Window Manager«: schlanker, konfigurierbarer Window-Manager
`enlightenment`	Window-Manager mit »Eye-Candy«
`sawfish`	Konfigurierbarer Window-Manager von GNOME

Tabelle 7.1 Übersicht über verschiedene Window-Manager

Welchen Window-Manager Sie verwenden, hängt ganz von Ihren persönlichen Vorlieben ab. Der einzige Window-Manager, der auf jedem (auch kommerziellem) Unix vorhanden ist, ist der `twm`. Dies ist zunächst die einzige Option für eine einheitliche Systemumgebung. Das Aussehen und die Bedienung entsprechen allerdings nicht mehr dem Stand der Dinge.

Bei kommerziellen Systemen ist derzeit der Motif-Window-Manager Standard. Sie können unter Linux den `fvwm` im Motif-Kompatibilitätsmodus verwenden, den Motif-Clone `lesstif` verwenden oder einen kommerziellen `mwm` einsetzen. Aber auch andersherum kann es sinnvoll sein, auf den anderen Systemen den `fvwm` zu installieren; oft ist dies allerdings nicht möglich.

7.6.1 Window-Manager und die Desktops GNOME und KDE

Die Projekte GNOME und KDE haben es sich zur Aufgabe gemacht, den Benutzern einen einfach und einheitlich zu bedienenden Desktop zur Verfügung zu stellen. Der Window-Manager ist ein Aspekt, der die Benutzung des Systems deutlich beeinflusst. Im Rahmen von KDE wird ein eigener Window-Manager implementiert, der sich daher sehr gut in das übrige System einpasst. Wenn man alle Features von KDE nutzen will, so kommt man zurzeit nicht um die Verwendung des KDE-eigenen Window-Managers herum.

Im Rahmen von GNOME geht man dieses Problem anders an. Im Prinzip kann jeder beliebige Window-Manager verwendet werden, es ist jedoch eine Programmierschnittstelle veröffentlicht, die es jedem Window-Manager erlaubt, die zusätzlichen Features des GNOME-Desktops zu unterstützen und auszunutzen.

7.7 Allgemeine X11-Kommandozeilen-Optionen

Viele X11-Anwendungsprogramme, alle Programme, die mittels des X-Toolkits (der Bibliothek `libXt`) entwickelt wurden, »erben« einige Kommandozeilenoptionen, die beim Initialisieren der Bibliothek automatisch ausgewertet werden. Auch wenn ein Anwendungsprogramm laut seiner Manpage keine Möglichkeit hat, z. B. die Größe und Lage eines Fensters beim Aufruf festzulegen, ist dies oft möglich, weil die übergebenen Optionen zunächst vom X-Toolkit ausgewertet und auch verstanden werden. Im Folgenden werden einige dieser allgemeingültigen Kommandozeilenoptionen kurz dargestellt. Alle Optionen werden in der Manpage zu `X(1)` erläutert. Beachten Sie, dass der Programmierer möglicherweise die Bibliothek `libXt` nicht verwendet hat und diese Optionen daher nicht erkannt werden.

Bildschirm
: Mit der Option `-display` werden der Name des Hosts und die Bildschirmnummer des X-Servers angegeben. Dabei wird eine eventuell vorhandene Umgebungsvariable `DISPLAY` ignoriert. Als Parameter kann z. B. `jupiter:0.0` angegeben werden. Normalerweise wird die Umgebungsvariable `DISPLAY` den korrekten Wert enthalten, es sei denn, Sie benutzen X im Netz. Genaueres zur Verwendung von X im Netz finden Sie im Abschnitt 7.8.2, »Benutzerbezogenen Zugriff erlauben mit `xauth`«.

Größe und Lage von Fenstern
: Mit der Option `-geometry` wird die anfängliche Größe und Lage eines Fensters spezifiziert. Eine gültige Geometrie wäre z. B. `-geometry 81x24+0+0`. Das Fenster mit der Größe 81 mal 24 Pixel (`xterm` verwendet hierbei Textspalten und Zeilen) wird in der linken oberen Ecke des Bildschirms geöffnet. Das Programm `xwininfo` gibt unter anderem diese Informationen aus, platzieren Sie also das Fenster an der gewünschten Stelle, und starten Sie das Programm `xwininfo`. Die ausgegebene `geometry` verwenden Sie dann beim nächsten Start als Parameter für die Option `-geometry`.

  Mit `-0-0` als Position wird die linke untere Ecke des Bildschirms als Startpunkt für die linke untere Ecke des Fensters verwendet. Wenn Sie als Startpunkt die Breite bzw. Höhe des Bildschirms angeben, so startet das Programm auf dem jeweils rechts bzw. unten gelegenen Desktop.

Farben
: Sowohl die Vorder- als auch die Hintergrundfarbe eines X-Clients können mit der Option `-fg` oder `-foreground` bzw. `-bg` oder `-background` verändert werden.

Fonts
: Mit der Option `-fn` bzw. `-font` lässt sich der zu verwendende Font festlegen.

Fenster-Titel
: Zur besseren Unterscheidung können mit der Option `-title` Fenstern desselben Programms unterschiedliche Titel zugewiesen werden. Diese Option verwende ich häufig, wenn ich bestimmte Programme wie z. B. `top` auf verschiedenen Rechnern starte.

X-Ressourcen
: Beim Start des Programms können, z. B. zum Testen verschiedener Einstellungen, einzelne Ressourcen mit der Option `-xrm` überschrieben werden. Beim permanenten Einsatz sollten diese Einstellungen in der Datei `~/.Xresources` eingetragen werden.

7.8 Zugriffskontrolle

X ist ein netzwerktransparentes Window-System. Programme können auf dem einen Rechner (z. B. eine leistungsfähige Workstation) laufen und ihre Fenster auf einem einfachen X-Terminal, einem Linux-Rechner oder einem X-Server unter Windows darstellen. Der Server, der die Fenster darstellt, kann durch zwei Verfahren bestimmt werden:

- Das Setzen der Umgebungsvariablen `DISPLAY`. Dabei wird der Host-Name des Servers, gefolgt von der Nummer des Displays angegeben. Nach dem Befehl `export DISPLAY=jupiter:0` werden die neu gestarteten Clients die

Fenster auf dem Rechner `jupiter` darstellen. Dabei wird der erste X-Server verwendet, der dort gestartet wurde.

- Die Standardkommandozeilenoption `-display` zur Angabe des X-Servers. Der Befehl `xterm -display jupiter:0` öffnet ein Shell-Fenster auf dem Rechner `jupiter`. Die Shell selbst (und das Programm `xterm`) laufen dabei auf dem lokalen Rechner.

Es ist aber nicht sinnvoll, wenn jeder Benutzer auf jeden X-Server zugreifen kann – leider ist das jedoch bei vielen kommerziellen X-Servern oder X-Terminals die Standardeinstellung. Zum einen können Bildschirmausgaben und Tastatureingaben mit Programmen wie `xspy` mitgelesen werden, zum anderen können Clients auch Tastatureingaben untergeschoben werden. Dies kann ein echtes Sicherheitsproblem darstellen, da man dieses als Benutzer nicht in jedem Falle mitbekommt.

Insbesondere wenn Sie unter X ein Passwort eingeben müssen (z. B. wenn Sie `su` ausführen), sollten Sie im `xterm` die Option »secure Keyboard« anwählen (Strg-linke Maustaste). Nur damit können Sie sicherstellen, dass nur ein Programm die Eingaben erhalten kann.

Ein X-Server, der über TCP/IP verfügbar ist, belegt den Port 6000 plus die Nummer des Displays. Der X-Server zum Display :0 belegt also Port 6000, der X-Server zum Display :1 den Port 6001. Mit Hilfe von Scan-Tools lässt sich einfach feststellen, auf welchen Rechnern ein X-Server läuft. Die Server des XFree86-Projekts schreiben Log-Dateien, in denen, je nach Server-Konfiguration, alle oder alle fehlgeschlagenen Verbindungen protokolliert werden. Damit ist es in einigen Fällen möglich, Hack-Versuche zu erkennen.

Wenn dann der Zugriff auf den X-Server auch für Unberechtigte erlaubt ist, dann ist es sehr einfach, den Benutzer bei der Arbeit zu stören. Starten Sie einfach genügend `xeyes` auf seinem Display. Sie können aber genauso einfach mit `xwd` einen Blick auf seinen Desktop werfen oder mit `xlsclients` eine Liste der aktuell laufenden X-Clients erstellen. Mit dem Programm `xkill` können Sie jeden beliebigen Client beenden. Listing 7.3 zeigt ein Beispiel dafür. Ups.

```
(linux):~$ export DISPLAY=victim:0
(linux):~$ xlsclients
victim  kfm
victim  gnome-terminal
victim  kmix -miniicon kmix.xpm
victim  kpanel
(linux):~$ xlsclients -l -a
...
Window 0x340000b:
  Machine:  victim
  Name:  KMix
  Icon Name:  KMix
```

```
  Command:  kmix -miniicon kmix.xpm
  kmix.xpm
  Instance/Class:  kmix/kmix
...
(linux):~$ xkill -id 0x340000b
xkill:  killing creator of resource 0x340000b
```

Listing 7.3 Das Beenden von beliebigen X-Clients mit `xkill`

7.8.1 Rechnerweise Zugriff erlauben mit xhost

Um trotz der Sicherheitsbedenken die Flexibilität von X zu erhalten, sind ausgeklügelte Sicherheitsmechanismen notwendig. Bis zum Release 4 (X11R4) war nur die Zugriffskontrolle über den Rechnernamen möglich. Da auch heute noch viele Rechner mit X11R4 im Einsatz sind, wird dieses Verfahren hier erläutert. Bei moderneren X-Releases sollten Sie das im nächsten Abschnitt vorgestellte Verfahren (`xauth`) verwenden. In der Praxis wird trotz aller Sicherheitsbedenken häufig `xhost` verwendet.

Standardmäßig ist der Zugriff von anderen Systemen aus nicht erlaubt; das Eingeben des Kommandos `xhost +` erlaubt den Zugriff von beliebigen Rechnern aus. Einige kommerzielle X-Terminals oder X-Server verwenden diese Einstellung, die Manipulationen Tür und Tor öffnet. Wenn Sie diese Einstellung irgendwo entdecken, sollten Sie sie sofort entfernen. Es ist zwar bequem, sich um diese Dinge nicht kümmern zu müssen, ruiniert aber die Sicherheit des gesamten Systems.

Der Befehl `xhost -` verhindert dementsprechend den allgemeinen Zugriff. In der Regel wird man den Zugriff nur für einen bestimmten Rechner mit dem Befehl `xhost` *`Host-Name`* freigeben. Unmittelbar nachdem der Client sein Fenster geöffnet hat, können Sie das Öffnen weiterer Fenster von diesem Rechner aus mit `xhost -`*`Host-Name`* verhindern. Mit dieser Methode können Sie das Zeitfenster verkleinern, in dem Sie angreifbar sind, aber wirklich sicher ist das Verfahren nicht.

7.8.2 Benutzerbezogenen Zugriff erlauben mit xauth

Wie bereits oben erwähnt, ist das host-basierte Verfahren nicht besonders sicher. Jeder Benutzer mit einem Account auf dem entsprechenden Rechner kann sich Zugriff zu diesem X-Server verschaffen. Daher wurde mit X11R5 ein weiteres Autorisierungsverfahren implementiert. Dabei wird die Zugriffsberechtigung nicht durch den Rechnernamen bestimmt, sondern durch eine Art Passwort (ein Magic-Cookie). Die Verwaltung der Zugriffsberechtigungen erfolgt mit dem Programm `xauth`. Beim Start des X-Servers (bzw. der Anmeldung mit `xdm`) wird ein zufälliger Schlüssel erzeugt und in der Datei `~/.Xauthority` gespeichert.

Jeder Client, der diesen Cookie kennt, ist zum Zugriff auf den Server berechtigt. Wenn Sie also den »magischen Keks« im Klartext auf einen anderen Rechner übertragen, dann kann er prinzipiell mit einem Packet-Sniffer abgehört und für einen Einbruch verwendet werden. Der Magic-Cookie wird von `xdm` bei jeder Anmeldung neu erzeugt, so dass eine alte Zugangsberechtigung nicht noch einmal verwendet werden kann.

Haben Sie auf den entsprechenden Rechnern das gleiche Home-Verzeichnis (z. B. via NFS), so haben Client und Server direkten Zugriff auf den Schlüssel. Läuft der Client auf einem Rechner, der keinen Zugriff auf den Schlüssel hat, so muss dieser vorher übertragen werden. Das Listing 7.4 zeigt ein Beispiel für die Anwendung von `xauth`. Wenn Sie oft mit verschiedenen Rechnern arbeiten, können Sie sich entsprechende Aliasnamen definieren. Beachten Sie, dass Magic-Cookies nicht in der Kommandozeile zu `xauth` auftauchen sollten. Andernfalls kann der Schlüssel, z. B. aus der Prozessliste, ausgelesen werden.

```
(linux):~> xauth list
jupiter/unix:0  MIT-MAGIC-COOKIE-1 66e15be3b61c812e0df5
(linux):~> xauth extract - | rsh typhon xauth merge -
(linux):~> rsh typhon xterm -display `hostname -f`:0
```

Listing 7.4 Die Verwendung von `xauth`

7.8.3 Secure Shell unter X

Wie wir gesehen haben, ist die sichere Verwendung von X mit etwas Aufwand verbunden. Viele Anwender schreiben sich Skripten, die zunächst die magischen Kekse auf einen fremden Rechner transportieren und dann via Remote-Shell das entsprechende Programm starten, wobei die `DISPLAY`-Variable entsprechend angepasst werden muss.

Im Prinzip ist dieses Skript für spezielle Anwendungen relativ einfach. Wenn es aber unter mehreren Betriebssystemen und über Domain-Grenzen hinweg (das kann wiederum zu Sicherheitsproblemen führen) funktionieren soll, dann wird das Skript recht schnell sehr kompliziert. Außerdem werden die magischen Kekse oft im Klartext übertragen und ein Angreifer ist in der Lage, das X-Protokoll auf dem Netzwerk mit einem Sniffer mitzulesen und zu analysieren.

Wenn Sie häufiger die Display-Umleitung verwenden, dann sollten Sie in jedem Fall die `ssh` verwenden. Leider ist diese nicht Bestandteil der üblichen Unix-Systeme, aber der Installationsaufwand lohnt sich in jedem Fall. Zunächst wird die Authentifizierung verbessert, indem nicht mehr die IP-Adresse, sondern ein Public-Key-Verfahren verwendet wird. Der zweite wichtige Aspekt ist, dass die gesamte Sitzung verschlüsselt wird. Damit können auch mit einem Packet-Sniffer keine Daten mehr ausgespäht werden. Die Verschlüsselung gilt auch für

die X-Umleitung, die von der `ssh` automatisch verwendet wird. Und das Beste am Schluss: `ssh` ist aufrufkompatibel zur `rsh`, so dass Ihre Benutzer sich nicht umgewöhnen müssen.

7.9 Tools und nützliche Programme für X

Für X und die Desktop-Umgebungen stehen nicht nur viele Anwendungen wie Textverarbeitungen, Editoren oder Tabellenkalkulationen zur Verfügung. Beinahe genauso wichtig sind die Tools, die im täglichen Einsatz immer wieder benötigt werden. Viele dieser Tools gehören zur Standarddistribution von X11, andere gehören zur Contrib-Software, wieder andere werden auf anderen Wegen verteilt.

Die Standard-Tools gehören zur X-Distribution. Einige Hersteller ersetzen diese Programme durch eigene Programme oder liefern kein derartiges Tool mit. So wird das Standard-XTerm, das Shell-Fenster unter X, häufig ersetzt (`aixterm` unter AIX, `dxterm` unter Digital-Unix oder ULTRIX). Leider sind diese Terminals häufig nicht korrekt in den Termcap- oder Terminfo-Dateien der anderen Systeme eingetragen. Nur das Vorhandensein der Standard-Tools ermöglicht daher in einem heterogenen Netz eine einheitliche Arbeitsumgebung.

- Als Shell-Fenster wird `xterm` verwendet, unter Linux manchmal auch `rxvt`, das weniger Speicher benötigt. Die Desktops GNOME und KDE verfügen jeweils über eigene Terminal-Emulatoren, man kann jedoch jederzeit ein normales `xterm` starten.
- Das Programm `xprop` gibt Informationen (Properties) zu Clients aus.
- Die Eigenschaften eines Fensters werden von `xwininfo` angezeigt. Diese Informationen enthalten beim Root-Window (dem Hintergrund) auch die aktuell verwendete Farbtiefe.
- Mit dem Programm `xwd` kann der aktuelle Bildschirminhalt in einer Datei gespeichert werden. Ein solcher Screenshot kann mit dem Programm `xwud` wieder angezeigt werden. Diese Programme sind nicht komfortabel, gehören aber zur Standarddistribution von X11 und sollten daher auf jedem System verfügbar sein. Im nächsten Abschnitt finden Sie Verweise auf komfortablere Programme.
- Verschiedene Einstellungen wie Tastaturklick, Sound oder der Bildschirmschoner können mit dem Programm `xset` durchgeführt werden.
- Wenn Sie Informationen über Ihren Server benötigen, starten Sie das Programm `xdpyinfo`. Informationen über einen Client erhalten Sie mit dem Programm `xwininfo`.

- Das Programm `editres` kann zur Anzeige und Veränderung von X-Ressourcen verwendet werden. Nur zur Anzeige kann das Programm `viewres` verwendet werden.
- `xev` zeigt die X-Events an, die z. B. bei Tastendrücken erzeugt werden.
- Mit `xmodmap` kann man die Tastaturbelegung verändern. Ein grafisches Frontend ist `xkeycaps`, das allerdings nicht zum X-Core gehört, sondern als Contrib-Software verteilt wird.
- `xman` ist ein einfaches Programm zur Anzeige von Manpages. Viele Benutzer verwenden `tkman`, das allerdings Tcl/Tk, eine Skript-Sprache mit X-Anbindung, voraussetzt.
- Mit `xmag` können Sie sich Teile des Bildschirms vergrößert darstellen lassen.
- Weitere Tools sind `xclock` zur Anzeige der Uhrzeit, `xlock` zum Sperren des Bildschirms, `xload` zur Anzeige der Auslastung des Systems, `xeyes` als auf den Mauszeiger schielendes Augenpaar und `xconsole` zum Lesen von Meldungen des Betriebssystems.

8 Datensicherung

Only wimps use tape backup: real men just upload their important stuff on ftp, and let the rest of the world mirror it.

Linus B. Torvalds

8.1 Notwendigkeit der Datensicherung

Anwender wollen kein Backup – Anwender wollen Restore.

Ein ungenannter Systemverwalter

Die Durchführung der Datensicherung zählt für viele Benutzer zu den ungeliebten, weil vermeintlich unproduktiven Tätigkeiten. Dennoch sollte sich ein Systemadministrator, und das ist auch der Betreiber eines Ein-Benutzer-Systems, ernsthafte Gedanken über die Sicherheit seiner Daten machen. Eine Reihe von Benutzerfehlern und Problemen können dafür verantwortlich sein, dass Daten verloren gehen oder verfälscht werden.

An eine gute Datensicherung sind eine Reihe von Kriterien anzulegen: Die Sicherung sollte möglichst unbeaufsichtigt ablaufen, die Sicherungsmedien sollten von den Programmen selbst verwaltet werden und die Daten sollten einfach und zuverlässig wiederhergestellt werden können. Bei ernsthafter Computernutzung ist eine Datensicherung unverzichtbar, will man nicht das Risiko eines schmerzhaften Datenverlusts eingehen.

Aufgrund von Unvorsichtigkeiten bei der Verwendung von »gefährlichen« Befehlen können erhebliche Datenmengen verlorengehen. Unter Unix kann einfach ein kompletter Verzeichnisbaum, z. B. mit dem Befehl `rm -rf *`, ohne Nachfrage gelöscht werden.

Zur Vermeidung solcher Probleme können, anstelle der gefährlichen Befehle, Aliasnamen mit vorsichtigeren Einstellungen verwendet werden. Für die Programme `rm`, `cp` und `mv` finden Sie entsprechende Anwendungen im Listing 8.1.

```
(linux):~> alias rm 'rm -i -v'
(linux):~> alias cp 'cp -i -v -b'
(linux):~> alias mv 'mv -i -v -b'
```

Listing 8.1 Beispiele für Aliasnamen

Die Option `-i` (`--interactive`) sorgt für eine Nachfrage beim Löschen oder Überschreiben von Dateien. Die Option `-v` (`--verbose`) veranlasst die Ausgabe der Dateinamen. Bei den Befehlen `cp` und `mv` kann zusätzlich mit der Option `-b` (bzw. `--backup`) die Erstellung von Sicherheitskopien eingestellt werden. Es kön-

nen auch mehrere Versionen einer Datei als Sicherungskopie vorgehalten werden. Die genaue Verwaltung von Sicherheitskopien wird mit der Umgebungsvariablen `VERSION_CONTROL` oder der Option `-V` (bzw. `--version-control`) eingestellt. Nähere Informationen dazu liefert `man cp`.

Das Programm `rm` löscht Dateien unwiderruflich. Unter Linux existiert kein Programm `undelete`. Die Funktionalität kann entweder durch einige Shell-Skripten bzw. Aliasnamen oder durch ein spezielles `delete`/`undelete`-System erreicht werden (z. B. vom MIT). Leider haben praktisch alle Lösungen dieses Problems, solange sie nicht vom Kernel implementiert sind, Nachteile, so dass viele Anwender mit wirklich gelöschten Dateien leben.

Ein weiteres Problem ist, dass mit einer einfachen Shell-Funktion, nämlich der Ausgabeumleitung, eine Datei ohne Rückfrage überschrieben wird. Dies kann bei der Verwendung der `bash` mit der Einstellung `set -o noclobber` verhindert werden. Eine bestehende Datei kann nur noch mit der Angabe von `>|` anstelle von `>` überschrieben werden. Benutzer der `tcsh` können das Überschreiben von Dateien aufgrund einer Ausgabeumleitung mit dem Befehl `set noclobber` erreichen. Die Datei wird überschrieben, wenn statt `>` die Zeichenfolge `>!` verwendet wird. Die Kombination `>|` bewirkt bei der `tcsh`, dass sowohl die Standardausgabe als auch die Standardfehlerausgabe gemeinsam umgeleitet werden.

Der Systemadministrator kann unter Unix alle Dateien löschen oder verändern. Daher sollte die Benutzerkennung `root` nur zur Systemadministration verwendet werden, aber nicht für die tägliche Arbeit. Ein nicht privilegierter Benutzer kann nur eine begrenzte Menge von Dateien löschen oder verändern, so dass die Gefahr des Komplettverlustes aller Daten nicht gegeben ist. Viele Einsteiger in Unix verwenden die Benutzerkennung `root` auch zur täglichen Arbeit. Bei der Verwendung einer nicht privilegierten Benutzerkennung erhält man manchmal die Meldung »Permission denied«. Erscheint diese Meldung beim Start von Programmen, die für jeden Benutzer ausführbar sein sollen, dann ist das System schlecht konfiguriert. Die Arbeit, die in die korrekte Konfiguration investiert werden muss, zahlt sich langfristig (durch die erhöhte Sicherheit aufgrund der Berechtigungsprüfung des Kernels) aus.

Ein Systemadministrator sollte, besonders vor »gefährlichen« Aktionen, stets die Kommandozeile mehrfach überprüfen, bevor ein Befehl ausgeführt wird. So bietet es sich bei der Erstellung von Schleifen oder anderen Befehlen, die viele Dateien bearbeiten sollen, an, diese zunächst nur »trocken« ablaufen zu lassen, indem vor dem eigentlichen Befehl ein `echo` eingetragen wird (Listing 8.2).

```
(linux):~# find . -name ~ -exec echo rm {} \;
(linux):~# find . -name ~ -exec rm {} \;
(linux):~# find . -name ~ -ok rm {} \;
```

Listing 8.2 Trockenes Ausführen gefährlicher Befehle

```
(linux):~# find . -name ~
(linux):~# rm $(find . -name ~)
```

Listing 8.3 Trockenes Ausführen gefährlicher Befehle, eine weitere Möglichkeit

Der erste Befehl zeigt die Liste aller zu bearbeitenden Dateien an. Im zweiten Befehl werden diese Dateien dann tatsächlich gelöscht. Eine andere Möglichkeit, die Ausgabe des oberen Befehls weiterzuverwenden, ist das Umleiten der Ausgabe in eine Shell. Der dritte Befehl in Listing 8.2 sorgt dafür, dass für jede gefundene Datei gefragt wird, ob der Befehl tatsächlich ausgeführt werden darf. Eine weitere Möglichkeit zum Löschen von Sicherheitskopien zeigt Listing 8.3.

Bei einem Stromausfall oder Systemabsturz werden mitunter nicht alle Daten auch auf der Festplatte abgelegt. Linux implementiert einen Schreib-Cache, in dem auch Schreibzugriffe gepuffert werden, bevor die Daten tatsächlich auf die Festplatte geschrieben werden. Damit ist das Schreiben auf die Festplatte wesentlich schneller für die Anwendung erledigt, so dass diese andere Funktionen ausführen kann.

Nach einem Systemabsturz kann sich das Dateisystem in einem nicht mehr wiederherstellbaren Zustand befinden, so dass einige Daten verloren sind. Im schlimmsten Fall ist auch ein Totalverlust der Daten denkbar, wenn die interne Struktur des Dateisystems nicht mehr konsistent ist. Das unter Linux sehr verbreitete Extended-2-Dateisystem ist in Hinblick auf eine weitgehende Reparierbarkeit implementiert worden, so dass diese Probleme relativ selten auftreten. Die meisten Fehler dieser Art sind bisher durch Hardware-Defekte aufgetreten.

Der erste auf dem Dateisystem vorhandene Superblock, der die wichtigsten Informationen über dieses Dateisystem enthält, kann beschädigt sein. Das Programm zur Prüfung von Extended-2-Dateisystemen, `e2fsck` oder `fsck.ext2`, kann mittels der Kommandozeilenoption `-b` angewiesen werden, eine Sicherheitskopie des Superblocks zu verwenden. Das Listing 8.4 zeigt ein mögliches Vorgehen. Der erste Superblock befindet sich im ersten Block, weitere Superblöcke findet man alle 8192 Blöcke innerhalb der Partition.

```
(linux):~# fsck.ext2 -f -v /dev/hdb1
e2fsck 0.5b, 14-Feb-95 for EXT2 FS 0.5a, 94/10/23
e2fsck: Bad magic number in super-block while trying to
open /dev/hdb1

The filesystem superblock is corrupt.  Try running e2fsck
with an alternate superblock using the -b option.  (8193
is commonly an alternate superblock; Hence,
'e2fsck -b 8193 <device>' may recover the filesystem.)
(linux):~# fsck.ext2 -f -v -b 8193 /dev/hdb1
```

Listing 8.4 Beispiel für einen beschädigten Superblock

Durch die Struktur des Linux-Verzeichnisbaums und die leistungsfähigen Dienstprogramme ist es relativ einfach möglich, die Benutzer- und Konfigurationsdateien zu sichern. Schon in der Standardinstallation stehen die dafür notwendigen Mittel zur Verfügung, wenn man z. B. `/home` auf einer eigenen Partition unterbringt.

Wichtig ist jedoch, dass nicht nur die Sicherung der Daten auf ein Medium ohne Fehler durchgeführt wird. Der Systemadministrator sollte auch die Wiederherstellung von einzelnen Dateien, und nach Möglichkeit auch die des gesamten Systems, testen. Erst danach kann man sicher sein, dass die Daten tatsächlich gesichert wurden und wiederhergestellt werden können. Dieser Test sollte sinnvollerweise nach Änderungen an der verwendeten System-Software (Kernel, Hardware-Treiber, Bibliotheken, Datensicherungssystem) oder der Hardware erneut durchgeführt werden.

Insbesondere ist es sinnvoll, eine Notfalldiskette zu erstellen, die alle notwendigen Treiber und Programme zur Wiederherstellung des Systems enthält. Dann kann zunächst ein Minimalsystem von der Diskette gebootet werden, so dass keinerlei Daten von der Festplatte benötigt werden. Damit ist es praktisch immer möglich, eine Datensicherung wiederherzustellen, selbst wenn das System auf der Festplatte nicht mehr lauffähig ist.

Es existieren einige frei verfügbare Pakete zur Erstellung von Notfalldisketten, die vom Systemadministrator an die lokalen Gegebenheiten angepasst werden müssen. Im Zweifelsfall sind die Boot- und Root-Diskette der entsprechenden Distribution ein guter Startpunkt. Moderne Systeme und Distributionen können auch von CD booten, das ist dann besonders einfach. Einige Pakete zur Erstellung von Notfall- und Boot-Disketten findet man auf dem `ftp`-Server `metalab.unc.edu` im Verzeichnis `/pub/Linux/system/Recovery`.

8.2 Plattenfehler überstehen mit RAID

Moderne Festplatten sind aus der Sicht des Betriebssystems relativ lange Zeit fehlerfrei. Wenn jedoch der erste Fehler gemeldet wird, dann wird es Zeit, diese Platte auszutauschen. In der Regel wird man noch eine Datensicherung durchführen und diese dann auf einer neuen Festplatte wieder einspielen.

In der Zeit, in der die Daten gesichert und zurückgespielt werden und die Festplatte ausgetauscht wird, ist kein Systembetrieb möglich. Das Ziel ist es also, selbst bei Auftreten eines Plattenschadens das System noch bis zur nächsten betriebsarmen Zeit laufen zu lassen, um dann in Ruhe den Austausch der Festplatte vornehmen zu können. Moderne SCSI-Adapter und Platten sind teilweise »Hot-Plug«-fähig, können also im laufenden Betrieb ausgetauscht werden, aber das geht nicht immer.

Mit Hilfe der RAID-Technologie (Redundant Array of Inexpensive/Independent Disks) hat man ein Verfahren entwickelt, das in der Lage ist, den Ausfall einer oder mehrerer Platten ohne Datenverlust zu überstehen. Leider schützt diese Technik nicht vor versehentlichem Löschen oder Inkonsistenzen nach einem Systemabsturz.

Die RAID-Technologie kennt eine Reihe von Levels, die die Art der Speicherung der Daten genauer beschreiben. Heute sind im Wesentlichen die folgenden Level im Gebrauch:

- RAID-Level 0 dient dazu, mehrere Platten oder Partitionen zu einer größeren (virtuellen) Partition zusammenzufassen. Dabei gibt es zwei Varianten: das einfache Aneinanderhängen der Teile (`concat`) und das so genannte Striping. Beim Striping werden aufeinander folgende Blöcke nicht auf einer Platte hintereinander angeordnet, sondern in Streifen über alle beteiligten Platten verteilt. Man erreicht hierbei eine größere Leistung, insbesondere wenn die Festplatten an verschiedenen SCSI-Adaptern angeschlossen sind. Hierbei sinkt jedoch die Datensicherheit, da mit einem Datenverlust zu rechnen ist, wenn eine der beteiligten Platten defekt ist.
- Die Datensicherheit wird mit RAID-1, dem so genannten Mirroring, verbessert. Hier wird jeder Block parallel auf zwei oder mehr Platten geschrieben, so dass beim Ausfall einer Festplatte weiter auf die Daten zugegriffen werden kann. Beim Lesen kann die Last auf mehrere Platten verteilt werden, so dass ein höherer Datendurchsatz möglich ist. Die Kosten für diese Lösung sind relativ hoch, da alle Festplatten doppelt ausgelegt werden müssen.
- Bei RAID-4 wird zusätzlich zu den Daten eine so genannte Parität gespeichert. Wenn eine Festplatte aus dem Plattenverbund ausfällt, so können die Daten aus allen anderen Platten immer noch rekonstruiert werden. Der wesentliche Nachteil dieser Lösung liegt darin, dass jeder Schreibvorgang auch auf der Paritätsplatte stattfinden muss. Vorteilhaft ist, dass vergleichsweise wenige zusätzliche Festplatten benötigt werden.
- Häufig wird RAID-5 verwendet, das die Vorteile von RAID-4 mit einer über alle Platten verteilten Parität verbindet. Dadurch sinkt der Datendurchsatz nicht in dem Maße, wie es bei RAID-4 der Fall ist.

Alle diese Lösungen können mittels spezieller Hardware (SCSI-Adapter mit einem entsprechenden Disk-Array) oder Software realisiert werden. Die aktuellen Entwicklerversionen des Linux-Kernels implementieren diese Funktionen.

Bevor Sie sich zu früh freuen: RAID ist nicht die Lösung aller Probleme. Mit RAID kann man sich gegen den Ausfall einer Festplatte absichern. Arbeitet jedoch der entsprechende Festplatten-Controller fehlerhaft, dann nützt auch die zuverlässigste Spiegelung nichts. Genauso wenig hilft RAID gegen Systemabstürze oder versehentliches Löschen der Daten. Sie sollten also auch mit RAID eine regelmäßige Datensicherung durchführen.

8.3 Medien zur Datensicherung

Bei den heute verbreiteten Festplattengrößen ist eine komplette Datensicherung auf Disketten praktisch unmöglich. Der dafür notwendige Zeit- und Datenträgeraufwand steht in keinem Verhältnis zur zu erwartenden Sicherheit. Dennoch ist die Sicherung von kleinen Datenmengen, die sehr häufig gesichert werden sollen, auf Disketten durchaus möglich und sinnvoll. Die Texte dieses Buches wurden nach jeder Änderung auf Diskette gesichert.

Für eine bequeme und automatische Datensicherung ist es notwendig, dass die Daten einer Sicherung auf ein Medium passen, so dass die Anwesenheit des Systemadministrators (z. B. zum Bandwechsel) nicht erforderlich ist. Damit können Datensicherungen nachts ohne Eingriffe durchgeführt werden.

Häufig werden Magnetbänder als Sicherungsmedien verwendet. Magnetbänder sind preiswert und trotz des sequentiellen Zugriffs ausreichend schnell. Es gibt eine Reihe von verschiedenen Standards für Magnetbänder und die entsprechenden Laufwerke bzw. Schnittstellen, an denen die Laufwerke angeschlossen werden.

8.4 Spezielle Datenträger

In seltenen Fällen werden auch Wechselplatten, MOs oder WORMs (Write Once Read Many Times) zur Datensicherung verwendet. Aufgrund der hier anfallenden Kosten werden diese Datenträger nur selten benutzt. Mit dem Aufkommen günstiger CD-Brenner und dem Preisverfall der Rohlinge werden wichtige Daten immer häufiger zur Datensicherung auf CD gebrannt.

8.5 Strategien zur Datensicherung

Die Strategie zur Datensicherung wird von vielen lokalen Faktoren bestimmt. Der Systemadministrator muss entscheiden, welche Daten wie oft gesichert werden sollen und welche Anzahl von Sicherungsmedien hierfür notwendig ist. Außerdem ist die Abfolge von vollständigen und inkrementellen Datensicherungen zu bestimmen. Bei vollständigen Sicherungen werden, bis auf einige Ausnahmen, alle Dateien gesichert. Bei einer inkrementellen Datensicherung werden alle Dateien, die seit der letzten Sicherung verändert wurden, erneut gesichert.

8.5.1 Umfang und Zeitpunkt der Datensicherung

Zunächst sollte die Frage geklärt werden, welche Daten wie häufig gesichert werden sollen. Dazu bietet sich bei vielen Linux- und Unix-Systemen eine Staffelung ähnlich der folgenden an. Die genauen Zeiträume und betroffenen Verzeichnisse oder Dateisysteme sollte der Systemadministrator aufgrund der lokalen Gegebenheiten festlegen.

In der Regel sollten Sie versuchen, die Daten in einer betriebsarmen Zeit zu sichern. Zum einen sind dann normalerweise weniger Daten Veränderungen unterworfen, zum anderen belastet die Datensicherung auch das System nicht unerheblich, so dass Sie Ihre Anwender weniger stören.

Für eine wirklich konsistente Datensicherung sollten Sie die Sicherung im Single-User-Mode durchführen. In der Praxis wird sich das aber in vielen Fällen nicht durchführen lassen. Außer bei der Sicherung von Datenbanken kann man aber in der Regel mit diesen Problemen leben.

8.5.2 Dateiklassen

Nicht alle Dateien müssen täglich gesichert werden. Bei vielen Dateien, die nur sehr selten verändert werden, kann eine wöchentliche Sicherung durchaus ausreichend sein. Bei anderen ist eine tägliche Sicherung bereits zu wenig. Häufig bietet sich die Unterscheidung der Dateien gemäß der folgenden Struktur an. Diese Entscheidung bleibt aber dem Systemadministrator überlassen.

8.5.3 Sicherung von Benutzerdaten

Benutzerdaten, die unter Linux meist im Verzeichnis `/home` abgelegt werden, werden laufend verändert und sollten häufig gesichert werden. In kommerziellen Umgebungen empfiehlt sich hier die tägliche (inkrementelle) Sicherung der veränderten Daten. Zusätzlich sollte mindestens einmal in der Woche eine vollständige Sicherung durchgeführt werden. Der Verlust der Arbeit, die von mehreren Programmierern oder Mitarbeitern an einigen Tagen erledigt wurde, kann erhebliche Kosten verursachen.

Auf privat genutzten Systemen kann, je nach Einschätzung des Benutzers oder Systemadministrators, eine wöchentliche oder monatliche Datensicherung sinnvoll sein. Je nach Datenmenge kann möglicherweise stets eine Komplettsicherung eingeplant werden, das vereinfacht das Handling ungemein. Längere Abstände zwischen zwei Datensicherungen sind selten sinnvoll, wenn Sie regelmäßig mit Ihrem System arbeiten.

8.5.4 Konfigurationsdateien

Ebenfalls variabel sind die in `/etc` und `/var` gespeicherten (Konfigurations-) Dateien. Hier ist zum Teil extrem viel Arbeit und Wissen versteckt, so dass diese Daten regelmäßig gesichert werden sollten. Dies trifft insbesondere für die Netzwerk- und Mail-Konfiguration zu.

Diese Dateien sollten auf jeden Fall vor einem System-Update gesichert werden. Es ist oft sinnvoll, diese Dateien zunächst in ein neues Verzeichnis (z. B. `/etc.old`) zu kopieren, so dass sie nach dem Update direkt eingesehen werden können. Damit lassen sich viele Einstellungen, die an einem Linux- oder Unix-System vorgenommen wurden, leicht erneut durchführen.

Gerade diese Einstellungen dienen zur Anpassung an die Bedürfnisse der Benutzer und erleichtern die tägliche Arbeit. Solange diese Anpassungen nicht zumindest vom Systemadministrator geprüft wurden, ist die neue Installation noch kein vollwertiger Ersatz für die alte Installation. Moderne Distributionen wie Debian, Red Hat oder SuSE sind updatefähig, so dass Sie jetzt in vielen Fällen eine Neuinstallation vermeiden können.

8.5.5 Die /usr-Partition

Die auf der `/usr`-Partition gespeicherten Daten und Programme sind relativ statisch und können meist vom Installationsmedium neu eingespielt werden. Dennoch kann es auch hier sinnvoll sein, in größeren Zeitabständen, z. B. vor dem Update wichtiger Software-Pakete, eine Datensicherung durchzuführen.

Die Verfügbarkeit der Installation, z. B. auf CD, bedeutet aber noch lange nicht, dass diese Daten auch ohne Probleme von der CD gelesen werden können. Es ist immer ein Glücksspiel, sich auf genau eine Sicherung zu verlassen, die möglicherweise fehlerbehaftet sein kann.

Es gibt Administratoren, die das `/usr`-Dateisystem mit der Option `ro` für Read-Only einhängen. Da das System nicht auf diese Partition schreibt, ist die Wahrscheinlichkeit für einen Datenverlust geringer. Außerdem wird nach einem System-Crash kein `fsck` durchgeführt, so dass das System bereits nach sehr kurzer Zeit wieder verfügbar sein kann. Allerdings lohnt sich dieses Verfahren nur dann, wenn Sie wirklich selten Programme austauschen oder andere Änderungen durchführen.

8.5.6 Lokal installierte Programme

In einigen Fällen, wenn besonders viele Programme lokal auf der `/usr/local`-Partition installiert und wesentliche Veränderungen an den Programmen vorgenommen wurden, kann es sinnvoll sein, auch diese Partition (regelmäßig) zu

sichern. Bei nur wenigen Programmen kann es einfacher sein, diese neu zu installieren. Dazu sollten selbstverständlich die Quellen der Programme gesichert oder die Installationsmedien aufgehoben werden.

8.5.7 Nicht sicherungswürdige Verzeichnisse

Die Verzeichnisse `/tmp`, `/var/tmp` und andere Verzeichnisse zum Speichern temporärer Dateien (z. B. `/usr/local/tmp`) sowie das Pseudo-Dateisystem `/proc` müssen nicht gesichert werden. Daher sollte man diese Verzeichnisse stets von der Datensicherung ausschließen. Je nach Entscheidung des Systemadministrators kann dazu auch ein Dateisystem wie `/var/cache` zählen.

8.6 Abfolge von inkrementeller und vollständiger Datensicherung

Die Durchführung einer inkrementellen Datensicherung ist sinnvoll, wenn zwischen den einzelnen Sicherungen nur relativ wenige Dateien verändert werden. Zunächst muss mit einer vollständigen Datensicherung begonnen werden, damit jede Datei auf mindestens einem Medium gesichert wurde. Anschließend müssen nur die jeweils veränderten Dateien erneut gesichert werden.

Die vollständige Wiederherstellung einer Datensicherung ist aufwändig, wenn die letzte vollständige Sicherung bereits weit zurückliegt und in der Zwischenzeit viele inkrementelle Datensicherungen erfolgt sind. Im Falle eines Datenverlusts muss zunächst die komplette Sicherung zurückgeladen werden und anschließend in der richtigen Reihenfolge alle seitdem erfolgten inkrementellen Sicherungen. Daher sollten regelmäßig vollständige Datensicherungen durchgeführt werden, um die Anzahl der inkrementellen Sicherungen zu beschränken. In der Praxis hat sich z. B. eine vollständige Sicherung in der Woche bei täglicher inkrementeller Sicherung bewährt.

Eine andere Möglichkeit ist die Durchführung einer vollständigen Sicherung jeden Monat, einer wöchentlichen inkrementellen Sicherung aller seit der letzten monatlichen Sicherung veränderten Daten sowie die tägliche inkrementelle Sicherung bezüglich der letzten Wochensicherung. Werden bei der inkrementellen Sicherung fast alle Dateien erneut gesichert, so ist es wesentlich sinnvoller, eine Komplettsicherung durchzuführen. Der Bedarf an Sicherungsmedien unterscheidet sich in diesem Fall kaum, aber die Handhabung ist wesentlich einfacher.

8.6.1 Anzahl der benötigten Medien zur Datensicherung

Im nächsten Schritt sollte die Anzahl der Mediensätze für die Datensicherung festgelegt werden. Als Mindestanforderung sind zwei unabhängige Mediensätze zu verwenden, besser jedoch mehr.

Die einfachste Variante ist die Verwendung der Medien des jeweils ältesten Backups für die aktuelle Sicherung. Dabei sollten aber aus Gründen der Datensicherheit mindestens drei Medien verwendet werden. Vorstellbar ist z. B. folgendes Szenario: Eine Datensicherung wird gestartet, aber aufgrund eines Fehlers oder Benutzereingriffs abgebrochen. Anschließend muss eine Datei aus der letzten Datensicherung zurückgeladen werden. Dieses Medium ist jedoch defekt und verursacht Lesefehler. Bei nur zwei Bändern ist dann keine verwendbare Datensicherung verfügbar, die Daten sind verloren.

Wenn bestimmte Daten für längere Zeit aufbewahrt werden müssen, so z. B. monatliche Sicherungen für jeweils ein Jahr und mindestens eine jährliche Sicherung permanent, so sind natürlich mehr Medien notwendig. Unternehmen, die von der elektronischen Datenverarbeitung abhängig sind, bringen oft ausgewählte Sicherungen in einem feuerfesten Schrank oder in einem externen Unternehmen unter, um auch vor Feuer oder anderen Katastrophen gefeit zu sein. Wichtig ist in jedem Fall, dass die magnetischen Medien kühl, trocken und außerhalb von magnetischen Feldern gelagert werden.

8.6.2 Probleme bei der Datensicherung und beim Restore

Datensicherung kostet Geld, manchmal auch viel Geld. Auf der anderen Seite ist die Datensicherung ein Schutz gegen zu großen Datenverlust. Zu Ihrer Backup-Strategie gehört daher auch, gegen welche Risiken Sie sich wie absichern wollen und was Ihnen dieser Schutz wert ist.

Ein offensichtliches Problem ist die Dauer einer Datensicherung. Wenn man Daten über das Netzwerk sichert, dann möchte man das Netz nicht tagsüber belasten, sondern wird die Backups in der Nacht durchführen. Bei großen Datenmengen und vielen Servern kann es schon vorkommen, dass die Nacht zu kurz ist. In diesem Fall müssen Sie analysieren, wo der Engpass ist (Netzwerk, Bandlaufwerke, Datenmengen), und dann entsprechende Maßnahmen ergreifen.

Bei Online-Sicherungen, wenn also das System in der Zeit zur Verfügung steht, dann kann es dazu kommen, dass die gesicherten Daten inkonsistent und damit unbrauchbar sind. Viele Datenbanken kennen so genannte Online-Backups, in diesem Fall werden inkonsistente Daten gesichert, die mit Hilfe der Redo- oder Transaction-Logs wiederhergestellt werden können. Bei anderen Applikationen müssen Sie eventuell eigene Strategien entwickeln.

Manche Storage-Systeme bieten so genannte Snapshots an, diese lassen sich in Sekunden erstellen. Die Applikation schreibt weiter in das Dateisystem, die Sicherungssoftware kann den Snapshot in Ruhe auf die Sicherungsmedien kopieren. Sie benötigen allerdings genügend freien Plattenplatz.

Ein weniger offensichtliches Problem sind die Restore-Zeiten. Insbesondere wenn Sie viele inkrementelle Sicherungen durchgeführt haben und die Daten auf sehr vielen Bändern verteilt sind, kann die Wiederherstellung sehr lange dauern, viel länger als die eigentliche Datensicherung. Eine Abhilfe kann sein, dass Sie zwischendurch (häufiger) eine vollständige Datensicherung einschieben.

Problematisch ist, dass Sie die wirklich notwendigen Zeiten für eine Wiederherstellung erst nach einem vollständigen Test wissen. Gerade bei großen Datenbanken, in denen viele Änderungen stattfinden, ist es nicht ungewöhnlich, dass man erst nach 24 oder mehr Stunden die Anwendung freigeben kann.

Eine einfache Datensicherung schützt Sie recht gut davor, dass Anwender versehentlich Daten löschen oder Dateien korrumpiert werden. Schwieriger ist das Disaster-Recovery, hier müssen die Daten extern gelagert (und regelmäßig aktualisiert) werden. Außerdem müssen Sie Methoden bereitstellen (und testen!), mit denen Sie die Daten auf leere Maschinen zurückspielen können. Auch das müssen Sie regelmäßig testen, sonst wird es nicht funktionieren, wenn Sie das benötigen.

8.6.3 Programme zur Datensicherung

In den folgenden Abschnitten werden eine Reihe von Programmen zur Datensicherung vorgestellt. Dabei wird nach Programmen unterschieden, die nur die Daten von der Festplatte lesen und auf ein Sicherungsmedium schreiben oder umgekehrt, und solchen Programmen, die dabei auch noch die Verwaltung von vollständigen und inkrementellen Datensicherungen durchführen. Diese Programme speichern eine Liste der gesicherten Dateien, so dass das entsprechende Medium direkt angefordert werden kann.

Low-level-Programme

Im einfachsten Fall kann die Datensicherung mit den dafür geeigneten Standard-Unix-Dienstprogrammen durchgeführt werden. Die einzelnen Programme werden mit einigen Beispielen vorgestellt, um die internen Abläufe, die ähnlich in den komplexeren Datensicherungssystemen benutzt werden, durchschaubarer zu machen.

Diese Programme sichern entweder alle Daten des Systems, einer Partition oder eines Verzeichnisbaums. Über einen entsprechenden `find`-Befehl (siehe Ab-

schnitt 9.2.1, »Suchen von Dateien mit dem Kommando `find`«) oder andere Parameter können nur die seit der letzten Datensicherung veränderten Dateien gesichert werden.

Dieser Weg führt zu einer Reihe von Skripten, mit denen die Handhabung der Programme und der verwendeten Medien vereinfacht werden soll. In den meisten Fällen ist es jedoch besser, ein Programmpaket zur Datensicherung zu verwenden. Viele dieser Pakete sind frei verfügbar, die Auswahl eines speziellen Systems ist von einer Reihe von Faktoren abhängig und sollte vom Systemadministrator lokal getroffen werden.

Das Programm tar

Das Programm `tar` (Tape Archiver) diente zunächst zur Speicherung von Archiven mit Dateien auf Magnetbändern. Heute werden auch Disketten, Dateien oder Netzwerklaufwerke unterstützt.

Ältere `tar`-Versionen konnten nur Dateien, Verzeichnisse und Links sichern, nicht jedoch die Gerätedateien (Devices in `/dev`). Dies war nur im Programm `cpio` implementiert. GNU-`tar` ist auch in der Lage, spezielle Dateien (Gerätedateien und Pipes) zu sichern, so dass nicht mehr `cpio` verwendet werden muss.

Auspacken von Archiven

Das Programm `tar` komprimiert die Daten nicht, sondern fasst nur die einzelnen Dateien in einem Archiv zusammen. Die Komprimierung kann (nur) bei GNU-`tar` durch die Option `-z` eingeschaltet werden. Beim Standard-`tar` muss die Ein- oder Ausgabe durch einen entsprechenden Komprimierer bearbeitet werden. Die Erweiterung des Dateinamens mit `.Z` steht dabei für die Komprimierung mit `compress`, die Erweiterung `.gz` für die Verwendung von `gzip`. Bei der Speicherung von komprimierten `tar`-Archiven auf DOS-Datenträgern wird die Erweiterung `.tar.gz` zu `.tgz` verkürzt. Analoges gilt für mit `compress` gepackte Archive, die statt `.tar.Z` dann `.taz` genannt werden.

Viele Programmpakete werden im `tar`-Format verbreitet. Zur Verringerung der zu übertragenden Datenmengen sind die Dateien in der Regel komprimiert. Als Packer wird entweder `compress` oder `gzip` verwendet. Da `gzip` wesentlich besser komprimiert und die mit `compress` gepackten Daten lesen kann, wird hier nur das Programm `gzip` verwendet. Das Paket `bc-1.02.tar.gz` wird vom Standard-Unix-`tar` mit dem in Listing 8.5 dargestellen Befehl entpackt.

```
(linux):~$ zcat bc-1.02.tar.gz | tar -xvf -
```

Listing 8.5 Auspacken einer `tar`-Datei

Die Optionen können hier mit dem Minuszeichen (`-`) eingeleitet oder auch direkt angegeben werden. Dabei bedeutet die Option `-x` (`--extract`), dass das entsprechende Archiv entpackt werden soll. Die Option `-v` (`--verbose`) bewirkt, dass

`tar` eine Liste der Dateien in dem Archiv auf die Standardausgabe ausgibt. Nach der Option `-f` (`--file`) wird der Dateiname des Archivs angegeben. Der Name `-` steht beim Lesen eines Archivs für die Standardeingabe und beim Schreiben für die Standardausgabe. Ist kein Dateiname angegeben, so wird als Standardwert `/dev/rmt0` verwendet. Dieser Wert kann mit der Umgebungsvariablen `TAPE` verändert werden.

Wird ein Dateiname der Form *`Host-Name`*`:`*`Device`* angegeben, so erfolgt die Datensicherung auf dieses Netzwerkgerät. Dazu ist wie bei dem Programm `rsh` oder `rlogin` der entsprechende Eintrag in die Datei `~/.rhosts` bzw. `/etc/hosts.equiv` notwendig. Besser ist es, wenn Sie die Daten mittels `ssh` übertragen (siehe Kapitel 18, »Die Secure Shell ssh«). Dazu geben Sie die Option `--rsh-command=ssh` an.

Soll auf dem entfernten Rechner eine andere Benutzerkennung verwendet werden, so kann diese mit derselben Syntax angegeben werden wie bei dem Programm `rcp`. In Listing 8.6 wird der Floppy-Streamer des Rechners `atlantis` mit der Benutzerkennung `hein` benutzt.

```
(linux):~> tar -zxvf hein@atlantis:/dev/rft0
```

Listing 8.6 Daten auf ein entferntes Bandlaufwerk speichern

Das Archiv kann auch zunächst entkomprimiert und dann mit `tar` bearbeitet werden (Listing 8.7). Dabei wird jedoch mehr Plattenplatz belegt, weil das Archiv unkomprimiert vorliegt.

```
(linux):~> gunzip bc-1.02.tar.gz
(linux):~> tar -xvf bc-1.02.tar
```

Listing 8.7 Dekomprimieren und Auspacken in zwei Schritten

Die zusätzliche Option `-z`, die nur im GNU-`tar` implementiert ist, bewirkt, dass der Komprimierer `gzip` intern aufgerufen wird (Listing 8.8). Dadurch erübrigt sich der explizite Aufruf des Kompressionsprogramms. Alternativ kann mit der Option `-j` das Programm `bzip2` verwendet werden, das noch besser komprimiert.

```
(linux):~> tar -zxvf bc-1.02.tar.gz
```

Listing 8.8 Automatisches Entpacken »on-the-fly«

Erstellen von Archiven

Zum Erstellen von Archiven dient die Option `-c` (`--create`). Nach der letzten Option (meist `-f`, gefolgt von einem Dateinamen) wird eine Liste der zu archivierenden Dateien und Verzeichnisse angegeben. Das obige Archiv kann aus den Dateien neu erstellt werden durch den Befehl in Listing 8.9.

```
(linux):~> tar -zcvf bc-1.02.tar.gz bc-1.02
```

Listing 8.9 Erzeugen eines `tar`*-Archivs*

Bei der Datensicherung auf Magnetband ist es sinnvoll, wenn das Band möglichst lange ununterbrochen läuft. Jede neue Positionierung kostet unverhältnismäßig viel Zeit. Daher kann es sinnvoll sein, auf die Kompression der Daten vollständig zu verzichten und die Blockgröße (Option `-b` bzw. `--block-size`) entsprechend zu vergrößern.

Ein Archiv kann mit der Option `-t` (`--list`) überprüft werden. Wenn gleichzeitig die Option `-v` (`--verbose`) angegeben ist, wird eine erweiterte Liste der im Archiv enthaltenen Dateien ausgegeben.

```
(linux):~> tar -ztvf bc-1.02.tar.gz
```

Listing 8.10 Testen eines `tar`*-Archivs*

Mit der Option `-M` (`--multi-volume`) werden Archive erstellt, die sich über mehrere Datenträger erstrecken können. Dabei ist es jedoch nicht möglich, dieses Archiv zu komprimieren. Dies wäre auch nicht sinnvoll, da sonst nach einem Lesefehler auf dem Sicherungsmedium alle weiteren Daten nicht mehr lesbar wären.

Das Programm `tar` kann durch die Optionen `-g` (bzw. `--listed-incremental`) und `--newer` auch inkrementelle Datensicherungen durchführen. Dann muss jedoch der Anwender sich die Daten der letzten Sicherung notieren und eine Liste der gesicherten Dateien selbst verwalten. Dies wird im Normalfall durch Skripten realisiert. Es existieren eine Reihe von Backup-Skripten, die einfach verwendet werden können. Ein Beispiel für ein Skript zur inkrementellen Sicherung der `home`-Verzeichnisse ist in Listing 8.11 abgedruckt.

```
#!/bin/bash
# Inkrementelle Datensicherung
now=$(date)
then=$(cat date.dump)
tar --create  --verbose --file /dev/rft0 --block-size 126 \
        --after-date $then \
        --label Dump from $then to $now \
      /home
echo $now > date.dump
```

Listing 8.11 Beispiel für ein Skript zur inkrementellen Datensicherung

Zur Überprüfung der Datensicherung sollte stets die Option `-W` (bzw. `--verify`) angegeben werden. Sollen einzelne Dateien von der Sicherung ausgeschlossen werden, so kann mit der Option `-X` eine Datei angegeben werden, die eine Liste mit regulären Ausdrücken enthält. Die Dateien, deren Namen zu den regulären Ausdrücken passen, werden nicht in das Archiv mit aufgenommen. Im Listing 8.12 werden alle Objekt- und Backup-Dateien nicht gesichert.

```
(linux):~> cat exclude
*~
*.o
(linux):~> tar -zcvXf exclude archiv.tgz .
```

Listing 8.12 Ausschließen von Dateien aus der Sicherung

Spezielle Funktionen

Das Programm `tar` kann auch verwendet werden, um ganze Verzeichnisbäume zu kopieren. Programme wie `cp` oder `rcp` lösen beim Kopieren die symbolischen Links auf und kopieren die Datei selbst anstelle des Links. Bei der Verwendung von `tar` bleiben diese Links erhalten (Listing 8.13). Seien Sie vorsichtig und packen Sie keine absoluten Pfade ein. GNU-`tar` entfernt zwar den führenden Schrägstrich und speichert relative Pfade (was man mit der Option `-P` bzw. `--absolute-names` verhindern kann), andere `tar`-Versionen tun das jedoch nicht.

```
(linux):~> tar -cf - . | ( cd /tmp && tar -xvf -)
```

Listing 8.13 Kopieren von Verzeichnisbäumen mit `tar`

Das Starten einer neuen Shell auf der rechten Seite des Pipe-Symbols kann unterbunden werden, indem mittels der Option `-C` (`--directory`) in das dort angegebene Verzeichnis gewechselt wird (Listing 8.14).

```
(linux):~> tar -cf - . | tar -xvCf /tmp -
```

Listing 8.14 Kopieren von Verzeichnisbäumen mit `tar`

Bei einem Netzwerk kann der eine `tar` auf einem anderen Rechner gestartet werden (Listing 8.15). Damit können dann Verzeichnisbäume über ein Netzwerk kopiert werden. Bei der Verwendung von `ssh` wird nach einem Passwort gefragt, bei `rsh` müsste die Anmeldung ohne Passwort konfiguriert sein. Wenn Sie `tar`-Archive als `root` auspacken, dann können Sie mit der Option `-p` (`--same-permissions` oder `--preserve-permissions`) die Berechtigungen aus dem Archiv übernehmen.

```
(linux):~> ssh Host tar -cf - . | tar -xvCf /tmp -
```

Listing 8.15 Verzeichnisse über Netzwerk kopieren

Hier werden Dateien einfach kopiert. Zur Verringerung der Netzlast kann es sinnvoll sein, einen Abgleich zwischen den Verzeichnissen vorzunehmen und nur die veränderten Dateien zu übertragen bzw. überzählige Dateien zu löschen. Dazu kann das Programm `rdist` oder `rsync` verwendet werden.

Das Programm `tar` ist bei verschiedenen Systemen auch in der Lage, Dateien mit »Löchern« (*holes*, *sparse files*) zu erzeugen. Dabei werden in den Dateien ent-

haltene Null-Zeichen nicht gespeichert, was z. B. bei Shared Libraries einigen Plattenplatz sparen kann. Dazu wird die Option `-S` (`--sparse`) verwendet, die aber nicht auf allen Unix-Systemen korrekt implementiert ist.

8.6.4 Das Programm cpio

Ein ebenfalls verbreitetes Programm zur Datensicherung ist `cpio`. Dieses Programm benötigt als Eingabe eine Liste der zu sichernden Dateien. Daher wird `cpio` oft gemeinsam mit `find` (siehe Abschnitt 9.2.1, »Suchen von Dateien mit dem Kommando `find`«) eingesetzt. In der GNU-Distribution ist im Paket `cpio` auch das Programm `mt` enthalten. Dieses Programm dient zur Steuerung des Bandlaufwerks (Spulen und Löschen von Bändern, Statusabfrage des Streamers).

Ein Archiv auf einer Diskette wird z. B. durch den Befehl in Listing 8.16 erstellt. Das Programm `find` generiert zunächst eine Liste der zu sichernden Dateien. Es sind alle Möglichkeiten zur Selektion verfügbar, die `find` anbietet. Die Kombination von `cpio` und `find` ist daher sehr flexibel. Die hier verwendeten Optionen des Programms `cpio` sind `-o` (`--create`) für die Erstellung eines neuen Archivs, `-v` (`--verbose`) für die Ausgabe des Namens jeder gesicherten Datei und `-O`, gefolgt von einem Datei- oder Gerätenamen. Ist die Option `-O` nicht angegeben, wird das Archiv auf die Standardausgabe geschrieben. Auch hier ist es möglich, ein Gerät im Netzwerk als Ausgabegerät zu verwenden. Die Syntax entspricht dabei der beim Programm `tar` erläuterten.

```
(linux):~> find . | cpio -o -v -O /dev/fd0H1440
```

Listing 8.16 Erstellen eines `cpio`-Archivs

Das in Listing 8.16, erstellte Archiv kann mit dem Befehl aus Listing 8.17 wieder eingelesen werden. Die Option `-i` (`--extract`) bedeutet, dass ein Archiv eingelesen und die Dateien daraus extrahiert werden sollen. Die Option `-I`, gefolgt von einem Datei- oder Gerätenamen, weist `cpio` an, dass das Archiv nicht von der Standardeingabe, sondern aus der entsprechenden Datei oder dem angegebenen Gerät gelesen werden soll.

```
(linux):~> cpio -i -v -I /dev/fd0H1440
```

Listing 8.17 Einlesen eines `cpio`-Archivs

Auch hier kann es zunächst sinnvoll sein, sich ein Inhaltsverzeichnis des Archivs zu beschaffen. Dazu wird die Option `-t` (`--list`) verwendet (Listing 8.18).

```
(linux):~> cpio -t -v -I /dev/fd0H1440
```

Listing 8.18 Inhaltsverzeichnis eines `cpio`-Archivs

8.6.5 Das Programm afio

Das Programm `afio` ist eine deutlich verbesserte Variante von `cpio`. Die Kommandozeilenoptionen ähneln denen von `cpio`, so dass hier auf eine ausführliche Beschreibung der einzelnen Optionen verzichtet wird.

Eine wesentliche Erweiterung ist, dass einzelne Dateien (mit `gzip`) komprimiert werden können, bevor sie in das Archiv aufgenommen werden. Diese Art der Komprimierung, die mit der Kommandozeilenoption `-Z` aktiviert wird, ist wesentlich sicherer, als das gesamte Archiv zu komprimieren. Bei einem Lesefehler ist nur die aktuelle Datei nicht wiederherstellbar. Die fehlerhafte Datei wird übersprungen und die nächste Datei aus dem Archiv extrahiert und anschließend entkomprimiert. Aufgrund der höheren Datensicherheit trotz Komprimierung wird dieses Programm von einer Reihe von Paketen zur Datensicherung verwendet.

8.6.6 Das Programm dump

Das Programm `dump` ist von anderen Unix-Systemen bekannt. Da auch die interne Struktur des Dateisystems angesprochen wird, ist für jedes Dateisystem eine angepasste Version von `dump` notwendig. Für Linux existieren Portierungen für das Extended-2-Dateisystem, das Minix-Dateisystem und `xfsdump` für XFS. Das Programm `dump` für das Minix-Dateisystem kann nur vollständige Sicherungen erzeugen und wird im Folgenden nicht weiter betrachtet.

Die Kernel-Entwickler raten von der Verwendung von `dump` ab, da aufgrund von Änderungen im Kernel veraltete Daten gesichert werden können. Damit ist der Datensicherung eines aktuell verwendeten Dateisystems nicht zu trauen – meist funktioniert es, aber wenn nicht, hat man ein großes Problem.

Das Programm `dump` ist zum einen ein Low-level-Programm, da es z. B. von dem Paket `amanda` benutzt wird. Zum anderen ist es aber auch ein High-level-Programm, weil in der Datei `/etc/dumpdates` das Datum und der Level der letzten Datensicherung gespeichert werden und damit die Verwaltung der verschiedenen Sicherungsstufen (inkrementellen Sicherungen) durchgeführt wird. Die Datensicherung wird mit dem Programm `restore` wieder zurückgespielt.

Eine vollständige Sicherung der Home-Verzeichnisse auf ein Floppy-Tape erfolgt durch den Befehl in Listing 8.19.

```
(linux):~# dump 0Bbuf 120000 1000 /dev/rft0 /home
```

Listing 8.19 Sicherung mit `dump`

Der Parameter `0` (`dump`-Level) steht für einen vollständigen Dump des Dateisystems. Die verschiedenen `dump`-Level können für inkrementelle und differentielle Sicherungen verwendet werden. Die Option `u` dient zur Aktualisierung der Da-

tei /etc/dumpdates, in der eine Liste aller erfolgreichen dump-Läufe geführt wird. Anhand dieser Liste werden auch inkrementelle Sicherungen durchgeführt.

Wenn das Programm dump die Bandlänge nicht korrekt erkennt und astronomisch viele Bänder anfordert, so müssen entweder die Bandlänge und die Schreibdichte mit den Optionen s und d angegeben oder die Größe und Anzahl der Blöcke auf dem Band mit den Optionen B und b eingestellt werden.

Die Wiederherstellung der Daten kann interaktiv (i) oder ohne Benutzereingriff erfolgen (r, Listing 8.20). Dabei verwaltet restore die einzuspielenden inkrementellen Datensicherungen anhand der in der Datei /etc/dumpdates festgehaltenen Daten.

```
(linux):~# restore rf /dev/rft0
```

Listing 8.20 Zurückspielen einer dump*-Sicherung mit* restore

Bei der interaktiven Wiederherstellung erhält man eine einfache Shell, in der mittels ls und cd navigiert werden kann. Einzelne Dateien oder Verzeichnisse können mit add angefordert oder später mit delete wieder aus der Anforderung entfernt werden. Mit dem Befehl extract wird das Band eingelesen und die entsprechenden Dateien werden wieder hergestellt.

8.7 High-level-Programme

Die Verwendung von Low-level-Programmen hat einige Nachteile. So muss der Systemadministrator Skripten zur Erstellung von vollständigen und inkrementellen Sicherungen schreiben. Dabei muss auch eine Verwaltung von verschiedenen Mediensätzen durchgeführt werden. Außerdem ist es sinnvoll, dass eine Liste aller gesicherten Dateien geführt wird, so dass bei der Wiederherstellung einzelner Dateien sofort bekannt ist, welches Magnetband eingelegt werden muss.

Aktuelle Linux-Distributionen enthalten eine Reihe von Programmen zur Datensicherung. Sie können Programme wie floppybackup, kbackup oder taper verwenden, um auf Bandlaufwerke oder Disketten zu sichern. Aufgrund der heute üblichen Datenmengen setzt sich die Datensicherung auf CDs langsam durch.

8.7.1 Das Datensicherungssystem amanda

amanda (Advanced Maryland Automated Network Disk Archiver, *http://www.amanda.org/*) ist ein Datensicherungssystem, das für große Datenmengen, die auf vielen Rechnern gespeichert sind, konzipiert wurde. Die Sicherung erfolgt dabei

zunächst mit Standardprogrammen wie `dump` und `tar`. Die Daten werden dann über das Netzwerk zum Backup-Server transportiert. Dort können die Daten zunächst auf einer Spool-Partition abgelegt werden und, um anschließend mit der maximal möglichen Geschwindigkeit auf das Band geschrieben zu werden.

Es ist eine Bandverwaltung implementiert, die das unbeabsichtigte Überschreiben eines Bands verhindern soll und dafür sorgt, dass beim Zurückladen der Datensicherung auch das richtige Band eingelegt ist.

Diese Software ist ein komplettes, konfigurierbares System zur Datensicherung, das aus einer Reihe von Programmen besteht. Die Programme sind in C geschrieben und recht portabel. Sie unterteilen sich in die Server-Programme, die nur auf dem Backup-Server installiert werden müssen, und die Client-Programme. Die Server-Programme sind aufgrund der verwendeten Systemschnittstellen nur bedingt portabel. Die auf den Clients installierten Programme sind durch die bereits erfolgte Portierung auf diverse Unix-Derivate praktisch überall lauffähig.

Für ein privates Linux-System ist diese Software sicherlich überdimensioniert. In großen Netzwerken ermöglicht `amanda` die Sicherung aller beteiligten Workstations, auch wenn diese unter den verschiedensten Unix-Varianten betrieben werden.

Die Installation ist deutlich aufwändiger als bei anderen Backup-Programmen, da spezielle Einstellungen für den Datentransfer über das Netzwerk notwendig sind. Außerdem besteht `amanda` aus einer Reihe von Programmen für die Durchführung und das Zurückladen der Datensicherung, aber auch aus Programmen zur Administration des Systems. Da ein Systemadministrator, der ein Datensicherungskonzept für ein Netzwerk erstellen soll, wesentlich tiefer in die Materie einsteigen muss, als das hier möglich ist, werden hier die Installation und Konfiguration nicht weiter beschrieben. Die Dokumentation zu `amanda` ist ausführlich und beschreibt die Installation und Konfiguration. Das Amanda-Kapitel aus [Preston1999] finden Sie auch unter *http://www.backupcentral.com/amanda.html*.

8.8 Tipps und Tricks zur Datensicherung

Datensicherung gehört zu den unbeliebtesten Aufgaben in der Computerwelt. In einer heilen Welt braucht man sie nicht, in der Realität wird man aber früher oder später von der Notwendigkeit eingeholt. Im Folgenden finden Sie einige Überlegungen zum Umgang mit Datensicherungen.

- Machen Sie nicht nur täglich (häufig) eine Datensicherung, sondern testen Sie auch regelmäßig einen Restore. Andernfalls müssen Sie im Ernstfall damit rechnen, Daten zu verlieren.
- Werden alle wichtigen Daten gesichert? Prüfen Sie das gelegentlich nach, insbesondere wenn Sie dateisystemweise sichern und Sie neue Dateisysteme manuell in die Konfiguration aufnehmen müssen. Automatisieren Sie diese Prüfung.
- Es gibt Unternehmen, die einen neuen Rechner erst in Betrieb nehmen, wenn dieser durch ein »Crash-Recovery« neu aufgebaut wurde. Der Rechner wird zunächst installiert und getestet. Nach einem Backup wird das Betriebssystem neu installiert (via Netzwerk und ohne Operatoreingriff) und anschließend die Datensicherung wieder eingespielt. Aufwändig, vermittelt aber die Sicherheit, dass das Crash-Recovery mindestens einmal funktioniert hat. Informationen zu einem »Bare-Metal-Recovery« unter Linux finden Sie unter *http://www.backupcentral.com/linux-bare-metal-recovery.html*.
- Prüfen Sie den Erfolg der Datensicherung, am besten automatisch. Gehen Sie Fehlern unbedingt nach, man kann nie wissen, wann man die Sicherung benötigt.
- Planen Sie ein Crash-Recovery: Welche Systeme sind wichtig und müssen möglichst schnell wieder in Betrieb gehen? Was ist an Hard- und Software und Know-how notwendig? Eine Liste der eingerichteten Partitionen und Informationen zu Erweiterungskarten, belegten Interrupts und ähnliche Informationen können sehr hilfreich sein. Drucken Sie diese Daten aus oder sammeln Sie diese automatisch an einem sicheren Ort.
- Prüfen Sie, wie lange Sie für einen Restore benötigen. Können Sie die in den Service-Level-Agreements (SLAs) vereinbarten Zeiten zum Wiederanlauf der Anwendung einhalten? Rechnen Sie damit, dass zunächst Hardware repariert oder ausgetauscht werden muss und erst dann mit der Wiederherstellung begonnen werden kann.
- Wenn Sie zunächst auf Festplatten sichern und dann auf Bänder migrieren, dann kann die Restore-Zeit deutlich über den Sicherungszeiten liegen. Auch diesen Fall sollten Sie unbedingt testen und dokumentieren.
- Bei Produkten, die nur inkrementell sichern, können die Dateien am Ende auf sehr vielen Bändern verteilt sein. In diesem Fall sollten Sie gelegentlich eine Vollsicherung einschieben, um die Bandwechsel zu vermeiden.

Viele Informationen, Tools und Tipps zur Datensicherung finden Sie unter *http://www.backupcentral.com/*. Das Linux Documentation Project hat die »Linux Complete Backup and Recovery HOWTO« veröffentlicht, dort findet sich u. a. ein nützliches Skript zur Partitionierung.

9 Unix-Tools

»Keep it simple, stupid«
Ein Designziel von Unix

9.1 Small is beautiful

Ein großer Vorteil von Unix sind die vielen mitgelieferten Tools, die die Arbeit am Computer erleichtern. Einige dienen dazu, Dateien zu finden oder Programme regelmäßig auszuführen. Andere können Textdateien relativ komfortabel und schnell bearbeiten. Für praktisch jede (einfache) Aufgabe existiert ein kleines, einfach zu bedienendes Programm. Verschiedene Programme können flexibel miteinander kombiniert werden, was die Einfachheit, Eleganz und Leistungsfähigkeit von Unix ausmacht.

Mit der Shell, den Unix-Tools und `awk` oder anderen Skriptsprachen lassen sich leicht Skripten oder Programme schreiben, die dem Systemverwalter die lästigen Routineaufgaben, wie zum Beispiel das regelmäßige Kürzen der Log-Dateien, abnehmen können. Auch viele andere Aufgaben, wie die Bearbeitung von Textdateien jeder Art, z. B. für Ad-hoc-Auswertungen, können mit den Unix-Dienstprogrammen mit relativ geringem Aufwand durchgeführt werden. Dazu müssen jedoch die Möglichkeiten der einzelnen Tools zumindest grundsätzlich bekannt sein, damit man für das Problem das passende Werkzeug oder einen geeigneten Lösungsweg wählen kann.

Ein Unix-System kann als »Werkzeugkasten« mit einer Reihe von einfachen Werkzeugen betrachtet werden. Die Kombination von verschiedenen Programmen, die einzeln nur einen Bruchteil der Arbeit erledigen können, kann dem Programmierer oder Systemverwalter oft stundenlange stupide Tätigkeiten ersparen. Die Kombination der Programme erfolgt durch Kontrollstrukturen, z. B. der Shell, und die einfache Weiterleitung der Daten mittels Pipes. In diesem Fall spielen einige Designentscheidungen eine wichtige Rolle:

- Jedes Programm soll eine (einfache) Aufgabe erledigen.
- Programme lesen von der Standardeingabe. Das kann eine Datei, eine Pipe oder das Terminal des Benutzers sein.
- Programme schreiben ihre Ergebnisse auf die Standardausgabe. Das kann eine Datei, eine Pipe oder das Terminal des Benutzers sein.
- Programme dürfen keine zusätzlichen Ausgaben auf der Standardausgabe erzeugen. Für Fehlermeldungen wird die Standardfehlerausgabe verwendet.
- Programme geben über den Erfolg oder Nicht-Erfolg des Laufs durch den Rückgabewert (Return-Code oder Exit-Code) Aufschluss.

Die Unix-Utilities können sehr einfach miteinander kombiniert werden und damit Aufgaben lösen, an die die Programmierer der Utilities niemals gedacht hätten. Bei größeren Projekten oder Programmen sollte man aber dennoch über die Programmierung in C oder einer anderen Programmiersprache nachdenken. Shell-Programme sind langsam und trotz aller Bemühungen relativ fehleranfällig, dafür sehr gut für »Schnellschüsse« und einfache Prototypen geeignet.

Die meisten der Standardprogramme, die unter Linux verwendet werden, entstammen dem GNU-Projekt. Das Ziel des GNU-Projekts (GNU is Not Unix), das von Richard M. Stallman ins Leben gerufen wurde und jetzt von der Free Software Foundation verwaltet wird, ist die Implementation eines verbesserten, freien Unix-Betriebssystems (The Hurd, ausgeschrieben »Herd of Unix Replacing Daemons«). Diese GNU-Programme sind freie Neuimplementierungen der bekannten Unix-Programme. Dabei wird viel Wert darauf gelegt, die in der originalen Implementierung vorhandenen Beschränkungen, wie maximale Zeilenlänge oder die Verarbeitung nur von druckbaren ASCII-Zeichen, zu entfernen. Außerdem ist die Konformität zum POSIX-Standard ein wesentliches Ziel, sofern sie der Weiterentwicklung von The Hurd nicht im Wege steht.

Dies ist die Ursache dafür, dass GNU-Tools sich oft etwas anders als die entsprechenden Programme anderer (älterer) Unix-Systeme (z. B. auf BSD- und System-V-Derivaten) verhalten. Bei inkompatiblen Erweiterungen beachten die GNU-Programme in der Regel die Option `--posix` oder die Umgebungsvariable `POSIXLY_CORRECT`. Dadurch werden die Erweiterungen abgeschaltet und das Programm verhält sich genauso, wie es im Standard festgelegt ist, auch wenn dies manchmal nicht dem entspricht, was der Benutzer erwartet. Ein Beispiel hierfür ist der `df`-Befehl. Der POSIX-Standard legt fest, dass die Ausgabe in 512-Byte-Blöcken erfolgt, das GNU-Programm `df` verwendet als Standard die Ausgabe in den gewohnteren Kbyte. In der Regel sind die Abweichungen und Erweiterungen als solche dokumentiert, so dass man dennoch in der Lage ist, relativ portable Skripten zu schreiben.

GNU-Erweiterungen sind oft nützlich und komfortabel, auch wenn sie dem POSIX-Standard widersprechen oder Erweiterungen dazu sind. Auf vielen kommerziellen Systemen installiert daher der Systemverwalter die GNU-Tools zusätzlich zu den bereits vorhandenen Unix-Versionen. Ein weiterer Vorteil ist, dass die GNU-Tools auf allen unterstützten Plattformen praktisch identisch sind. Auf verschiedenen kommerziellen Unix-Systemen findet man oft unterschiedliche und inkompatible Versionen der verschiedenen Programme, möglicherweise mit verschiedenen, kaum zu umgehenden Fehlern. In diesem Fall dienen die GNU-Programme oft dazu, eine einheitliche Oberfläche für Anwender oder Skripten zu schaffen.

9.2 Dateiverwaltung

Ein komplettes Linux-System besteht aus mehreren tausend Dateien. Die Suche nach einer Datei kann da manchmal fast aussichtslos erscheinen. Zum Glück gibt es aber Tools, die die Suche nach Dateien erleichtern. Das (Unix-)Kommando `find` sucht im aktuellen Verzeichnisbaum Dateien und Verzeichnisse nach bestimmten Kriterien. Eine andere Möglichkeit besteht darin, das GNU-Tool `locate` zu verwenden. Dieses durchsucht eine mit `updatedb` erstellte Datenbank nach dem Dateinamen.

9.2.1 Suchen von Dateien mit dem Kommando find

Mit dem Kommando `find` können Sie Dateien anhand vieler verschiedener Kriterien im Dateisystem suchen und finden. `find` kann eine Liste der gefundenen Dateien ausgeben, die dann von anderen Programmen, wie zum Beispiel dem `awk` (siehe Abschnitt 9.6, »Die Programmiersprache `awk`«), weiterverarbeitet werden kann. Oder man gibt dem Kommando `find` einen anderen Unix-Befehl an, der für jede gefundene Datei ausgeführt wird.

Dem Kommando `find` können als erster Parameter ein oder mehrere Ausgangsverzeichnisse angegeben werden, andernfalls beginnt die Suche im aktuellen Verzeichnis. Es werden alle dort enthaltenen Unterverzeichnisse durchsucht. Die Suchtiefe kann durch die Option `-maxdepth` eingeschränkt werden, mit der Option `-xdev` oder `-mount` lässt sich die Suche auf ein Dateisystem begrenzen.

Danach können Sie verschiedene Suchkriterien angeben, die für die Dateien gelten sollen. Mögliche Kriterien sind z. B. der Dateiname (oder Suchmuster dafür), der Typ der Datei oder die letzte Zugriffs- oder Änderungszeit oder die Statusänderungszeit. Um zum Beispiel nach dem Namen einer Datei zu suchen, gibt man `-name` *`Dateiname`* ein. Eine tabellarische Übersicht über die wichtigsten Kriterien finden Sie in Tabelle 9.1. Beachten Sie, dass sich GNU-`find` deutlich von anderen `find`-Versionen unterscheidet.

Option	Bedeutung
`-mindepth`	Minimale Pfadlänge
`-maxdepth`	Maximale Pfadlänge
`-xdev`	Nur das aktuelle Dateisystem durchsuchen
`-atime`	Suche nach der Accesstime der Datei
`-mtime`	Suche nach der Modifikationszeit
`-ctime`	Suche nach der Zeit der letzten Statusänderung
`-mmin`	Zeit in Minuten statt in Tagen
`-anewer` *`Datei`*	Modifikation neuer als von der Datei *`Datei`*
`-empty`	Findet nur leere Dateien oder Verzeichnisse

Option	Bedeutung
`-uid` *uid*	Dateieigentümer hat die UID *uid*
`-user` *name*	Die Datei gehört dem Benutzer mit dem Namen `name`
`-gid` bzw. `-group`	Analog für die Gruppenzugehörigkeit
`-inum` *i-Node*	Die Datei hat die angegebene i-Node-Nummer
`-name` *Muster*	Der Name der Datei entspricht *Muster*
`-newer` *Datei*	Findet Einträge neuer als *Datei*
`-nouser` bzw. `-nogroup`	Benutzernummer bzw. Gruppennummer nicht bekannt
`-regex` *Muster*	Regulärer Ausdruck für das *gesamte* Suchergebnis
`-perm` *Mode*	Berechtigungen des Verzeichniseintrags
`-size` *Größe*	Größe der Datei, Default in 512-Byte-Blöcken
`-type` *Typ*	Typ der Datei, siehe Tabelle 9.2
`-a` bzw. `-o`	Verknüpfung der Bedingungen mit UND bzw. ODER
`!` bzw. `-not`	Negierung der folgenden Bedingung
`( ... )`	Erzwingung einer bestimmten Auswertungsfolge

Tabelle 9.1 Übersicht über die wichtigsten `find`*-Kriterien*

Die verschiedenen Ausdrücke werden implizit durch UND verknüpft. Der in Tabelle 9.1, angegebene Parameter `-a` ist also nur in seltenen Fällen notwendig. Die erzwungene Reihenfolge der Auswertung durch die runden Klammern ist eher notwendig, wenn man mit komplexen, mit UND und ODER verknüpften Bedingungen arbeitet. In der Praxis findet man derartige Konstruktionen in der Regel in Skripten versteckt, manuell werden meist nur recht einfache Suchen durchgeführt.

Parameter	Bedeutung
`b`	Block-Gerät
`c`	zeichenorientiertes Gerät
`d`	Verzeichnis (directory)
`p`	Named Pipe (FIFO)
`f`	Reguläre Datei
`l`	Symbolischer Link
`s`	Socket

Tabelle 9.2 Parameter für die `-type`*-Option von* `find`

Mit einem Parameter muss dem Kommando `find` noch mitgeteilt werden, was zu tun ist, wenn eine passende Datei gefunden wurde. Hier besteht die Möglichkeit, einfach nur den Dateinamen auszugeben (Option `-print`) oder andere Unix-Programme, mit dem gefundenen Dateinamen als Parameter, zu starten (Option `-exec`). Das GNU-Programm `find` verwendet als Standardaktion `-print`,

wenn nichts angegeben wurde; andere Unix-Systeme tun dies auch, – oder auch nicht. Wenn Sie portabel bleiben wollen, geben Sie stets `-print` mit an. In Listing 9.1, wird nach allen Dateien im Verzeichnis `/etc` gesucht, deren Name dem Muster `*net*` entspricht.

```
(linux):/$ find /etc -name '*net*' -print
/etc/rc.d/rc.inet1
/etc/rc.d/rc.inet2
/etc/inet
/etc/inetd.conf
/etc/networks
```

Listing 9.1 Suche nach Dateien mit `find`

Durch den Parameter `/etc` wird das Verzeichnis bestimmt, das durchsucht werden soll. Aufgrund des Parameters `-name '*net*'` sucht `find` nach Dateinamen, deren Name dem Suchmuster `*net*` entspricht. Der zu suchende Dateiname sollte in einfache Anführungszeichen gesetzt werden, damit die Shell, die `find` aufruft, die Sonderzeichen nicht interpretiert. Durch den Parameter `-print` wird `find` angewiesen, den kompletten Dateinamen auszugeben.

Wenn Sie die Option `-printf` angeben, so können Sie die Ausgabe durch eine Formatangabe modifizieren. Dadurch können Sie z. B. das Änderungsdatum, den Eigentümer oder die Dateigröße in die Ausgabe aufnehmen. Die Manpage zu `find` enthält eine vollständige Übersicht über die möglichen Formatierungen. Außerdem sind weitere Ausgabeformate einstellbar. So kann mit `-print0` kein Zeilenvorschub, sondern ein Null-Byte als Trennzeichen zwischen den einzelnen Namen verwendet werden. Das kann für den Umgang mit Dateinamen notwendig sein, in denen ein Zeilenvorschub enthalten ist.

Anstatt die gefundenen Dateinamen nur aufzulisten, kann ein Unix-Kommando mit dem Dateinamen als Parameter gestartet werden. Dies geschieht durch die Verwendung von `-exec`. Dabei wird für jede gefundene Datei ein neuer Prozess erzeugt. Enthält das Suchergebnis viele Dateien, so sollte einer der ebenfalls in Listing 9.2 angegebenen Befehle verwendet werden, die nur einen oder wenige zusätzliche Prozesse erzeugen. Dabei wird entweder ein Mechanismus der Shell oder das Programm `xargs` (siehe Abschnitt 9.3.3, »Überlange Parameterzeilen«) verwendet.

```
(linux):~$ find . -name '*.bak' -exec rm -f {} \;
(linux):~$ rm -f $(find . -name '*.bak')
(linux):~$ find . -name '*.bak' | xargs rm -f
```

Listing 9.2 Ausführen von Befehlen mit `find`

Alle Befehle in Listing 9.2 löschen alle Dateien mit der Endung `.bak`. Der Parameter `-exec` führt für jede gefundene Datei das Kommando `rm -f` aus. Die beiden geschweiften Klammern in der ersten Zeile bilden den Platzhalter für den gefundenen Dateinamen. Durch das Semikolon erkennt der Befehl `find` das Ende des Parameters `-exec`. Da die Shell das Semikolon als Trennzeichen zwischen verschiedenen Befehlen auswerten würde, muss es z. B. mit einem Backslash (`\`) maskiert werden.

Wenn Sie sich nicht sicher sind, ob der angegebene Befehl wirklich bei allen Dateien ausgeführt werden soll, verwenden Sie die Option `-ok` anstelle von `-exec`. Hier fragt `find` bei jeder gefundenen Datei nach, ob das Ausführen des angegebenen Kommandos erwünscht ist.

Ein weiterer interessanter Verwendungszweck der Option `-exec` ergibt sich aus der Verbindung mit dem Kommando `grep`. So ist es zum Beispiel leicht möglich, Dateien zu finden, in denen eine bestimmte Textstelle vorhanden ist. Listing 9.3 listet alle Dateien auf, in denen der eigene Host-Name eingetragen ist. Das Beispiel kann je nach Umfang des Dateisystems recht lange laufen, da alle Verzeichnisse durchsucht werden.

```
(linux):~$ find / -type f -exec grep -l $(hostname) {} \;
/etc/profile
/etc/HOSTNAME
/etc/hosts
/etc/nntpserver
/etc/ftpaccess
```

Listing 9.3 `find` in Kombination mit `grep`

In Kombination mit dem Kommando `cpio` (Abschnitt 8.6.4, »Das Programm `cpio`«) oder `afio` (Abschnitt 8.6.5, »Das Programm `afio`«) kann man mit `find` auch leicht ein einfaches Datensicherungssystem erstellen. Ein sehr einfaches Beispiel sehen Sie in Listing 9.4. In der Regel ist jedoch die Verwendung eines kompletten Sicherungssystems die bessere Wahl (siehe auch Kapitel 8, »Datensicherung«).

```
(linux):~# find . -print|afio -oZf /dev/fd0
./.profile
./.emacs
./diary
...
(linux):~# touch .backup
(linux):~# find . -newer .backup -print|afio -oZf /dev/fd0
./.bash_history
```

Listing 9.4 Datensicherung mit `find` und `cpio`

Nach der Installation einer Linux-Distribution müssen noch einige Dateien manuell angepasst werden, z. B. die Datei `/etc/printcap`. Nach und nach werden immer mehr Änderungen durchgeführt, um das System an die eigenen Bedürfnisse anzupassen. Dies geschieht entweder manuell oder mit den Tools der Distribution. Um nun für eine Sicherung herauszufinden, welche Dateien alle verändert wurden, kann man gleich nach der Installation eines Linux-Systems in `/root` die Datei `Install_Date` anlegen (siehe Listing 9.5).

```
(linux):~# date > /root/Install_Date
```

Listing 9.5 Anlegen einer Datei, die das Installationsdatum enthält

Es ist nicht notwendig, das Datum in der Datei zu sichern, aber man kann so das Datum immer wieder herstellen, falls das Änderungsdatum der Datei versehentlich, z. B. mit `touch`, verstellt wurde. Dann kann man später jederzeit alle veränderten Dateien wiederfinden (siehe Listing 9.6). Ein weiteres, möglicherweise interessantes Datum ist der letzte Neustart des Systems, das man z. B. in dem Skript `rc.local` in eine Datei speichern kann.

```
(linux):~# find / -newer /root/Install_Date -print
/etc/profile
/etc/printcap
```

Listing 9.6 Alle nach der Installation geänderten Dateien finden

Praktisch alle Distributionen lassen sich ohne Probleme und Neuinstallation auf eine neue Version aufrüsten. Trotzdem kann es sinnvoll sein, auf diese Art und Weise eine Sicherung seiner Anpassungen durchzuführen. Man kann alle geänderten Dateien auf einen neuen Rechner übertragen und dort anpassen oder eine Datensicherung durchführen. Oft reicht es aber schon zu wissen, welche Dateien geändert wurden, um mehr über das System zu erfahren.

Das Programm `find` ist ein sehr flexibles und leistungsfähiges Tool für die Suche nach Dateien. Der Benutzer kann nach Namen, Größe, Eigentümer und anderen Bedingungen selektieren, Bedingungen verknüpfen und eine oder mehrere Aktionen auslösen. Nützlich kann die Erstellung einer Dateiliste mit der Aktion `-printf` sein, die neben dem Namen noch weitere Informationen, wie Dateieigentümer, Berechtigungen oder die Größe enthält. Die vollständige Dokumentation zu `find` finden Sie in der Manpage bzw. der Dokumentation im Info-Format.

9.2.2 Suchen mit locate

Ein Nachteil des Kommandos `find` ist, dass die Suche in großen Dateisystemen oder gar auf Netzlaufwerken sehr lange dauert. Aber es gibt noch eine andere Methode, Dateien zu finden: das GNU-Programm `locate`.

Das Kommando `locate` durchsucht statt des Dateisystems eine Datenbank nach dem gewünschten Dateinamen. Diese Unix-Datenbank wird mit dem Kommando `updatedb` erzeugt. Erst nachdem eine Datenbank erstellt ist, kann mit dem `locate`-Befehl gearbeitet werden. Die Suche geht nun sehr viel schneller als mit dem `find`-Kommando, aber möglicherweise ist die Datenbank nicht auf dem letzten Stand, so dass nicht alle Dateien gefunden oder bereits gelöschte Dateien immer noch angezeigt werden. `locate` warnt den Benutzer, wenn die Datenbank älter als acht Tage ist. Ein weiterer Nachteil gegenüber `find` ist, dass nur die Suche nach Mustern im Dateinamen unterstützt wird.

Das Kommando `updatedb` durchsucht alle Verzeichnisse, die in den Umgebungsvariablen `SEARCHPATHS` und `NFSPATHS` angegeben sind. Mit dem Suchmuster in der Variablen `PRUNEREGEX` können Dateinamen oder ganze Pfade davon ausgeschlossen werden, in die Datenbank aufgenommen zu werden. Standardmäßig sind dies die Verzeichnisse `/tmp`, `/usr/tmp` und `/var/tmp`. Es empfiehlt sich, das `/proc` Verzeichnis auch in diesen Ausdruck mit einzubauen. Das `/proc`-Dateisystem enthält nur Daten, die den internen Status des Systems darstellen, und diese ändern sich sehr schnell.

Sie sollten zunächst die Home-Verzeichnisse der Benutzer von der Suche ausnehmen, da auch nicht lesbare Verzeichnisse vom Systemverwalter durchsucht würden. Wenn Sie Ihren Benutzern auch eine Datenbank über alle Home-Verzeichnisse zur Verfügung stellen möchten, starten Sie zumindest diesen Suchlauf als Benutzer `nobody`. Andernfalls kann jeder Benutzer auch eine zusätzliche Datenbank für seine Verzeichnisse anlegen und standardmäßig von `locate` durchsuchen lassen. Das Listing 9.7 enthält ein Beispiel für die Erstellung und Abfrage einer eigenen `locate`-Datenbank.

```
(linux):~> updatedb --localpaths=/home/jochen \
? --output=update.db
(linux):~> locate -d update.db:/var/lib/locatedb passwdtab
```

Listing 9.7 Erstellen einer eigenen `locate`-Datenbank

Das Kommando `updatedb` sollte in die `crontab` eingetragen werden (siehe Listing 9.36), damit die Datenbank regelmäßig auf den neuesten Stand gebracht wird. Die Datenbank befindet sich üblicherweise in dem Verzeichnis `/var/lib`. Der Name (`locatedb`) der Datenbank, die normalerweise durch `updatedb` erzeugt wird, kann durch Setzen der Variablen `LOCATE_DB` überschrieben werden. So kann man sich beispielsweise für jede CD eine Datenbank anlegen. Dem Kommando `locate` können mit dem Parameter `-d` auch mehrere Datenbanknamen, durch Doppelpunkte getrennt, angegeben werden.

Im folgenden Beispiel (siehe Listing 9.8) wird eine Datenbank unter dem Namen `cd_linux2_1.codes` im Verzeichnis `~/locate` angelegt, die den Inhalt der CD enthält. Nachdem die Datenbank angelegt ist, kann mit dem Befehl `locate` mit

der Option `-d ~/locate/cd_linux2_1.codes` in dieser Datenbank gesucht werden. Bei Freshmeat (*http://freshmeat.net/*) finden Sie ausgefeilte Programme für diesen Zweck.

```
(linux):~$ export LOCATE_DB=~/locate/cd_linux2_1.codes
(linux):$ export SEARCHPATHS='/CD'
(linux):$ export PRUNEREGEX=' '
(linux):$ updatedb
(linux):$ locate -d ~/locate/cd_linux2_1.codes ftape
```

Listing 9.8 Aufbereitung einer Suchdatenbank für eine CD

Die Nachteile dieses Systems liegen darin, dass die Unix-Datenbank möglichst häufig aktualisiert werden muss, da sonst neuere Dateien nicht gefunden werden können oder bereits gelöschte Dateien immer noch angezeigt werden. Auch ist die Suche selbst nicht so flexibel wie beim `find`-Kommando. Es kann nicht nach Datum, Eigentümer oder dergleichen gesucht werden, sondern nur nach dem Dateinamen. Außerdem ist ein Lauf von `updatedb` relativ aufwändig, da ein `find` über die gesamte Festplatte durchgeführt wird.

Alles in allem lohnt sich der Einsatz von `locate` immer dann, wenn Sie viele Benutzer auf Ihrem System haben und der Rechner über genügend »ruhige« Zeiten verfügt, in denen die Datenbank neu aufgebaut werden kann. Wenn Sie auf Ihrem System feststellen, dass Sie häufig Dateien anhand ihres Namens suchen, dann lohnt sich `locate` auf einem privaten System; bei mir wird `updatedb` alle vier Stunden aufgerufen.

9.2.3 Wo ist das Programm xyz?

Sucht man den kompletten Namen einer ausführbaren und im Suchpfad (!) stehenden Datei, so kann man die langwierige Suche mit `find` umgehen. Anstatt das Programm `find` zu verwenden, sollte man das Kommando `which` benutzen. Bei der `tcsh` ist `which` ein interner Befehl der Shell. Benutzer der `bash` können den Aliasnamen `type -path` verwenden. In manchen Unix-Systemen heißt das Kommando auch `ewhich`, `whereis` oder `whence`.

Das Kommando `which` (Listing 9.9) durchsucht den kompletten Pfad, der in der Variablen `PATH` steht, nach dem gewünschten Dateinamen. Die Reihenfolge, in der der Pfad abgearbeitet wird, entspricht genau der, die auch die Shell benutzen würde.

```
(linux):~$ which gawk
/usr/bin/gawk
```

Listing 9.9 Die Verwendung von `which`

Mit dem Kommando `whereis` kann man nicht nur Programme, sondern auch die zugehörigen Manpages finden. Mit der Option `-b` wird nur nach Programmen und mit der Option `-m` nur nach den Manpages gesucht. Weitere Optionen und die durchsuchten Pfade finden Sie in der Manpage zu `whereis`.

Mit dem `bash`-Kommando `type -all -path` kann man feststellen, ob zwei verschiedene Versionen eines Programms im Pfad liegen, z. B. in `/usr/bin` und `/usr/local/bin`, und welche davon gestartet werden würde. Dies festzustellen, kann hilfreich sein, wenn man ein Programm neu installiert hat und dennoch die alte Version gestartet wird.

Ein Programm wie `which` ist immer dann sinnvoll, wenn man den Pfad zu einem Programm benötigt. Das kann bei der Installation und Anpassung von Shell-Skripten genauso der Fall sein wie bei der Fehlersuche. Programme wie `strings` oder `ldd` arbeiten mit Dateien und nicht mit Programmen aus dem Suchpfad.

Die Shell `zsh` kennt für `which` den Shortcut `=`. Wenn Sie in der `zsh` das Shell-Skript `~/scripts/dtree` bearbeiten wollen und das Verzeichnis `~/scripts` im Suchpfad enthalten ist, dann können Sie den Befehl `vi =dtree` verwenden.

9.2.4 Was ist denn das für eine Datei?

Ein Unix-System besteht aus praktisch unüberschaubar vielen Dateien. Genauso unüberschaubar ist die Anzahl der Dateitypen. Es gibt Textdateien, Binärprogramme, Shell-Skripten, Bilder und viele mehr. Oft kann man dem Namen, genauer gesagt der Erweiterung, entnehmen, was das für eine Datei ist, manchmal versagt aber dieser Trick. Unter Unix ist es zwar üblich, dass die Endung den Typ der Datei angibt, im Gegensatz zu Windows ist es aber nicht notwendig.

Das Programm `file` öffnet die Datei und versucht, in der Datei `/etc/magic` einen passenden Eintrag zu finden. Anhand von »magischen Kennungen« und Ähnlichem wird versucht, den Dateityp zu bestimmen. Manchmal ist die Ausgabe von `file` überraschend genau, manchmal greift das Programm auch daneben. Das Listing 9.10 zeigt einige Beispiele für die Ausgaben von `file`.

```
(linux):~$ file /vmlinuz /bin/ls announce
/vmlinuz: Linux/x86 kernel image, version 2.0.30
/bin/ls:  ELF 32-bit LSB executable, Intel 80386, version 1,
dynamically linked, stripped
announce: International language text
```

Listing 9.10 Das Programm `file`

Die Ausgaben von `file` sind nur so gut wie die zugrunde liegende Datenbank in `/etc/magic`. Gerade kommerzielle Systeme haben hier oft relativ kleine und

alte Versionen. Moderne Linux-Systeme enthalten sehr gute Versionen dieser Datei. Oft findet man auch einzelne Dateitypen in neuen Programmen definiert, so dass man diese Datei einfach ergänzen kann.

9.2.5 Wo kommt die Datei denn her?

Heutige Linux-Distributionen basieren auf so genannten Paketen, die jeweils einen Teil des Systems enthalten. Die Verwaltung der Pakete wird von einem Paket-Manager wie `rpm` (Red Hat Package Manager) oder `dpkg` (Debian Package Manager) übernommen. In den Paketen sind Programme, Dokumentationen und Konfigurationsdateien enthalten, zum Teil werden bei der Installation einfache Konfigurationsprogramme gestartet. Ein wesentlicher Vorteil der Installation mittels Paketen ist, dass diese einfach zu aktualisieren sind und sich vollständig aus dem System entfernen lassen. Praktisch jede Datei im System war ursprünglich in einem Paket enthalten.

Wenn man nicht weiß, zu welchem Paket eine Datei gehört, kann die entsprechende Funktion des Paket-Managers verwendet werden. Bei `rpm` kann neben dem Namen des Pakets auch die zugehörige Beschreibung oder die Liste aller in diesem Paket enthaltenen Dateien angezeigt werden. Das Listing 9.11 zeigt einige Beispiele. Die RPM-HowTo ist genauso lesenswert wie die entsprechende Webseite (*http://www.rpm.org*).

```
(linux):~$ rpm -qf /usr/bin/units
units-1.0-10
(linux):~$ rpm -qif /usr/bin/units
Name        : units        Distribution: Manhattan
Version     : 1.0               Vendor: Red Hat Software
...
(linux):~$ rpm -qlf /usr/bin/units
/usr/bin/units
/usr/lib/units.lib
/usr/man/man1/units.1
```

Listing 9.11 Abfrage der Paket-Datenbank mit `rpm`

Die Option `-q` allein prüft, ob das entsprechende Paket auf dem System installiert ist, und gibt die Versionsnummer des Pakets aus. Wird wie im ersten Befehl in Listing 9.11 zusätzlich die Option `-f` verwendet, dann ist der folgende Parameter nicht der Name eines Pakets sondern ein Dateiname. `rpm` sucht dann in seiner Datenbank nach einem installierten Paket, das diese Datei enthält, und gibt den Paketnamen aus.

Der zweite Befehl im Listing 9.11 verwendet zusätzlich die Option `-i`, die eine recht ausführliche Beschreibung des Pakets anzeigt. Aus Platzgründen ist die Ausgabe hier nicht vollständig abgebildet. Im letzten Beispiel wird mit Hilfe der

Option `-l` eine Liste aller in dem gesuchten Paket enthaltenen Dateien angezeigt. `rpm` hat noch viele weitere Funktionen, die dem Systemverwalter das Leben sehr erleichtern können, schauen Sie einfach mal in die Manpage.

Die Debian-Distribution verwendet nicht `rpm` sondern `dpkg` als Paket-Manager. Auch dieser bietet die eben erläuterten Funktionen, allerdings mit Hilfe anderer Optionen. Das Listing 9.12 zeigt einige Aufrufe. Für die Installation und Verwaltung von Paketen kann auch `apt` verwendet werden. `apt` ist eine Familie von Tools, die als Frontend zu `dpkg` eingesetzt werden. Außerdem gibt es natürlich sowohl für KDE als auch für GNOME entsprechende grafische Frontends.

```
(linux):~$ dpkg -S /usr/bin/units
units: /usr/bin/units
(linux):~$ dpkg -l units
Gewünscht=Unbekannt/Installieren/R=Entfernen/P=Säubern/Halten
| Status=Nicht/Installiert/Config/U=Entpackt/Fehlgeschl.
Konf./Halb install.
|/ Fehler?=(keiner)/Halten/R=Neuinst. notw/X=beides (Status,
Fehler: GROß=schlecht)
||/ Name        Version     Beschreibung
+++-===========-===========-==============================
ii  units       1.77-1      converts
between different systems of units.
(linux):~$ dpkg -L units
[...]
/usr/bin/units
/usr/share/misc/units.dat
/usr/share/info/units.info.gz
/usr/share/man/man1/units.1.gz
/usr/share/doc/units/changelog.gz
[...]
```

Listing 9.12 Abfrage der Paketdatenbank mit `dpkg`

9.3 Andere kleine Helfer

Es gibt noch ein paar andere kleine Helfer, die hauptsächlich in Shell-Programmen verwendet werden. Für die effiziente Erstellung von Shell-Skripten muss man zumindest die prinzipiellen Möglichkeiten der Utilities kennen. Die genaue Syntax der Befehle kann in der Regel schnell in der Manpage nachgeschlagen bzw. ausprobiert werden. Machen Sie sich keine Sorgen darüber, dass Sie sich die Optionen nicht merken können – dafür gibt es das Online-Handbuch in den Manpages. Die GNU-Utilities reagieren auf die Option `--help` und geben dann eine recht knappe Hilfeseite aus. Andere Unix-Utilities erkennen die Option `-h` oder `-?` und geben eine kurze Übersicht über die Syntax aus.

9.3.1 Der basename einer Datei

Das Kommando `basename` trennt den Dateinamen von einem kompletten Pfad ab. Der Befehl `basename` kann z. B. in Kombination mit dem `find`-Kommando Verwendung finden. Der Befehl `find` liefert, wie oben beschrieben, den kompletten Pfad der gesuchten Dateien. Soll nur der Dateiname ausgegeben werden, so kann dies mit dem Kommando `basename` erfolgen (siehe Listing 9.13).

```
(linux):~$ find / -name '*.conf' -exec basename {} \;
ld.so.conf
syslog.conf
host.conf
inetd.conf
resolv.conf
lilo.conf
dosemu.conf
```

Listing 9.13 Die erste Verwendung von `basename`

Das Kommando `basename` kann auch eine angegebene Erweiterung (Extension) vom Dateinamen abtrennen. Ein Beispiel finden Sie in Listing 9.14. Damit ist es möglich, eine Reihe von Dateien mit einer anderen Erweiterung zu versehen. Zu diesem Zweck kann allerdings auch das Programm `mmv` verwendet werden, das diesen Aufgabentyp flexibler erledigen kann.

```
(linux):~$ find / -name '*.conf' -exec basename {} '.conf' \;
ld.so
syslog
host
inetd
resolv
lilo
dosemu
```

Listing 9.14 Die zweite Verwendung von `basename`

9.3.2 Der Pfad einer Datei – dirname

Das Kommando `dirname` arbeitet ähnlich wie der Befehl `basename`, es wird aber der Pfadname ohne den Dateinamen ausgegeben. Wenn der als Parameter übergebene Name kein Verzeichnis enthält, dann wird ein Punkt (`.`) für das aktuelle Verzeichnis angezeigt. Ein Beispiel finden Sie in Listing 9.15.

```
(linux):~$ dirname /etc/rc.d/rc.local
/etc/rc.d
(linux):~$ dirname /vmlinuz
/
(linux):~$ dirname vmlinuz
.
```

Listing 9.15 Die Verwendung des Programms `dirname`

9.3.3 Überlange Parameterzeilen

Wenn man Ausgaben, z. B. die des `find`-Befehls, als Eingabe von anderen Befehlen verwenden will, kann es passieren, dass die maximal erlaubte Länge der Kommandozeile (bei Linux 2.0 128 Kbyte) nicht ausreicht und die Meldung »Commandline too long« ausgegeben wird. Die einzige Abhilfe ist, die Daten nicht vollständig in der Kommandozeile zu übergeben, sondern sie in handliche Teile zu zerlegen und erst dann zu übergeben. Das Kommando `xargs` erzeugt aus der Standardeingabe eine Kommandozeile, die an den angegebenen Befehl weitergeleitet wird. Wenn die Kommandozeile die maximal erlaubte Länge überschreitet, wird der angegebene Befehl mehrfach nacheinander oder gleichzeitig mit jeweils einem Teil der Parameter aufgerufen. Die Fehlermeldung »Commandline too long« wird bei der Verwendung von `xargs` nie wieder auftauchen.

Wenn man mit dem `find`-Befehl in Kombination mit dem `grep`-Befehl Dateien nach Inhalten durchsuchen will, so bekommt man von `grep` zwar die Zeilen ausgegeben, in denen sich der gesuchte Text befindet, aber nicht den Dateinamen. Hier gibt es mehrere Möglichkeiten:

- Übergeben Sie als zusätzlichen Parameter die Datei `/dev/null` an `grep`.
- Genügt die Ausgabe der Dateinamen, so können Sie die Option `-l` an `grep` übergeben.
- Mehrere Dateien können gleichzeitig mit `xargs` durchsucht werden. Ein Beispiel finden Sie in Listing 9.16.

Dadurch, dass der `grep`-Befehl nicht für jede Datei einzeln aufgerufen wird, wie es `find` mit der Option `-exec` macht, werden jetzt auch die Dateinamen mit ausgegeben. Außerdem ist diese Variante »freundlicher« zu anderen Benutzern, da weniger Systemressourcen in Form von Prozessen, Speicher und Prozessorzeit benötigt werden.

```
(linux):~$ find /etc -type f -print | xargs grep LUENE02
/etc/hosts:192.168.10.1 LUENE02.goe.de LUENE02
/etc/HOSTNAME:LUENE02.goe.de
/etc/rc.d/rc.inet1:/sbin/ifconfig dummy LUENE02
/etc/rc.d/rc.inet1:/sbin/route add LUENE02
```

```
/etc/hosts,v:127.0.0.1          LUENE02.goe.de LUENE02
/etc/hosts,v:127.0.0.1  LUENE02.goe.de LUENE02
```

Listing 9.16 Die Verwendung von `xargs`

`xargs` funktioniert in der Regel sehr gut, in der Kombination mit `find` und Dateinamen kann es jedoch dazu führen, dass Dateinamen, die einen Zeilenvorschub enthalten, falsch interpretiert werden. Die einzige Abhilfe ist, bei `find` statt `-print` die `-print0`-Aktion zu verwenden. Damit `xargs` die Parameter richtig interpretiert, muss hier die Option `-0` angegeben werden. Da das Null-Zeichen in Dateinamen nicht erlaubt ist, kann hier ein Benutzer des Systems keinen Unsinn mehr anstellen.

Oft stellt sich das Problem, eine Reihe von Dateien umzubenennen. Leider kann der Befehl immer nur eine Datei umbenennen (`mv *.c *.c.bak` funktioniert aufgrund der Expansion der Sterne durch die Shell nicht). Stattdessen kann man entweder eine kleine `for`-Schleife verwenden, den Befehl `mmv` oder den Befehl `xargs` mit der Option `-n1 -i`. Die Beispiele in Listing 9.17 kopieren aus dem aktuellen Verzeichnis alle Dateien, die mit `*.c` enden, in das Verzeichnis `backups` und versehen jede Datei mit der Endung `.bak`.

Dies ist ein Beispiel dafür, dass unter Unix oft mehrere Wege zum Erfolg führen. Diese Wege können sich aber in Sachen Komfort und Ressourcenverbrauch zum Teil extrem voneinander unterscheiden. Echte Unix-Kenner kombinieren die zur Verfügung stehenden Tools schnell und elegant und gehen dabei sparsam mit den Systemressourcen um.

```
(linux):~/src$ ls *.c | xargs -n1 -i cp {} backups/{}.bak
(linux):~/src$ for i in *.c ; do
>     mv $i  backup/$(basename $i .c).bak ;
> done
```

Listing 9.17 Verschieben bzw. Umbenennen von mehreren Dateien

Durch die Option `-n1` des Befehls `xargs` wird jeder Dateiname einzeln an den `cp`-Befehl übergeben. Durch die Option `-i` ersetzt `xargs` alle geschweiften Klammern durch den aktuellen Dateinamen (ähnlich wie beim Kommando `find` mit der Option `exec`). Anstelle des Kopierbefehls könnte z. B. auch der Vergleich der Dateien mittels `diff` stehen.

9.3.4 Verarbeiten von Textdateien

Zur Verarbeitung von Textdateien stellt Unix einige Programme zur Verfügung. Praktisch alle Konfigurationsdateien, Programmquellen und Textdateien sind einfache ASCII-Dateien, die ohne spezielle Programme bearbeitet werden können. Dies ist einer der Gründe, warum es unter Unix so viele effiziente Werkzeu-

ge in diesem Bereich gibt. Mit `sed`, `tr`, `awk` und anderen hier vorgestellten Tools lassen sich Textdateien einfach be- und verarbeiten. Es gibt noch viele andere Programme, die hilfreich sein können, wie z. B. `perl` oder Python.

`awk` und `sed` sind Werkzeuge, die eine Skriptsprache zur einfachen Verarbeitung von Textdateien zur Verfügung stellen. Je nach Anwendungszweck kann man `awk` oder `sed` bevorzugen. Will man bestimmte Daten aus Textdateien herausfiltern oder weiterverarbeiten, bietet es sich an, dies mit `awk` zu erledigen. Soll eine Textdatei stellenweise verändert werden, kann das am besten mit `sed` getan werden.

Mit dem Kommando `grep` kann nach Textstellen in Textdateien gesucht werden. Andere kleine Tools wie z. B. `cut` oder `tr` erledigen einfache Aufgaben, die auch mit `awk` oder `sed` lösbar sind, aber durch die Kompaktheit der Programme ist die Ausführungszeit oft erheblich kürzer.

In Dateien nach Textstellen suchen

Der Befehl `grep` (global regular expression print) durchsucht beliebige Textdateien nach einem angegebenen Suchmuster, einem so genannten *Regulärer Ausdruck* (*Regular Expression*). Solche Suchmuster werden von verschiedenen Programmen in ähnlicher Form verwendet, daher wird die Erläuterung der Suchmuster in einem gesonderten Abschnitt beschrieben (siehe Abschnitt 9.4, »Suchmuster (Regular Expressions)«).

Das Kommando `grep` arbeitet zeilenorientiert. Das heißt, die Eingabedatei oder die Standardeingabe wird Zeile für Zeile auf eine Übereinstimmung mit dem Suchmuster überprüft. Alle Zeilen, die zu dem Suchmuster passen, werden auf die Standardausgabe ausgegeben. Das Beispiel in Listing 9.18 gibt alle Zeilen der Datei `/etc/services` aus, die die Zeichenfolge `ftp` enthalten.

```
(linux):~$ grep ftp /etc/services
ftp             21/tcp
tftp            69/udp
sftp            115/tcp
```

Listing 9.18 Eine einfache Anwendung von `grep`

Das Kommando `grep` beachtet, wie fast alle Unix-Tools, Groß- und Kleinschreibung. Soll der `grep`-Befehl die Groß- und Kleinschreibung ignorieren, so muss der Parameter `-i` (`--ignore-case`) angegeben werden. Sollen alle Zeilen ausgegeben werden, die das angegebene Suchmuster nicht enthalten, so kann dies durch den Parameter `-v` (`--revert-match`) geschehen.

GNU-`grep` implementiert zwei weitere Varianten, `egrep` (bzw. `grep -E` oder `grep --extended-regexp`) und `fgrep` (bzw. `grep -F` oder `grep --fixed-strings`). Die Unterschiede liegen darin, dass `grep` die »Basic Regular Expressions« verarbeitet.

`egrep` kann zusätzlich mit den »Extended Regular Expressions« umgehen. `fgrep` ist das alte »fast«-`grep`, das nur konstante Strings sucht.

`grep` kann mit der Option `-n` (`--line-number`) dazu veranlasst werden, zusätzlich zu den Fundstellen auch die Zeilennummer auszugeben. Die Option `-l` (`--files-with-matches`) sorgt dafür, dass `grep` nur die Namen der Dateien, die das Suchmuster enthalten, ausgibt, nicht aber den Text, der in der Zeile enthalten ist. Bei der Suche in Texten ist es oft sinnvoll, zusätzlich zur Fundstelle einige Zeilen Kontext anzuzeigen, dazu dient z. B. die Option `-C` (`--context`). Mehr zu den Kommandozeilenoptionen von `grep` finden Sie in der zugehörigen Manpage.

Um mit `gzip` komprimierte Dateien durchsuchen zu können, gibt es den Befehl `zgrep`. Mit diesem Befehl muss die Datei nicht erst entpackt und anschließend wieder komprimiert werden, um darin zu suchen. Genauso können komprimierte Dateien mit `zcat` oder `zmore` angezeigt werden. Unterschiede zwischen komprimierten Dateien können mit `zdiff` angezeigt werden.

Transformieren von Zeichen mit tr

Das Kommando `tr` kann einzelne Zeichen schneller transformieren als `sed` oder `awk`. `tr` ist allerdings auf die Bearbeitung einzelner Zeichen ausgelegt und kennt keine Suchmuster oder Kontextabhängigkeiten. Will man zum Beispiel alle Kleinbuchstaben in einer Datei als Großbuchstaben ausgeben, so erfolgt das mit dem Aufruf aus Listing 9.19.

```
(linux):~$ echo 'Hallo' | tr '[a-z]' '[A-Z]'
HALLO
(linux):~$ echo 'Hallo' | tr '[:lower:]' '[:upper:]'
HALLO
```

Listing 9.19 Die Verwendung von `tr`

Die Parameter ähneln regulären Ausdrücken, repräsentieren aber nur einfache Zeichenketten. In Tabelle 9.3 finden Sie eine Übersicht über die erlaubten Zeichenklassen.

Zeichenklasse	Bedeutung
`alnum`	Die Buchstaben a–z und Ziffern
`alpha`	Die Buchstaben a–z in Groß- und Kleinbuchstaben
`blank`	Horizontaler Whitespace (Leerzeichen und Tabulatorzeichen)
`cntrl`	Steuerzeichen
`digit`	Ziffern
`graph`	Druckbare Zeichen ohne Leerzeichen
`lower`	Kleinbuchstaben

Zeichenklasse	Bedeutung
`print`	Druckbare Zeichen inklusive Leerzeichen
`punct`	Satzzeichen
`space`	Horizontaler oder vertikaler Whitespace
`upper`	Großbuchstaben
`xdigit`	Hexadezimale Ziffern (0–9 und A–Z)

Tabelle 9.3 Zeichenklassen für das Programm `tr`

Sollen z. B. alle Ziffern aus einer Datei herausgefiltert werden, so kann dies wie in Listing 9.20 geschehen. Die Option `-d` (`--delete`) bedeutet, dass die angegebenen Zeichen aus der Eingabedatei herausgelöscht werden sollen. Eine häufige Anwendung für `tr` ist das Löschen des überflüssigen Carriage-Return (`\r`) aus Windows-Texten.

```
(linux):~$ echo 'Die Geheimzahl ist 4711!' | tr -d '[0-9]'
Die Geheimzahl ist !
```

Listing 9.20 Eine weitere Verwendung von `tr`

Mit `tr` können Sie nicht nur lesbare Zeichen modifizieren, sondern Zeichen auch als Oktalwerte eingeben. Für viele häufig zu bearbeitende Zeichen sind außerdem spezielle Symbole definiert, die aus einem Backslash (`\`), gefolgt von einem Zeichen, bestehen. Tabelle 9.4, enthält eine Übersicht über die gebräuchlichsten Sonderzeichen. Eine vollständige Tabelle aller Sonderzeichen finden Sie in der Manpage zu `tr`.

Zeichen	Bedeutung
`\f`	Seitenvorschub, Control-F
`\r`	Carriage-Return, Control-M
`\t`	Das Tabulatorzeichen, Control-T
`\\`	Ein einfacher Backslash
`\ooo`	Oktale Zeichendarstellung

Tabelle 9.4 Sonderzeichen für `tr`

Als abschließendes Beispiel finden Sie in Listing 9.21 den Aufruf für die so genannte Cäsar-Chiffre (auch als ROT13 bekannt). Bei dieser sehr einfachen Kodierung wird jeder Buchstabe durch den im Alphabet um 13 Stellen verschobenen Buchstaben ersetzt. Das Entschlüsseln erfolgt einfach durch die erneute Anwendung der Kodierung – Sicherheit kann man damit nicht erreichen. Gelegentlich wird dieses Verfahren im Usenet verwendet, um die Pointe eines Witzes oder eventuell beleidigende Passagen unkenntlich zu machen.

```
(linux):~$ echo 'Hallo Welt!' | tr a-zA-Z n-za-mN-ZA-M
Unyyb Jryg!
(linux):~$ echo 'Unyyb Jryg!' | tr a-zA-Z n-za-mN-ZA-M
Hallo Welt!
```

Listing 9.21 Implementation der Cäsar-Chiffre mit `tr`

Zwischen den `tr`-Programmen in BSD- und System-V-Systemen existieren subtile Unterschiede, die die Portierung von Shell-Skripten, die `tr`-Aufrufe enthalten, erschweren. Die Unterschiede und das genaue Format der Parameterstrings sind in der Manpage zu `tr` ausführlich beschrieben. Dort finden Sie auch eine Reihe von Beispielen.

Felder in Zeilen bearbeiten mit cut

Um bestimmte Teile einer Zeile aus einer Datei auszugeben, kann das Kommando `cut` verwendet werden. Dafür eignet sich zwar auch `awk`, `cut` ist aber oft schneller, da es nicht so flexibel und leistungsfähig wie `awk` ist. Das Kommando `cut` kann sowohl bestimmte Felder, die durch ein bestimmtes Zeichen, z. B. durch den Doppelpunkt wie in der Datei `/etc/passwd`, getrennt sind, isolieren als auch zeichenweise ausschneiden. Das Beispiel in Listing 9.22 zeigt, wie aus einer Datei die Zeichen an der Stelle 1 bis 11 ausgegeben werden.

```
(linux):~$ cut -b1-11 artikel.dat
11340556671
11366671332
30113434287
```

Listing 9.22 Zeichenweises Abschneiden mit `cut`

Im nächsten Beispiel (siehe Listing 9.23) sollen die Namen aller Benutzer aus der Datei `/etc/passwd` ausgegeben werden. Das Feld, in dem der Name steht, ist das fünfte Feld. Alle Felder sind durch den Doppelpunkt (`:`) getrennt. Der Parameter `-d` (–delimiter) setzt das Feldtrennzeichen (*Delimitor*). Mit dem Parameter `-f5` (`--fields=5`) wird das fünfte Feld ausgewählt, das den ausgeschriebenen Benutzernamen (Real-Name) enthält.

```
(linux):~$ cut -d: -f5 /etc/passwd
Systemverwalter
New-Admin
Carsten Schabacker
Karl Eichwalder
```

Listing 9.23 Felder aus der Datei `/etc/passwd` *ausschneiden*

Zeichensatzkonvertierung

Zur Umsetzung von Textdateien zwischen verschiedenen Zeichensätzen gibt es eine Reihe von Programmen. Im Rahmen des GNU-Projekts wurde `recode` entwickelt, mit dem Texte aus den und in die unterschiedlichsten Zeichensätze konvertiert werden können. Darunter befinden sich Zeichensätze wie ASCII (IBM-PC-Codepage 437), ISO-8859-1 (ISO-Latin-1) und EBCDIC (IBM-Großrechner). Auch Textformate wie LaTeX oder HTML (Hypertext Markup Language) sind als mögliche Kodierung verwendbar. Die Zuordnung von Zeichen aus unterschiedlichen Zeichensätzen wird dabei durch Tabellen gesteuert. Eine Änderung der Tabellen erfordert jedoch eine Neu-Übersetzung des Programms.

Eine einfache Verwendung ist die Konvertierung von Texten aus dem und in das DOS-Format. Dabei sind zwei Umsetzungen durchzuführen.

- Unter DOS wird meist die IBM-Codepage 437 verwendet, unter Unix der ISO-Latin-1-Zeichensatz. Buchstaben, die auch durch den ASCII-Zeichensatz definiert sind, benötigen keine Umsetzung, Umlaute werden aber z. B. unterschiedlich kodiert.
- Unter Windows wird das Zeilenende einer Textdatei durch die Zeichenfolge CR+LF (Carriage Return und Line Feed), unter Unix jedoch nur durch LF dargestellt.

Damit man sich nicht die Namen der zugehörigen Zeichensatztabellen merken muss, kann man sich z. B. Aliasnamen definieren (Listing 9.24), die für diesen Zweck völlig ausreichen.

```
(linux):~$ alias fromdos='recode ibmp:lat1'
(linux):~$ alias todos='recode lat1:ibmp'
```

Listing 9.24 Aliasnamen für `recode`

Eine ausführliche Dokumentation zur Implementierung und Arbeitsweise von `recode` finden Sie in der Info-Dokumentation, die Sie zum Beispiel mit dem Editor Emacs lesen können. Der POSIX.2-Standard verlangt die Implementierung des `iconv`-Programms, das Bestandteil der GNU-`libc` ist. Bei `iconv` wird die Startkodierung mit der Option `-f` (`--from-code`) und die Zielkodierung mit der Option `-t` (`--to-code`) angegeben. Mit der Option `-l` (`--list`) erhalten Sie eine Übersicht über die implementierten Kodierungen.

Textdateien vergleichen

Mit dem Programm `diff` lassen sich Dateiinhalte vergleichen. Häufig fragt man sich, worin sich Dateien unterscheiden. Das kann bei Sicherungskopien, verschiedenen Versionen einer Datei oder bei Änderungen und Anpassungen, die andere Benutzer durchgeführt haben, sinnvoll sein. In Listing 9.25 finden Sie zwei Dateien, die in den folgenden Beispielen verwendet werden.

```
(linux):~/examples$ cat hello.c
include <stdio.h>
main()
{
  printf(Hello World!\n);
}
(linux):~/examples$ cat hallo.c
include <stdio.h>
main()
{
  printf(Hallo Welt!\n);
}
```

Listing 9.25 Anwendungen für das Programm `diff`

Das Programm `diff` kann die Unterschiede zwischen zwei Dateien sehr flexibel darstellen. Das betrifft sowohl das Umfeld der Unterschiede (nur die betreffenden Zeilen oder einige Zeilen davor und danach) als auch die Darstellung selbst. Die beiden Dateien können nebeneinander auf dem Bildschirm ausgegeben werden. Einige Beispiele für die Anwendung des Befehls `diff` finden Sie im Listing 9.26.

```
(linux):~/examples$ diff hello.c hallo.c
4c4
<   printf(Hello World!\n);
---
>   printf(Hallo Welt!\n);
(linux):~/examples$ diff -u hello.c hallo.c
--- hello.c     Sun Nov  5 10:44:09 1995
+++ hallo.c     Sun Nov  5 10:45:04 1995
@@ -1,5 +1,5 @@
 include <stdio.h>
 main()
 }
-  printf(Hello World!\n);
+  printf(Hallo Welt!\n);
 }
(linux):~/examples$ diff -y --width=40 hello.c hallo.c
include <stdio.h>              include <stdio.h>
main()                          main()
{                               {
  printf(Hello World!\n);   |    printf(Hallo Welt!\n);
}                               }
```

Listing 9.26 Anwendungen für das Programm `diff`

Im ersten Beispiel wird ein einfacher `diff` verwendet. Die Ausgabe ist ein `ed`-Skript, das zur automatischen Änderung auf anderen Rechnern verwendet werden kann. Ein Nachteil dieser Skripten ist, dass der Kontext der Änderungen nicht dargestellt wird. Daher ist diese Funktion nur bei absolut identischen Dateien sinnvoll – nur hier wird das `ed`-Skript funktionieren.

Besser ist die Erstellung von Context-`diff`s (`-C`, `--context`) bzw. Unified-`diff`s (zweites Beispiel im Listing 9.26, Option `-u` bzw. `--unified`). Die Erstellung von Context-`diff`s ist ein Bestandteil des Standard-Unix-Programms, Unified-`diff`s sind eine Erweiterung der GNU-Version. Ich persönlich halte Unified-`diff`s für besser lesbar. In beiden Fällen können Sie die Änderungen mittels `patch` in leicht modifizierte Dateien einspielen. Beachten Sie, dass die `patch`-Programme einiger Unix-Systeme keine unified-`diff`s verarbeiten können. Wenn Sie also Patches zwischen verschiedenen Systemen austauschen müssen, dann sollten Sie sich auf Context-`diff`s beschränken.

Der dritte Befehl im Listing 9.26 zeigt die Darstellung der beiden Versionen nebeneinander (`-y` oder `--side-by-side`). Dies ist bei Dateien mit kurzen Zeilen und einem Terminal mit vielen Spalten oft geeignet, um schnell einen Überblick über die Änderungen zu bekommen. Der Editor Emacs hat spezielle Modi (`ediff` und `emerge`), um komfortabel die Unterschiede zwischen Dateien anzuzeigen und möglicherweise die Dateien entsprechend anzupassen. Dabei werden die Unterschiede unter X in verschiedenen Farben angezeigt und Änderungen können mit einem Tastendruck durchgeführt werden. `ediff` kann unter Emacs für zwei oder drei Dateien (`ediff`, `ediff3`), Puffer (`ediff-buffers`, `ediff-buffers3`) oder Verzeichnisse (`ediff-directories`, `ediff-directories3`) verwendet werden. Daneben können auch Revisionen aus Versionsverwaltungen wie RCS oder CVS (siehe Kapitel 11, »Source- und Konfigurations-Management«) verglichen werden.

Beim Aufruf von `ediff` fragt Emacs nach zwei Dateien, die miteinander verglichen werden sollen. Diese Dateien werden in zwei Puffer geladen, Puffer `A` und Puffer `B`. Mit einzelnen Tastendrücken können Sie sich nun in diesen Dateien bewegen. Die Taste [?] liefert ein Menü mit der Tastaturbelegung. Kopieren Sie sich eine beliebige Datei (z. B. `/etc/profile`) zweimal und modifizieren Sie eine Kopie an beliebigen Stellen. Anschließend verwenden Sie `ediff`, um diese Änderungen anzusehen und eventuell auch in der zweiten Datei durchzuführen.

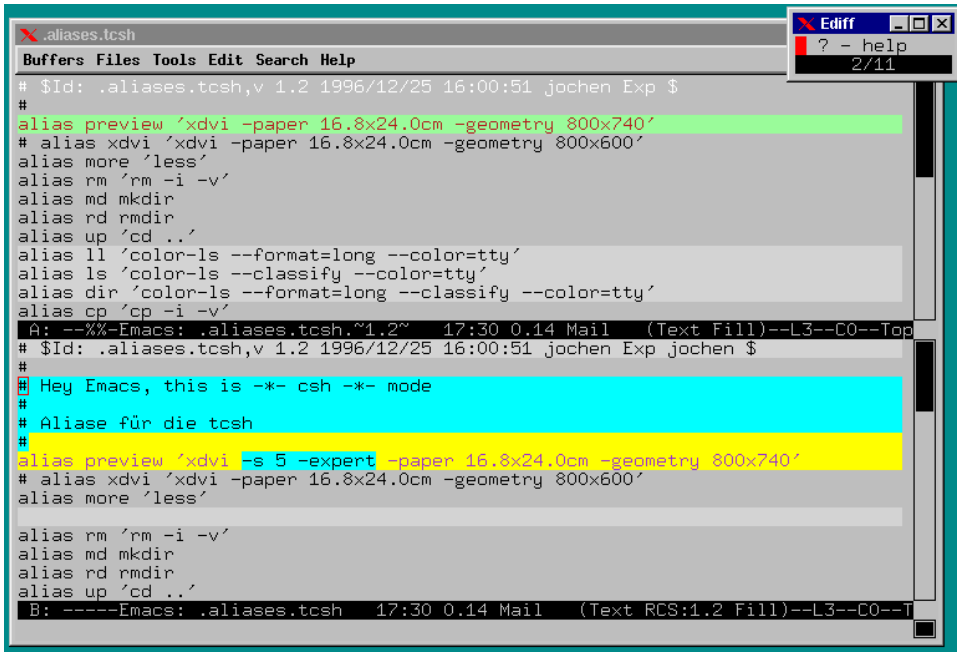


Abbildung 9.1 Der `ediff`-Mode des Emacs in Aktion

Patches erzeugen und einspielen

Viele Programme, die unter Linux eingesetzt werden, werden im Quellcode verteilt. Wenn der Autor eine neue Version herausgibt, so sind oftmals die Änderungen im Vergleich zum gesamten Quellcode relativ klein. In solchen Fällen ist es sinnvoll, nur die Änderungen zu veröffentlichen bzw. von einem `ftp`-Server zu holen, um die Netzbelastung und Übertragungskosten möglichst gering zu halten. Eine Datei, die nur die Änderungen zwischen zwei Versionen enthält, wird Diff oder Patch (Flicken) genannt. Diese Änderungen können mit dem Kommando `patch` automatisch durchgeführt werden.

Ein weiterer Vorteil vom Patches ist, dass diese relativ einfach zu lesen sind und einige Anwender oder Mitentwickler diese Änderungen begutachten und kommentieren. Damit erhofft man sich, dass Patches zunächst beim »Peer-Review« für gut befunden werden und erst dann den Weg ins Release schaffen. Voraussetzung ist, dass eine Entwicklergemeinde miteinander arbeitet und der ursprüngliche Autor eventuell die Änderungen überarbeiten muss.

Diese Patch-Dateien werden mit dem Programm `diff` erzeugt. Dabei können sowohl normale Diffs als auch Context- (`-C`) oder Unified-Diffs (`-u`) erzeugt werden. Normale Diffs sind im Vergleich kleiner als Context- oder Unified-Diffs, können aber nur in absolut identische Dateien eingebracht werden, da das Um-

feld zu den Änderungen fehlt. Context- und Unified-Diffs enthalten (in einem etwas unterschiedlichen Format) jeweils einige Zeilen Kontext zu den Änderungen, so dass die Änderungen notfalls von Hand eingebaut werden können. In der Regel wird man die Änderungen allerdings mit dem Programm `patch` einbauen.

Die Änderungen zwischen den einzelnen Versionen des Linux-Kernels werden als Patches veröffentlicht, so dass diese Patches hier als Beispiel dienen sollen. Zunächst betrachten wir mit Hilfe des Programms `head` (siehe auch den Abschnitt 9.8.5, »Den Anfang von Dateien anzeigen«) den Beginn einer solchen Datei (in Listing 9.27). Der Patch enthält die Namen der geänderten Dateien mit dem Pfad `linux`.

Wird der Patch im Verzeichnis `/usr/src` angewandt, so muss der komplette Pfad erhalten werden. Dies geschieht mit Hilfe der Option `-p0`. In der Regel kann diese Option auch weggelassen werden. Werden aber Dateien neu angelegt, so werden diese im falschen Verzeichnis angelegt.

Wird der Patch im Verzeichnis `/usr/src/linux` angewandt, so kann mit der Option `-p1` der erste Verzeichnisname entfernt werden. Die Angabe der Option `-s` bewirkt, dass nur bei einem Fehler eine Bildschirmmeldung ausgegeben wird.

```
(linux):/usr/src$ zcat 1.2-patches/patch-1.2.13.gz | head
diff -u -r -N v1.2.12/linux/Makefile linux/Makefile
--- v1.2.12/linux/Makefile      Tue Jul 25 12:39:54 1995
+++ linux/Makefile      Wed Jul 26 09:43:44 1995
@@ -1,6 +1,6 @@
 VERSION = 1
 PATCHLEVEL = 2
-SUBLEVEL = 12
+SUBLEVEL = 13

ARCH = i386
```

Listing 9.27 Beginn eines Kernel-Patch

9.4 Suchmuster (Regular Expressions)

Ein *Suchmuster* (*Regulärer Ausdruck*, engl. *Regular Expression*) ist ein Ausdruck, eine Schablone, die mit einer Zeichenkette verglichen wird. Eine Zeichenkette kann einer Schablone entsprechen, muss es aber nicht. Oft werden Suchmuster verwendet, um festzustellen, ob eine Zeichenkette dem Suchmuster entspricht (z. B. vom Kommando `grep`), oder um bestimmte Zeichenketten durch andere zu ersetzen (z. B. mit dem Kommando `sed`).

Die Suchmuster werden von vielen Unix-Programmen, wie z. B. `awk`, `sed`, `grep` oder `perl` verwendet. Jedes dieser Programme verwendet einen Teil der hier beschriebenen Ausdrücke. Zum Teil wird die Bedeutung leicht abgewandelt. In eigenen Programmen können Sie die entsprechenden Funktionen (`regcomp()`, `regexec()`, `regerror()` und `regfree()`) der C-Bibliothek verwenden. Mehr Informationen über diese Funktionen finden Sie in der entsprechenden Manpage. In [Friedl1998] werden reguläre Ausdrücke sehr ausführlich behandelt.

Ein Suchmuster ist eine Schablone, die aus Zeichen besteht. Manche dieser Zeichen haben eine besondere Bedeutung. Diese Zeichen werden im folgenden Text »Jokerzeichen« (oder »Wildcards«) genannt. Einige dieser Jokerzeichen stehen für ein bestimmtes oder für ein beliebiges Zeichen, andere für eine Menge von Zeichen. Sollen diese Jokerzeichen ihre besondere Bedeutung verlieren, so muss ein Backslash (`\`) vorangestellt werden.

Zunächst werden die Jokerzeichen beschrieben, die einem einzelnen Zeichen entsprechen. Die am häufigsten verwendeten Jokerzeichen sind die Zeichen, die sich selbst entsprechen, wie z. B. das `a`. Listing 9.28 zeigt die Verwendung eines Suchmusters, das aus drei solchen Zeichen besteht. Es werden alle Zeilen aus der Datei `datei` ausgegeben, die die Zeichenfolge `abc` enthalten.

```
(linux):~$ grep abc datei
```

Listing 9.28 Ein einfacher regulärer Ausdruck

Das nächste oft verwendete Zeichen ist der Punkt (`.`). Dieses Zeichen ist ein Jokerzeichen mit einer besonderen Bedeutung: Es ist der Platzhalter für ein beliebiges Zeichen. Das Listing 9.29 gibt alle Zeilen aus, die den Buchstaben `a` enthalten, dann ein beliebiges Zeichen und anschließend das Zeichen `c`, also `abc` und `afc`, aber nicht `ac`.

```
(linux):~$ grep 'a.c' datei
```

Listing 9.29 Noch ein einfacher regulärer Ausdruck

Soll geprüft werden, ob eine Buchstabenfolge aus einer bestimmten Menge von Zeichen an einer bestimmten Stelle besteht, so kann dies durch das Zusammenfassen in eckige Klammern (`[` und `]`) geschehen. Das erste Beispiel in Listing 9.30 liefert alle Zeilen aus der Datei, die den Buchstaben `a`, dann entweder `d`, `e` oder `f` und schließlich den Buchstaben `c` enthalten. Im Beispiel ist der reguläre Ausdruck jetzt in einfache Anführungszeichen eingeschlossen – das ist sinnvoll, damit nicht jedes einzelne Shell-Sonderzeichen einzeln maskiert werden muss.

```
(linux):~$ grep 'a[def]c' datei
(linux):~$ grep 'a[a-z]c' datei
(linux):~$ grep 'a[0-9]c' datei
```

Listing 9.30 Weitere Beispiele für reguläre Ausdrücke

Das zweite Beispiel liefert alle Zeilen, die einen kleinen Buchstaben zwischen dem `a` und dem `c` enthalten. Durch einen Bindestrich können Bereiche angegeben werden. So entspricht der Ausdruck `[a-z]` den ASCII-Kleinbuchstaben, `[A-Z]` sind die ASCII-Großbuchstaben. Es können auch Bereiche kombiniert werden: `[a-z0-9]` entspricht allen Kleinbuchstaben und allen Ziffern.

Das dritte Beispiel liefert alle Zeilen der Datei, die zwischen dem Buchstaben `a` und `c` keine Ziffer haben. Das Caret-Zeichen (`^`) als erstes Zeichen einer Zeichenmenge besagt, dass der Ausdruck in den eckigen Klammern logisch umgedreht wird.

Bei so genannten Extended Regular Expressions, wie sie von `egrep` verwendet werden, können als Menge auch Zeichenklassen wie z. B. `[:lower:]` oder `[:upper:]` angegeben werden. Diese Zeichenklassen enthalten, wenn z. B. die deutsche Lokale verwendet wird, auch die Umlaute. Beachten Sie, dass diese Zeichenklassen innerhalb einer Menge angegeben werden müssen, so dass der resultierende reguläre Ausdruck je zwei öffnende bzw. schließende eckige Klammern enthält. In Tabelle 9.5 finden Sie eine Übersicht über alle Zeichenklassen.

Zeichenklasse	Bedeutung
`[:alpha:]`	Alle Buchstaben
`[:upper:]`	Alle Großbuchstaben
`[:lower:]`	Alle Kleinbuchstaben
`[:digit:]`	Alle Ziffern
`[:xdigit:]`	Alle Hexadezimalziffern
`[:space:]`	Alle Leerzeichen-Äquivalente
`[:punct:]`	Alle Trennzeichen
`[:alnum:]`	Alle Buchstaben und Ziffern
`[:print:]`	Alle druckbaren Zeichen
`[:graph:]`	Druckbare Zeichen ohne das Leerzeichen
`[:cntrl:]`	Alle Steuerzeichen
`[:blank:]`	Leerzeichen oder Tabulator

Tabelle 9.5 Zeichenklassen in regulären Ausdrücken

Weitere sehr wichtige Jokerzeichen sind das Sternchen (`*`) und das Plus (`+`). Diese Zeichen sind Wiederholungsoperatoren. Der Ausdruck vor dem Sternchen darf beliebig oft oder auch gar nicht vorkommen. Der Ausdruck vor einem Plus muss mindestens einmal auftauchen, darf sich aber beliebig oft wiederholen. Der Ausdruck im nächsten Beispiel (siehe Listing 9.31) listet alle Zeilen auf, die nach dem Buchstaben `a` eine beliebige Anzahl von Ziffern und dann den Buchstaben `c` enthalten. Es wird auch der Ausdruck `ac` gefunden, da der Buchstabe `b` beliebig oft, also auch gar nicht auftauchen kann.

```
(linux):~$ grep 'a[0-9]*c' datei
```

Listing 9.31 Wiederholungen in regulären Ausdrücken

Andere nützliche Jokerzeichen sind der Zirkumflex (^, Caret-Zeichen) am Beginn und das Dollarzeichen ($) am Ende eines regulären Ausdrucks. Diese beiden Zeichen stehen für den Zeilenanfang und das Zeilenende. Sollen z. B. alle Kommentarzeilen aus einem Shellprogramm ausgegeben werden, so kann das durch den in Listing 9.32 angegebenen Ausdruck geschehen. Um alle Leerzeilen zu finden, kann man den Ausdruck ^$ verwenden.

```
(linux):~$ grep '^#.*'
```

Listing 9.32 Das letzte Beispiel für reguläre Ausdrücke

Tabelle 9.6 enthält eine Übersicht über die wichtigsten Jokerzeichen. Reguläre Ausdrücke werden unter Unix ständig verwendet (z. B. im Editor, am Shell-Prompt oder als Parameter zu grep). Durch den geschickten Einsatz von regulären Ausdrücken lassen sich viele Aufgaben wesentlich vereinfachen. Andererseits können reguläre Ausdrücke recht kompliziert werden und sind dann nur noch schwer nachvollziehbar.

Ausdruck	Bedeutung
.	*Ein* beliebiges Zeichen
[abc]	Eine Menge bestimmter Zeichen an dieser Stelle
[^abc]	Wie oben, aber entgegengesetzte Bedeutung
*	Wiederholt den vorangegangenen Ausdruck beliebig
+	Wiederholt den vorangegangenen Ausdruck mindestens einmal
^	Zeilenanfang
$	Zeilenende

Tabelle 9.6 Reguläre Ausdrücke

Weitere Sonderzeichen werden in der Manpage zu grep erklärt. Wenn Sie mehr zu regulären Ausdrücken erfahren möchten, finden Sie eine ausführliche Dokumentation in [Friedl1998].

9.5 Kommandos automatisch starten

Oft ist es erwünscht, dass z. B. rechenintensive Programme oder Programme, die exklusiven Zugriff auf den Rechner benötigen, zu einem späteren Zeitpunkt, z. B. in der Nacht, ausgeführt werden. Das kann mit dem Kommando at ver-

anlasst werden. Sollen bestimmte Programme regelmäßig zu einer bestimmten Zeit gestartet werden, so kann das Kommando `crontab` verwendet werden, mit dem die Steuertabellen für `cron` verwaltet werden.

9.5.1 Kommandos später ausführen (at)

Um ein Kommando einmalig zu einer bestimmten Uhrzeit auszuführen, kann man das Kommando `at` verwenden. Anwendungen des `at`-Kommandos sind z. B. das Ausführen einer aufwändigen Datenbankverarbeitung zu einem Zeitpunkt, zu dem der Rechner nicht besonders stark ausgelastet ist. Zu diesem Zweck würde sich übrigens auch der Befehl `batch` besonders gut eignen, da er die angegebenen Befehle erst dann ausführt, wenn die Systembelastung unter eine bestimmte Grenze gesunken ist. Listing 9.33 zeigt, wie man um 20:00 Uhr die Datei `/etc/nologin.ttyS1` löschen kann.

Um welche Uhrzeit der Befehl `at` genau gestartet wird, hängt davon ab, wie oft der `atrun`-Befehl ausgeführt wird. Dieser wird in meinem System z. B. nur alle fünf Minuten aufgerufen (siehe den Abschnitt 9.5.4, »Kommandos regelmäßig ausführen (`crontab`)«).

```
(linux):~# at 20:00
>rm -f /etc/nologin.ttyS1
>Ctrl-D
Job c00c95ff4.00 will be executed using /bin/sh
```

Listing 9.33 Start eines Programms mit `at`

Der Befehl `at` erwartet die auszuführenden Befehle von der Standardeingabe. Nach dem Eingeben der Befehle muss die Tastenkombination Strg+d als Endemarkierung gedrückt werden. Damit wird der Auftrag in die Jobqueue (Warteschlange) eingefügt. Um sich zu vergewissern, ob der Befehl `at` das Kommando auch richtig abgespeichert hat, kann man mit dem Kommando `atq` die Warteschlange auflisten lassen (Listing 9.34).

```
(linux):~# atq
Date                    Owner   Queue   Job
20:00:00 02/03/95       root    c       c00c95ff4.00
```

Listing 9.34 Ausgabe des Befehls `atq`

Das Datum und die Uhrzeit geben an, wann der Auftrag ausgeführt werden soll. Der entsprechende Eigentümer des Auftrags sowie eine Identifikationsnummer werden aufgelistet. Soll ein Job doch nicht ausgeführt werden, dann kann man diesen mit dem Kommando `atrm` wieder aus der Jobqueue entfernen (siehe Listing 9.35).

```
(linux):~# atrm c00c95ff4.00
```

Listing 9.35 Entfernen eines `at`*-Jobs mit* `atrm`

Das Kommando `at` akzeptiert die Zeitangaben in verschiedenen Formaten. Die Uhrzeit kann in der Form `HH:MM` oder `HHMM` angegeben werden. Es wird auch `midnight` (Mitternacht), `noon` (Mittag) oder `teatime` (4 Uhr nachmittags) akzeptiert.

Neben der Zeitangabe kann auch das Datum des Tages angegeben werden, wann das Programm die Aktionen starten soll. Das Datum kann z. B. im Format `DD.MM.YY` oder `MM/DD/YY` angegeben werden. Man kann die Zeitangaben auch relativ zur aktuellen Zeit oder zum aktuellen Datum machen. Soll ein Kommando z. B. in zwei Stunden gestartet werden, so muss man `+2 hours` als Zeit vorgeben. Mit `+2 days` kann man ein Kommando in zwei Tagen starten.

Das Kommando `at` liest die auszuführenden Befehle normalerweise von der Standardeingabe. Es kann aber auch eine Datei (Shell-Skript) durch die Option `-f` *`Datei`* angegeben werden. Wenn das auszuführende Kommando Ausgaben auf die Standardausgabe oder die Standardfehlerausgabe machen will, werden diese dem Benutzer, der das `at`-Kommando ausgelöst hat, per Unix-Mail zugestellt.

Der Systemverwalter kann die Verwendung des Kommandos `at` mit den zwei Dateien `/etc/at.allow` und `/etc/at.deny` kontrollieren. Existiert die Datei `/etc/at.allow`, so dürfen nur die Benutzer das Kommando `at` ausführen, die in dieser Datei aufgelistet sind. Wenn diese Datei nicht existiert, aber die `/etc/at.deny` angelegt ist, so dürfen alle Benutzer das Kommando `at` verwenden, außer denen, die in `/etc/at.deny` aufgelistet sind. Fehlen beide Dateien, so darf nur der Systemverwalter (`root`) das Kommando `at` verwenden.

9.5.2 Batch-Verarbeitung simulieren mit batch

Das Programm `at` führt den angegebenen Befehl in praktisch jedem Fall zur angegebenen Uhrzeit aus. Ist der Rechner zu diesem Zeitpunkt extrem belastet, so möchte man manchmal die Ausführung des Prozesses verhindern bzw. zu einem späteren Zeitpunkt einplanen. Der Befehl `batch` plant ein Programm wie `at` ein, führt dieses aber nur aus, wenn die Auslastung des Rechners dies zulässt. Ansonsten ist die Bedienung analog zum Befehl `at`.

9.5.3 Serialisierung von Programmen

Eine weitere Anforderung kann die Serialisierung einiger Programme sein. Ein Beispiel hierfür ist die Verwendung eines kommerziellen Compilers, der zu einem Zeitpunkt auf einem System nur einmal gestartet sein darf. Zur Abarbei-

tung von make-Jobs kann man z. B. eine Print-Queue einrichten, die die entsprechenden Compiler-Aufrufe durchführt. Da in dieser Queue stets nur ein Job aktiv sein kann, werden alle Anforderungen serialisiert.

Für Linux-Anwender sollte es nur selten notwendig sein, zu solchen Maßnahmen zu greifen. Auf anderen Systemen, wo viele Programme durch entsprechende Sperren und Lizenzmanager gesichert sind, kann die Serialisierung aber sinnvoll sein.

9.5.4 Kommandos regelmäßig ausführen (crontab)

Das Kommando at führt ein Kommando nur einmal aus. Sollen bestimmte Kommandos regelmäßig zu bestimmten Zeiten ausgeführt werden, so kann man dies durch einen Eintrag in die crontab-Datei erreichen. Dazu wird der gleichnamige Befehl crontab verwendet. Mit der crontab-Datei lassen sich regelmäßig durchzuführende Routineaufgaben automatisch erledigen. Anwendungsbeispiele sind z. B. das Erzeugen von Indexdateien, die Überwachung des Systems nach verschiedenen Kriterien oder das Löschen von Log-Dateien.

Die crontabs, die mit dem Kommando crontab angelegt werden, werden im Verzeichnis /var/spool/cron abgespeichert. Diese Dateien dürfen nicht direkt geändert werden, sondern müssen durch das Kommando crontab mit der Option -e bearbeitet werden. Es wird dann der Editor, der in der Umgebungsvariablen EDITOR bzw. VISUAL eingetragen ist, mit der aktuellen crontab gestartet. Nach der Änderung der Datei wird diese wieder in das Systemverzeichnis kopiert und der crond-Dämon veranlasst, diese Datei neu einzulesen. Der Dämon crond führt die Befehle in diesen Dateien regelmäßig aus.

Weitere Optionen von crontab sind -l zur Anzeige der aktuellen Tabelle und -r zum Ersetzen der Datei. Achtung: Die Option -r löscht die aktuelle Tabelle, wenn kein Dateiname einer neuen Tabelle angegeben wird.

Anzeigen einer crontab

Eine vorhandene crontab kann durch den Befehl crontab mit der Option -l ausgegeben werden. Die in Listing 9.36 gezeigte crontab führt das Kommando atrun alle fünf Minuten aus (siehe Abschnitt 9.5.1, »Kommandos später ausführen (at)«). Außerdem wird jeden Tag um 22:00 Uhr die Datei /etc/nologin.ttyS1 gelöscht und um 7:06 Uhr wieder angelegt[1].

```
(linux):~# crontab -l
# Run the 'atrun' program every 5 minutes
# This runs anything that's due to run from 'at'.
# See man 'at' or 'atrun'.
```

1. Diese Datei wird von mgetty ausgewertet.

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * *   /usr/lib/atrun
00 22 * * * rm -f /etc/nologin.ttyS1
06 07 * * * touch /etc/nologin.ttyS1
# generate database for locate
0 04,12 * * * updatedb
```

Listing 9.36 Beispiel für eine `crontab`*-Datei für* `root`

Bei der Ausführung der Programme werden automatisch die Umgebungsvariablen `LOGNAME` und `HOME` auf die Werte aus der Datei `/etc/passwd` gesetzt, die Variable `SHELL` jedoch stets auf `/bin/sh`. Die Login-Skripten oder andere Initialisierungen der Shell werden nicht durchlaufen.

Weitere Variablen können durch den Eintrag *`Variable`* `=` *`Wert`* in der `crontab`-Datei gesetzt werden. Dabei wird die Variable `MAILTO` direkt von `cron` verwendet, um den Mail-Empfänger der Programmausgaben zu ermitteln. Ist kein `MAILTO` angegeben, dann wird der Eigentümer der `crontab` verwendet. Wenn der Eintrag eine leere Zeichenkette enthält, dann wird keine Mail versendet.

Aufbau der crontab

Alle Zeilen, die mit der Raute (`#`) beginnen, sind Kommentarzeilen und werden vom `cron`-Dämon (`crond`) ignoriert. Die ersten fünf durch Leerzeichen getrennten Wörter bestimmen, wann das Kommando ausgeführt werden soll. Die erste Zahl sind die Minuten, danach folgen die Stunden, anschließend der Tag, der Monat und schließlich der Wochentag. Ein Sternchen (`*`) bedeutet, dass der Befehl zu jeder Zeit ausgeführt werden soll. Nach den Zeitangaben folgt schließlich das auszuführende Kommando. Es können auch Bereiche, z. B. `20-23`, angegeben werden.

Soll ein Programm alle zehn Minuten ausgeführt werden, so kann die `crontab` einen der in Listing 9.37 angegebenen Einträge enthalten. Beachten Sie, dass nur der erste Eintrag mit allen `cron`-Versionen funktioniert. Die Notation `/10` ist eine Erweiterung des unter Linux häufig verwendeten Vixie-`cron`. Eine ausführliche Beschreibung des Formats einer `crontab`-Datei sowie der Vixie-Erweiterungen finden Sie in der Manpage `crontab(5)`.

```
# Der folgende Befehl wird alle zehn Minuten gestartet
# Dieser Eintrag ist Standard auf System-V-Rechnern
0,10,20,30,40,50 * * * * touch /tmp/cron_is_running
# Alle zehn Minuten starten, Vixie-Cron-Erweiterung
0-59/10 * * * * touch /tmp/cron_is_running.2
# Und noch etwas einfacher
*/10 * * * * touch /tmp/cron_is_running.3
```

Listing 9.37 Befehle alle zehn Minuten ausführen

Manche cron-Versionen verlangen eine Leerzeile am Ende der Datei, vertragen aber keine Kommentare oder Leerzeilen. Im Zweifelsfall finden Sie unter /var/log/cron bzw. der System-Log-Kategorie daemon weitere Informationen.

Die Benutzung von cron kann durch die Dateien /etc/cron.allow und /etc/cron.deny reguliert werden. Wenn eine Datei /etc/cron.allow existiert, dann dürfen nur die Benutzer, die in dieser Datei aufgeführt sind, dieses Tool benutzen. Wenn die Datei /etc/cron.deny existiert, so werden die Benutzer von der Verwendung von cron ausgeschlossen, die in dieser Datei aufgeführt sind. Wenn keine der Dateien existiert, so kann je nach Kompilierung des cron-Dämons entweder jeder Benutzer oder nur root dieses Tool benutzen.

Weitere Beispiele

In Listing 9.38 soll ein Programm morgens um 6 Uhr und abends um 22 Uhr gestartet werden, aber nur wochentags (Montag-Freitag). Dasselbe Programm kann natürlich in weiteren Einträgen erneut verwendet werden. So lassen sich einfach unterschiedliche Rhythmen für Arbeitstage und Wochenenden definieren.

```
0 6,22 * * * 1-5 /usr/lib/uucp/uucico -r1 -spertron
```

Listing 9.38 Zweimal täglicher Aufruf von Montag bis Freitag

Ein Programm soll wie in Listing 9.39 an jedem 1. des Monats um 2:00 Uhr gestartet werden. Dabei ist jedoch keine Einschränkung auf spezielle Wochentage oder gar Feiertage möglich. Wenn Sie einen Wochentag, wie z. B. Sonntag, angeben, dann wird das Programm zusätzlich an allen Sonntagen ausgeführt. Es gibt keine einfache Möglichkeit, Jobs immer am letzten Tag eines Monats ausführen zu lassen.

```
0 2 1 * * /root/scripts/clean_logs
```

Listing 9.39 Monatliches Ausführen eines cron-Jobs

Um ein Programm jeden Sonntag auszuführen, kann der in Listing 9.40 angegebene Eintrag verwendet werden. Sie können Monate und Wochentage als Nummer und als ausgeschriebenen Namen verwenden. cron erkennt die dreistelligen Kürzel der englischen Monatsnamen und Tagesbezeichnungen. Wenn Sie diese Notation verwenden, sind Aufzählungen nicht mehr erlaubt.

```
0 1 * * sun /root/scripts/clean_up
```

Listing 9.40 Job einmal in der Woche ausführen

Wenn Sie auf einem aktiv benutzten System arbeiten, dann kann es sinnvoll sein, cron-Jobs nicht zu »runden« Zeiten wie 12:00 Uhr einzuplanen, sondern Zeiten

wie 11:37 Uhr zu verwenden. Wenn fünf Programme um Mitternacht gestartet werden, dann belastet das den Rechner mehr, wie wenn man diese über einen längeren Zeitraum verteilt startet.

Mehr Informationen zu `cron` und seinen Möglichkeiten finden Sie in der Manpage. Unter Linux wird in der Regel der sehr gute Vixie-`cron` von Paul Vixie verwendet, der über einige Erweiterungen im Vergleich zum Standard-Unix-`cron` verfügt.

Cronjobs und Pakete

Auf der einen Seite hat der Systemverwalter die Freiheit, Cronjobs so einzuplanen, wie es ihm sinnvoll scheint. Auf der anderen Seite darf man heute erwarten, dass bei der Installation eines Pakets eventuell notwendige Jobs mit sinnvollen Vorgaben automatisch eingeplant werden. Dafür wäre es notwendig, eine bestehende `crontab` zu überschreiben oder zu modifizieren. Besonders lästig wird es bei Upgrades: Wie kann man den Cronjob wiederfinden und modifizieren?

Bei der Debian-Distribution wird das Problem dadurch gelöst, dass Pakete in das Verzeichnis `/etc/cron.d` Crontab-Fragmente ablegen können, die dann zur passenden Zeit ausgeführt werden können. Da jedes Paket eine eigene Datei verwendet, kann man Änderungen durch den Systemverwalter erkennen und gegebenenfalls nachfragen und bei einer unveränderten Datei eine neue Version installieren.

Cron und Rechner, die nicht permanent laufen

Das Programm `cron` geht davon aus, dass es permanent läuft. Sollte das nicht der Fall sein, z. B. weil der Rechner ausgeschaltet war, dann gehen die in der Zeit zu startenden Jobs einfach verloren. Für viele Systeme und viele Jobs ist das kein großes Problem, für einen Rechner, der jeden Abend oder am Wochenende ausgeschaltet wird, kann es zu einem Problem werden. Viele Jobs zur Systemverwaltung werden entweder Nachts oder am Wochenende gestartet, so dass z. B. keine Log-Dateien rotiert werden.

Das Programm `anacron` bietet hier eine Abhilfe. In der Datei `/etc/anacrontab` werden Jobs hinterlegt, die nach einer bestimmten Anzahl Tage gestartet werden sollen. Jobs, die häufiger gestartet werden sollen, können nicht mit `anacron` verwaltet werden, genauso wie `anacron` die Ausführung zu einem bestimmten Tag nicht garantieren kann.

Um das System nicht zu überlasten, kann zusätzlich zu jedem Job eine Verzögerung angegeben werden. Das System wartet nach einem Systemstart diese Zeit, bevor der Job gestartet wird. Listing 9.41 zeigt ein Beispiel, wie es in der Debian-Distribution verwendet wird.

```
# These replace cron's entries
#Tage  Min  Befehl
1       5    cron.daily   nice run-parts --report /etc/cron.daily
7      10    cron.weekly  nice run-parts --report /etc/cron.weekly
30     15    cron.monthly nice run-parts --report /etc/cron.monthly
```

Listing 9.41 Die Datei `/etc/anacrontab`

cron in der LSB

Bei LSB-konformen Systemen hat der Systemverwalter die alleinige Kontrolle über die systemweite `/etc/crontab`. Pakete dürfen diese Datei nicht verändern, sondern sollen ihre `crontab`-Einträge in eigenen Dateien unter `/etc/cron.d/` ablegen. Diese Dateien sind als Konfigurationsdatei zu markieren, so dass ein Upgrade diese nicht ungefragt überschreibt.

9.6 Die Programmiersprache awk

Mit dem Kommando `awk` lassen sich strukturierte Textdateien verarbeiten und z. B. als formatierte Tabelle ausgeben. Der Name `awk` setzt sich aus den Initialen der Erfinder dieser Sprache zusammen: Alfred V. Aho, Peter J. Weinberger und Brian W. Kernighan. Mit dem Kommando `awk` steht eine der Programmiersprache C in gewisser Weise ähnliche und doch ganz andere Interpretersprache zur Verfügung.

Ein `awk`-Programm besteht aus Funktionsblöcken, die wie in C in geschweifte Klammern eingeschlossen sind. Zu jedem Funktionsblock kann eine Bedingung angegeben werden, wann (für welche Zeilen) diese Funktion ausgeführt werden soll. Diese Bedingung kann ein so genannter regulärer Ausdruck (*Regular Expression*) oder eine »normale« Bedingung (z. B. ein Vergleich) sein. Der Funktionsblock wird für jede Zeile ausgeführt, die dieser Bedingung entspricht. Der Funktion wird als Parameter `$0` die komplette Eingabezeile übergeben, so dass diese dort verarbeitet werden kann.

Wenn keine Bedingung angegeben wird, dann werden die Anweisungen in dem Funktionsblock für jede Zeile der Eingabe ausgeführt. Wenn mehrere Bedingungen zutreffen, so werden die Funktionsblöcke aller passenden Bedingungen ausgeführt.

Es gibt noch zwei spezielle Bedingungen: `BEGIN` und `END`. Der Funktionsblock, der hinter `BEGIN` angegeben ist, wird ausgeführt, bevor die erste Zeile der Eingabedateien eingelesen wird. Dort ist der Platz zum Initialisieren von Variablen oder für andere Vorbereitungen.

Die Anweisungen hinter der Bedingung END werden nach dem Lesen und Verarbeiten der letzten Zeile der Eingabedateien ausgeführt. Hier können zum Beispiel Fußtexte oder berechnete Summen ausgegeben oder andere abschließende Bearbeitungen durchgeführt werden.

9.6.1 Ein erstes Beispielprogramm in awk

Die ganze Beschreibung hört sich komplizierter an, als der Vorgang tatsächlich ist. Zum besseren Verständnis folgt in Listing 9.42 ein kleines Beispiel. Für alle Zeilen der Datei /etc/services, die den String ftp enthalten, wird der angegebene Funktionsblock ausgeführt. Dieser gibt einfach nur die komplette aktuelle Zeile aus.

Das Beispiel hat das gleiche Ergebnis, wie beim Kommando grep beschrieben. awk kann diese Zeile aber noch weiterverarbeiten. Soll z. B. nur der Servicename ausgegeben werden und nicht die komplette Zeile, so muss in dem Beispiel nur $0 gegen $1 ausgetauscht werden. Damit wird nur das erste Wort der Zeile ausgegeben.

```
(linux):~$ awk '/ftp/ { print $0; }' /etc/services
ftp             21/tcp
tftp            69/udp
sftp            115/tcp
```

Listing 9.42 Ein erstes awk*-Beispiel*

Um das awk-Programm herum können Sie sich eine Schleife vorstellen, die für jede Zeile der Eingabe durchlaufen wird. Um so lästige Dinge wie das Öffnen und Schließen von Dateien und das Erkennen des Dateiendes braucht sich ein awk-Programm in vielen Fällen nicht zu kümmern.

9.6.2 Trennen von Feldern

Der awk trennt die Eingabezeile in Wörter auf. Auf diese Wörter kann durch die Parameter *$1*, *$2*, *$3* usw. zugegriffen werden. Die Zeichen, die zur Trennung der Wörter verwendet werden, stehen in der Variablen *FS* (Field Separator). Standardmäßig werden die Wörter durch Leerzeichen getrennt, daran kann der Programmierer aber sehr einfach Änderungen vornehmen.

Entweder wird das Trennzeichen in der speziellen BEGIN-Funktion oder über den Kommandozeilenparameter -F*Zeichen* verändert. Das Listing 9.43 gibt nur das erste Wort der entsprechenden Zeile aus. Die Wörter werden jetzt durch den Schrägstrich (/) voneinander getrennt. Als Beschreibung für die Trennsymbole sind auch reguläre Ausdrücke erlaubt. Damit kann awk an viele Eingabeformate einfach angepasst werden.

```
(linux):~$ awk -F'/' '/ftp/ { print $1; }' /etc/services
ftp             21
tftp            69
sftp            115
```

Listing 9.43 Ein zweites awk*-Beispiel*

Sie können den Feldtrenner auch ändern, ohne die Kommandozeilenoption `-F` zu benutzen. In diesem Fall verwenden Sie eine `BEGIN`-Funktion, in der die Variable *FS* gesetzt wird. Das entsprechende Beispiel finden Sie in Listing 9.44.

```
(linux):~$ awk 'BEGIN { FS=/} /ftp/{ print $1 }' /etc/services
ftp             21
tftp            69
sftp            115
```

Listing 9.44 Ein drittes awk*-Beispiel*

Für größere `awk`-Programme ist die Kommandozeile zu `awk` nicht der geeignete Ort. Zum einen hat man keinen vernünftigen Editor zur Hand (womöglich Emacs mit dem `awk`-Modus), zum anderen steht das Programm später möglicherweise nicht mehr zur Verfügung. Ein `awk`-Programm kann dann entweder mit `awk -f` *`Datei`* oder, falls der `#!`-Mechanismus benutzt wird und das Skript les- und ausführbar ist, direkt ausgeführt werden. Listing 9.43 würde dann z. B. wie in Listing 9.45 geschrieben werden. Diese Variante ist bei größeren Programmen auch deutlich übersichtlicher und erlaubt die Verwendung von ausführlichen Kommentaren und die Verwaltung der Skripten mittels RCS, CVS oder anderen Versionsverwaltungen.

```
#!/usr/bin/awk -f
BEGIN {
  FS=/;
}
/ftp/ {
  print $1;
}
```

Listing 9.45 Ein awk*-Skript*

Wenn Sie Emacs als Editor für `awk`-Skripten verwenden, dann hilft Ihnen der `awk-mode`. Funktionsblöcke werden automatisch korrekt eingerückt, Kommentare werden erkannt. Kommentare werden in `awk` mit einer Raute eingeleitet und reichen bis zum Zeilenende. Dieser Modus ist ein etwas veränderter C-Modus, da sich `awk` und C sehr ähnlich sind. Wenn Sie das Font-Lock-Paket verwenden, dann werden Kommentare, Zeichenketten und Schlüsselwörter auch farblich hervorgehoben.

9.6.3 Interne Variablen

awk kennt neben *FS* und *$n* noch weitere interne Variablen mit besonderer Bedeutung. Auf diese Variablen kann der Programmierer lesend zugreifen. Schreiben ist nur für einige Variablen erlaubt. In Tabelle 9.7 finden Sie die wichtigsten Variablen mit ihrer Bedeutung aufgelistet. Das Skript in Listing 9.46 gibt die Anzahl der Zeilen der Datei /etc/services aus und verwendet dazu die interne Variable NR.

Variable	Bez. (engl.)	Bez. (deut.)	Bemerkung
FS	Field-Separator	Feldtrennzeichen	Default: Leerzeichen
NF	Number of Fields	Anzahl der eingelesenen Felder	nur zur Ausgabe
RS	Record-Separator	Satztrennzeichen	Standardwert ist »\n« (Zeilenvorschub)
NR	Number of Records	Anzahl der bisher eingelesenen Datensätze	nur zur Ausgabe

Tabelle 9.7 Interne awk-Variablen

```
(linux):~$ awk 'END { print NR; }' /etc/services
184
```

Listing 9.46 Verwendung der internen Variable NR

Sie können mit awk viele andere Unix-Befehle mit wenig Aufwand simulieren. Das kann dann sehr hilfreich sein, wenn das System beschädigt wurde und nur noch wenige Dinge benutzbar sind. Die Info-Datei zu gawk enthält einige Beispiele dafür.

9.6.4 Ausgabebefehle

Die Ausgabe von Text kann mit den awk-Kommandos print oder printf erfolgen. Der Befehl printf kann Variablen formatiert ausgeben. Die Syntax der Formatierungsanweisungen entspricht der der gleichnamigen C-Funktion. Ein Beispiel für die Formatierung von Zahlen und Zeichenketten finden Sie im Listing 9.50. Eine vollständige Übersicht über die möglichen Formatieroptionen finden Sie in den Info-Seiten bzw. der Manpage zu gawk.

9.6.5 Arithmetische Operationen

Die Programmiersprache `awk` kennt die meisten in C bekannten arithmetischen Operationen, wie `+`, `-`, `*`, `++` und `+=`. Damit ist es möglich, auch kompliziertere Berechnungen durchzuführen. `awk` kennt keine typisierten Variablen, es kann also mit jeder Variablen gerechnet werden, sofern sie eine Zahl enthält.

Bei Bedingungen zu Funktionsblöcken können auch mehrere Bedingungen verknüpft werden, wie es auch in C durchgeführt wird (`||` für ODER, `&&` für UND sowie `!` für NOT). Zusammen mit den verfügbaren Kontrollstrukturen (`if`, `while` `do` und `for`) ist `awk` eine vollständige Programmiersprache.

`awk` hat immer dann Vorteile, wenn Texte vergleichsweise einfach bearbeitet werden sollen und die Entwicklung schnell abgeschlossen sein muss. Man erkauft sich diese Vorteile durch eine relativ (verglichen mit Compilersprachen wie C) langsame Ausführungsgeschwindigkeit. In vielen Fällen und insbesondere bei »Einmalaktionen« ist die Geschwindigkeit von `awk` vollkommen ausreichend.

9.6.6 Interne Funktionen

`awk` hat auch eingebaute Funktionen, sowohl für arithmetische Operationen, wie `exp`, `sqrt` usw., als auch für Operationen mit Zeichenketten, wie z. B. `substr` oder `index`. Eine ausführliche Beschreibung aller internen Funktionen und Beispiele dazu finden Sie in der Info-Dokumentation zu GNU-`awk` oder in der Manpage `gawk(1)`.

Schauen Sie einfach mal in die Info-Seiten von `gawk`, um sich einen Überblick über die internen Funktionen zu verschaffen. Sie können sie sich natürlich nicht alle merken, aber Sie sollten ein Gefühl dafür bekommen, welche Funktionen es gibt und wie Sie diese einsetzen können.

9.6.7 Assoziative Arrays

`awk` kennt so genannte *assoziative Arrays*. Im Gegensatz zu normalen Variablen, die eine Zahl oder Zeichenkette enthalten, können in Arrays (oder Feldern) (praktisch) beliebig viele Werte gespeichert werden. Der Zugriff auf einen einzelnen Wert erfolgt dabei über einen Index. In den assoziativen Arrays von `awk` wird kein numerischer Index verwendet, sondern ein beliebiger Wert (egal ob Zahl oder Zeichenkette). Damit lassen sich auch komplizierte Suchvorgänge einfach programmieren. Das Listing 9.47 enthält ein Beispiel für die Verwendung assoziativer Arrays. Bekommen Sie keinen Schreck, hier werden tatsächlich viele Funktionen verwendet, die Sie bisher noch nicht kennen gelernt haben.

```
#!/usr/bin/awk -f
# Prüfen des Plattenplatzes auf ausreichenden Freiplatz
# Aufruf:  df -k -P | $0

BEGIN {
  FS = "[ \t]*";          # Leerzeichen oder Tabulatoren
  while ( (getline < "/etc/freespace.conf") > 0 ) {
    if ( ($1 != "") && ($2 != "")) {
      min_free[$1] = $2; # kein Kommentar
    }
  }
}

{
  fs=$1; size=$2; used=$3; free=$4; pct_used=$5; mount=$6;
  if ( min_free[mount] != "" ) {
    if ( free <= min_free[mount] )
      printf("mount, free, min_free[mount]);
  }
}
```

Listing 9.47 Assoziative Arrays in awk

Im BEGIN-Teil wird die Konfigurationsdatei /etc/freespace.conf gelesen und die Daten werden in das interne Array min_free eingelesen. Als Index wird der Name eines Dateisystems verwendet – es können in awk also auch Zeichenketten als Indizes verwendet werden. Das Einlesen geschieht mit dem Befehl getline, das Ende der Datei wird mit Hilfe einer while-Schleife erkannt.

Im awk-Skript wird die Ausgabe des Befehls df -k -P als Standardeingabe erwartet. Für jedes Dateisystem wird mit Hilfe des assoziativen Arrays geprüft, ob ein minimaler Freiplatz angegeben wurde, und dieser entsprechend geprüft.

In einem Programm kann stets über alle Elemente des Arrays eine Schleife ausgeführt werden, egal ob ein numerischer Index verwendet wird oder nicht. Zur Fehlersuche verwendete ich zunächst den in Listing 9.48 angegebenen END-Teil.

```
END {
  for ( i in min_free )
    printf("FS }
```

Listing 9.48 Eine Schleife über ein assoziatives Array

9.6.8 Eine sinnvolle Anwendung von awk

Zum Abschluss folgt nun eine sinnvolle Anwendung von `awk`, die deutlich macht, für welchen Zweck `awk` am besten geeignet ist. Ich habe mit dem SQL-Befehl »unload« Daten aus einer Datenbank exportiert. Als Feldtrennzeichen habe ich die Raute (#) verwendet. Die Datendatei hat den in Listing 9.49 dargestellten Inhalt.

```
134076#Skywalker#Luke#11256.01
334521#Solo#Han#12.56
422110#Vader#Darth#1198.00
```

Listing 9.49 Testdaten für das letzte Beispiel

Ein `awk`-Skript (siehe Listing 9.50) soll zur formatierten Ausgabe der Daten verwendet werden. Außerdem soll der Umsatz der Kunden summiert und ausgegeben werden. Die Ausgabe des Skripts finden Sie im Listing 9.51.

```
#!/usr/bin/awk -f
BEGIN {
  FS="";
  sum=0;
  print( "Kd.Nr.  Name             Vorname           Umsatz");
  print( "------  ---------------  ---------------- --------");
}
{
  printf( "%d  %-15s  %-15s  %8.2f\n", $1, $2, $3, $4);
  sum+=$4;
}
END {
  print( "------  ---------------  ---------------- --------");
  printf("                         Gesamtsumme:    %9.2f\n",sum);
  print( "                                         =========");
}
```

Listing 9.50 Ein letztes `awk`-Beispiel

```
Kd.Nr.  Name             Vorname           Umsatz
------  ---------------  ----------------  --------
134076  Skywalker        Luke             11256.01
334521  Solo             Han                 12.56
422110  Vader            Darth             1198.00
------  ---------------  ----------------  --------
                         Gesamtsumme:     12466.57
                                          =========
```

Listing 9.51 Ausgabe des Programms

Haben Sie Lust auf awk bekommen? Dann testen Sie es einfach einmal. Beachten Sie aber, dass GNU-awk gegenüber dem Standard-awk einige (nützliche) Erweiterungen enthält. Neben GNU-awk gibt es noch eine Reihe von anderen awk-Versionen. So finden Sie auf vielen kommerziellen Unix-Systemen oawk (das historische Programm) und nawk (die entsprechende neue Version). Nützliche Bücher sind [Robinson 2001] und [Dougherty 1997].

9.7 Textdateien bearbeiten mit sed

sed steht für »Stream-Editor«. Der sed ist ein waschechter Editor, aber anders als »normale« Editoren, wie z. B. der Emacs, arbeitet sed nicht interaktiv, sondern kommandoorientiert. Das heißt, sed werden in einer Befehlsdatei oder der Kommandozeile Editierbefehle vorgegeben, die dann für die Eingabedatei oder die Standardeingabe ausgeführt werden.

Das Listing 9.52 zeigt ein kleines Beispiel für die Anwendung von sed. Der Befehl ps zeigt die gerade laufenden Programme des Benutzers an. In der ersten Zeile steht normalerweise eine Überschriftszeile, die bei der automatischen Weiterverarbeitung stört[2]. Diese erste Zeile soll nicht ausgegeben werden. Also wird die Ausgabe des Befehls ps durch eine Pipe an den sed weitergeleitet. Der sed-Befehl 1d bedeutet, dass die erste Zeile gelöscht werden soll.

```
(linux):~$ ps | sed '1d'
   95 p 1 SW    0:04 (bash)
11109 p 6 S     0:06 -bash
11110 p 5 SW    0:05 (bash)
11111 p b SW    0:04 (bash)
11811 p 5 S     2:02 emacs
12716 p 6 R     0:00 ps
12717 p 6 R     0:00 sed 1d
```

Listing 9.52 Ein einfaches Beispiel für sed

Alle sed-Befehle bestehen aus nur einem Buchstaben. Die Befehle werden durch ein Zeilenende in der Kommandodatei oder durch ein Semikolon getrennt. Jedem Befehl kann eine Adresse oder ein Adressbereich zugeordnet werden. Die Adresse kann durch Angabe der Zeilennummern, im obigen Beispiel die 1, festgelegt werden oder es kann ein Suchmuster (regulärer Ausdruck) angegeben werden. Der Befehl wird dann für jede Zeile ausgeführt, die diesem Suchmuster entspricht. Nach der Angabe der Adresse folgt direkt der Befehl. Im obigen Beispiel ist es der Befehl d (für »delete«). Manche Befehle erwarten Parameter, die direkt nach dem Befehl angegeben werden.

2. ps kennt eine Option -h zur Unterdrückung dieser Zeile, das soll uns in diesem Beispiel aber nicht stören.

Der sed arbeitet die Eingabe zeilenweise ab. Soll ein Befehl für eine bestimmte Zeilennummer ausgeführt werden, so wird diese dem Befehl vorangestellt. Es kann aber auch ein Bereich von Zeilen angegeben werden. Der Anfang und das Ende des Bereichs werden durch ein Komma getrennt. Beispiele für die Verwendung von Zeilennummern finden Sie im Listing 9.53. Die Ausgaben der Befehle sind hier aus Platzgründen nicht abgedruckt – probieren Sie die Befehle einfach einmal aus.

```
(linux):~> head /etc/services | sed -e '1d'
(linux):~> head /etc/services | sed -e '1,5d'
(linux):~> head /etc/services | sed -e '$d'
```

Listing 9.53 Beispiele für die Verwendung von Zeilennummern

Das erste Beispiel löscht, wie bereits dargestellt, die erste Zeile aus der Eingabe. Der zweite Befehl löscht die Zeilen 1 bis 5 aus der Eingabe, das nächste Beispiel löscht die letzte Zeile. Das Dollarzeichen ist der Platzhalter für die letzte Zeile.

Ist die Zeilennummer unbekannt, für die ein sed-Befehl ausgeführt werden soll, so kann ein regulärer Ausdruck als Suchmuster verwendet werden. Hier kann auch ein Adressbereich angegeben werden, indem zwei reguläre Ausdrücke durch ein Komma getrennt werden. Es ist natürlich möglich, den Beginn mit einer Zeilennummer und das Ende mit einem regulären Ausdruck oder umgekehrt zu beschreiben. Das Listing 9.54 enthält einige Beispiele für die Verwendung von Suchmustern in sed, die Ergebnisse sind ebenfalls nicht abgedruckt.

```
(linux):~> head /etc/services | sed -e '/^#/d'
(linux):~> head /etc/services | sed -e '/tcp/d'
(linux):~> head /etc/services | sed -e '/[A-Z]/d'
(linux):~> head /etc/services | sed -e '/tcpmux/,/ftp/d'
```

Listing 9.54 Suchmuster in sed

Der erste Aufruf in Listing 9.54 löscht alle Zeilen, die eine Raute (#) in der ersten Spalte enthalten, aus der Eingabedatei. In vielen Unix-Dateien werden auf diese Weise Kommentare markiert, die so sehr einfach entfernt werden können.

Das zweite Suchmuster in Listing 9.54 löscht alle Zeilen, die das Wort »tcp« enthalten, und das dritte Beispiel löscht alle Zeilen, die mindestens einen Großbuchstaben enthalten. Das letzte Beispiel löscht einen Block von Zeilen aus der Eingabedatei, wobei die erste Zeile die Zeichenkette »tcpmux« und die letzte »ftp« enthält.

Neben dem Kommando d für »Zeile löschen« gibt es noch weitere Kommandos, die nun im Folgenden einzeln erläutert werden. Allen diesen Kommandos muss, wie eben beschrieben, eine Adresse angegeben werden, für die diese Kommandos ausgeführt werden. Es werden nur die wichtigsten Kommandos beschrieben, eine vollständige Übersicht finden Sie in der Manpage.

Weiterführende Informationen sowie eine praxisorientierte Einführung in awk und sed finden Sie in [Dougherty1997] (englisch). Beide Bücher sind Standardwerke zu diesem Thema und lesenswert.

`a\`
`text` (für »append«)
Dieser Befehl hängt an die angegebene Adresse den Text *text* an.

`d` (für »delete«)
Es wird die komplette Zeile zur angegebenen Adresse gelöscht.

`i\`
`text` (für »insert«)
Der Text *text* wird vor der angegebenen Adresse ausgegeben.

`q` (für »quit«)
Bei Erreichen der angegebenen Adresse wird der sed beendet.

`r` *`Dateiname`* (für »read file«)
Der Inhalt der Datei *Dateiname* wird hinter der angegebenen Adresse angehängt.

`s/`*`regex`*`/`*`replacement`*`/`*`flags`* (für »search and replace«)
Dieser Befehl entspricht der »Suchen-und-Ersetzen«-Funktion der meisten Texteditoren. Es wird das Textstück, das dem Suchmuster *regex* entspricht, durch die in *replacement* angegebene Zeichenkette ersetzt. Dieser Befehl ist der wohl am häufigsten verwendete sed-Befehl. In Kombination mit dem find-Befehl kann man in vielen Dateien die gleichen Änderungen vornehmen.

Ein Beispiel: Man will den Host-Namen ändern, aber man weiß nicht, in welchen Dateien dieser versteckt ist. Also lässt man den sed in allen von find gefundenen Dateien den alten Host-Namen (im Beispiel jupiter) gegen den neuen Host-Namen austauschen (typhon):

```
(linux):~$ find / -type f -exec sed 's/jupiter/typhon/g' {} \;
```

Mit dem Parameter *flags* kann man angeben, ob dieser Austausch immer durchgeführt werden soll (Flag »g«) oder ob der Austausch maximal null- bis neunmal ausgeführt werden soll. Durch das Flag 1 wird z. B. nur das erste Vorkommen von *regex* durch *replacement* ersetzt.

`y/`*`zeichen1`*`/`*`zeichen2`*`/` (für »yank«)
Alle Zeichen, die in *zeichen1* vorkommen, werden durch die Zeichen in *zeichen2* ersetzt. Damit lassen sich Zeichen transformieren, genau wie beim Kommando tr (siehe Abschnitt »Transformieren von Zeichen mit tr« in Kapitel 9.3.7). Denkbar wäre z. B. ein Umlautefilter für DOS-Zeichensätze nach ISO-Latin-1 oder umgekehrt.

sed wird häufig verwendet, um automatisch Änderungen an Konfigurationsdateien (z. B. bei der Installation von Software) durchzuführen. Die configure-Skripten des GNU-Projekts erzeugen mit Hilfe von sed die systemabhängigen Makefiles aus allgemeinen Eingabedateien.

Ein weiteres Beispiel für die Verwendung von sed ist das Programm dtree. Es wird eine grafische Darstellung der Dateisystemhierarchie angezeigt. Das Programm, das aus dem Linux-Journal vom September 1996 stammt, ist im Listing 9.55 abgedruckt.

```
#!/bin/sh
( cd ${1-.} ; pwd )
find ${1-.} -type d -print \
 | sort -f \
 | sed -e "s,^${1-.},," \
       -e "/^$/d" \
       -e "s,[^/]*/\([^/]*\)$,\`-----\1," \
       -e "s,[^/]*/,|       ,g"
```

Listing 9.55 Das Programm dtree *als Anwendung von* sed

Der Programmablauf sieht zunächst sehr verwirrend aus, ist aber bei genauerer Betrachtung relativ einfach. Im ersten Schritt wird der Name des obersten Verzeichnisses in der Liste ausgegeben. Der zweite Befehl durchsucht die Festplatte nach Verzeichnissen, die unterhalb des als Parameter angegebenen Verzeichnisses liegen. Diese Liste wird mit sort sortiert und dann durch eine Reihe von sed-Befehlen aufbereitet.

Der erste sed-Befehl löscht aus allen Zeilen den (gemeinsamen) Startpfad. Dieser ist in der gesamten Liste konstant und durch die bereits erfolgte Ausgabe nicht weiter interessant. Der zweite sed-Befehl (/^$/d) löscht alle Leerzeilen. Die nächsten beiden Befehle sorgen für die grafische Aufbereitung der Darstellung: Zunächst wird für jedes neue Unterverzeichnis ein waagrechter Strich erzeugt, anschließend ein senkrechter Strich ergänzt.

Dieses Programm ist nicht perfekt, da die grafische Darstellung nicht immer korrekt ist. In jedem Fall gewinnt man jedoch einen guten Überblick über die Struktur des Verzeichnisses – und das Programm zeigt, wie man mit einer Kombination von einfachen Befehlen eine bestimmte Aufgabe lösen kann.

Ein Beispiel für ein komplexes sed-Skript finden Sie in den Quellen von GNU-sed: eine Version des Taschenrechners dc, vollständig mit sed-Kommandos implementiert. Dieses Skript ist nicht nur ein Kunstwerk, sondern wird auch im Rahmen der Regressionstests verwendet.

9.8 Weitere nützliche Utilities

Der Werkzeugkasten Unix ist noch lange nicht erschöpft. Viele weitere Programme können Unix-Benutzern und Administratoren das Leben weiter erleichtern. Dazu gehören weitere Programmier- und Skriptsprachen, wie z. B. `perl` (Practical Extraction and Report Language), `python` oder `tcl` (Tool Command Language), und Utilities. Eine komplette Aufzählung oder gar Referenz würde den Rahmen dieses Buchs sprengen, so dass wir hier nur noch wenige Programme kurz vorstellen.

Eine ausführliche Beschreibung der Optionen zu den hier vorgestellten Programmen finden Sie in den entsprechenden Manpages. Hier geht es darum, einen Überblick über die zur Verfügung stehenden Programme zu bekommen, um das richtige Werkzeug für eine Aufgabe auswählen zu können. Anschließend schlägt man die entsprechenden Optionen nach und probiert diese gegebenenfalls mehrfach (z. B. in einer Pipe) aus, bis das gewünschte Ergebnis erreicht ist.

Die wichtigsten Utilities finden Sie in den Paketen `fileutils`, `sh-utils` und `textutils` aus dem GNU-Projekt. Wenn Sie wissen, dass es ein Programm gibt, das für Ihre Aufgabe geeignet sein könnte, können Sie die passenden Optionen aus der Dokumentation entnehmen. Aus diesem Grunde sollten Sie zumindest eine grobe Beschreibung der Programme aus diesen Paketen gesehen haben, z. B. die aus der `whatis`-Datenbank.

9.8.1 Dateien sortieren

Eine der am häufigsten vorkommenden Aufgaben in der EDV ist das Sortieren von Daten. Zum einen sind sortierte Daten oft übersichtlicher, zum anderen ist es manchmal für die weitere Verarbeitung unbedingt erforderlich, dass die Daten sortiert sind. Ausgereifte Sortieralgorithmen sind bereits im Programm `sort` implementiert.

Das Programm `sort` kann Dateien sortieren, zusammenmischen und auf Sortiertheit prüfen. Dabei kann nach bestimmten Feldern oder Spalten sortiert werden, die auch numerische Werte enthalten können. Außerdem kann die Sortierreihenfolge umgekehrt werden. Damit lassen sich praktisch alle Sortieraufgaben auch in Shell-Skripten erledigen.

Normalerweise sortiert das Programm `sort` nach Feldern, die durch einen leeren String zwischen einem Nicht-Space-Zeichen und einem White-Space getrennt sind. Das Trennzeichen kann mit der Option `-t` geändert werden. Die Sortierfelder werden mit der Option `-k` bestimmt. Dabei kann das Feld und darin ein Offset in der Form `-k` *`Feld`*`[.`*`Offset`*`]` angegeben werden. Natürlich können Sie auch rückwärts (Option `-r`) oder nach Zahlen (Option `-n`) sortieren.

`sort` hat zwei weitere Modi: `-c`, um eine Datei auf Einhaltung der Sortierreihenfolge zu prüfen, und `-m`, um mehrere Dateien zusammenzumischen. Wenn Sie große Dateien bearbeiten wollen, dann ist es möglich, dass der Platz im `/tmp`-Verzeichnis nicht ausreicht. Mit der Umgebungsvariablen `TMPDIR` oder der Option `-T` können Sie ein anderes Verzeichnis angeben. Die Umgebungsvariable `TMPDIR` wird auch von anderen Programmen bei der Erzeugung von temporären Dateien verwendet.

9.8.2 Mehrfachzeilen finden oder löschen mit uniq

Bei einigen Aufgaben ist es nicht sinnvoll, wenn identische Zeilen (gemäß einem bestimmten Feld) mehrfach auftauchen. Doppelte Zeilen können mit dem Befehl `uniq` gelöscht werden. Manchmal sind nur die Zeilen interessant, die doppelt (Option `-d` bzw. `--repeated`) oder einfach (Option `-u` bzw. `--unique`) in der Datei gespeichert sind. Die Anzahl der Wiederholungen wird ausgegeben, wenn die Option `-c` bzw. `--count` angegeben wird.

Über verschiedene andere Optionen (`--skip-fields` und `--skip-chars` sowie `--check-chars`) kann der Teil der Zeile, der für den Vergleich herangezogen wird, genauer bestimmt werden. Standardmäßig wird die gesamte Zeile verwendet.

Dieser Befehl wird oft im Zusammenhang mit `sort` verwendet. Wenn mehrere Dateien zusammengemischt werden, ist es vielfach störend, wenn Zeilen doppelt auftauchen. In diesen Fällen sortiert man die Daten und bereinigt die Ausgabe danach mit Hilfe des Programms `uniq`.

9.8.3 Sortierte Dateien vergleichen

Das Programm `diff` findet Unterschiede in beliebigen Textdateien und generiert eine Ausgabe, die maschinell weiterverarbeitet werden kann. Beim Vergleich sortierter Dateien kommt es häufig nicht auf die maschinelle Änderbarkeit an, sondern mehr auf eine übersichtliche Darstellung. Hier kommt das Programm `comm` ins Spiel. Die Darstellung umfasst drei Spalten: In der ersten Spalte findet man die Zeilen, die nur in der ersten Datei enthalten sind, in der zweite Spalte analog die Zeilen für die zweite Datei. In der dritten Spalte werden alle Zeilen ausgegeben, die in beiden Dateien enthalten sind.

Mit den Optionen `-1`, `-2` bzw. `-3` kann die Anzeige der ersten, zweiten oder dritten Spalte ausgeblendet werden. Beachten Sie, dass die Ausgabe nur bei sortierten Dateien sinnvoll ist, da `comm` sonst aus dem Tritt kommt.

9.8.4 Umgebungsvariablen verändern

In Skripten oder bei regelmäßig (via `cron`) zu startenden Programmen müssen oft spezielle Umgebungsvariablen vorgegeben werden. Dazu kann das Programm `env` verwendet werden. Damit können problemlos Umgebungsvariablen temporär für einen Prozess und dessen Kindprozesse verändert werden.

In Listing 9.56 finden Sie ein einfaches Beispiel. Hier wird das Programm `printenv` verwendet, das einzelne oder alle Umgebungsvariablen ausgibt. Mit diesem Programm können Sie sich schnell einen Überblick über die gesetzten Umgebungsvariablen verschaffen – nützlich wenn Sie in einem Cronjob die Umgebungsvariablen überprüfen wollen.

```
(linux):~$ printenv LANG
de_DE
(linux):~$ env LANG=C printenv LANG
C
```

Listing 9.56 Ein Beispiel für die Verwendung von `env`

Ein weiteres Einsatzgebiet für den Befehl `env` ist das dynamische Linken mit einer speziellen Bibliothek. So kann man für einzelne Programme, die z. B. eine ältere `libc`-Version benötigen als das übrige System, die Variable `LD_LIBRARY_PATH` setzen. Der dynamische Linker verwendet diese Variable, um die benötigten Bibliotheken zu laden, sofern das Programm nicht `setuid` oder `setgid` ist. Wäre das Programm mit speziellen Rechten versehen und könnte jeder Benutzer einen anderen Suchpfad nach Shared-Libraries angeben, so könnte beliebiger Programmcode mit erweiterten Rechten ausgeführt werden.

Ein weiteres Einsatzgebiet ist das temporäre Ausschalten des National Language Support, indem die Variable `LC_ALL` auf den Wert »C« gesetzt wird. Dies ist immer dann sinnvoll oder notwendig, wenn einzelne Programme NLS nicht oder fehlerhaft verwenden, der Rest des Systems jedoch entsprechend eingerichtet ist. Im Laufe der Zeit sollte es aber immer seltener notwendig sein, derartige Maßnahmen zu ergreifen.

Wenn Sie eine Bourne-Shell-kompatible Shell wie die `bash`, `ksh` oder `zsh` verwenden, dann können Sie auch die in Listing 9.57 angegebene Notation verwenden. Diese funktioniert allerdings nicht in der `csh` oder `tcsh`.

```
(linux):~$ LANG=en_US printenv LANG
en_US
```

Listing 9.57 Temporäre Umgebungsvariablen in der `bash`

9.8.5 Den Anfang von Dateien anzeigen

Oft reicht es, nur den Anfang einer Datei anzuzeigen, um den Inhalt oder die Aktualität der Datei festzustellen. Der Start von Programmen wie `more` oder `less` ist in diesem Fall vollkommen unnötig. Das kleine Programm `head` erfüllt genau diesen Zweck und zeigt standardmäßig nur die ersten zehn Zeilen der Datei an. Mit der Option `-n` *`Zeilen`* (`--lines`) oder kurz `-`*`Zeilen`* kann eine beliebige Anzahl Zeilen angezeigt werden. Mit der Option `-c` *`Bytes`* (oder `–bytes`) kann eine bestimmte Anzahl Bytes angezeigt werden.

9.8.6 Das Ende einer Datei anzeigen

Das Gegenstück zu `head` ist `tail`, das das Ende einer Datei anzeigt. BSD-Versionen von `tail` sind bei relativ kleinen Dateien in der Lage, diese in umgekehrter Reihenfolge der Zeilen auszugeben. Das GNU-Projekt verwendet zu diesem Zweck das Programm `tac` (`cat` rückwärts ...).

Mit der Option `-n` *`Zeilen`* (`--lines`) werden analog zu `head` die letzten Zeilen angezeigt. Wird die Option `+n` *`Zeilen`* angegeben, so wird ab dieser Zeile gedruckt. Damit lassen sich Dateien auch mit `tail` in mehrere Teile zerlegen. In der Regel wird man dazu aber das Programm `split` verwenden.

Oft möchte man eine Log-Datei beobachten, die immer länger wird. Programme wie `more` lesen die Datei komplett ein, so dass neu hinzukommende Meldungen nicht angezeigt werden. Das Programm `tail` zeigt mit der Option `-f` (`--follow`) immer das aktuelle Ende der Datei an. Das Programm `less` bietet bei der Dateianzeige diese Funktion mit der Tastenkombination Shift+F.

9.8.7 Zeichenketten in Binärdateien ausgeben

Mit dem Programm `strings` lassen sich Zeichenketten aus Binärdateien herausfiltern. Das kann z. B. nützlich sein, um festzustellen, auf welche Dateien das Programm zugreifen will, oder um »versteckte« Funktionen zu entdecken. Das GNU-`strings` (aus den `binutils`) liest standardmäßig nicht alle Teile eines Programms, so dass der Schalter `-a` angegeben werden sollte.

9.8.8 Systemaufrufe von Programmen verfolgen

Mit dem Linux-Tool `strace`[3] lassen sich kompilierte Programme »debuggen«. Es werden alle Systemaufrufe mitprotokolliert. Damit kann man z. B. feststellen, auf welche Dateien das Programm zugreift. Die Ausgaben von `strace` können

3. Unter Solaris heißt ein ähnliches Programm `truss`.

mit dem Kommandozeilenparameter `-o Dateiname` in die angegebene Datei geschrieben werden. Außerdem ist es möglich, die zu protokollierenden Systemaufrufe mit `-esyscall` einzuschränken.

Diese Funktion ist immer dann nützlich, wenn ein Programm eine unvollständige oder unverständliche Fehlermeldung ausgibt, wie etwa »file not found« ohne Angabe des entsprechenden Dateinamens. Wenn man nicht weiß, welche Dateien von einem Programm als Konfigurationsdateien verwendet werden, dann kann ein Aufruf von `strace` darüber Aufschluss geben. Diese Funktion kann allerdings nur verwendet werden, wenn das zu verfolgende Programm nicht `setuid` oder `setid` ist.

9.8.9 Funktionsaufrufe in Programmen verfolgen

Die Verfolgung von Systemaufrufen, wie sie durch `strace` möglich ist, hilft nicht in jedem Fall weiter, da nicht jeder Funktionsaufruf innerhalb eines Programms in einen Systemaufruf umgesetzt wird. Analog zu `strace` verfolgt `ltrace` den Aufruf von Bibliotheksfunktionen anstelle von Systemaufrufen und zeigt möglicherweise ebenfalls nützliche Informationen an.

9.8.10 Dateien in verschiedenen Formaten ausgeben

Manchmal ist es notwendig, eine Datei genauer zu untersuchen. In vielen Fällen kommt man mit Programmen wie `strings` relativ weit, aber manchmal benötigt man wirklich den genauen Überblick über die in der Datei enthaltenen Zeichen. Für derartige Fälle kann das Programm `od` verwendet werden. Als Default gibt `od` die Daten in oktaler Darstellung aus, man kann jedoch eine andere Darstellung wählen. Das Listing 9.58 zeigt einige einfache Beispiele für die Formatierung der Ausgabe mit `od`.

```
(linux):~> echo 'Hallo Welt' | od
0000000 060510 066154 020157 062527 072154 000012
0000013
(linux):~> echo 'Hallo Welt' | od -x
0000000 6148 6c6c 206f 6557 746c 000a
0000013
(linux):~> echo 'Hallo Welt' | od -a
0000000   H   a   l   l   o  sp   W   e   l   t  nl
0000013
```

Listing 9.58 Das Programm `od`

Wenn Sie Binärdateien editieren müssen, so bietet `emacs` mit dem `hexl-mode` einen entsprechenden Modus an. Sie können einen Buffer mit `M-x hexl-mode` in eine Hex-Darstellung umschalten. Ein Beispiel für diese Darstellung finden Sie

in Abbildung 9.2. Die erste Spalte enthält den Offset in dem Buffer, der mittlere Teil die Darstellung der Zeichen als Hexadezimalcode und der rechte Teil die Anzeige als Text.

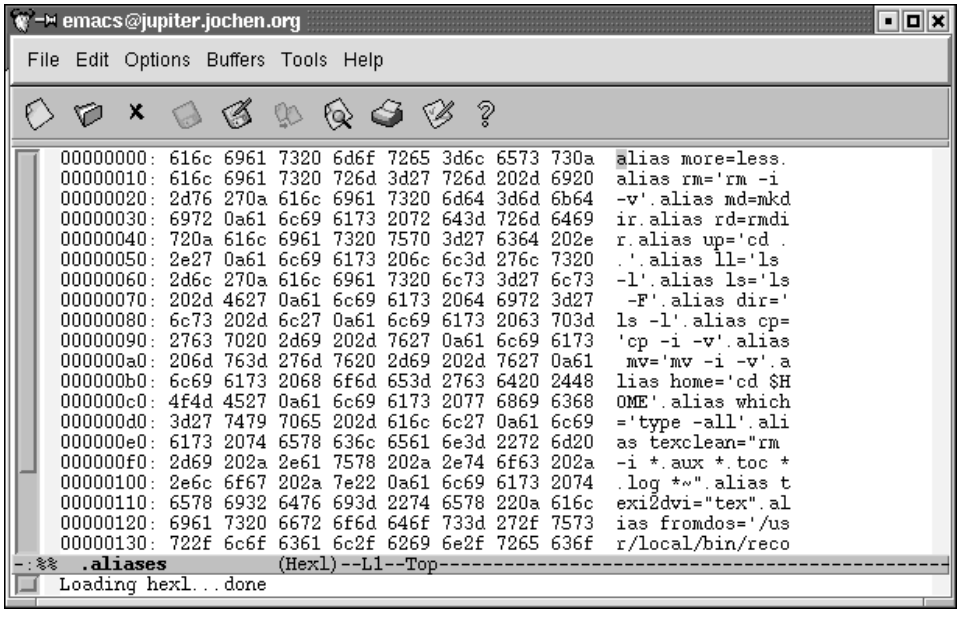


Abbildung 9.2 Der hexl-mode in Aktion

Einfache ASCII-Zeichen (ohne Umlaute) können einfach durch Drücken der entsprechenden Tasten eingefügt werden, so dass dieses Zeichen das vorher an dieser Stelle vorhandene Zeichen überschreibt. Ist eine Taste nicht mit `self-insert` belegt, kann dieses Zeichen mittels Strg+q eingefügt werden. Wenn Sie nach Strg+q Ziffern eingeben, so werden diese als oktale Darstellung eines Zeichens interpretiert.

Mit Meta+Strg+x, Meta+Strg+o bzw. Meta+Strg+d können Sie ein Zeichen in Hexadezimal-, Oktal- bzw. Dezimaldarstellung einfügen. Den `hexl-mode` verlassen Sie mit Strg+c Strg+c (`hexl-mode-exit`).

Neben Emacs gibt es eine Reihe von Programmen, wie z. B. `bpe` (Binary Patch Editor), mit denen Binärprogramme editiert werden können. Beachten Sie aber, dass das Verändern von binären Dateien in jedem Fall nicht ungefährlich ist. Sie sollten daher vor den Änderungen eine Sicherheitskopie der betreffenden Datei erstellen.

9.9 Die Shell als Bindeglied zwischen den verschiedenen Programmen

Shells sind nicht nur eine Benutzerschnittstelle, sondern auch relativ leistungsfähige Programmiersprachen. Jeder Unix-Benutzer hat eine Login-Shell, die bei der Anmeldung automatisch gestartet wird. Im Prinzip ist dies ein ganz normales Programm. Oft wird hier z. B. automatisch ein Branchenpaket gestartet. Für die interaktive Benutzung eines Unix-Systems wird hier eine Shell gestartet. Verbreitet sind die Bourne- oder Korn-Shell sowie die C-Shell und deren jeweilige Ableger.

Die Shell übernimmt zunächst die Benutzerinteraktion und startet die gewünschten Programme. Außerdem können alle Shells auch als Programmiersprache verwendet werden – zum einen um die Initialisierungsdateien zu verfassen, aber auch für weitere Shell-Skripten.

Kontrollstrukturen wie Verzweigungen und Schleifen sind in der Shell intern realisiert. Für fast alle anderen Funktionen werden die üblichen Unix-Programme verwendet. Aus Performance-Gründen sind jedoch viele Funktionen wie `echo` oder `pwd` intern realisiert. Da der POSIX-Standard diese Funktionen beschreibt, sind in der Regel interne und externe Kommandos kompatibel.

Der folgende Abschnitt wird einige Funktionen zur Programmierung mit der `bash` vorstellen, die aber auch für die interaktive Bedienung oft sehr hilfreich sind. Beachten Sie, dass andere Unix-Systeme häufig nur über eine Original-`sh` oder `ksh` verfügen, so dass einzelne Erweiterungen der `bash` nicht überall verwendet werden können.

Praktisch alle modernen Unix-Systeme unterstützen den `#!`-Mechanismus zum Starten von Shell-Skripten. Dazu muss nach dieser Zeichenfolge der Pfad zum Interpreter komplett angegeben werden. Dadurch ist bei Skripten, die unverändert auf verschiedenen Systemen laufen sollen, die Wahl des Interpreters beschränkt. Oft kommen nur die `/bin/sh` oder die `/bin/csh` in Frage, wobei man die `csh` als Skript-Sprache meiden sollte. Gründe für diese Einstellung finden Sie in dem Text *Why csh is considered harmful* von Tom Christiansen[4]. Dieser Text wird regelmäßig, z. B. in der Newsgruppe *news:comp.unix.shell*, gepostet und findet sich im FAQ-Archiv auf `rtfm.mit.edu`.

Auf praktisch allen modernen Unix-Systemen ist außerdem die Korn-Shell `ksh` verfügbar, unter Linux wird die Reimplementierung `pdksh` verwendet. Die Skript-Sprache ist zu derjenigen der `sh` kompatibel, unter Linux sind jedoch die `bash` und die `tcsh`, eine erweiterte und verbesserte Version der C-Shell, wesentlich weiter verbreitet.

4. *mailto:tchrist@mox.perl.com*

Für die interaktive Benutzung sind interaktive Funktionen wie Aliasnamen, History und ein guter Kommandozeileneditor wichtig. In Shell-Skripten spielen diese Funktionen praktisch keine Rolle. Mehr Informationen zur Auswahl einer Login-Shell finden Sie im Abschnitt 4.9.1, »Auswahl einer interaktiven Shell«. In den folgenden Abschnitten geht es um die Verwendung der `bash` als Skript-Sprache.

9.9.1 Ein- und Ausgabeumleitung und Pipes

Auf der Kommandozeile und in Skripten werden die Funktionen der Ein- und Ausgabeumleitung oft verwendet. Dabei wird der Umstand ausgenutzt, dass fast alle Unix-Programme aus der Standardeingabe lesen und auf die Standardausgabe ausgeben. Dabei ist es wichtig, dass die Programme sich an eine wesentliche Regel halten: Sie dürfen keine Ausgaben zusätzlich erzeugen (z. B. Überschriftszeilen oder Hinweise). Die Shell stellt verschiedene Methoden bereit, hier einzugreifen.

Die (Standard-)Ausgabe eines Programms kann unmittelbar als Eingabe für das nächste Programm verwendet werden. Dazu dient der senkrechte Strich (`|`, auch Pipe-Symbol genannt), der zur Trennung der Befehle verwendet wird. Im Listing 9.59 wird die Überschrift aus der Ausgabe von `ps` entfernt. Der Befehl `ps` kennt dazu auch die Option `-h`, so dass man in diesem Fall in der Praxis auf die Verwendung von `tail` verzichtet.

```
(linux):~$ ps | tail -n +2
  143 v02 S     0:06 -tcsh
  144 v03 S     0:13 -tcsh
  145 v04 S     0:08 (tcsh)
  153 v04 S     4:47 emacs master.tex
  219 v03 R     0:00 ps
  220 v03 S     0:00 tail -n +2
```

Listing 9.59 Die Datenübergabe mit einer Pipe

Oft müssen Sie die Ausgaben eines Befehls aufbewahren, um sie z. B. in eine Textdatei aufzunehmen. Auch diese Funktion ist in der Shell mit dem Größer-Zeichen (`>`) implementiert. Geben Sie als Dateinamen `/dev/null` an, so wird die Ausgabe ignoriert. Das Listing 9.60, zeigt ein Beispiel für die Verwendung einer Pipe und der Ausgabeumleitung.

```
(linux):~$ ps | tail -n +2 > ps.out
```

Listing 9.60 Die Verwendung einer Pipe

Wenn Sie keine besonderen Vorkehrungen treffen, können Sie auf diese Weise jede Datei, auf die Sie Schreibrechte haben, überschreiben. Sie machen doch eine Datensicherung, oder? Innerhalb von Shell-Skripten ist das Überschreiben ohne Nachfrage in der Regel sinnvoll, bei der interaktiven Nutzung können Sie dieses mit der Shell-Option `noclobber` verhindern.

Bei der `bash` (und der `sh` bzw. `ksh`) kann die Standardfehlerausgabe, auf der nur Fehlermeldungen ausgegeben werden, einzeln erfasst werden. Dazu dient die Zeichenfolge `2>` anstelle des Größer-Zeichens (`>`). Wenn Sie beide Ausgabekanäle gemeinsam erfassen möchten, leiten Sie zunächst die Standardausgabe mit `>` um. Anschließend lenken Sie die Fehlerausgabe mit `2>&1` auf diesen Kanal um. Die Zeichenfolge `&1` steht für den Filedeskriptor der Standardausgabe, `&2` ist die Standardfehlerausgabe und `&0` die Standardeingabe. Die `csh` ist hier nicht so flexibel wie die `sh`. Das ist einer der Gründe, warum man keine `csh`-Skripten schreiben sollte.

Eine weitere Funktion ist das Lesen aus einer Datei anstelle der Standardeingabe. Dazu wird das Kleiner-Zeichen (`<`) verwendet. Viele Programme verwenden, wenn als Dateiname ein Minuszeichen (`-`) angegeben wird, die Standard- ein- bzw. Standardausgabe anstelle einer Datei. Dies ist nur bei Programmen notwendig und sinnvoll, die standardmäßig nicht von der Standardeingabe lesen oder auf die Standardausgabe schreiben. Ein Beispiel für derartige Programme ist `tar`, das ein Band benutzt, falls kein Dateiname angegeben wird.

Im Listing 9.61 wird die Datei `.emacs` als E-Mail an den Benutzer `root` verschickt. Dabei wird als `Subject:` der Text »meine .emacs-Datei« verwendet.

```
(linux):~$ mail -s 'meine .emacs-Datei' root < .emacs
```

Listing 9.61 Eingabeumleitung in der Shell

In Shell-Skripten wird oft darauf verzichtet, eine Datei zu erstellen, die im nächsten Schritt dann als Eingabe verwendet wird. Der Text wird direkt (als *Here-Dokument*) in das Skript aufgenommen. Die Zeichen `<<`, gefolgt von einer beliebigen Zeichenkette, weisen die Shell an, den Text bis zum Auftreten dieser Zeichenfolge als Standardeingabe zu verwenden. Dabei werden Shell-Variablen expandiert, aber keine anderen Sonderzeichen (Listing 9.62).

```
cat << EOF
Das ist der Text, der ausgegeben
werden soll.  Er kann (fast) beliebig lang sein.
Das Skript wurde vom Benutzer »$USER« gestartet.
EOF
```

Listing 9.62 Ein Beispiel für ein »Here«-Dokument

9.9.2 Expandieren von speziellen Zeichen

Die Shell kennt eine Reihe von Zeichen, die bei der Eingabe oder in Shell-Skripten eine besondere Bedeutung haben. Hier spielt eine Rolle, dass die Shell fast alles als String betrachtet. Das gilt auch für vom Benutzer eingegebene Befehle oder Shell-Skripten. In diesem String werden zunächst die Sonderzeichen gesucht und bearbeitet und anschließend wird der entsprechende Befehl ausgeführt.

Das Verfahren, wie aus der Eingabe die auszuführenden Befehle erzeugt werden und wann welche Ersetzungen durchgeführt werden, ist standardisiert. Der POSIX-Standard beschreibt hier das Verfahren, das sich historisch mit der `sh` entwickelt hat. Dabei wurden auch einige Entscheidungen, die man heute nicht mehr so treffen würde, festgeschrieben.

Dateinamenexpansion

Bei der Eingabe eines Befehls kann man entweder den oder die Dateinamen von Hand eingeben oder die Dateinamenvervollständigung der Shell benutzen. Oft wird es jedoch so sein, dass man eine Gruppe von Dateien gemeinsam bearbeiten möchte. Diese Gruppe ist häufig durch einen Namensbestandteil identifiziert. Für diese Funktion kann die *Dateinamenexpansion* verwendet werden.

Zur Expansion werden verschiedene Sonderzeichen (Wildcards oder Jokerzeichen) verwendet. Zunächst steht ein Sternchen (`*`) für beliebig viele Zeichen eines Dateinamens, auch für einen oder mehrere Punkte. Im Gegensatz zu DOS steht also ein `*` allein für (fast) alle Dateien. Ausgenommen sind diejenigen Dateien, die mit einem Punkt (`.`) beginnen. Das Fragezeichen (`?`) steht für genau ein Zeichen in dem Dateinamen. Im Gegensatz zu DOS können die Jokerzeichen an beliebigen Stellen im Muster auftauchen, nicht nur am Ende. Auch der Punkt (`.`) hat außer der oben erwähnten Besonderheit unter Unix keine spezielle Bedeutung.

Die Expandierungen erfolgen durch die Shell, so dass der Befehl `ls` bei der Eingabe `ls *.c` am Shellprompt bereits eine Liste von C-Dateien als Parameter erhält. Sie können daher in Notfällen, wenn z. B. auf einer Notfalldiskette kein `ls` vorhanden ist, immer noch eine Liste der Dateien im aktuellen Verzeichnis erstellen, indem Sie den internen Shell-Befehl `echo *` verwenden. Die Expansion durch die Shell ist dafür verantwortlich, dass es relativ schwierig ist, eine Gruppe von Dateien mit einem Befehl umzubenennen. Hierzu kann das Tool `mmv` verwendet werden.

Wollen Sie die Expandierung verhindern, so können Sie entweder jedes einzelne Sonderzeichen mit einem Backslash (`\`) entwerten oder die Zeichenkette in einfache Anführungszeichen einschließen. Andernfalls würde das Programm be-

reits expandierte Shell-Sonderzeichen als Parameter erhalten und sicher nicht das gewünschte Ergebnis liefern. Wenn Sie nicht sicher sind, ob ein Programm wirklich die richtigen Parameter erhält, verwenden Sie den Befehl `echo`.

Bei der Dateinamenexpansion können Sie auch Zeichenmengen, z. B. mit `[A-Z]`, angeben. Diese Expansionen sind keine regulären Ausdrücke, aber diesen relativ ähnlich. Wird der Ausdruck in doppelte Anführungszeichen eingeschlossen, so werden Shell-Variablen expandiert, aber keine Sonderzeichen wie z. B. das Sternchen.

Dateinamen unter Unix können beinahe beliebige Zeichen enthalten. Auch die Verwendung von Leerzeichen oder Jokerzeichen ist möglich, aber nicht empfehlenswert. Bei der Verwendung dieser Zeichen ist immer besondere Vorsicht geboten, andernfalls können unverhofft die merkwürdigsten Dinge passieren, wenn die Shell derartige Sonderzeichen vom Benutzer ungewollt expandiert.

Will man einen Dateinamen, der ein Leerzeichen enthält, als Parameter übergeben, so muss man diesen in Anführungsstriche einschließen. Gute Shells wie die `bash` tun dies bei der Dateinamenvervollständigung automatisch. Der POSIX-Standard legt fest, dass Dateinamen auf jedem POSIX-kompatiblen System aus den Buchstaben von `a` bis `z`, den entsprechenden Großbuchstaben sowie Ziffern und einigen Sonderzeichen wie Bindestrich und Unterstrich zusammengesetzt werden können – Umlaute sind bei manchen Programmen oder anderen Unix-Systemen mit Vorsicht zu genießen.

Versuchen Sie, einen Ausdruck zu erstellen, der alle Dateien in einem Verzeichnis erfasst, aber nicht die Verzeichnisse `.` und `...` Überlegen Sie sich auch, wie Sie eine Datei, deren Namen ein Leerzeichen enthält, ansprechen. Ein gefährlicher Test für Unix-Neulinge ist es, die Datei `-rf *` anzulegen und löschen zu lassen. Neulinge vergessen oft, die Sonderzeichen in Anführungszeichen einzuschließen, und löschen das gesamte Verzeichnis inklusive aller Unterverzeichnisse. Als letzte Übung, auch wenn sie nichts mehr mit der Dateinamenexpansion zu tun hat, sollten Sie versuchen, die Datei `-i` zu löschen.

Nun folgt die Auflösung zu den oben gestellten Fragen. Der erste Versuch, einen Ausdruck zu erstellen, der auf alle Dateien in einem Verzeichnis passt, könnte z. B. `.* *` sein, was aber die Verzeichnisse `.` und `..` einschließt. An der zweiten Stelle derjenigen Dateinamen, die mit einem Punkt beginnen, muss ein Zeichen stehen, das kein Punkt ist. Der zweite Versuch ist dann `.[^.]* *`, was aber Dateinamen, die mit zwei Punkten beginnen, nicht expandiert. Also benötigt man zusätzlich noch das Suchmuster `..?*`. Der gesamte Ausdruck ist also `* .[^.]* ..?*`, wobei außerdem kein Dateiname doppelt angegeben wird. Viel einfacher ist es aber, die dafür vorhandene `ls`-Option `-A` (bzw. `--almost-all`) zu verwenden.[5]

5. Die Unix-FAQ enthält eine etwas andere Lösung, eine Beschreibung, wie man zu dieser Lösung kommt und Betrachtungen zur Portabilität.

Eine Datei, deren Name ein Leerzeichen enthält, können Sie mit der Shell bearbeiten, indem Sie den Namen in einfache oder doppelte Anführungszeichen einschließen. Alternativ können Sie das Leerzeichen mit Hilfe eines Backslash entwerten, aber das wird recht schnell unübersichtlich. Wie bereits oben erwähnt, schließt die `bash` Dateinamen, die ein Leerzeichen enthalten, bei der Verwendung der Dateinamenvervollständigung automatisch in Anführungsstriche ein.

Bei der dritten Aufgabe interpretiert das Programm `rm` den Namen `-i` als Option. Der einfachste Weg ist die Angabe eines relativen Pfads zur Datei, also `rm ./-i`. Alternativ kann die Zeichenfolge `--` verwendet werden, um Optionen und Parameter zu trennen. Der resultierende Befehl ist dann `rm -- -i`.

Shell-Variablen

Neben der Expansion von Dateinamen kennt die Shell Variablen und Parameter. Beide werden durch ein Dollarzeichen (`$`) markiert. Shell-Variablen werden anhand ihres Namens (z. B. `$PATH`) identifiziert, Parameter von Shell-Skripten oder Shell-Funktionen aufgrund ihrer Position in der Eingabezeile (z. B. `$1` für den ersten Parameter). Daneben existieren eine Reihe von speziellen Variablen; eine Übersicht dazu finden Sie in Tabelle 9.8.

In der Manpage zur `bash` finden Sie eine Reihe von weiteren Informationen, die Sie bei der intensiveren Beschäftigung mit der Shell-Programmierung über Variablen kennen sollten. Abschnitt »Variablensubstitution« enthält weitere Informationen zur Bearbeitung von Variablen durch die Shell.

Variable	Bedeutung
`$*`	Alle Parameter
`$@`	Alle Parameter als einzelne Worte
`$#`	Anzahl der Parameter
`$?`	Status (Rückgabewert) des letzten Kommandos
`$$`	Prozess-ID der aktuellen Shell, z. B. für temporäre Dateien
`$!`	Prozess-ID des letzten Hintergrundprozesses
`$0`	Name der Shell oder des Shell-Skripts
`$1-$9`	Erster bis neunter Parameter, alle weiteren Parameter sind erst nach `shift` zugänglich

Tabelle 9.8 Spezielle Shell-Variablen

Die Bedeutung der einzelnen Variablen in der Tabelle 9.8 ist relativ klar, mit Ausnahme des Unterschieds zwischen `$*` und `$@`. Wenn Sie `$*` verwenden, dann entspricht das in etwa der Verkettung aller Parameter durch Leerzeichen. Ist einer der Parameter beispielsweise `"Ich bin da"`, dann wird dieser Parameter zu

drei Worten expandiert. Wenn die Shell-Variable `IFS` gesetzt ist, dann wird das erste Zeichen dieses Wertes als Trennzeichen verwendet.

Die Variable `$@` expandiert dagegen zu `"$1" "$2" ...`, das oben angegebene Beispiel expandiert also nur zu einem Parameter für den nächsten Aufruf. Das entsprechende Beispiel finden Sie im Listing 9.63. Im ersten Aufruf der Shell-Funktion `zweite` wird durch die doppelten Anführungszeichen exakt ein Parameter übergeben. Der zweite Aufruf expandiert zu zwei Parametern, wobei der erste Parameter aus einer Zeichenkette mit drei Wörtern besteht. Der letzte Aufruf expandiert zu insgesamt vier einzelnen Wörtern. Wenn Sie größere Shell-Skripten mit Funktionen und verschachtelten Aufrufen erstellen, dann können diese Informationen sehr nützlich sein.

```
(linux):~$ zweite() { echo "$1 * $2 * $3 * $4" }
(linux):~$ erste() { zweite "$*"; zweite "$@"; zweite $* }
(linux):~$ erste "Da bin ich" "nochmal"
Da bin ich nochmal *  *  *
Da bin ich * nochmal *  *
Da * bin * ich * nochmal
(linux):~$ IFS=/
(linux):~$ erste Da/bin/ich nochmal
Da/bin/ich/nochmal *  *  *
Da/bin/ich * nochmal *  *
Da * bin * ich nochmal *
```

Listing 9.63 Expansion von Parametern und `IFS`

Einige Variablen werden bereits durch die Shell oder das System vorbelegt und können in Login-Skripten oder eigenen Programmen verwendet werden. Eine (unvollständige) Übersicht finden Sie in Tabelle 9.9. Besonders erwähnenswert ist hier die Variable `IFS` (Internal Field Separator). Diese Variable bestimmt, ob und an welchen Stellen die Shell z. B. die Parameter trennt. Der letzte Aufruf in Listing 9.63 zeigt ein einfaches Beispiel.

Variable	Bedeutung
`HOME`	Home-Verzeichnis (manchmal auch als Tilde ~ angegeben)
`PATH`	Suchpfad nach Programmen
`USER`	Der eigene Benutzername
`LOGNAME`	Der Login-Name
`SHELL`	Die verwendete Shell
`PS1`	Definition des Prompts
`MAIL`	Die Mailbox-Datei
`IFS`	Internal Field Separator

Tabelle 9.9 Vorbelegte Shell-Variablen

Bei der Expansion von Variablen gibt es noch einige weitere Besonderheiten. Soll unmittelbar nach der Variablen weiterer Text folgen, so muss der Variablenname in geschweifte Klammern eingeschlossen werden (`${PS1}text`).

Brace-Expansion

Ein weiteres Sonderzeichen sind die geschweiften Klammern (`{}`, »Braces«). Die Erweiterung (*Brace-Expansion*) erfolgt auf textueller Basis. Mit dieser Funktion ist es recht einfach möglich, eine Reihe von Dateien mit derart beschriebenen Namen neu zu erzeugen. Das Listing 9.64 zeigt einige Beispiele für die Verwendung dieser Expansion.

```
(linux):~$ echo Datei{1,2,3,4}
Datei1 Datei2 Datei3 Datei4
(linux):~$ echo Datei{1,2,3}{a,b}
Datei1a Datei1b Datei2a Datei2b Datei3a Datei3b
```

Listing 9.64 Brace-Expansion

Erst nach dieser Expansion werden andere Sonderzeichen, wie Sternchen und Fragezeichen, oder Variablen expandiert. Bei der Brace-Expansion spielen existierende Dateien keine Rolle, ganz im Gegensatz zu den Jokerzeichen `*` und `?`.

Ausgabe eines Kommandos als Parameter

Oft ist es nützlich oder notwendig, die Ausgabe eines Programms (z. B. von `find`) als Parameter eines anderen Programms zu verwenden. Für diese *Kommandosubstitution* existieren zwei Schreibweisen, die im Prinzip zueinander äquivalent sind (Listing 9.65). Die umgekehrten einfachen Anführungszeichen (Backquotes, `` ` ``) sind in der Regel einfacher zu tippen, haben aber den Nachteil, dass sie nicht geschachtelt werden können und möglicherweise aus dem POSIX-Standard entfernt werden.

```
(linux):~$ rm -f `find . -name \*.orig`
(linux):~$ rm -f $(find . -name \*.orig)
```

Listing 9.65 Backquotes und deren Alternative

Verwenden Sie `$(Kommando)`, werden Zeilenvorschübe durch Leerzeichen ersetzt. Meistens ist das gewünscht, aber wenn Sie `"$(Kommando)"` verwenden, dann bleiben Zeilenvorschübe erhalten.

9.9.3 Rechnen in Shell-Skripten

Obwohl die Shell im Wesentlichen nur mit Strings umgeht, kann man in Skripten oder am Prompt auch rechnen. Dies geht jedoch, wie fast alle anderen Funk-

tionen auch, auf Kosten der Geschwindigkeit. Ab einer gewissen Größe sollte man darüber nachdenken, eventuell eine andere Skript-Sprache wie z. B. `perl` einzusetzen oder ein entsprechendes C-Programm zu implementieren.

Die ersten beiden Beispiele in Listing 9.66 verwenden die internen Funktionen der `bash`. Alternativ kann auch das externe Kommando `expr` verwendet werden, das im dritten Beispiel benutzt wird. Hier müssen die Zahlen und Rechenzeichen durch Leerzeichen getrennt werden. Wenn Sie multiplizieren möchten, dann denken Sie daran, dass das Sternchen ein Shell-Sonderzeichen ist und entsprechend maskiert werden muss.

```
(linux):~$ echo $[1+2]
3
(linux):~$ echo $((1+2))
3
(linux):~$ expr 1 + 2
3
```

Listing 9.66 Rechnen in Shell-Skripten

9.9.4 Kontrollstrukturen

Neben der Verwaltung und Expansion von Variablen können mit der Shell auch komplexe Abläufe programmiert werden. Dazu dienen die Kontrollstrukturen `if`, `case`, `for`, `while` und `until`, die aus der Shell eine leistungsfähige Programmiersprache machen.

Die unauffälligste Kontrollstruktur ist die Sequenz. Die einzelnen Befehle werden dabei durch Semikola (`;`) getrennt. Am Zeilenende muss kein Trennzeichen angegeben werden, die Shell interpretiert dies in der Regel als Ende des Befehls. Wenn Sie eine Sequenz von Befehlen in runde Klammern `()` einschließen, werden diese Befehle von einer neu gestarteten Shell ausgeführt. Änderungen an Umgebungsvariablen oder ein Verzeichniswechsel haben so keine Auswirkungen auf die Shell oder das Skript.

if-Verzweigung

Mit der `if`-Verzweigung kann eine Bedingung geprüft und entsprechend behandelt werden. Die Bedingung wird oft, aber nicht immer durch das Programm `test` (oder `[`) geprüft. Dabei stellt der Rückgabewert des aufgerufenen Programms das Entscheidungskriterium dar. Daher kann praktisch jedes Unix-Programm als Bedingung eingesetzt werden, wenn der Rückgabewert passend gesetzt wird.

Das Programm `test` kann viele Bedingungen prüfen und diese auch logisch miteinander verknüpfen. Die wichtigsten Optionen für `test` finden Sie in Tabelle 9.10 alle Möglichkeiten dieses Programms sind in der Manpage dokumentiert.

Option	Bedeutung
`-d` *Verzeichnis*	Das Verzeichnis existiert?
`-f` *Datei*	Die (reguläre) Datei existiert?
`-r` *Datei*	Die Datei ist lesbar?
`-s` *Datei*	Die Datei ist größer als null Byte?
`-w` *Datei*	Die Datei ist beschreibbar?
`-x` *Datei*	Die Datei ist ausführbar?
String1 `=` *String2*	Die Strings sind gleich.
`!` *Ausdruck*	Logisches Nicht
Ausdruck1 `-a` *Ausdruck2*	Logisches Und
Ausdruck1 `-o` *Ausdruck2*	Logisches Oder
Ausdruck1 `OP` *Ausdruck2*	Diverse numerische Vergleiche

Tabelle 9.10 Auszug aus den Optionen von `test`

In vielen Shell-Skripten, wie auch im Listing 9.67 wird die eckige öffnende Klammer als Aliasname für `test` benutzt. Im Dateisystem sind beide Programme mittels eines Hardlinks miteinander verknüpft. Das einzige, was Sie beachten müssen, ist, dass Sie die geöffnete Klammer wieder schließen.

```
if [ -x /sbin/kerneld ] ; then
    echo -n "Starting kerneld "; /sbin/kerneld; echo "done"
else
    echo "/sbin/kerneld not found."
fi
```

Listing 9.67 Verzweigung mit `if`

Beachten Sie, dass in der Regel das Programm `/usr/bin/test` vor dem Programm `./test` ausgeführt wird. Normalerweise sollte das aktuelle Verzeichnis (`.`) nicht (schlimmstenfalls bei nicht privilegierten Benutzern hinten) im Pfad enthalten sein. Andernfalls ist es möglich, dass z. B. ein Programm eines anderen Benutzers aus dem `tmp`-Verzeichnis gestartet wird. Der Sicherheitsgewinn, der sich ergibt, wenn man das aktuelle Verzeichnis aus dem Pfad entfernt, wiegt die Unannehmlichkeit des (seltenen) Tippens von `./` bei weitem auf. Ein beliebter Fehler ist, dass Benutzer ein Testprogramm `test` nennen und dann das gleichnamige Systemprogramm starten.

Einfache Verzweigungen können Sie auch mit den Kontrolloperatoren `&&` und `||` realisieren. Dabei ist die Ausführung des zweiten Befehls abhängig vom Rückgabewert des ersten. Der Operator `&&` führt den folgenden Befehl nur aus,

wenn das erste Kommando mit dem Rückgabewert 0 beendet wurde. Der Operator || startet das zweite Kommando nur, wenn das erste einen Rückgabewert ungleich 0 liefert. Einige einfache Beispiele finden Sie im Listing 9.68. Oft werden diese Funktionen für einfache Fehlerabfragen verwendet.

```
(linux):~$ [ -f /etc/inittab ] && echo 'inittab existiert'
inittab existiert
(linux):~$ false || echo 'wird ausgeführt'
wird ausgeführt
```

Listing 9.68 Die Operatoren && und ||

Manchmal benötigt man ein Programm, das stets dasselbe Ergebnis liefert. Unter Unix können Sie hier die Programme `true` und `false` verwenden. Wenn es darum geht, permanent eine bestimmte Zeichenkette auszugeben, dann leistet das Programm `yes` gute Dienste.

Mehrfachverzweigungen mit case

Die Shell verfügt neben der einfachen Verzweigung, die auf Rückgabewerten von Programmen basiert, auch über eine *Mehrfachverzweigung* (`case`). Diese basiert auf einem *Pattern-Matching*. Es wird hier nach einer Übereinstimmung zwischen der angegebenen Variablen und dem Suchmuster gesucht und der passende `case`-Zweig durchlaufen. Beim Vergleich des Werts mit dem Muster gelten dieselben Regeln wie bei der Expansion von Dateinamen. Im Listing 9.69 finden Sie eine einfache Anwendung.

```
case "$1" in
    start|START)
        echo "start"
        ;;
    stop|STOP)
        echo "stop"
        ;;
    *)
        echo "Unbekannter Parameter"
        exit 1
        ;;
esac
```

Listing 9.69 Mehrfachverzweigung mit `case`

Wenn als Parameter im Listing 9.69 die Zeichenkette `start` oder `START` angegeben wird, dann wird die erste Verzweigung durchlaufen. Wenn man möglichst wenig angeben will, dann könnte man als Muster beispielsweise `sta*` angeben, so dass tatsächlich nur noch drei Buchstaben angegeben werden müssen. Das

letzte Muster * ist der Zweig, der durchlaufen wird, wenn kein anderes Suchmuster gepasst hat. Im Listing 9.69 wird eine Fehlermeldung ausgegeben und das Programm beendet.

Wiederholungen und Schleifen

Ein weiteres Konstrukt, das eine Programmiersprache bereitstellen muss, sind Wiederholungen und Schleifen. Auch hier verfügt die bash über eine Reihe von unterschiedlichen Möglichkeiten.

Schleifen über Aufzählungen

Mit der for-Schleife kann für eine Reihe von Wörtern eine Folge von Befehlen ausgeführt werden. Das Listing 9.70 zeigt dafür ein einfaches Beispiel. Zunächst wird der Ausdruck *.c zu einer Liste der C-Dateien im aktuellen Verzeichnis expandiert. Anschließend werden für jede Datei die Befehle im Schleifenkörper ausgeführt.

```
for i in *.c ; do
    cp $i $i.old
done
```

Listing 9.70 Die for*-Schleife in der* bash

Die for-Schleife iteriert über Wörter. Eine FOR-Schleife, wie man sie aus BASIC kennt, ist in der Shell nicht implementiert. Man kann jedoch mit einem externen Programm wie seq zunächst die zu durchlaufenden Werte generieren und dann mit Hilfe einer for-Schleife abarbeiten.

Wiederholung bis zur Erfüllung einer Bedingung

Die Shell-Befehle until und while dienen zur Ausführung von Schleifen, bis (until) oder solange (while) die danach folgende Bedingung erfüllt ist. Anschließend wird die Schleife beendet und die Ausführung mit dem nächsten Befehl fortgesetzt. Die Syntax der Befehle ist:

```
while list; do list; done
until list; do list; done
```

Listing 9.71 Syntax von while- *und* until- *Schleifen*

Die Schleife kann auch mit dem Befehl break abgebrochen werden. Das kann bei aufgetretenen Fehlern sinnvoll sein oder wenn die Bedingung nicht am Schleifenanfang oder -ende überprüft werden kann, sondern nur mitten in der Schleife. Ein Beispiel für eine while-Schleife finden Sie in Listing 9.72.

```
(linux):~$ cat while-beispiel
#!/bin/sh
par=1
```

```
while [ $# -ne 0 ]; do
    echo "Parameter $par: »$1«"
    par=$(expr $par + 1)
    shift
done
(linux):~$ ./while-beispiel eins zwei "drei vier"
Parameter 1: »eins«
Parameter 2: »zwei«
Parameter 3: »drei vier«
```

Listing 9.72 Eine einfache `while`*-Schleife*

Shell-Funktionen

Die `bash` verfügt über *Shell-Funktionen*, die allerdings von der Standard-Shell `sh` nicht unterstützt werden. Sollten Sie dennoch Funktionen einsetzen, was für die Übersichtlichkeit in Shell-Skripten sehr sinnvoll ist, verlieren Sie jedoch die einfache Portabilität zu anderen Systemen, die (noch) keine POSIX-kompatible Shell wie die Korn-Shell `ksh` haben.

Auch für das interaktive Arbeiten sind Shell-Funktionen nützlich. In einem Aliasnamen können Sie nur die gesamte Kommandozeile bearbeiten und nicht einzelne Parameter verwenden oder an diese weitere anhängen. In solchen Fällen erweist sich eine Shell-Funktion als die Rettung, dort können Sie auch weitere Funktionen wie Verzweigungen einfach und übersichtlich unterbringen. Ein entsprechendes Beispiel finden Sie im Listing 9.73.

```
(linux):~$ alias abc='echo $* danach'
(linux):~$ abc eins zwei drei
danach eins zwei drei
(linux):~$ unalias abc
(linux):~$ function abc () { echo $* danach }
(linux):~$ abc eins zwei drei
eins zwei drei danach
```

Listing 9.73 Eine einfache Shell-Funktion

In Shell-Funktionen verwendete Variablen sind global, es sei denn, sie wurden mit dem Schlüsselwort `local` als lokale Variablen deklariert. Die `bash` erkennt anhand der runden Klammern, dass eine Shell-Funktion definiert wird, so dass das Schlüsselwort `function` nicht angegeben werden muss.

Variablensubstitution

Variablen werden mit dem Dollar-Zeichen (`$`) expandiert. Anstelle des Variablennamens wird der Inhalt der Variablen eingesetzt. Der Variablenname kann optional in geschweifte Klammern eingeschlossen werden, um das Ende des Na-

mens zu markieren. Die Shell hat aber auch Möglichkeiten, die Expansion von Variablen weiter zu beeinflussen.

- `${Variable:-Wort}` Wenn die Variable nicht gesetzt oder leer ist, dann wird der Wert von *Wort* eingesetzt. Die Shell führt vorher noch die Tildenexpansion, Variablenexpansion, Kommandosubstitution und arithmetische Berechnungen aus.
- `${Variable:=Wort}` Zusätzlich zu den oben angegebenen Funktionen wird der Variablen *Variable* noch das Wort nach der Expansion von *Wort* zugewiesen.
- `${Variable:?Wort}` Wenn die Variable nicht gesetzt oder leer ist, dann wird der Text, der sich durch die Expansion von *Wort* ergibt, auf die Standardfehlerausgabe ausgegeben. Andernfalls wird der Wert der Variablen eingesetzt.
- `${Variable:+Wort}` Wenn die Variable nicht gesetzt oder leer ist, wird nichts ersetzt. Andernfalls wird das Ergebnis der Expansion von *Wort* eingesetzt.
- `${#Variable}` Die Länge des Variableninhalts in Zeichen.
- `${Variable#Wort}` `${Variable##Wort}` Die Variable wird expandiert und anschließend wird, falls das Pattern *Wort* auf den Beginn des Textes passt, dieser entfernt. Bei der #-Variante wird der kleinste passende String entfernt, bei ## der größte.
- `${Variable%Wort}` `${Variable%%Wort}` Hier wird nicht der Anfang, sondern das Ende des Strings betrachtet.

Zum Abschluss dieses Abschnitts finden Sie im Listing 9.74 einige Beispiele zur Variablenexpansion. Überlegen Sie sich, welcher Befehl welche Ausgabe erzeugt und warum. Dieses Wissen kann manchmal sehr nützlich sein, wenn man nur schwer nachvollziehbare Probleme bzw. Fehler in Shell-Skripten oder Kommandozeilen aufspüren will.

```
(linux):~$ export VAR="echo Das ist ein Test"
(linux):~$ echo $VAR ; echo \$VAR ; echo "$VAR"
(linux):~$ echo "\$VAR" ; echo '$VAR'
(linux):~$ echo '\$VAR' ; echo `$VAR`
(linux):~$ echo $VAR'abc' ; echo $VAR{a,b,c} ; echo $VARabc
(linux):~$ echo ${VAR}abc ; echo ${VAR}{a,b,c}
```

Listing 9.74 Beispiele für Quoting und Variablen in der Shell

9.9.5 Tipps zur Programmierung

Shell-Skripten dienen oft zur automatischen Durchführung wiederkehrender Aufgaben. In der Regel werden Sie diese Aufgaben zunächst »von Hand« durchführen. Später, wenn Sie wissen, welche Befehle in welcher Reihenfolge (und un-

ter welchen Bedingungen) auszuführen sind, können Sie mit der Erstellung eines Skripts beginnen.

Gehen Sie bei der Programmierung eines Skripts schrittweise vor und testen Sie regelmäßig den bisher erstellten Teil. Dadurch kommen Sie Fehlern, wie z. B. falschen und nicht geschlossenen Anführungszeichen relativ schnell auf die Schliche.

Lagern Sie mehrfach benötigte Programmteile in eigene Funktionen aus. Das sind oft eine Funktion, die eine Fehlermeldung ausgibt und das Skript (mit einem Returncode ungleich Null) beendet, und eine Funktion, die die Verwendung (Syntax) erläutert. Derartige Funktionen können Sie in eine eigene Datei auslagern und dann in verschiedenen Shell-Skripten »sourcen«, also mittels `. Datei` einlesen.

Verwenden Sie keine fest vorgegebenen Dateinamen (z. B. für temporäre Dateien), sondern einheitlich eine Variable. In diesem Fall können Sie den Pfad und den Namen der Datei einfach ändern. Für temporäre Dateien sollten Sie auf alle Fälle die Prozess-ID der Shell in den Dateinamen aufnehmen, so dass auch mehrere Skripten parallel laufen können.

Wenn Ihr Skript nicht so abläuft, wie Sie es sich vorgestellt haben, so können Sie das Skript entweder mit `sh -x` *`Skript`* ausführen oder in die `#!`-Zeile ein `-x` aufnehmen. Die Shell protokolliert dann jede ausgeführte Zeile, so dass Sie relativ schnell zum Kern des Problems vorstoßen können. Alternativ können Sie auch den Befehl `set -x` direkt im Skript verwenden.

Hier wird eine relativ große Ausgabe erzeugt, außerdem dauert der Ablauf des Skripts dadurch relativ lange. Es kann daher sinnvoll sein, die Ausgabe mittels `script` in eine Datei zu schreiben oder im Shell-Modus des Emacs aufzubewahren. Dort können Sie dann in Ruhe den Ablauf nachvollziehen und dem Fehler vergleichsweise schnell auf die Spur kommen.

Schreiben Sie kleine, überschaubare Programme oder Funktionen und überprüfen Sie die Ergebnisse von aufgerufenen Funktionen. Eventuell bauen Sie sogar in die verschiedenen Funktionen Fehlersuchhilfen ein, so dass Sie ohne Änderung des Skripts in bestimmten Funktionen ausführlichere Ausgaben erzeugen können.

9.10 Prozesse und Jobs

Unter Unix können gleichzeitig viele Programme vieler verschiedener Benutzer aktiv sein. Darunter befinden sich interaktive Programme, die von angemeldeten Benutzern verwendet werden, Programme, die im Hintergrund von Benutzern gestartet wurden, die schon lange nicht mehr angemeldet sind, oder System-Dämonen zur Bereitstellung von Diensten. Zur Prozessverwaltung stellen sowohl das Unix-System als auch die Shell einige Funktionen bereit.

9.10.1 Prozessverwaltung

Jeder Prozess wird unter Unix eindeutig durch seine *Prozessnummer* (Process-ID, kurz PID) identifiziert. Diese Nummern werden beim Erzeugen eines neuen Prozesses mit `fork()` vom Kernel in der Regel in aufsteigender Reihenfolge vergeben. Auf anderen Systemen (zum Beispiel AIX) wird manchmal ein anderes Verfahren zur Erzeugung der Prozessnummern verwendet. Ist die größte Prozessnummer erreicht, so wird wieder von vorn gezählt; bereits belegte Nummern werden natürlich übersprungen. Nach einem `fork()` existieren zwei fast[6] identische Prozesse.

Soll nach dem `fork()` ein anderes Programm gestartet werden, so wird dazu in der Regel eine Funktion aus der `exec`-Familie verwendet. Diese Funktionen ersetzen das gerade laufende Programm durch das neu zu startende. Die Prozessnummer bleibt dabei erhalten. Unter Unix existiert keine C-Funktion, die diese beiden Schritte automatisch ausführt. Damit ist die Programmierung etwas aufwändiger, aber durch die verschiedenen Funktionen aus der `exec`-Familie ist eine flexible Steuerung der Umgebung und der Parameter möglich, mit denen der neue Prozess gestartet wird.

Auch die Shell kennt den `exec`-Befehl, der die Shell durch ein anderes Programm ersetzt. Wenn Sie die Fehlermeldung »`no more processes`« beim Aufruf eines Befehls erhalten, dann können Sie mit diesem Befehl zumindest noch ein Programm starten. Danach ist allerdings die Shell beendet, so dass Sie diesen einen Befehl mit Bedacht wählen sollten. Hier kann z. B. `top` (table of processes) eine gute Idee sein, da man das System damit untersuchen und von dort aus Prozesse beenden kann.[7]

Neuere Linux- und Unix-Versionen kennen neben Prozessen noch den Begriff des *Threads*. Threads sind Ausführungspfade innerhalb eines Programms. Die Verwendung von Threads ist z. B. bei rechenintensiven Aufgaben innerhalb eines interaktiven Programms sinnvoll. Je nach Anforderung müssen verschiedene Systembereiche zwischen den Threads eines Prozesses gemeinsam benutzt werden. Außerdem ist eine Synchronisation mittels Semaphoren erforderlich. Linux unterstützt die Programmierung mit Threads durch den Systemaufruf `clone(2)` und die Bibliothek `linuxthreads`, die in der GNU-C-Bibliothek (`glibc`) enthalten ist.

6. Der Unterschied besteht darin, dass der Vaterprozess als Rückgabewert von `fork()` die Prozess-ID des Kindprozesses bekommt, das Kind aber den Wert 0 erhält.
7. Unter AIX heißt ein vergleichbares Programm `topas`, als Freeware ist das Programm `monitor` verfügbar.

Prozessstatus ps

Der Status und viele weitere Informationen zu Prozessen können mit dem Befehl `ps` (process status) angezeigt werden. Ohne Parameter aufgerufen, zeigt `ps` nur dem Benutzer gehörende Prozesse an, denen ein Terminal zugeordnet ist. Die Option `x` zeigt auch solche Prozesse an, die kein zugeordnetes Terminal besitzen.

Die Prozesse anderer Benutzer werden angezeigt, wenn die Option `a` angegeben wird.[8] Oft verwendet man Befehle wie `ps aux | more`[9], um einen Überblick über das System und die darauf laufenden Prozesse zu gewinnen. Die Option `u` sorgt für die Anzeige des Benutzers, der Startzeit und einiger weiterer Informationen.

Eine weitere wichtige Option ist `w`, die mehrfach angegeben werden kann und mehr Text der Kommandozeilen der laufenden Prozesse ausgibt. Sie ist sinnvoll, wenn man wissen möchte, mit welchen Parametern ein bestimmter Hintergrundprozess gestartet wurde. Andererseits kann jeder Benutzer die Kommandozeile aller laufenden Programme erfahren, was oft aus Sicherheits- oder Datenschutzgründen nicht erwünscht ist.

Das Linux-`ps` kennt die Option `f`, die die Prozesstabelle als Baum darstellt. Alternativ können Sie den Befehl `pstree` verwenden. In manchen Fällen kann diese Übersicht sehr nützlich sein.

Weitere Optionen können Sie der Manpage zu `ps` entnehmen. Beachten Sie, dass von System-V abgeleitete Unix-Systeme eine Reihe von anderen Schaltern verwenden.

Prozesstabelle

Neben einigen anderen Programmen ist `top` (table of processes) sehr beliebt. Es werden die aktivsten Prozesse angezeigt, so dass man leicht feststellen kann, welche zurzeit aktiven Prozesse die CPU und den Speicher belegen. `top` benutzt, wie auch `ps`, das `proc`-Dateisystem, um die Daten über die Prozesse anzuzeigen.

Unter Linux kann die Darstellung innerhalb von `top` mit verschiedenen Tastenkombinationen verändert werden. Außerdem kann die Sortierung der Prozesstabelle beeinflusst werden und es besteht die Möglichkeit, Prozesse zu `renice`-n oder ein Signal zu senden. Tabelle 9.11 zeigt einen Überblick über die Möglichkeiten von `top`.

8. Ja, einige `ps`-Optionen werden tatsächlich ohne Minuszeichen verwendet.
9. Unter System-V heißt der Befehl `ps -ef`.

Taste	Bedeutung
Leer bzw. Strg-L	Auffrischen der Anzeige
f bzw. F	Hinzufügen oder Löschen von Feldern aus der Anzeige
o bzw. O	Ändern der Anordnung der angezeigten Felder
h bzw. ?	Ausgabe der Hilfeseite
S	Zeiten in Summen- oder Differenzdarstellung anzeigen
i	Darstellung untätiger Prozesse ein- bzw. ausschalten
c	Kommandoname oder Kommandozeile anzeigen
l	Load-Durchschnitt anzeigen oder ausblenden
m	Informationen über Hauptspeicher ein-/ausblenden
t	Summary anzeigen/ausblenden
k	Signal an einen Prozess senden
r	Einen Prozess »renicen«
P	Sortieren nach aktuellem relativen CPU-Verbrauch
M	Sortieren nach Speicherbelegung
T	Sortieren nach Zeit bzw. kumulierter Zeit
u	Nur einen bestimmten Benutzer anzeigen
n bzw. #	Anzahl der angezeigten Prozesse festlegen
s	Wartezeit bis zur nächsten Anzeige festlegen
W	Speichern der Konfiguration in der Datei `~/.toprc`
q	Das Programm verlassen

Tabelle 9.11 Tastaturbelegung in `top`

Neben der Darstellung der Prozesstabelle im Textmodus gibt es auch verschiedene Programme für die Anzeige der Daten unter X. Insbesondere haben sowohl KDE als auch GNOME entsprechende Programme implementiert.

Ein weiteres Programm, das den Status des Systems darstellt, ist `yamm`. Neben den Funktionen von `top` gibt es viele weitere Anzeigen, entweder system- oder prozessbezogen. Wenn Ihnen `top` nicht ausreicht, werfen Sie einen Blick auf `yamm`. Ein weiteres nützliches Tool zum Monitoring ist `vmstat`, das eine Statistik zur Speichernutzung anzeigt.

Prozessbaum

Zu jedem Prozess speichert der Kernel, von welchem Prozess dieser erzeugt wurde. Man spricht hier auch vom Vater-Prozess (parent-prozess). Die neu gestarteten Prozesse heißen Kind- oder Tochter-Prozesse (child-processes). Mit Hilfe des Kommandos `pstree`, das es unter anderen Unix-Systemen nicht gibt, kann man sich einen Überblick über die Hierarchie in der Prozesstabelle verschaffen. Rufen Sie es einfach mal auf.

Prozessmanagement

Ein Programm, das ein kontrollierendes Terminal hat, kann in der Regel mit Strg-C (dem Interrupt-Character, siehe auch die Ausgabe von `stty -a`) abgebrochen werden. Viele Prozesse (z. B. Netzwerkdämonen) unter Unix haben jedoch kein Terminal assoziiert und können daher nicht mit Tastenkombinationen kontrolliert werden. Es ist notwendig, auch diese Prozesse ansprechen zu können. Unter anderem dazu dient das Konzept der *Signale*. Ob und wie ein Programm auf Signale reagiert, ist implementationsabhängig und in der Regel in der Manpage dokumentiert. Die Standardreaktion ist für viele Signale das Programmende.

Das Programm `kill` dient nicht nur zum Abbruch von Programmen, sondern einfach zum Senden von Signalen. Signale können entweder numerisch oder symbolisch angegeben werden. Die wichtigsten Signale finden Sie in Tabelle 9.12 alle Signale gibt der Befehl `kill -l` aus. Eine Sonderfunktion hat das Signal 0: Hier prüft der Befehl `kill` nur, ob der betreffende Prozess existiert, und setzt den Returncode entsprechend. Diese Funktion ist in Shell-Skripten zur Systemüberwachung und -verwaltung oft sehr nützlich.

Signal	Nummer	Bedeutung
	0	Dummy-Signal, nur Prozess suchen
SIGHUP	1	Hang-Up, Konfiguration neu einlesen
SIGKILL	9	Unbedingtes Programmende
SIGUSR1	10	Benutzerdefiniertes Signal
SIGSEGV	11	Ungültige Speicherreferenz
SIGUSR2	12	Benutzerdefiniertes Signal
SIGTERM	15	Normales Programmende
SIGCONT	18	Fortsetzen der Ausführung nach SIGSTOP
SIGSTOP	19	Anhalten des Prozesses

Tabelle 9.12 Die wichtigsten Signale

Häufig müssen Sie einem Dämonen ein Hang-Up-Signal (`SIGHUP`) senden. Dadurch werden viele Dämons veranlasst, ihre Konfigurationsdatei neu zu lesen. Dazu können Sie die Prozessnummer des Dämonen mit `ps` erfahren. Dieser Prozessnummer senden Sie dann das Signal mit `kill -HUP` *`PID`*. Viele Dämonen speichern die Prozess-ID in einer Datei, die auch direkt zum Senden des Signals verwendet werden kann, z. B. mit dem Befehl `kill -HUP $(cat /var/run/syslog.pid)`. Das Kommando `killall`, das eine Linux-Spezialität[10] ist, ermög-

10. Auf anderen Systemen werden *alle* Prozesse beendet – danach ist das System erstmal unbrauchbar ...

licht es, einem Prozess ein Signal zu schicken, auch wenn man nur den Programmnamen kennt, also z. B. `killall -HUP syslogd`.

Gelegentlich ist das Signal `0` nützlich, es veranlasst den Kernel dazu, nach einem passenden Prozess zu suchen. Das Programm `kill` liefert den Return-Code `0`, wenn der Prozess existiert, ansonsten einen Fehlercode. Damit ist es einfach möglich, in einem Shell-Skript andere Prozesse zu überwachen. Das Listing 9.75 zeigt ein Beispiel dafür. Tun Sie das aber nicht auf fremden Rechnern, ohne dem Systemverwalter Bescheid zu sagen.

```
(linux):$ crontab -l
* * * * * killall -0 setiathome || setiathome
```

Listing 9.75 Prüfen und Neustarten eines Prozesses

In der Shell und in Shell-Programmen können Sie mit dem Befehl `trap` Signale bearbeiten. Dies kann notwendig sein, wenn Sie verhindern möchten, dass ein Benutzer ein Shell-Skript wie die Anmelde-Skripten unterbricht. Listing 9.76 zeigt den interaktiven Einsatz von `trap`.

```
(linux):~$ trap 'echo Signal »SIGHUP« empfangen' SIGHUP
(linux):~$ kill -HUP $$
Signal »SIGHUP« emfpangen
```

Listing 9.76 Signale mit Shell-Mitteln bearbeiten

Die Prozessnummer `-1` hat eine besondere Bedeutung: Das entsprechende Signal wird an alle Prozesse gesendet, für die der Benutzer die entsprechende Berechtigung hat. Um schnell alle seine Prozesse zu beenden, kann man den Befehl `kill -9 -1` verwenden. Als Systemverwalter sollte man das besser nicht versuchen, danach ist das System in einem nicht mehr brauchbaren Zustand.

Als Systemverwalter kann man jedoch schnell die Prozesse eines anderen Benutzers aus dem System entfernen, indem man den Befehl `su - `*`Name`*` -c 'kill -9 -1'` verwendet. Das ist nicht besonders nett und sollte nur in Notfällen getan werden. Alternativ können Sie das Kommando `skill` verwenden, das nach verschiedenen Kriterien Prozesse sucht und Signale verschickt. Mögliche Kriterien sind unter anderem Benutzernamen, Terminal- oder Programmname. Eine nützliche Option ist `-i` (für »interactive«), bei der vor dem Senden des Signals nach Erlaubnis gefragt wird.

9.10.2 Jobs und Jobcontrol

Viele Shells stellen unter modernen Unix-Systemen, die über *Jobcontrol* verfügen, verschiedene Funktionen bereit, um mit Jobs komfortabel umgehen zu können. Unter Unix ist ein Job ein Prozess, der im Hintergrund gestartet wird. Unter Job-

control versteht man die Funktionen, mit denen Jobs gesteuert und beeinflusst werden. Diese Begriffe haben nichts mit der von Großrechnern bekannten »Job Control Language« (JCL) zu tun.

Programme können mit einem kaufmännischen Und (`&`) in den Hintergrund geschickt werden. Ausgaben erscheinen jedoch auf dem Terminal, von dem aus das Programm gestartet wurde. Erwartet das Programm eine Eingabe, so wird das Programm angehalten, das System gibt je nach Shell und Konfiguration eine entsprechende Meldung aus.

Das Programm kann mit `fg` (für »foreground«) wieder in den Vordergrund geholt werden. Haben Sie mehrere Jobs gestartet (eine Liste erhalten Sie mit `jobs`, siehe auch `help jobs` in der `bash`), so können Sie mit `%`*`Jobnummer`* einen Job auswählen. Die Zeichenfolge `%%` steht dabei für den zuletzt gestarteten Job und `%`*`Name`* für das im Hintergrund gestartete Programm mit diesem Namen. Diese Zeichenfolgen können auch bei der Prozessverwaltung mittels `kill` verwendet werden.

Haben Sie ein Programm gestartet, das Sie später im Hintergrund fortsetzen möchten, so unterbrechen Sie das Programm mit Strg-Z (siehe auch `stty -a`, »Suspend-Character«). Anschließend können Sie das Programm mit `bg` (für »background«) in den Hintergrund verlagern.

Wollen Sie größere Programme, die länger laufen und viel Speicher benötigen, im Hintergrund abarbeiten lassen, können Sie dem Programm mit `nice` eine niedrigere Priorität zuweisen. Damit wird das interaktive Arbeiten anderer Benutzer wesentlich weniger beeinträchtigt.

Je höher der `nice`-Wert ist, desto »netter« sind Sie zu anderen Benutzern. Der Benutzer `root` kann als einziger auch einen negativen Nice-Wert vorgeben. Diese Prozesse werden dann beim Scheduling bevorzugt. Haben Sie den Aufruf mit `nice` vergessen, so können Sie mit dem Programm `renice` oder `top` die Priorität auch später noch verändern (nur der Systemadministrator darf dabei die Priorität erhöhen).

Hintergrundprozesse erhalten, wenn ein Benutzer der `bash` sich ausloggt, das Signal `SIGHUP`. Das führt in der Regel zum Programmende, was oft nicht gewünscht ist. Für diese Fälle muss das Programm mit `nohup` gestartet werden. Das ist bei Jobs in der `tcsh` stets der Fall. Hat man das `nohup` vergessen, hilft der Befehl `disown`, mit dem das Programm intern von der Verteilung der entsprechenden Signale ausgeschlossen wird.

Beispiele für Programme, die lange laufen und oft im Hintergrund gestartet werden, sind `ftp`-Übertragungen von großen Datenmengen, Simulationsprogramme oder andere CPU-aufwändige Programme und X-Clients. In allen Fällen hat man den Vorteil, dass man sich ausloggen kann und die Prozesse weiterlaufen oder dass man zumindest die Shell weiter verwenden kann.

9.10.3 Von Programmen benutzte Dateien

In verschiedenen Situationen muss man feststellen, welche Programme auf eine bestimmte Datei oder ein Dateisystem zugreifen. Unter Linux kennt man dafür den Befehl `fuser`, für Linux und viele andere Unix-Systeme ist das Programm `lsof` verfügbar.

Programme wie `lsof` zeigen nicht nur die Prozesse und die von diesen benutzten Dateien an, sondern auch Sockets (für Netzwerkverbindungen). Immer, wenn Sie die Meldung »Text file busy« oder »Device is busy« erhalten, können Sie mit `lsof` eine Liste der Prozesse erstellen, die diese Datei oder dieses Dateisystem verwenden und geeignete Maßnahmen ergreifen.

`lsof` hat eine Reihe von Parametern, mit denen die Daten der Prozesse bzw. Dateien selektiert werden, und Parameter, die die Listaufbereitung steuern. Damit ist es möglich, für spezielle Anwendungen die Ausgabe so aufzubereiten, dass sie einfach mit Skripten verarbeitet werden kann.

Das Programm `lslk` zeigt Sperren zu Dateien (so genannte Locks) an. Wenn man ein Problem mit Dateisperren vermutet, dann kann dieses Programm eine echte Hilfe sein. Das Programm `lslk` erhalten Sie wie `lsof` von *ftp://vic.cc.purdue.edu/pub/tools/unix/lslk*.

Ein weiteres nützliches Programm ist `socklist`, das eine Liste der aktiven Netzwerk-Sockets anzeigt und die Prozessnummer des betreffenden Programms.

9.11 Small is Beautiful – auch heute noch?

Die Idee der vielen kleinen Tools, die auf sehr einfachem Weg miteinander kommunizieren können, hat einen Charme, dem man sich nur schwer entziehen kann. Durch die sehr gute Modularisierung konnte beispielsweise das GNU-Projekt über mehrere Jahre hinweg Unix neu implementieren und hatte stets etwas vorzuzeigen.

Bei den früher üblichen Rechnern war an grafische Oberflächen oder integrierte Pakete kaum zu denken, heute sieht das aber anders aus. Nach und nach zeigt sich, dass nach der Kommunikation mittels Pipes etwas Neues, Leistungsfähigeres benötigt wird. Was wird das sein? Auf der einen Seite wird sehr häufig XML verwendet, um Text mit Struktur zu versehen. Der Nachteil bei dieser Repräsentation ist, dass die Daten immer noch serialisiert und beim Empfänger wieder in interne Datenstrukturen transformiert werden müssen. Für den Dateitransfer keine schlechte Idee, aber nicht unbedingt das Mittel der Wahl bei der Interprozesskommunikation.

Andere Systeme bieten Komponenten, die in eigene Programme eingebunden werden können, oder Schnittstellen wie DCOM oder CORBA, um auch über Netzwerke auf Objekte zugreifen zu können. Mit den Desktop-Umgebungen KDE und GNOME werden vergleichbare Schnittstellen (KParts und Bonobo) auch unter Unix implementiert. Die Frage ist jedoch, ob die Benutzung so einfach werden kann, dass eventuell die kleinen Werkzeuge, die wir in diesem Kapitel kennen gelernt haben, diese Funktionalität irgendwann verwenden können oder bereitstellen werden.

In jedem Fall ist der große Vorteil der »Werkzeug-Philosophie« eine Schwäche von Unix: Es funktioniert sehr lange sehr gut, so dass der Druck zu neuen Entwicklungen und dem praktischen Einsatz davon lange nicht spürbar war. Andere Systeme wie Windows scheinen hier einen gewissen Vorsprung zu haben.

10 Werkzeuge (nicht nur) für Programmierer

10.1 ... sondern auch für Anwender und Systemverwalter

Die heute verbreiteten Linux-Distributionen enthalten eine ganze Reihe von Programmen und Paketen, aus denen der Anwender auswählen kann. Daher erscheint es zunächst nicht sinnvoll, diese oder andere Programme selbst zu übersetzen. Es sind aber nicht alle Programme bereits als Binärversionen für Linux erhältlich und bei Fehlerkorrekturen sind oft nur die Änderungen des Quelltextes verfügbar. Dieses Kapitel stellt die Möglichkeiten vor, die der Anwender bei der Installation und Entwicklung von Programmen hat. Diese Möglichkeiten sind nicht nur für Programmierer interessant, sondern auch für Anwender, die häufig wiederkehrende Aufgaben zu erledigen haben.

Für Linux sind eine Reihe von Programmiersprachen verfügbar. Am weitesten verbreitet ist C, da der Kernel weitgehend in dieser Sprache geschrieben ist. Die Installation des C-Compilers ist unter Linux beinahe ein Muss, da nur so ein an die eigene Hardware angepasster Kernel erstellt werden kann. Außerdem sind die meisten frei verfügbaren Unix-Programme in C implementiert.

Es sind viele andere Programmiersprachen verfügbar: C++, Objective-C, Pascal (GNU-Pascal oder Pascal-to-C), FORTRAN (GNU-FORTRAN oder FORTRAN-to-C), Lisp, Skript-Sprachen wie Perl, Tcl und Python. Oft sind Programme in einer dieser Sprachen implementiert, so dass dann der entsprechende Interpreter oder Compiler installiert sein muss, wenn diese Programme zum Einsatz kommen sollen. Das ist einer der Gründe, weshalb die Programme des GNU-Projekts in der Regel in der Sprache C implementiert wurden.

10.2 Das Programm make

Das Programm `make` dient zunächst zur Kompilation von großen Programmen, die aus vielen Modulen bestehen. Nach Änderungen an den Quelltexten werden nur die Programmteile neu übersetzt, die verändert wurden. Das spart im Vergleich zur Komplettübersetzung oft eine Menge Zeit. Daher werden die Quellen fast jedes Programms mit einem `Makefile` ausgeliefert, das die Übersetzung steuert.

GNU-make verwendet folgende Dateien als Steuerdateien für den Übersetzungsprozess, sofern diese Datei existiert: GNUMakefile, Makefile bzw. makefile. Andere Dateien können Sie mit der Option -f angeben. Wenn Sie in Ihrem Makefile spezielle Funktionen von GNU-make benutzen, so sollten Sie diese Datei besser GNUMakefile nennen.

Beim Aufruf von make wird das in der Datei Makefile angegebene Default-Ziel (Target) erzeugt. Wird ein zusätzlicher Parameter angegeben, so wird dieses Target erzeugt. Zur Erläuterung betrachten wir hier ein einfaches Beispiel (siehe Listing 10.1).

```
# Ein einfaches Makefile für prune
all: prune

prune: prune.c
        gcc -O -s prune.c -o prune

install: prune
        install -m 755 prune /usr/local/sbin/prune
        echo "edit /etc/prune.conf to suit your system"
```

Listing 10.1 Ein einfaches Makefile

Der Befehl make erzeugt das Default-Target all. Das Default-Target ist das erste Target im Makefile. Targets stehen immer in der ersten Spalte und werden mit einem Doppelpunkt beendet. Es besteht eine Abhängigkeit von der Datei bzw. dem Target prune. Abhängigkeiten stehen auf der rechten Seite des Doppelpunkts. Kommentare in einem Makefile beginnen mit einer Raute # und enden am Zeilenende.

Im Target prune, das von der Datei prune.c abhängig ist, wird der C-Compiler gcc aufgerufen, wenn die Datei prune.c neuer ist als die Datei prune. Für diese einfache Funktion kennt make bereits eine so genannte Standardregel, die angewendet würde, wenn hier kein Befehl angegeben wäre. Die Liste aller Standardregeln kann mit make -p ausgegeben werden. Befehle werden nach einem Tabulatorzeichen eingetragen. Werden anstelle des Tabulatorzeichens Leerzeichen verwendet, so geben manche make-Versionen eine Fehlermeldung aus.

Der Befehl make install führt die Befehle aus, die im Target install angegeben sind. Zunächst wird jedoch überprüft, ob das Programm prune neu erstellt werden muss. Der Befehl install kopiert das Programm an die vom Programmierer festgelegte Stelle im Verzeichnisbaum und kann dabei entsprechende Berechtigungen setzen.

Listing 10.2 zeigt ein Makefile, in dem sowohl Variablen verwendet werden als auch die Standardregeln zur Übersetzung anstelle des Aufrufs von gcc. Dieses Makefile hat nun schon eine gewisse Ähnlichkeit mit den üblicherweise verwendeten Makefiles.

```
CFLAGS=-O2
LDFLAGS=-s
INSTALLDIR=/usr/local/sbin

all: prune

install: prune
        install -m 755 prune $(INSTALLDIR)/prune
        echo "edit /etc/prune.conf to suit your system"

prune: prune.o

clean:
        rm -f prune.o prune
```

Listing 10.2 Ein etwas komplizierteres `Makefile`

Im ersten Abschnitt werden einige Variablen gesetzt, die später dann in den Regeln oder Abhängigkeiten verwendet werden können. Mit der Variable `CFLAGS` werden die Optionen für den C-Compiler festgelegt. Hier ist es die Option `-O2` für die Optimierung. Die Variable `LDFLAGS` enthält die Optionen für den Linker. Hier wird die Option `-s` verwendet, die alle Symbolinformationen zur Fehlersuche aus dem Programm entfernt. Sie können diese Variablen auch beim Aufruf des `make`-Programms als Parameter mitgeben, beispielsweise als `make CFLAGS=-O`.

In diesem `Makefile` wird das Programm `prune` in zwei Schritten erzeugt: Zunächst wird der C-Quelltext in das so genannte Objektformat (Dateiname `prune.o`) übersetzt. Anschließend wird diese Objektdatei mit der Standard-C-Bibliothek zusammengelinkt, um ein lauffähiges Programm zu erzeugen. Beides sind Standardregeln, die Definition aller Standardregeln können Sie sich mit dem Befehl `make -p` anzeigen lassen.

Im `install`-Target wird das Programm in das Verzeichnis `INSTALLDIR` kopiert. Neu wurde das Target `clean` aufgenommen, das bereits erzeugte Programme oder Objektdateien löscht. Der nächste `make`-Aufruf muss dann das Programm komplett neu übersetzen, was zum Beispiel bei Konfigurationsänderungen sinnvoll sein kann. Ein weiteres, häufig verwendetes Target ist `dist`, das in der Regel eine `tar`-Datei mit dem Quelltext erstellt. Um ein anderes Target als das Default-Target zu erzeugen bzw. die dort hinterlegten Befehle auszuführen, ruft man `make` mit dem Namen des entsprechenden Targets als Parameter auf, beispielsweise mit `make clean`.

Bei größeren Programmen müssen oft einzelne Parameter vor der Übersetzung angepasst werden. Das sind z. B. die Namen von Konfigurations- oder Lock-Dateien oder die Installationsverzeichnisse. In der Regel findet man diese Variablen am Beginn des Makefiles, in dem auch oft Kommentare zur Bedeutung der Variablen zu finden sind.

Das Programm `make` hat eine Reihe von Optionen, die den Ablauf von `make` beeinflussen können. Wichtig und nützlich ist die Option `-n`, die bewirkt, dass `make` die notwendigen Befehle nicht ausführt, sondern nur ausgibt. So kann man z. B. einfach überprüfen, welche Dateien wohin installiert werden, wenn zunächst der Befehl `make -n install` anstelle von `make install` verwendet wird.

`make` kann mehrere Programme gleichzeitig starten, so dass auf Rechnern mit genügend Speicher die CPU möglichst vollständig ausgelastet werden kann. Die Option `-j` erlaubt die parallele Ausführung mehrerer Programme. Folgt auf diese Option eine Zahl, so wird die Anzahl der gleichzeitig von `make` gestarteten Programme auf diese Zahl limitiert. Ohne diese Zahl wird die Anzahl nicht limitiert. Beim Start von vielen Prozessen geht der »Load«, also die Anzahl der Prozesse, die um Prozessorzeit konkurrieren, in die Höhe. Mit der Option `-l`, gefolgt von einer Zahl, werden keine neuen Prozesse gestartet, wenn noch Prozesse laufen und der Load größer ist als diese Zahl. Wird keine Zahl angegeben, so wird der Load nicht beachtet. Wenn man also parallele `make`-Prozesse startet, muss man auf den belegten Hauptspeicher und die Prozessorauslastung achten, um Geschwindigkeitsvorteile zu erreichen. Besonders im Vorteil sind hier natürlich Mehrprozessorrechner.

`make` ist nicht nur für Programmierer interessant. Immer, wenn nach der Änderung einer oder mehrerer Dateien bestimmte Befehle ausgeführt werden müssen, kann `make` diese Arbeiten sehr vereinfachen. `make` wird beim Network Information Service (siehe Kapitel 23, »Network Information Service«) verwendet, um die Maps aktuell aus den Systemdateien zu erzeugen. Andere vorstellbare Einsatzgebiete sind die Generierung von Indizes auf Web-Servern oder die maschinelle Erstellung von Grafiken aus gespeicherten Daten. Bei einem korrekten `Makefile` sind die generierten Dateien auf dem aktuellen Stand, ohne dass der Anwender sich mit den dafür notwendigen Befehlen auseinander setzen muss.

```
report.dvi: report.tex bild1.eps bild2.eps
        latex report

%.eps:%.fig
        fig2dev -L ps $< > $@
```

Listing 10.3 Ein Makefile für LaTeX

Das Listing 10.3 zeigt ein Beispiel für die Verwendung von `make` zur Erzeugung der DVI-Datei `report.dvi`. Diese Datei ist von dem Quelltext `report.tex` und den Bildern `bild1.eps` und `bild2.eps` abhängig. Diese Bilder wurden mit dem Programm `xfig` erzeugt und dann in das eps-Format (Encapsulated PostScript) umgewandelt. Die Verwendung von `make` automatisiert die Umwandlung der Bilder und sorgt dafür, dass in die DVI-Datei stets die aktuellen Bilder eingebunden sind.

Dieses Beispiel zeigt auch einige weitere Funktionen von make. Das Symbol $< repräsentiert die Dateien, die als Eingabe benötigt werden, und das Symbol $@ wird zur Laufzeit durch den Namen der zu erzeugenden Datei ersetzt. Diese Symbole werden in einer Regel verwendet, die allgemein die Abhängigkeiten zwischen Bildern (%.fig) und deren eps-Format (%.eps) beschreibt. Damit muss nicht für jedes Bild eine eigene Regel angegeben werden.

Diese Informationen sollen Ihnen helfen, fremde Makefiles zu verstehen. Bevor Sie beginnen, eigene Makefiles zu entwickeln, lesen Sie das Handbuch zu GNU make, das im Info-Format verteilt wird. Dort wird ausführlich die Verwendung von make und die Erstellung von Makefiles dokumentiert. Wenn Sie häufig Makefiles editieren, so können Sie den makefile-mode des Editors Emacs verwenden, da er eine Reihe von nützlichen Funktionen bereitstellt.

Für die Programme des GNU-Projekts gibt es ein Dokument (standards.texi), das neben der C-Programmierung und dem dort zu praktizierenden Stil auch die wichtigsten Funktionen für ein Makefile vorschreibt. Man muss nicht mit jeder Vorschrift dort einverstanden sein, das Dokument enthält aber eine Reihe von möglicherweise nützlichen Hinweisen.

Neben GNU-make gibt es noch weitere make-Programme. pmake stammt aus dem BSD-Umfeld und bietet im Vergleich zum Standard-make ebenfalls einige nützliche Erweiterungen, die von vielen BSD-Programmen verwendet werden. Leider sind GNU-make und pmake nicht kompatibel, so dass man für fremde Programme manchmal das passende make verwenden muss. Interessante Erweiterungen sind make-Programme, die die Übersetzung auf mehrere Rechner verteilen, so dass man vorhandene Ressourcen möglicherweise besser ausnutzen oder die insgesamt notwendige Zeit verringern kann.

10.2.1 Das Programm imake

Das Programm imake ist Bestandteil des X-Window-Systems und wird daher oft verwendet, um X-Programme zu übersetzen. Dabei wird für das entsprechende Programm ein Imakefile erstellt, in dem die zu kompilierenden Programme und Bibliotheken beschrieben werden. Anhand von systemspezifischen Konfigurationsdateien wird aus dem Imakefile ein Makefile erzeugt, das dann für die Übersetzung des Programms verwendet wird.

In der Regel wird mit dem Befehl xmkmf das systemspezifische Makefile erzeugt. Sind die Quellen des Programms über mehrere Verzeichnisse verteilt, so können mit dem Befehl xmkmf -a die Makefiles auch in den Unterverzeichnissen automatisch erzeugt werden. Dazu startet xmkmf intern den Befehl make Makefiles und anschließend make includes und make depend.

```
SYS_LIBRARIES = -lXext -lX11
 EXTRA_INCLUDES = -I/net/packages/X11R5/contrib/lib/xpm
LOCAL_LIBRARIES = -L/usr/X11/lib -lXpm
           SRCS = xpinguin.c
           OBJS = xpinguin.o

ComplexProgramTarget(xpinguin)
```

Listing 10.4 Ein einfaches `Imakefile`

Das Listing 10.4 zeigt ein einfaches `Imakefile` für das Programm `xpinguin`. Im ersten Teil werden einige Variablen gesetzt, die das Programm und die notwendige Systemumgebung beschreiben. In der letzten Zeile wird festgelegt, dass das generierte `Makefile` das Programm `xpinguin` erzeugen soll. Beim Aufruf von `xmkmf` wird aus dieser Datei mit Hilfe des C-Präprozessors `cpp` ein geeignetes `Makefile` generiert. Durch die Verwendung des `cpp` ist es nicht einfach, Kommentare im `Makefile` unterzubringen, auch ist das Verfahren nicht so flexibel wie die Verwendung von `autoconf`.

Die Konfigurationsdateien für das Programm `imake` finden Sie im Verzeichnis `/usr/X11R6/lib/X11/config`. Die Datei `linux.cf` enthält alle notwendigen Anpassungen für Linux. Eigene Anpassungen können in der Datei `site.def` vorgenommen werden; dies ist in der Regel jedoch nicht notwendig. Eine komplette Einführung in die Verwendung und Konfiguration von `imake` finden Sie z. B. in [DuBois1996].

10.2.2 Andere Programme als Ersatz für make

`make` ist schon lange im Einsatz, sehr stabil und den meisten Unix-Programmierern bekannt. Dennoch gibt es einige Probleme, so ist die Syntax der Makefiles gewöhnungsbedürftig und das Erstellen derselben nicht einfach. Außerdem ist es problematisch, portable Makefiles zu schreiben, die auf verschiedenen Systemen eingesetzt werden können.

Das GNU-Projekt verwendet zum Erzeugen der Makefiles eine Reihe von Tools: `autoconf` erzeugt ein `configure`-Skript, das vor der Übersetzung ein systemspezifisches Makefile erzeugt. `automake` hilft dem Programmierer bei der Erstellung des Makefile-Template. Damit sind GNU-Programme in der Regel sehr portabel, erfordern aber einigen Aufwand.

Aus dem Apache/Java-Umfeld kommt `ant` als ein Ersatz für `make` (*http://jakarta.apache.org/ant/*). Noch wird es außerhalb von Java praktisch nicht eingesetzt, aber das könnte sich ändern. Bei `ant` werden die Regeln im XML-Format spezifiziert und können plattformunabhängig sein. Das scheint mir einer der größten Vorteile von `ant`.

10.2.3 Installation von Programmen

Eine Tätigkeit, mit der ein Unix-Systemadministrator immer wieder konfrontiert wird, ist die Installation von neuen Programmen oder neuen Versionen bereits vorhandener Programme. Praktisch alle unter Linux verwendeten Programme sind frei, d. h., der Quellcode ist verfügbar. Da diese Software meist auch unter anderen Unix-Systemen eingesetzt wird, muss sie auf verschiedenen Plattformen übersetzt werden können. Dazu müssen die Programme portabel zwischen verschiedenen Systemen sein.

Binärversionen von Programmen installieren

Unter Linux und Unix erhält man viele Programme mit Quellcode. Alles begann mit der Verbreitung von Unix an Universitäten und der offenen Weiterentwicklung z. B. in Berkeley. Ein weiterer Grund ist, dass im Laufe der Zeit Unix auf viele verschiedene Hardware-Plattformen portiert wurde und der Austausch von Binärprogrammen technisch nicht möglich war. Der ursprüngliche Autor eines Programms hat in der Regel nur Zugang zu wenigen verschiedenen Plattformen und ist daher häufig nicht in der Lage, entsprechende Binärversionen zu erstellen.

Unter Windows erhält man auch Freeware-Programme in der Regel nur als fertiges, ausführbares Programm, zusammen mit einem entsprechenden Installationsprogramm. Bei derart verbreiteten und homogenen Systemen ist das für viele Programmierer ausreichend und einfach genug. Unter Linux ist dies nicht so ausgeprägt, man erhält jedoch in letzter Zeit immer häufiger auch vorkompilierte Programme.

Wenn die Distribution das gewünschte Programm mitbringt, so kann man es einfach mit den entsprechenden Tools der Distribution (z. B. `dpkg` für Debian oder `rpm` für Red Hat) installieren. Wenn die Distribution das entsprechende Programm nicht enthält, dann kann es durchaus sein, dass ein anderer Anwender das Programm bereits übersetzt hat und anderen Anwendern zur Verfügung stellt. Oftmals findet man diese Programme im Contrib-Bereich der entsprechenden Distribution. Ich bevorzuge es dabei, den Quellcode des Pakets zu holen, ihn dann selbst zu übersetzen und mit Hilfe der Paketverwaltung zu installieren.

Es ist vorteilhaft, Software selbst zu übersetzen, damit die Programme vollständig an das eigene System angepasst sind. Außerdem findet man im Quellcode in der Regel etwas mehr Dokumentation als in einem Binärpaket. Die Installation mit Hilfe der Paketverwaltung sorgt dafür, dass ein Upgrade oder Löschen des Pakets einfach und sicher durchgeführt werden kann.

Ist ein Programm nicht im passenden Format für die Distribution verfügbar, dann kann es trotzdem sein, dass eine Binärversion als `tar`-Datei zur Verfügung steht. Diese kann mit `tar zxvf` *`tar-Datei`* ausgepackt werden. Dieses Verfahren hat eine Reihe von Nachteilen: Das Programm wird nicht mit Hilfe der Paketver-

waltung installiert und kann daher in der Regel nicht mit einem Befehl gelöscht werden. Ob das Programm, z. B. durch die Abhängigkeit von bestimmten Bibliotheken, stabil läuft, kann nur ein ausführlicher Test zeigen, und möglicherweise vertragen sich einige Einstellungen mit anderen im System installierten Programmen nicht.

Wenn Sie genügend Plattenplatz haben und es nicht um wirklich komplexe und große Programmpakete geht, dann sollten Sie das Programm, wie in den nächsten Abschnitten beschrieben, selbst übersetzen. Neben der guten Anpassung des Programms an das eigene System lernen Sie außerdem Ihr System besser kennen.

Die Installation von GNU-Programmen

Das Ziel der Free Software Foundation (FSF) ist die Erstellung eines freien, Unix-ähnlichen Betriebssystems (The Hurd). Alle wesentlichen Unix-Programme sind bereits entwickelt und werden rund um die Welt unter den unterschiedlichsten Unix-Versionen eingesetzt. Viele der unter Linux eingesetzten Programme sind die GNU-Versionen der bekannten Unix-Dienstprogramme. Daher wird Linux oft als GNU-System auf Linux-Basis bezeichnet. Ohne diese Programme wäre Linux nicht das, was es heute ist.

Die GNU-Programme sind sehr portabel und leistungsfähig, so dass sie auf vielen Rechnern zusätzlich zu den Standard-Unix-Dienstprogrammen installiert sind. Die Portabilität wird dadurch erreicht, dass vor der Übersetzung des Programms das System untersucht und das Programm entsprechend angepasst wird. Die Untersuchung des Systems wird mit dem Skript `configure` gestartet. Das GNU-Projekt hat einige Standards zur Erstellung von Programmen und `Makefiles` gesetzt, an die sich fast alle GNU-Programme halten. Den GNU-Coding-Standard finden Sie auf jedem GNU-Mirror in der Datei `standards.texi`.

Das von `autoconf` erzeugte `configure`-Skript untersucht das System und übersetzt eine Reihe von Testprogrammen. Anschließend wird ein `Makefile` erstellt, das die Übersetzung des Programms steuert. Zunächst stelle ich aber einige Beispiele für die Konfiguration von GNU-Programmen vor. Wir beginnen mit der Konfiguration von `gzip`, dem GNU-Ersatz für das Unix-`compress` (siehe Listing 10.5). `gzip` packt wesentlich besser und hat den Vorteil, dass für die Verwendung keine Lizenz aufgrund bestehender Patente notwendig ist.

```
(linux):~$ tar zxf /tmp/gzip-1.2.4.tar.gz
(linux):~$ cd gzip-1.2.4
(linux):~/gzip-1.2.4$ ./configure
```

Listing 10.5 Auspacken und Konfigurieren eines GNU-Programms

Das Skript gibt eine Reihe von Meldungen aus, mit denen der Ablauf der Konfiguration verfolgt werden kann. Am Ende des configure-Laufs wird ein Makefile erstellt, mit dem das Programm übersetzt werden kann. Die Anzahl der ausgegebenen Meldungen kann mit den Optionen --verbose bzw. --quiet beeinflusst werden.

GNU-Programme werden normalerweise in das Verzeichnis /usr/local installiert. Unter Linux werden die GNU-Programme oft wie die Standardprogramme in das Verzeichnis /usr installiert. Dies wird mit der Option --prefix erreicht (siehe Listing 10.6). Mit dem Befehl ./configure --help erhalten Sie eine Liste mit allen unterstützten Optionen. Außerdem finden Sie die möglichen Optionen (z. B. für den Pfad zu den X-Bibliotheken) mit weiteren Erläuterungen in der Datei INSTALL oder README.

```
(linux):~/gzip-1.2.4$ ./configure --prefix=/usr
```

Listing 10.6 Die Option --prefix für den configure-Aufruf

Anschließend kann das Programm mit der Eingabe von make kompiliert und mit make install installiert werden. GNU-Programme werden normalerweise so übersetzt, dass ein Debugger benutzt werden kann, um Fehler zu finden. Für den durchschnittlichen Systemadministrator ist dies unnötig und belegt viel Platz auf der Festplatte. Es existieren zwei Möglichkeiten, dies abzustellen. Zum einen können einige Umgebungsvariablen gesetzt werden, das kann z. B. in den Initialisierungsdateien der Shell eingetragen werden (siehe Listing 10.7).

```
(linux):~/gzip-1.2.4$ setenv CFLAGS=-O2
(linux):~/gzip-1.2.4$ setenv LDFLAGS=-s
```

Listing 10.7 Zusätzliche Flags für den C-Compiler und Linker

Die Variablen können zum anderen auch beim Aufruf des make-Programms zusätzlich zum entsprechenden Target angegeben werden (siehe Listing 10.8). Für alle GNU-Programme können Sie derartige Einstellungen in der Datei /etc/config.site hinterlegen. Diese Datei wird von allen configure-Skripten eingelesen und kann entsprechende Variablen enthalten.

```
(linux):~/gzip-1.2.4$ make CFLAGS=-O2 LDFLAGS=-s
(linux):~/gzip-1.2.4$ su
(linux):~/gzip-1.2.4# make CFLAGS=-O2 LDFLAGS=-s install
```

Listing 10.8 Übergabe von speziellen Flags an make

Nun ist das Programm installiert. Je nach Shell muss eventuell die Hash-Tabelle neu aufgebaut werden, in der die Shell den Pfad zu Programmen speichert. Bei der tcsh erfolgt das mit dem Befehl rehash, bei der bash mit hash -r.

Einige GNU-Pakete, wie z. B. der GNU-C-Compiler (GCC) oder der Editor Emacs, können mit einer Reihe von weiteren Optionen konfiguriert werden. Genauere Informationen über die möglichen und notwendigen Optionen des `configure`-Skripts finden Sie in der Datei `INSTALL`, die in jedem Paket enthalten ist.

Das Standardformat für die Dokumentation der GNU-Programme ist Texinfo. Damit können aus einem Quelltext sowohl eine gedruckte Dokumentation als auch eine Online-Dokumentation erstellt werden. Oft werden aber zur Referenz auch Manpages installiert, die nur in aller Kürze die Optionen der Programme erklären.

Viele GNU-Programme können mit dem Befehl `make uninstall` wieder aus dem System entfernt werden. Voraussetzung dafür ist, dass der Quellcode des installierten Pakets noch zur Verfügung steht bzw. die entsprechenden `Makefiles` noch vorhanden sind oder mit dem passenden `configure`-Aufruf wieder erstellt werden können. Im Vergleich zu einer ausgefeilten Paketverwaltung, über die heute praktisch alle Linux-Distributionen verfügen, ist dieses Verfahren nicht besonders komfortabel, aber es ist wesentlich besser als nichts.

Das GNU-System zur Generierung der Binärprogramme ist sehr ausgefeilt, leistungsfähig und stabil. Wenn Sie selber unter Unix Programme entwickeln, sollten Sie es sich einmal in Ruhe ansehen. Das System besteht aus den Programmen `autoconf` zur Erzeugung des `configure`-Skripts, `automake` zur Erzeugung der Datei `Makefile.in` und `libtool` zum Erstellen von Shared Libraries. Mit diesem Tools ist es oft sehr einfach, ein Programm auf ein anderes Unix-System zu portieren.

GNU-Programme sind nicht abhängig vom verwendeten Betriebssystem oder der Version, sondern von so genannten »Features«. Das Programm `configure` prüft, welche Features das System unterstützt. Entsprechend wird die Datei `config.h` erstellt. Der Quellcode ist entsprechend aufbereitet, entweder wird das Feature benutzt, neu implementiert oder bestimmte Funktionen des Programms werden nicht übersetzt. Die Dokumentation zu `autoconf` beschreibt das System recht ausführlich.

Die Installation von anderen Programmen

Die Installation von Programmen, die nicht zum GNU-Projekt gehören, verläuft zum Teil sehr unterschiedlich. Eine genaue Beschreibung der Installation finden Sie oft in den Dateien `README` oder `INSTALL`. Die Lektüre dieser Dateien ist vor der Installation unbedingt notwendig. Oft existiert auch eine Datei, die Tipps und Tricks für einzelne Betriebssysteme enthält.

Eine Möglichkeit, Programme unter verschiedenen Unix-Systemen zu übersetzen, ist die Erstellung eines `Makefile` für jedes Betriebssystem. Das passende `Makefile` ist dann zu kopieren und eventuell anzupassen. In anderen Program-

men ist ein großes Makefile enthalten, in dem die entsprechenden Konfigurationen für die verschiedenen Systeme eingetragen sind. Hier ist dann mit einem Editor die richtige Einstellung zu aktivieren.

In vielen Fällen sind die Programme bereits für die Übersetzung und Installation unter Linux vorbereitet. Manchmal ist dies nicht der Fall, oft sind für Linux auch keine Änderungen notwendig. In der Regel können die Einstellungen für ein POSIX-System verwendet werden. Kennt das Makefile nur System-V- und BSD-basierte Systeme, so sollten Sie zunächst die System-V-Einstellungen probieren. Damit sollten sich bereits viele der heute frei verfügbaren Programme übersetzen lassen. Müssen Sie größere Änderungen am Quellcode oder den Makefiles vornehmen, so senden Sie diese an den Programmierer. Damit können in einer späteren Version andere von Ihren Erfahrungen profitieren.

In letzter Zeit werden relativ viele Programme auf die Verwendung von GNU-configure umgestellt. Zunächst ist das ein relativ hoher Aufwand, dadurch lässt sich jedoch die Portabilität deutlich verbessern.

Den Überblick behalten

Solange Sie nur Programme mit der Pakteverwaltung Ihrer Distribution installieren, können Sie dieses Tool auch zum Suchen, Updaten oder Löschen von Paketen verwenden. Sobald man selbst Programme installiert, entweder weil die Distribution diese nicht enthält oder man eine neuere (oder ältere) Version benötigt, dann steht man vor einem Dilemma: Wohin installieren und wie den Überblick behalten?

Als Konvention hat sich eingebürgert, dass der Systemverwalter Programme in /usr/local installiert und die Distribution dort keine Änderungen vornimmt. Das geht so lange gut, bis man etliche Programme dort installiert hat und nicht mehr weiß, wo welche Datei herkommt. Da würde nun die Paketverwaltung der Distribution helfen, aber es bedeutet zumindest Arbeit, lokal installierte Programme in ein Paket zu packen.

Abhilfe kann der Einsatz von stow (*http://www.gnu.org/software/stow/stow.html*) schaffen. Programme werden übersetzt, aber nicht mit make install installiert, sondern unter /usr/local werden alle benötigten Links angelegt. Bei einem Update wird der Quellcode in einem neuen Verzeichnis ausgepackt, neu kompiliert und die Links werden neu erstellt. Ein Fallback auf die alte Version ist damit schnell möglich. Auch das Löschen eines Programmpakets ist einfach: das Übersetzungsverzeichnis löschen und alle »dangling symlinks« (symbolische Links, deren Ziel nicht existiert) entfernen.

10.3 Editoren

Unix stellt eine sehr leistungsfähige Umgebung für Programmierer zur Verfügung. Diese besteht zum einen aus dem Kernel, der die Verwaltung der CPU, der Dateisysteme und der Hardware übernimmt. Der Kernel stellt dem Programmierer fast alle Funktionen zur Verfügung, die benötigt werden. Aus Gründen der Portabilität und der einfacheren Programmierung stellt die `libc` eine etwas höhere Abstraktionsebene dar. Programmierer von Anwendungen benutzen also weitgehend die Funktionen der `libc`.

Ein weiterer Bestandteil eines Unix-Systems sind die vielen Dienstprogramme. Diese Programme sind jeweils für eine spezielle Funktion entwickelt worden. Durch die Verwendung verschiedener dieser Programme können mit relativ geringem Aufwand eine Reihe von Funktionen ausgeführt werden. Das Bindeglied zwischen den Programmen sind entweder Pipes oder Funktionen der Shell.

Programmierer benötigen neben diesen Dientsprogrammen und Compilern auch einen Editor, um Programme, Skripten, Konfigurationsdateien und Dokumentationen zu editieren. Dabei ist es sinnvoll, wenn der Editor leistungsfähig und in die Unix-Umgebung integriert ist. Auf jedem Unix-System ist der Editor `vi` (oder ein Clone) vorhanden. Dieser Editor ist relativ klein, aber sehr leistungsfähig und gut in die Unix-Umgebung integriert. Einige Tipps und die wichtigsten Befehle zu diesem Editor werden in Anhang A, »Der Standardeditor `vi`« vorgestellt. Ich gehe hier zunächst auf den Editor Emacs ein. Anschließend werden weitere Editoren kurz vorgestellt, die für Programmierer nützlich sein könnten.

10.3.1 Der Editor Emacs

Ein weit verbreiteter Editor ist (GNU-)Emacs. Aufgrund der Leistungsfähigkeit und der Möglichkeiten, die dieser Editor aufweist, wird er in einem eigenen Kapitel beschrieben (Kapitel 6, »Der Editor Emacs«). Hier werden nun einige spezielle Funktionen vorgestellt, die besonders für Programmierer interessant sind. Die hier getroffenen Aussagen gelten natürlich genauso auch für den XEmacs.

Zu beinahe jeder Programmiersprache gibt es einen entsprechenden Emacs-Modus. Ist für die gewünschte Programmiersprache kein Modus in der Standardversion von Emacs enthalten, so finden Sie diesen Modus möglicherweise in einem Emacs-Lisp-Archiv (*http://www.elisparchive.net/*). In einem solchen Modus werden Tasten so belegt, dass das Editieren von Quelltexten möglichst einfach wird. Darüber hinaus stehen Funktionen zur Übersetzung und zum Testen von Programmen zur Verfügung.

Besonders nützlich kann die automatische Einrückung sein. Damit sind Verzweigungen und Schleifen beinahe immer richtig eingerückt und manche logische Fehler bereits an der Einrückung zu erkennen. Wird zusätzlich noch ein Syntax-Highlighting durchgeführt, kann man sich recht schnell auch in fremden Quelltexten zurechtfinden.

Um Tipparbeit zu sparen, kann möglicherweise ein `abbrev-mode` verwendet werden. Für viele Modi wurden bereits entsprechende Abkürzungstabellen erstellt, in denen die Schlüsselwörter der jeweiligen Sprache enthalten sind.

Um bei größeren Projekten den Überblick zu behalten, lässt sich mit dem Programm `etags` eine `TAGS`-Datei erzeugen. Diese Datei enthält Querverweise (eine Cross-Referenz) zwischen den Quelltextdateien und den dort implementierten Funktionen. Mit Hilfe dieser Datei können beliebige Funktionen in beliebigen Quelldateien mit nur wenigen Tastendrücken erreicht werden. Eine Übersicht über die Tastenkombinationen finden Sie in Tabelle 10.1. Tags können z. B. bei Projekten in C, Emacs-Lisp oder LaTeX verwendet werden.

Taste	Funktion	Beschreibung
Meta+.	`find-tag`	Tag suchen
Strg+u Meta+.	`find-tag`	Nächstes Vorkommen
	`tags-search`	Dateien mit Tag laden
	`tags-query-replcae`	Suchen und Ersetzen
Meta+,	`tags-loop-continue`	Suche fortsetzen
	`list-tags`	Tags anzeigen
	`tags-apropos`	Suchen

Tabelle 10.1 Tastenkombinationen für Tags

Der Tags-Modus kann auch eine `TAGS`-Datei verwenden, die in einem anderen Verzeichnis gespeichert ist. Diese Datei kann manuell geladen oder aufgrund von Einträgen in der Variablen `tag-table-alist` ausgewählt werden. Mit Hilfe des Tags-Modus erhalten Sie einen guten Überblick über die Implementierung des Emacs. Erzeugen Sie eine `TAGS`-Datei im Emacs-Lisp-Verzeichnis mit dem Befehl `etags *.el`. Diese Datei können Sie mit Meta+x `visit-tags-table` laden und dann verwenden. Mit dem in Listing 10.9 gezeigten Befehl in der Datei `~/.emacs` kann die `TAGS`-Datei automatisch bei jeder Emacs-Lisp-Datei geladen werden.

```
(setq tag-table-alist
   '(("\\.el$" . "/usr/local/share/emacs/19.29/lisp/")))
```

Listing 10.9 `TAGS`-Datei automatisch laden

Nach der Erstellung des Programms muss dieses kompiliert werden. Auch hier bietet Emacs wieder gute Unterstützung beim Aufruf des Compilers. Mit dem Befehl Meta+x compile, der natürlich auch auf eine Tastenkombination wie z. B. Strg+c Strg+c gelegt werden kann, wird der Compiler (genauer: das Kommando, das in der lokalen Variablen compile-command festgelegt ist) gestartet. Sind bei der Übersetzung Fehler aufgetreten, so kann mit der Tastenkombination Strg+c Strg+` zur nächsten Fehlerstelle gesprungen werden.

Ist das Programm dann übersetzt, so kann es ebenfalls unter Emacs getestet werden. Dazu existiert eine Schnittstelle zum GNU-Debugger gdb. In einem Emacs-Fenster wird der Quelltext des Programms angezeigt; der aktuelle Befehl ist dabei markiert. Der Quelltext kann direkt in diesem Buffer verändert werden, wenn dies notwendig sein sollte. In einem zweiten Fenster können alle Befehle des Debuggers verwendet werden. Den Debugger starten Sie mit dem Befehl Meta+x gdb.

Zum Austausch von Änderungen mit anderen Programmierern oder Anwendern wird das Programm diff zur Erstellung eines Patch verwendet, der dann mit dem Programm patch in den Quelltext eingebaut wird. Beim Abgleich zwischen zwei oder drei Versionen eines Programms bietet Emacs mit den Modi ediff und emerge ebenfalls Unterstützung an. Mit dem ediff-Mode können bequem einzelne Änderungen angesehen und eventuell übernommen werden. Unter X werden die Unterschiede dabei farblich markiert.

Dies alles ist nur ein sehr kleiner Ausschnitt aus den Möglichkeiten, die Emacs einem Programmierer oder erfahrenen Anwender bietet. Sich diese Funktionen zu erarbeiten, ist mitunter nicht einfach und manchmal zeitaufwendig, aber man kann zunächst mit einfachen Editierfunktionen beginnen und je nach Bedarf und Zeit dazulernen. Wenn Sie einen Modus besonders häufig verwenden (z. B. den sh-mode bei der Entwicklung von Shell-Skripten), dann werfen Sie einen Blick in die Dokumentation und den Quellcode des Modus. Dort finden Sie immer wieder nützliche Funktionen und Einstellungen erklärt.

10.3.2 Der Editor jed

Vielen Anwendern ist emacs zu groß und zu langsam. Der Editor jed ist im Vergleich zu Emacs sehr klein. Die Benutzerschnittstelle lässt sich so eingestellen, dass sie an emacs, WordStar oder EDT erinnert, so dass DOS- oder VAX-Anwender sich eher mit jed zurechtfinden.

Wie Emacs ist jed sowohl auf der Konsole als auch unter X lauffähig, verfügt über Syntax-Highlighting (auch auf der Konsole) und kann mittels der C-ähnlichen Sprache slang erweitert und konfiguriert werden. Es stehen bereits eine Reihe von Modi zur Verfügung, wenn auch nicht so viele wie bei Emacs.

Wenn Sie nach einem kleinen, schnellen und trotzdem leistungsfähigen Editor suchen, dann kann `jed` genau das Richtige sein. Wie Emacs ist er ausführlich dokumentiert und kann recht einfach konfiguriert und erweitert werden. Wenn Sie `jed` im Emacs-kompatiblen Modus verwenden, dann ist es später auch nicht schwer, auf Emacs umzusteigen. Für einen ständigen Wechsel zwischen beiden Editoren reicht die Kompatibilität aber nicht aus, so dass man sich auf Dauer für einen der beiden entscheiden sollte.

Der Editor Nedit

Nedit (*http://www.nedit.org/*) ist ein kleiner, leistungsfähiger Editor, der von vielen Programmierern gerne verwendet wird. Er ist schnell und bietet für viele Programmiersprachen Syntax-Highlighting und verfügt über eine Makro-Sprache. Wenn Ihnen `vi` und Emacs nicht zusagen, probieren Sie ihn einfach mal aus.

10.4 Integrated Development Environments (IDE)

Unter Windows verfügt praktisch jeder Compiler über eine eigene Entwicklungsoberfläche. Diese ist in der Regel gut in den Compiler integriert und versorgt den Programmierer mit den meisten benötigten Funktionen. Nachteilig ist, dass für eine andere Sprache oft eine andere Umgebung und ein anderer Editor und Debugger verwendet wird, so dass der Programmierer sich umgewöhnen muss und Projekte in mehreren Programmiersprachen einen größeren Aufwand verursachen.

Unter Unix verfolgte man bisher eine andere Philosophie: Viele kleine Tools ergeben zusammen die Oberfläche. Diese ist allerdings selten grafisch und in vielen Fällen nicht besonders gut integriert. Auf der anderen Seite kann man recht einfach einzelne Komponenten nach Bedarf austauschen.

Wie auch immer, mit den grafischen Oberflächen KDE und GNOME haben die IDEs auch endgültig unter Unix ihren Platz gefunden. Für KDE existiert `Kdevelop`, mit dem Projekte verschiedenster Art bearbeitet werden können. Wenn Sie eine Windows-ähnliche Entwicklungsumgebung suchen, dann ist das vermutlich Ihre Wahl.

11 Source- und Konfigurations-Management

11.1 Versionen, Revisionen und Management

Nicht nur Programmierer, sondern fast alle Computeranwender arbeiten mit Dateien, tauschen diese mit Kollegen, Freunden oder Geschäftspartnern aus und bearbeiten diese gemeinsam. Irgendwann kommt der Zeitpunkt, an dem man sich fragt, welches eigentlich die aktuelle Version einer Datei ist, wer diese wie verändert hat oder was der Unterschied zwischen zwei Versionen ist.

Diese Fragen manuell oder mit organisatorischen Mitteln zu klären, ist praktisch unmöglich. Die einzige Chance ist ein zentrales Repository anzulegen, auf das alle Zugriff haben und in dem alle Versionen gespeichert werden. Es gibt eine ganze Reihe von Programmen, die diese Funktionen bereitstellen. Wir werden uns hier einige davon genauer ansehen.

Eine Versionsverwaltung oder Source Code Management ist nicht die Lösung aller Probleme. Entwickler müssen immer noch miteinander reden, allen Beteiligten muss die Entwicklungsrichtung bekannt sein. Auch wenn eine Versionsverwaltung das Nachvollziehen von Fehlern erleichtert, ohne eine Testumgebung wird man beim Programmieren nicht auskommen.

Bei größeren Projekten wird die Entwicklung häufig aufgeteilt, zum Beispiel in eine stabile und eine Entwicklerversion, oder bestimmte Funktionen werden zunächst in einem eigenen Branch implementiert. Problematisch dabei ist, dass man bestimmte Änderungen, zum Beispiel wichtige Bugfixes, in allen Branches durchführen muss (und hoffentlich keinen vergisst). Ein weiteres Problem ist der Status der Branches. Sind alle Änderungen jetzt im Mainline? Welcher Branch enthält welche Version und wird er noch benötigt? Hier ist eine ausgeklügelte Dokumentation notwendig.

11.2 Revision Control System (RCS)

Das Revision Control System (RCS) wurde von Walter F. Tichy geschrieben und verwaltet verschiedene Versionen von Textdateien. Das RCS automatisiert das Speichern, Wiederherstellen, Dokumentieren, Identifizieren und Zusammenführen von verschiedenen Versionen von Textdateien. RCS kann für Textdateien verwendet werden, die regelmäßig geändert werden müssen, zum Beispiel Programme, Dokumentationen oder Konfigurationsdateien. Alle Änderungen und Versionsinformationen speichert das RCS in einer Datei, dem RCS-Archiv, ab.

In einem RCS-Archiv werden nur die Differenzen zwischen den Versionen abgespeichert, nicht die kompletten Dateien. Dadurch ist der Mehrbedarf an Plattenplatz für die Verwaltung von Dateien mittels RCS relativ gering. Dabei wird die letzte Version immer komplett abgelegt, so dass der (relativ häufige) Zugriff darauf sehr schnell erfolgt. Beim Zugriff auf ältere Versionen müssen zunächst die durchgeführten Änderungen rückgängig gemacht werden, so dass dies etwas länger dauern kann.

Die grundlegenden Kommandos des RCS sind einfach zu lernen. Die beiden wichtigsten Kommandos sind `ci` und `co`. `ci` ist eine Abkürzung für »check in«. Es speichert die Inhalte einer Datei in ein RCS-Archiv. `co` ist die Abkürzung für »check out« und kopiert Versionen aus dem RCS-Archiv.

11.2.1 Die Funktionen des RCS

- Alte Versionen werden platzsparend in einer Datei, im folgenden »RCS-Archiv« genannt, gespeichert. Vorgenommene Änderungen an der Textdatei zerstören nicht mehr das Original, weil vorangegangene Versionen immer verfügbar sind. Bestimmte Versionen können nach Versionsnummer, Datum, Autorennamen oder Status abgefragt werden.
- Neben den Veränderungen am Text werden auch der Autorenname, das Datum und die Zeit sowie eine Logbuchmitteilung abgespeichert. Diese Mitteilungen machen es einfach, herauszufinden, was mit einer Datei passiert ist, ohne dass Kollegen gefragt werden müssen. Damit ist die komplette Änderungshistorie einer Datei stets nachvollziehbar.
- Wenn zwei oder mehr Programmierer die gleiche Version ändern wollen, warnt RCS die Programmierer und verhindert dadurch, dass eine Veränderung die andere überschreibt. Bei großen Projekten kann dies allerdings zu einem Engpass führen. Modernere Programme kommen ohne Sperren aus.
- Mit RCS können mehrere Versionen eines Projekts in Form eines Baums verwaltet werden. Zunächst wird mit einer Version begonnen, die später möglicherweise in mehrere unterschiedliche Versionen (Test und Produktion oder für mehrere Kunden) aufgespalten wird. Diese Versionen können später wieder zusammengeführt werden (`merge`). Die Verwaltung von »Branches« kann recht schnell kompliziert werden, insbesondere ist es notwendig, über die Existenz der Branches den Überblick zu behalten.
- Die sich überlappenden Änderungen in mehreren Versionen eines Baums werden aufgezeigt, damit die dadurch entstehenden Probleme per Hand behoben werden können.
- Dateien werden automatisch identifiziert durch Versionen mit Angabe von Dateiname, Versionsnummer, Autorennamen, Datum mit Uhrzeit usw.

- Das RCS braucht wenig Plattenplatz für das Archiv. Es werden jeweils nur die Differenzen zwischen den Versionen abgespeichert. Sie sollten bei der Verwendung von RCS nach Möglichkeit keine globalen Ersetzungen durchführen oder das Programm `indent` verwenden, da in diesen Fällen die gespeicherten Differenzen extrem groß werden können.

11.2.2 Ein Beispiel für die Verwendung von RCS

Die Verwendung von RCS ist wesentlich einfacher, als man nach der Lektüre der entsprechenden Manpages erwartet. In den folgenden Abschnitten werden die wichtigsten Funktionen des RCS anhand von Beispielen vorgestellt. In Listing 11.1 finden Sie eine Datei, die mit RCS verwaltet werden soll.

```
/* hello.c */
main( void)
{       printf( "Hello world\n");
}
```

Listing 11.1 Eine Beispieldatei für die Verwendung von RCS

Diese Datei soll nun in dieser Ursprungsversion in das RCS-Archiv aufgenommen werden. Dazu rufen Sie das Kommando `ci` auf (Listing 11.2). Wenn Sie mit dem Editor Emacs arbeiten, so können Sie mit der Tastenkombination Strg+x v v (`vc-next-action`) eine Datei neu in ein RCS-Archiv aufnehmen. In der Modeline werden der Text `RCS:` und die RCS-Versionsnummer der Datei angezeigt, so dass man mit einem Blick sieht, dass die Datei unter RCS-Kontrolle ist.

```
(linux):~$ ci hello.c
hello.c,v  <--  hello.c
enter description, terminated with single '.' or end of file
NOTE: This is NOT the log message!
>> Mein erstes Programm in C
>> .
initial revision: 1.1
done
```

Listing 11.2 Aufnehmen einer Datei in das RCS-Archiv

Dieses Kommando erzeugt das RCS-Archiv `hello.c,v`, speichert den Inhalt von `hello.c` darin als Version 1.1 und löscht die Arbeitsdatei. Beim Speichern wird nach einer Programmbeschreibung gefragt, die ebenfalls im RCS-Archiv gespeichert wird. Alle weiteren Aufrufe von `ci` werden nach einem Log-Eintrag fragen. Dort sollten die vorgenommenen Änderungen kurz und präzise beschrieben sein. Werden hier ungenaue oder fehlerhafte Beschreibungen eingegeben, so ist das ganze Verfahren nicht besonders sinnvoll, man hat allerdings noch Zugriff auf die alten Versionen der Datei.

RCS-Archive sind die Dateien, die auf `,v` enden. Das `v` steht für Version(en). Die anderen Dateien sind die »Arbeitsdateien«. Um die Arbeitsdatei aus dem vorangegangenen Beispiel aus dem Archiv zu lesen, kann das Kommando `co` (siehe Listing 11.3) verwendet werden. Das RCS-Archiv wird in dem Verzeichnis angelegt, in dem die Datei bearbeitet wird. In größeren Projekten wird das schnell unübersichtlich, so dass die RCS-Archive in das Verzeichnis `./RCS` verschoben werden können. Die RCS-Programme finden das Archiv dort selbständig, auch `make` kann Quelltexte automatisch aus dem Archiv extrahieren.

```
(linux):~$ co hello.c
hello.c,v  -->  hello.c
revision 1.1
done
```

Listing 11.3 Die Verwendung von `co`

Das Kommando `co` extrahiert die letzte Version aus dem RCS-Archiv und erzeugt die Arbeitsdatei `hello.c`. Soll die Arbeitsdatei auch geändert werden, so muss das Kommando `co` mit der Option `-l` (für Lock) aufgerufen werden. Eine spezielle Version kann mit der Option `-r` *`Version`* aus dem Archiv extrahiert werden (siehe Listing 11.4).

```
(linux):~$ co -l hello.c
hello.c,v  -->  hello.c
revision 1.1 (locked)
done
```

Listing 11.4 Die Verwendung von `co` *mit Locking*

Jetzt hat die Arbeitsdatei `hello.c` Schreibrechte für den Benutzer gesetzt. Außerdem ist eine Sperre aktiviert, die verhindert, dass ein anderer Benutzer eine geänderte Version in das RCS-Archiv speichern kann. Wenn Sie Emacs verwenden, dann können Sie eine Datei mit der Tastenkombination Strg+x Strg+q (`vc-toggle-read-only`) aus dem Archiv auslesen, ändern und anschließend mit derselben Tastenkombination wieder in das Archiv zurückschreiben. Für die einfache Verwendung von RCS reichen diese Informationen bereits vollkommen aus.

Nach dem Auslesen aus dem Archiv ist unsere Beispieldatei geändert worden. In Listing 11.5 ist die Datei nach der Änderung abgedruckt.

```
/* hello.c */
main( void)
{       printf( "Hallo Welt\n");
}
```

Listing 11.5 Die geänderte Beispieldatei

Die geänderte Datei soll als neue Version in das RCS-Archiv aufgenommen werden. Dazu wird das Kommando `ci` erneut aufgerufen. Das Kommando `ci` fragt jetzt nach einer Logbuchmitteilung. Auch der Editor Emacs fragt in einem neuen Window nach einem solchen Eintrag, mit der Tastenkombination Strg+c Strg+c wird die neue Version in das Archiv aufgenommen.

```
(linux):~$ ci hello.c
hello.c,v  <--  hello.c
new revision: 1.2; previous revision: 1.1
enter log message, terminated with single '.' or end of file
>>Ins Deutsche übersetzt.
>>.
done
```

Listing 11.6 Einchecken einer geänderten Datei

Sollte die Fehlermeldung erscheinen, so hat der Benutzer versucht, eine Arbeitsdatei in das RCS-Archiv einzufügen, die vorher beim Auslagern nicht gesperrt war. Mit dem Kommando `rcs` mit der Option `-l` wird eine Sperre erzeugt, so dass die Arbeitsdatei nun doch noch in das Archiv gespeichert werden kann. Den entsprechenden Befehl finden Sie in Listing 11.7.

```
(linux):~$ rcs -l1.1 hello.c
rcs file: hello.c,v
1.1 locked
done
```

Listing 11.7 Der Befehl `rcs` im Einsatz

Nur mit der Verwaltung von Versionen ist in der Regel nicht allzuviel anzufangen. Besonders wenn ein neuer Fehler eingebaut wurde, ist es sinnvoll zu wissen, was zwischen zwei Versionen genau geändert wurde. Um Unterschiede zwischen zwei Versionen anzuzeigen, kann der Befehl `rcsdiff` verwendet werden. Der Befehl in Listing 11.8 zeigt die Unterschiede zwischen den Versionen 1.1 und 1.2 an.

```
(linux):~$ rcsdiff -r1.1 -r1.2 hello.c
rcs file: hello.c,v
retrieving revision 1.1
retrieving revision 1.2
diff -r1.1 -r1.2
3c3
< {       printf( "Hello world\n");
---
> {       printf( "Hallo Welt\n");
```

Listing 11.8 Der Befehl `rcsdiff`

Wird nur eine Versionsnummer angegeben, so wird die aktuelle Arbeitsdatei mit der angegebenen Version verglichen. Wird keine Versionsnummer angegeben, so wird die Arbeitsdatei mit der Default-Version verglichen (das ist in der Regel die letzte Version im Archiv). Da das Kommando `rcsdiff` den Unix-Befehl `diff` verwendet, ist es zum Beispiel auch möglich, so genannte Unified- (Option `-u`) oder auch Context-Diffs (Option `-c`) zu erstellen.

Innerhalb des Emacs können Sie die Funktion `ediff-revision` verwenden, um verschiedene RCS-Versionen einer Datei zu vergleichen. Es wird der `ediff`-Mode verwendet, mit dem man auch einzelne Dateien vergleichen kann (siehe auch Abschnitt »Textdateien vergleichen«).

11.2.3 Automatische Identifikation

RCS kann spezielle Kennzeichen, die RCS-Merker, in der Arbeitsdatei erzeugen. Um solch eine Kennzeichnung in eine Datei einzufügen, kann der Merker `$Id$` in den Text, z. B. in einen Kommentar, eingefügt werden. Dieser Merker wird vom RCS durch »$Id: *Dateiname Version Datum Zeit Autor Status* $« ersetzt.

Mit diesem Merker auf der ersten Seite in jedem Modul kann man jederzeit sehen, an welcher Version gerade gearbeitet wird. Die Merker werden vom RCS automatisch aktualisiert. Um diese Kennzeichnung auch in den Objektcode des Programms zu bekommen, muss diese Kennung in einen String eingebettet werden. Zum Beispiel geht das in der Programmiersprache C, wie in Listing 11.9 dargestellt.

```
static char rcsid[] =
"$Id: scm.xml,v 1.5 2002/06/23 11:15:16 jochen Exp $";
```

Listing 11.9 Einbinden der RCS-Identifikation in Binärprogramme

Das Kommando `ident` extrahiert diese Merker aus jeder Art von Datei, z. B. auch aus Objektdateien, Binärdateien oder Core-Dumps. So kann man herausfinden, welche Versionen von welchen Modulen in das Programm einkompiliert wurden.

Eventuell kann es auch nützlich sein, den Merker `$Log$` in die Source-Datei einzubauen. Dieser Merker sammelt alle Log-Einträge, die während des Einspeicherns mit `ci` eingegeben wurden. So ist die komplette Versionsgeschichte in dem Sourcefile mit abgespeichert, allerdings kann diese Liste recht schnell sehr lang werden. Es gibt eine Reihe solcher Merker, die bei der Manpage des Kommandos `co` aufgelistet sind.

11.2.4 RCS-Merker

RCS-Merker sind Zeichenketten, die mit dem Zeichen `$` beginnen. Dann folgt die Merkerbezeichnung. Danach folgt, wenn der Merker bereits ersetzt wurde, nach einem Doppelpunkt der Wert des Merkers und zum Schluss kommt noch ein `$`. Zunächst wird der Benutzer den Merker ohne Wert eingeben, zum Beispiel `$Id: scm.xml,v 1.5 2002/06/23 11:15:16 jochen Exp $`. `co` erkennt diese Merker und erweitert sie durch die aktuellen Werte. Folgende Merker werden von `co` in der Arbeitsdatei erkannt und ersetzt:

`$Author$`
: Der Login-Name des Benutzers, der die Version einspeichert.

`$Date$`
: Das Datum und die Uhrzeit in GMT, wann die Version eingespeichert wurde.

`$Header$`
: Ein Standardvorspann, der den kompletten RCS-Dateinamen, die Versionsnummer, das Datum, den Autorennamen, den Status und den Benutzernamen, der die Datei gesperrt hatte, enthält.

`$Id$`
: Ein Standardvorspann wie beim `$Header$`, es wird aber beim Dateinamen der Pfad nicht angegeben.

`$Locker$`
: Der Login-Name des Benutzers, der die Version gesperrt hat.

`$Log$`
: Die Logbuchmitteilung, die beim Einspeichern eingegeben werden kann. Bereits existierende Logbucheinträge werden nicht aus der Arbeitsdatei entfernt. Die neuen Einträge werden also hinzugefügt.

`$rcsfile$`
: Der Name des RCS-Archivs.

`$Revision$`
: Die Versionsnummer.

`$Source$`
: Der komplette Pfadname des RCS-Archivs.

`$State$`
: Der aktuelle Status der Version. Standardstatus ist »Exp«. Sinnvolle Bezeichnungen in der Programmentwicklung könnten zum Beispiel »Alpha« oder »Beta« für entsprechende Testversionen sein.

Nach dieser eher trockenen Einführung folgt nun wieder ein Beispiel. Zunächst finden Sie in Listing 11.10 den Header einer Datei, bevor die RCS-Merker expandiert wurden. Im Listing 11.11 ist derselbe Header abgedruckt, nachdem RCS die Merker expandiert hat.

```
/*
 * $Id$
 * $Revision$
 * $Date$
 * $Log$
 */
```

Listing 11.10 Ein Dateiheader für RCS

```
/*
 * $Id: hello.c,v 1.18 1995/01/15 16:01:00 cschaba Exp $
 * $Revision: 1.18 $
 * $Date: 1995/01/15 16:01:00 $
 * Revision 1.18  1995/01/15  16:01:00  cschaba
 * In der Funktion wird jetzt 0 zurueckgegeben.
 *
 * Revision 1.17  1995/01/08  17:46:41  cschaba
 * Verwendet jetzt die BSD-Header.
 * ...
 */
```

Listing 11.11 Ein Dateiheader mit expandierten RCS-Merkern

11.2.5 RCS und andere Programme

Die Zusammenarbeit zwischen RCS und Emacs haben wir bereits kennen gelernt. Der Editor XEmacs enthält das Paket `pcl-cvs`, das eine wesentliche Erweiterung der hier beschriebenen Schnittstelle zu `cvs` enthält. `cvs` selbst ist auf der Basis von RCS implementiert und hat besonders bei der Entwicklung mit mehreren Programmierern deutliche Vorteile gegenüber RCS.

Das Programm GNU-`make` kennt Standardregeln, um Dateien aus Archiven mittels RCS zu extrahieren. Dabei werden sowohl Archive im aktuellen Verzeichnis als auch im Verzeichnis `./RCS` unterstützt. Damit ist die Verwendung von RCS vollkommen transparent, selbst wenn man vergessen hat, eine Arbeitsdatei zum Übersetzen zu extrahieren.

11.2.6 Weitere rcs-Kommandos

RCS besteht noch aus einigen weiteren Kommandos. Im Folgenden stelle ich nur einige wenige Funktionen dieser Programme vor. Die vollständige Dokumentation finden Sie in den entsprechenden Manpages.

Änderungslogs mit rlog

Das Programm `rlog` zeigt die Änderungshistorie einer Datei an. Dabei kann nach einer Reihe von Kriterien selektiert werden, z. B. ändernder Benutzer (`-w`), Versionen (Option `-r`) oder Zeiträume (Option `-d`). Das genaue Format der Optionen und der dazugehörigen Parameter finden Sie in der Manpage zu `rlog`.

Archive verwalten mit RCS

Das Programm `rcs` dient ganz allgemein zur Verwaltung von RCS-Archiven. Archive können neu angelegt oder gesperrt werden. Der Eigentümer des Archivs kann Locks von anderen Benutzern aufheben oder die Zugriffsberechtigungen auf dieses Archiv verändern. Weiterhin ist es möglich, einen symbolischen Namen für eine Version zu verwenden. Die vollständige Beschreibung aller Optionen finden Sie in der Manpage.

Zusammenführen von mehreren Branches

Bei der Arbeit mit mehreren Branches kommt es häufig vor, dass Änderungen in einem Branch auf einem zweiten durchgeführt werden müssen. So soll z. B. eine Fehlerkorrektur auch in die aktuelle Entwicklungsversion einfließen, sofern an dieser Stelle nicht bereits etwas modifiziert wurde. Diese anstrengende und zeitaufwendige Arbeit kann mit dem Programm `rcsmerge` weitgehend automatisiert werden. Alle Optionen zu `rcsmerge` finden Sie in der Manpage.

Zum besseren Verständnis der Arbeitsweise von `rcsmerge` folgt hier ein Beispiel. Nachdem Sie das Programm `hello.c` mit der Version 2.2 ausgeliefert haben, haben Sie einen neuen Entwicklungszweig (Branch) mit der Version 3.3 aufgemacht. Nun kommen von einem anderen Programmierer einige Verbesserungen zur Version 2.2, die Sie gern in Ihre aktuelle Entwicklungsversion aufnehmen möchten. Speichern Sie die externen Änderungen zur Version 2.2 in der Datei `hello.c` und starten Sie den Befehl `rcsmerge` (siehe Listing 11.12).

```
(linux):~$ rcsmerge -p -r2.2 -r3.3 hello.c > hello.merged.c
```

Listing 11.12 Der Befehl `rcsmerge`

Anschließend finden Sie alle Änderungen, sofern sie automatisch durchgeführt werden konnten, in der Datei `hello.merged.c` wieder. Da jedoch möglicherweise in beiden Branches Änderungen an derselben Stelle im Programm vorgenommen wurden, sollten Sie diese Datei zunächst genauer prüfen.

Alternativ können Sie die externen Änderungen in einem eigenen Branch unterbringen und von dort aus verarbeiten. Ein Beispiel dafür wären die Befehle in Listing 11.13.

```
(linux):~$ ci -r2.2.1.1 hello.c
(linux):~$ co -r3.3 -j2.2:2.2.1.1 hello.c
```

Listing 11.13 Erstellen eines neuen Branch

Mit `rcsmerge` können Sie auch weiter zurückliegende Änderungen wieder aus Ihrem Programm entfernen. Dazu mischen Sie einfach die Änderungen zwischen den beiden älteren Versionen in umgekehrter Reihenfolge zu Ihrer aktuellen Arbeitsversion. Das Beispiel in Listing 11.14 entfernt die Änderungen zwischen den Versionen 3.2 und 3.1 aus `hello.c`.

```
(linux):~$ rcsmerge -r3.2 -r3.1 hello.c
```

Listing 11.14 Noch ein Beispiel für `rcsmerge`

ident – Dateien identifizieren

Das Kommando `ident` durchsucht die angegebene Datei oder die Standardeingabe nach RCS-Merkern. Diese Kennzeichen werden normalerweise von `co` automatisch gesetzt, können aber auch manuell in die Datei eingefügt werden. Mit der Option `-q` wird die Anzeige der Warnung, dass keine Merker gefunden wurden, ausgeschaltet. `ident` funktioniert mit Text- und Binärdateien.

```
(linux):~$ ident hello
hello:
    $Id: hello.c,v 1.18 1995/01/15 16:01:00 cschaba Exp $
```

Listing 11.15 RCS-Informationen aus einer Datei extrahieren

11.3 Das Concurrent Versions System

Bei großen Projekten ist es oft sinnvoll, den ganzen Source-Tree mit einem Befehl aus- bzw. einzuchecken. Diese Funktion ist (neben vielen anderen nützlichen Dingen) in CVS, dem »Concurrent Versions System«, implementiert. Die Autoren haben festgestellt, dass in den meisten Fällen nur ein Programmierer an einer speziellen Funktion arbeitet, so dass CVS auf Locks verzichtet.

Wenn beim Einchecken festgestellt wird, dass die Änderungen nicht maschinell eingepasst werden können, dann wird der Programmierer aufgefordert, diesen Konflikt manuell zu beseitigen. Man hat hier den Vorteil, dass sich Programmierer nur in den seltensten Fällen behindern.

Eine weitere nützliche Funktion ist ein CVS-Archiv, das im Internet verfügbar ist. Damit ist eine Versionskontrolle auch für Projekte möglich, bei denen kein NFS-Zugriff auf die entsprechenden Archive möglich ist. Diese Funktion wird für einzelne Projekte wie `FreeBSD` oder `lesstif` verwendet. Dabei unterscheidet

man zwischen lesendem Zugriff, der für viele Anwender möglich ist, und der Möglichkeit, Veränderungen vorzunehmen. Letztere ist oft auf nur wenige Entwickler beschränkt.

Wenn Sie Emcas verwenden, dann ist der Einsatz von CVS ähnlich unauffällig wie der Einsatz von RCS, insbesondere die Tastenkombinationen sind dieselben. Die wichtigsten Kommandos von CVS sind:

`cvs checkout` *`Module`*
: Auschecken eines Moduls aus dem Repository. Das Repository kann entweder mit der Kommandozeilenoption `-d` angegeben werden, oder in der Umgebungsvariablen `CVSROOT`. Es werden keine Sperren gesetzt, so dass mehrere Mitarbeiter gleichzeitig am Tree arbeiten können.

`cvs update`
: Aktuelle Änderungen aus dem CVS auslesen und in die Arbeitskopie einpflegen. Hierbei kann es zu Konflikten, sich überschneidenden Änderungen, kommen. Diese müssen vom Entwickler manuell aufgelöst werden.

`cvs commit`
: Die Änderungen aus der Arbeitskopie in das Repository festschreiben. Um Konflikte zu vermeiden, muss unmittelbar vorher `cvs update` ausgeführt werden. Erst nach dem Commit sind lokale Änderungen auch für andere sichtbar.

`cvs diff` *`Dateiname`*
: Vergleichen der aktuellen Version einer oder mehrerer Dateien mit dem Repository. Wird kein Dateiname angegeben, so wird ein »diff« des gesamten Tree erzeugt. Sie können Optionen für `diff` angeben oder diese dauerhaft in der Datei `.cvsrc` festlegen.

`cvs add` *`Dateiname`*
: Fügt eine Datei im Tree hinzu, diese wird mit dem nächsten `cvs commit` in das Repository übernommen. Eine gelöschte Datei kann mit `cvs remove` *`Dateiname`* aus dem Repository entfernt werden.

Mit diesen Befehlen werden Sie CVS weitgehend problemlos einsetzen können. Vielfach haben Sie Arbeitsdateien oder Ergebnisse von Compilerläufen im Arbeitsverzeichnis. Diese sollen von CVS nicht bearbeitet werden, nehmen Sie die Namen oder passende Wildcards in die Datei `.cvsignore` auf.

Zusätzlich zu einem Branch kann eine bestimmte Version mit einem Merker, einem »Tag« versehen werden. Dieser kann beim Checkout oder Merge verwendet werden. Tipps zum Umgang mit CVS allgemein und mit Branches und Tags finden Sie in den »CVS Best Practices« unter *http://www.tldp.org/REF/CVS-Best-Practices/html/index.html*.

11.3.1 Anlegen eines CVS-Repository

Ein CVS-Repository ist ein Verzeichnis, das mit einer Reihe von Dateien gefüllt ist, die das CVS zur internen Verwaltung benötigt. Ein Repository legen Sie mit dem Befehl `cvs init` an. Den Speicherort des Repository geben Sie entweder mit der Kommandozeilenoption `-d` an oder in der Umgebungsvariablen `CVSROOT`.

Im einfachsten Fall liegt das Repository lokal, so dass Sie ohne Weiteres darauf zugreifen können. Wenn Sie eine Benutzerkennung auf dem Rechner haben, wo das CVS-Repository liegt, so können Sie `ssh` zur Kommunikation verwenden (Listing 11.16). Viele CVS-Repositories sind global lesbar und verwenden dafür das `pserver`-Protokoll. Man kann dieses auch zum Checkin verwenden, ich bevorzuge allerdings die Kommunikation mit `ssh`.

```
(linux):~$ export CVS_RSH=/usr/bin/ssh
(linux):~$ export CVSROOT=:ext:jupiter.jochen.org:/usr/src
(linux):~$ cvs init
```

Listing 11.16 Anlegen eines Repository mit `ssh`

Nach dem Befehl in Listing 11.16 finden Sie im Verzeichnis `/usr/src/CVSROOT` ein leeres Repository, in das Sie zum Beispiel mit `cvs import` einen bestehenden Verzeichnisbaum importieren können. Wenn Sie einen `pserver`-Zugang aufsetzen wollen, dann sehen Sie bitte in die CVS-Dokumentation und beachten Sie die Hinweise zur Systemsicherheit.

11.4 Subversion – die zweite Generation von CVS

CVS ist in vielen Projekten im Einsatz – und hat sich bewährt. Auf der anderen Seite ist nicht alles Gold, was glänzt. CVS hat eine Reihe von Schwächen, die offensichtlichsten sind die fehlende Unterstützung beim Umbenennen von Dateien und dass ein Checkin nicht atomar ist. Das hat auch damit zu tun, dass CVS auf RCS basierte.

Subversion (*http://subversion.tigris.org/*) tritt an, die Defizite von CVS abzuschütteln und CVS abzulösen. Dabei soll der Umstieg so einfach wie möglich sein, es wird ein Perl-Skript mitgeliefert, um ein CVS-Repository komplett in ein Subversion-Repository zu überführen. Außerdem bietet Subversion WebDAV (Web enabled Distributed Authoring and Versioning, [rfc2518]), so dass viele Werkzeuge für diese Protokolle verwendet werden können.

Bei CVS werden die Revisionen je Datei nummeriert, bei Subversion gilt eine Revision für den gesamten Tree. Damit ist es eindeutig, über welche Revision man gerade redet. Bei CVS ändert sich die Revision einer Datei bei einer Änderung,

bei Subversion ist es wahrscheinlich, dass sich eine Datei zwischen den Revisionen nicht ändert.

Der Umstieg für den normalen Anwender ist unkompliziert, tauschen Sie einfach das Kommando `cvs` durch `svn` aus. Die Kommandos zum Checkout, Update und Commit funktionieren wie bei CVS. Listing 11.17 zeigt ein Beispiel für eine Editiersitzung mit `svn`.

```
(linux):~/src$ svn checkout http://svn.jochen.org/svn/demo
...
(linux):~/src$ cd demo
(linux):~/src/demo$ svn update
...
(linux):~/src/demo$ vi Datei
(linux):~/src/demo$ svn update
...
(linux):~/src/demo$ svn commit
...
```

Listing 11.17 Arbeiten mit `svn`

Im ersten Schritt müssen Sie sich eine Kopie des Repository zur Bearbeitung holen. Das erfolgt durch das Kommando `svn checkout` *`Repository`*. Mit dem Parameter `-d` können Sie einen anderen Verzeichnisnamen angeben. Nun können Sie nach Herzenslust Änderungen vornehmen, kompilieren und testen. Gelegentlich können Sie die Dateien mit einem neueren Stand aus dem Repository aktualisieren (`svn update`), das ist spätestens vor dem Commit notwendig. Das Update versucht, die Änderungen automatisch in der Arbeitskopie vorzunehmen, die Ausgabe des Befehls sagt, welche Dateien aktualisiert (`U`) bzw. lokal verändert wurden (`M`) oder einen Konflikt haben (`C`).

Nach einem Update mit Änderungen sollten Sie nochmal einen Funktionstest durchführen, bevor Sie schließlich Ihre Änderungen festschreiben. Der Befehl `svn commit` fragt nach einem Logeintrag und speichert Ihre Änderungen im zentralen Repository. Andere Anwender können diese dann mittels `cvs update` in ihre Arbeitskopie übernehmen. Mit diesen Befehlen können Sie bereits normal arbeiten.

Sie können Dateien in das Repository aufnehmen (`svn add`) oder daraus löschen (`svn remove`). Die Änderungen werden mit dem nächsten Commit in das Repository übernommen. CVS bietet keine Möglichkeit, Dateien im Repository umzubenennen, man wird dabei immer die Historie beschädigen. Subversion speichert auch Verzeichniseinträge im Repository, so dass auch Kopieren und Verschieben von Dateien und Verzeichnissen nachvollziehbar sind.

Mit `svn copy` *`Quelle Ziel`* kopieren Sie eine Datei, dabei wird in der Historie der neuen Datei auf die der alten verwiesen. Es können analog auch Verzeichnisse angegeben werden. Eine Datei benennen Sie mit `svn move` um (Listing 11.18).

Viele CVS-verwaltete Projekte versuchen auf Umbenennungen zu verzichten, weil dies nicht unterstützt wird. Auf der anderen Seite gibt es bei der Entwicklung gute Gründe, Dateien oder Verzeichnisse umzubenennen. Subversion kann hier Vorteile bieten.

```
(linux):~$ svn move
(linux):~$ svn status
```

Listing 11.18 Umbenennen von Dateien mit Subversion

Ein neues Repository erstellen Sie mit `svnadmin create` *`Verzeichnis`*. Anschließend können Sie Dateien aus einem bestehenden Verzeichnis mit `svn import` importieren (Listing 11.19). In vielen Fällen funktioniert `svn` wie `cvs`, es gibt aber subtile und deutliche Unterschiede. Diese finden Sie zum Beispiel in der Dokumentation auf der Subversion-Webseite erläutert. Noch ist die mitgelieferte Dokumentation nicht vollständig, Hilfe zu den einzelnen Kommandos erhalten Sie mit `svn help` *`Kommando`*.

```
(linux):~$ svnadmin create /usr/src/svn-repo
(linux):~$ svn import file:///usr/src/svn-repo workdir
```

Listing 11.19 Erstellen eines Subversion-Repository

Nach dem ersten Befehl ist das Verzeichnis `/usr/src/svn-repo` mit einigen Dateien und Verzeichnissen gefüllt, die die internen Verwaltungsinformationen enthalten. Mit dem zweiten Befehl importieren Sie bestehende Dateien aus dem Verzeichnis *`workdir`* in das Repository. Wie Sie sehen, verwendet Subversion URLs zur Adressierung des Repository. Damit können Sie auf Verzeichnisse zugreifen, die lokal oder auf einem Web-Server gespeichert sind. Damit kann Subversion recht einfach über Firewalls hinweg verwendet werden, denn dieser Port ist meist freigeschaltet. Bei CVS wird ein eigener oder der SSH-Port verwendet und diese sind meist nicht freigegeben.

Subversion ist noch nicht in der Version 1.0 freigegeben, allerdings benutzt das Projekt bereits seit einiger Zeit die eigene Software. Wenn Sie etwas Zeit haben, sich damit zu beschäftigen, und in der Testphase mit Datenverlusten leben können, dann sollten Sie Subversion mal ausprobieren. Das Projekt plant bald eine erste Beta-Version, dann werden hoffentlich auch Pakete für die verschiedenen Distributionen bereitstehen.

11.5 Andere Systeme zur Versionsverwaltung

Bei kommerziellen Unix-Systemen ist oft SCCS als Versionsverwaltung enthalten. Wenn Sie in Ihren Projekten dieses System einsetzen, dann können Sie unter Linux die GNU-Implementation CSSC verwenden. Wenn Sie mit einem neuen Projekt beginnen, dann sollten Sie sich in jedem Fall RCS und CVS ansehen.

Für die Verwaltung der Kernel-Quellen wird BitKeeper eingesetzt. Eine sehr leistungsfähige, kommerzielle Software, die für GPL-Projekte unter bestimmten Bedingungen kostenfrei eingesetzt werden darf. Wenn Sie die Lizenzbedingungen nicht schrecken (es gibt auch eine »normale« kommerzielle Version), dann ist auch dieses Programm einen Blick wert. Für meine persönlichen Zwecke ist bisher CVS ausreichend, ich werde aber vermutlich bald auf Subversion umsteigen.

12 XML unter Linux

12.1 Die Entwicklung von XML

Schon vor vielen Jahren erkannte man, dass die Struktur von Dokumenten von der Darstellung getrennt werden kann und dass dies in vielen Fällen auch sehr sinnvoll ist. In einem Dokument wird eine Auszeichnungssprache (Markup-Language) verwendet, dabei werden bestimmte Textteile markiert (ausgezeichnet). Bei einem visuellen Markup wird unmittelbar die Darstellung angegeben (z. B. fett, kursiv), bei semantischem Markup wird die Struktur des Dokuments angegeben (Kapitel, Überschrift etc.).

In 1989 wurde SGML (Standard Generalized Markup Language) zu einem internationalen Standard (ISO 8879:1986). Viele große Unternehmen haben ihr Dokumentenarchiv auf SGML aufgebaut – aufgrund der Komplexität von SGML und den beteiligten Tools wurde SGML lange Zeit eher stiefmütterlich behandelt. SGML legt fest, wie man eine Dokumentenstruktur (eine SGML-Anwendung, DTD, Document Type Description) definiert. Mit Hilfe der Stylesheet-Sprache DSSSL (Document Style Semantics and Specification Language, ISO/IEC 10179:1996) kann das Dokument in andere Formate überführt werden. Eine frei verfügbare DSSSL-Engine ist `jade`.

SGML führte lange Zeit ein Schattendasein. Mit der Entwicklung HTML (Hyptertext Markup Language) als SGML-Applikation lernten mehr Anwender die betreffende Notation kennen. Bei HTML ist das Markup ein Mittelding zwischen visuell und semantisch – weder Fleisch noch Fisch. Viele Web-Designer erwarten von HTML ein pixelgenaues Layout oder verlangen einen WYSIWYG-Editor, was es bei semantischem Layout nicht geben kann. Im Rahmen des Browser-Kriegs zwischen Netscape und Microsoft wurden verschiedene zusätzliche Tags definiert – die aber nicht jeder Browser darstellen konnte.

Die Herausforderung war nun, wie sich HTML erweitern ließe ohne die HTML-Tools zu verlieren und die Komplexität von SGML implementieren zu müssen. Die Extensible Markup Language (XML, *http://www.w3.org/XML/*) war geboren. Die nächste HTML-Version (XHTML) wird in XML statt SGML definiert sein.

12.2 Document Type Descriptions und Schemata

XML kennt verschiedene Arten von Dokumenten. Die erste Art heißt »wohlgeformt«. Dabei ist für jedes Start-Tag ein End-Tag in der passenden Reihenfolge vorhanden, siehe Listing 12.1. Die zweite Art ist »valid« gemäß einer bestimmten

Dokumentenstruktur. Eine derartige Struktur kann entweder in einer (SGML-kompatiblen) Document Type Description (DTD) oder einem XML-Schema festgelegt werden. Noch werden meist DTDs verwendet, in Zukunft wird man eher Schemata verwenden. Warum? Schemata werden wieder in XML-Syntax notiert – damit werden Tools einfacher und Autoren müssen nicht mehr verschiedene Formate beherrschen.

```
<?xml version='1.0' encoding='iso-8859-15'?>
<book>
  <title>Der Buchtitel</title>
  <author>Der Autor</author>
</book>
```

Listing 12.1 Ein einfaches XML-Dokument

Das Design einer DTD oder eines Schemas ist nicht einfach, in der Praxis wird man, je nach Anwendung, häufig bereits bestehende Strukturen verwenden können. Insbesondere für den Datenaustausch gibt es bereits viele (Branchen-)Standards, im Publishing-Umfeld kann man DocBook, TEI oder verlagseigene Vorgaben verwenden. Dieses Buch wurde mit Hilfe der DocBook-DTD geschrieben. Der Verlag hat eine eigene DTD, so dass das Manuskript am Ende mittels einer XSL-Transformation konvertiert wurde.

Sollten Sie eine eigene DTD entwickeln müssen, so legen Sie zunächst Ihre Anforderungen fest (damit Sie nicht mehr implementieren, als Sie eigentlich wollten). Sehen Sie sich einige Beispiele von DTDs an, insbesondere die DocBook-DTD ist ein Beispiel, wie Anpassungen relativ leicht möglich sein können – wenn das ein Designziel ist. Prüfen Sie eine Reihe von Dokumenten und reden Sie mit den Anwendern bzw. Autoren, um eventuell notwendige Anpassungen oder Erweiterungen zu erkennen. Die schönste Dokumentenstruktur nützt nichts, wenn diese nicht verwendet oder gar missbraucht wird.

12.3 Emacs und Markup-Languages

XML, das ist ja nur ASCII – und genau das ist der Trick: XML-Dateien können mit einem normalen Editor bearbeitet werden, sind einfach mit Unix-Tools manipulierbar und dennoch mit Programmen einfach zu bearbeiten. Wenn Sie Emacs verwenden, dann hilft Ihnen der `psgml-mode` (*http://psgml.sf.net/*) bei der Bearbeitung von SGML- und XML-Dateien. `psgml-mode` kann beliebige DTDs einlesen und Ihnen bei der Eingabe die erlaubten Tags anzeigen oder die Dokumentenstruktur prüfen. Die wichtigsten Tastenkombinationen finden Sie in Tabelle 12.1, eine vollständige Übersicht erhalten Sie in der `psgml`-Dokumentation und mit [Strg]+[h] [m].

Tasten	Funktion	Beschreibung
Strg+c #	sgml-make-character-reference	Wandelt das Zeichen unter dem Cursor in ein »Entity« um.
Strg+c +	sgml-insert-attribute	Fragt im Minibuffer nach einem Attribut und dessen Wert. Im Minibuffer erhalten Sie mit TAB eine Übersicht über die möglichen Eingaben.
Strg+c -	sgml-untag-element	Die Tags um den Cursor herum entfernen.
Strg+c /	sgml-insert-end-tag	Das letzte offene Tag schließen.
Strg+c <	sgml-insert-tag	Ein Tag einfügen, im Minibuffer können Sie die Vervollständigung nutzen.
Strg+c =	sgml-change-element-name	Die Tags um den Cursor herum ersetzen.
Strg+c RET	sgml-split-element	Das aktuelle Element aufteilen, so kann man damit sehr schnell aus einem Absatz zwei machen.
Strg+c Strg+a	sgml-edit-attributes	Alle Attribute des Elements um den Cursor herum werden zum Ändern bereitgestellt. Die Tastenkombination Strg+c Strg+c übernimmt die Änderungen in den Text.
Strg+c Strg+c	sgml-show-context	Zeigt im Minibuffer alle noch offenen Tags an.
Strg+c Strg+d	sgml-next-data-field	Springt zum nächsten Element, wo eine Texteingabe erfolgen darf.
Strg+c Strg+e	sgml-insert-element	Fügt ein Element ein. Sind Attribute notwendig, so werden diese abgefragt. Ist ein zweites Element in diesem notwendig (zum Beispiel der Titel bei einer Tabelle), so wird dieses automatisch eingefügt. In der Vervollständigung stehen nur die erlaubten Tags zur Verfügung.
Strg+c Strg+o	sgml-next-trouble-spot	Der Cursor springt zum nächsten Fehler gemäß der DTD.

Tasten	Funktion	Beschreibung
Strg+c Strg+p	`sgml-parse-prolog`	Neueinlesen der DTD.
Strg+c Strg+r	`sgml-tag-region`	Einfügen eines Tags um die Region herum.
Strg+c Strg+t	`sgml-list-valid-tags`	Listet alle an der Cursorposition erlaubten Tags auf.
Strg+c Strg+v	`sgml-validate`	Ruft das Kommando `sgml-validate-command` auf.
Strg+c Strg+y	`sgml-position`	Ähnlich wie `sgml-show-context`.

Tabelle 12.1 Tastaturbelegung des `psgml`*-Modus*

Wenn Sie ein Dokument in mehrere Dateien aufgeteilt haben, dann ist das Teildokument nicht valid gegenüber der DTD. Dennoch kann Ihnen `psgml` helfen: In Buffer-lokalen Variablen können Sie den Namen der Hauptdatei angeben, den Starttag des Dokuments und des Teildokuments. Damit können Sie alle `psgml`-Funktionen auch in Teildokumenten verwenden. Für dieses Kapitel verwende ich die Einstellungen aus Listing 12.2.

```
<!--
Keep this comment at the end of the file
Local variables:
mode: xml
sgml-parent-document: ("master.xml" "book" "chapter")
ispell-skip-sgml: t
sgml-indent-step:nil
sgml-insert-missing-element-comment: nil
End:
-->
```

Listing 12.2 Lokale Emacs-Variablen im `psgml`*-Mode*

Teildokumente können Sie mit Hilfe einer Entity-Referenz in das Hauptdokument einbinden. Listing 12.3 zeigt ein einfaches Beispiel.

```
<?xml version='1.0' encoding='iso-8859-15' standalone='no'?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.1.2//EN"

"file:///usr/share/sgml/docbook/dtd/xml/4.1.2/docbookx.dtd"
[<!ENTITY kapitel SYSTEM 'kapitel.xml'>
...
]>
<book>
...
```

```
 &kapitel;
...
</book>
```

Listing 12.3 Referenzieren eines Teildokuments

12.4 XML-Tools

Um XML-Dateien zu bearbeiten, gibt es eine ganze Reihe von Tools. In diesem Abschnitt werden wir uns einige ansehen. Mit Hilfe von verschiedenen Zusatzbibliotheken können Sie XML auch mit den verbreiteten Skript-Sprachen Perl, Python und Tcl bearbeiten.

`xmllint`
: `xmllint` gehört zur `libxml` (*http://www.xmlsoft.org/*) und prüft XML-Dateien auf Wohlgeformtheit und Übereinstimmung mit der DTD (sofern eine verwendet wird). Dabei können verschiedene Ausgaben erzeugt werden, so kann das Programm beispielsweise mit der Option `--html` HTML-Dateien einlesen und in XML konvertieren. Die Option `--sgml` kann verwendet werden, um DocBook/SGML-Dokumente einzulesen.

`xsltproc`
: `xsltproc` gehört zur `libxslt` (*http://www.xmlsoft.org/*) und ist ein schneller und guter XSL-Transformator.

  `xsltproc` ist ein in C geschriebener XSL-Prozessor. Dieser kann DocBook-Dokumente verarbeiten und ist, insbesondere im Vergleich zu den Java-Programmen, vergleichsweise schnell.

`rxp`
: `rxp` (*http://www.cogsci.ed.ac.uk/~richard/rxp.html*) ist ein XML-Parser.

`xerces`
: Ein vom Apache-Project entwickelter XML-Parser (*http://xml.apache.org/*).

`xalan`
: Ein vom Apache-Project entwickelter XSL-Processor (*http://xml.apache.org/*).

`nsgmls`
: `nsmgls` ist ein SGML-Parser, der auch für XML-Dokumente verwendet werden kann (*http://jclark.com/sp*). Aktuelle Entwicklungen erfolgen in OpenSP, das Sie unter *http://openjade.sf.net/* finden.

`jade`
: `jade` (*http://jclark.com/jade/*) ist ein frei verfügbarer DSSSL-Prozessor. James Clark entwickelt das Programm nicht mehr weiter, den Nachfolger OpenJade finden Sie unter *http://openjade.sf.net/*. `jade` kann SGML und XML-Dokumente

in verschiedene andere Formate konvertieren, unter anderem HTML, RTF und `jadetex`. Mit Hilfe von `jadetex` kann dieses Format in PDF oder PostScript umgewandelt werden.

`XML::Path`, `XML::Parser`
`XML::Path` und `XML::Parser` sind Perl-Module, die die XML-Path-Sprache (*http://www.w3c.org/TR/xpath*) und einen XML-Parser für Perl-Skripte bereitstellen. Darin enthalten ist ein Skript `xpath`, mit dem XML-Dateien ausgewertet werden können.

12.5 Stylesheets

Ein Dokument (oder strukturierte Daten) alleine machen XML noch lange nicht zu einem besseren Werkzeug als andere Tools. Um nicht für jede Verarbeitung von XML-Dateien ein eigenes Programm schreiben zu müssen, wurden Stylesheets entwickelt. Neben und für SGML wurde DSSSL entwickelt. DSSSL ist eine an Lisp angelehnte, relativ komplexe Sprache. Heute verwendet man meistens XSL (*http://www.w3.org/Style/XSL/*) oder Cascading Stylesheets (*http://www.w3.org/Style/CSS/*).

XSL-Stylesheets sind wiederum im XML-Format geschrieben. Damit müssen Tools nur einen XML-Parser mitbringen, nicht mehr einen SGML- und einen DSSSL-Parser. Listing 12.4 zeigt ein einfaches Stylesheet, mit dem ich aus dem Manuskript alle URLs extrahiere, um diese auf Aktualität zu prüfen.

```
<?xml version="1.0" encoding="iso-8859-15"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

  <xsl:output method="html"/>

  <xsl:template match="primaryie/ulink">
  </xsl:template>

  <xsl:template match="ulink">
    <p>
      <a>
        <xsl:attribute name="href">
          <xsl:value-of select="."/>
        </xsl:attribute>
        <xsl:value-of select="."/>
      </a>
    </p>
    <xsl:text disable-output-escaping = "yes">
    </xsl:text>
```

```
  </xsl:template>

  <xsl:template match="/">
    <head>
      <title>Bookmarks aus dem Buch</title>
    </head>
    <body>
      <h1>Bookmarks aus dem Buch</h1>
      <xsl:apply-templates select=".//ulink"/>
  </body>
  </xsl:template>

</xsl:stylesheet>
```

Listing 12.4 Ein einfaches XSL-Stylesheet

Das Template `match="/"` wird für das oberste Tag des Dokuments aufgerufen. Hier wird das Gerüst für die HTML-Datei generiert. Außerdem werden alle `ulink`-Tags prozessiert, diese enthalten in DocBook-Dokumenten die URLs. Das erste Template `match="primarie/ulink"` filtert Index-Einträge heraus – diese verweisen nur innerhalb des Manuskripts.

Mit einem XSL-Prozessor können Sie das Stylesheet aus Listing 12.4 auf ein Dokument anwenden. Listing 12.5 zeigt die Verwendung von `xsltproc`.

```
(linux):~$ xsltproc --nonet extract-url.xsl master.xml >
bookmarks.html
```

Listing 12.5 Eine XSL-Transformation mit `xsltproc` durchführen

Mit Hilfe des Perl-Skripts `xpath` können Sie eine brauchbare Liste auch mit einem Shell-Kommando erhalten (Listing 12.6). Die Option `-q` bewirkt, dass nur die betreffenden Nodes (und keine Trennzeilen) ausgegeben werden. Nach der Option `-e` folgt die XPath-Anfrage.

```
(linux):~$ xpath -q -e //ulink master.xml
```

Listing 12.6 Auswerten von XML-Dateien mit `xpath`

12.6 Die Emacs-Erweiterung xslide

Wie sollte es anders sein, Emacs hat auch für die Bearbeitung von XSL-Dateien eine passende Erweiterung. `xslide` (*http://www.menteith.com/xslide/*) fügt bei einem öffnenden gleich ein schließendes Tag ein. Der Anwender kann mit der Funktion `xsl-process` das Stylesheet ablaufen lassen.

Neue Tags werden automatisch eingerückt und zur Darstellung werden verschiedene Farben verwendet. Zur einfachen Eingabe sind passende Abkürzungstabellen definiert. Mit diesem Mode ist es recht einfach, XSL-Dateien zu bearbeiten. Auf meinem System habe ich die Tastenkombination Strg+c Strg+e mit der Funktion `xsl-insert-tag` (Listing 12.7), diese Tastaturbelegung bin ich vom `psgml-mode` gewohnt. Außerdem habe ich den `abbrev-mode` aktiviert, so dass Emacs Tags automatisch vervollständigt.

```
(defun my-xsl-mode ()
  "XSL IDE setup"
  (local-set-key (kbd "C-c C-e") 'xsl-insert-tag)
  (abbrev-mode 1))

(add-hook 'xsl-mode-hook 'my-xsl-mode)
```

Listing 12.7 Anpassungen für den `xsl-mode`

12.7 Die DocBook-Anwendung

Docbook (*http://docbook.sf.net/*, *http://www.docbook.org/*) ist eine relativ komplexe SGML- bzw. XML-Anwendung. Sie wurde von verschiedenen Gruppen entwickelt, um computerbezogene Dokumentation strukturiert speichern und verarbeiten zu können. Heute wird Docbook auch von vielen Projekten zur Dokumentation eingesetzt, besonders prominente Vertreter sind KDE und GNOME.

Neben Büchern und Artikeln wird Docbook auch für Manpages verwendet. Diese können sowohl nach `nroff` konvertiert als auch in andere Dokumente eingebunden werden. Außerdem existieren Erweiterungen zur Gestaltung von Webseiten (*http://docbook.sourceforge.net/projects/website/index.html*) und Vorträgen (Folien, *http://docbook.sourceforge.net/projects/slides/index.html*). Wenn man sich an die Arbeit mit den verschiedenen Tags und Strukturen gewöhnt hat, ist der Unterschied zwischen den verschiedenen Anwendungen relativ gering.

Norm Walsh hat sowohl DSSSL- und XSL-Stylesheets für DocBook implementiert. Diese erstellen HTML, RTF und FO-Format. FO steht für Formatting Objects und ist wieder ein W3C-Standard. Die Stylesheets sind so entworfen, dass Anwender diese möglichst einfach anpassen können. Beispiele finden Sie in der Dokumentation und in der DocBook-FAQ (*http://www.miwie.org/docbook-dsssl-faq.html* und *http://www.dpawson.co.uk/docbook/styling/styling.html*).

Für DocBook können Sie nicht nur einen Texteditor wie Emacs verwenden. Viele andere Programme bieten ebenfalls Unterstützung für DocBook. Wenn Sie lieber mit einer WYSIWYG-Umgebung arbeiten, dann werfen Sie einen Blick auf aktuelle Versionen von Abiword (*http://www.abisource.com/*), Conglomerate (*http://www.conglomerate.org/*) oder Lyx (*http://www.lyx.org/*).

13 Emulatoren unter Linux

13.1 Emulatoren allgemein

Linux ist ein weitgehend POSIX-kompatibles Unix-System. Diese Kompatibilität erstreckt sich jedoch nur auf die Portabilität von C-Quelltexten zwischen verschiedenen Unix-Systemen. Das heißt, dass viele Unix-Programme unter Linux laufen, wenn sie dort neu übersetzt werden. Für freie Unix-Software ist das in der Regel einfach, anders sieht es bei kommerzieller Software oder Programmen anderer Systeme aus.

Dennoch kann man unter Linux viele Programme ablaufen lassen, die entweder für ein anderes Betriebssystem oder gar für eine andere Rechnerarchitektur erstellt wurden. Dies geschieht mit Hilfe so genannter Emulatoren, die die Funktionen anderer Hardware oder Betriebssysteme auf die Funktionen des Linux-Systems abbilden.

Durch die große Komplexität sowohl von Linux als auch von anderen Betriebssystemen und Prozessoren ist eine Emulation oft nicht vollständig. Das liegt häufig auch daran, dass Funktionen nachgebildet werden müssen, die unter anderen Betriebssystemen verfügbar sind und benutzt werden, aber nicht dokumentiert sind.

Dennoch stehen auch unter Linux eine Reihe von Emulatoren für verschiedene Betriebssysteme oder Rechnerarchitekturen zur Verfügung. Im Folgenden werden die Emulatoren für PC-Hardware und Windows genauer vorgestellt. Es existieren weitere Emulatoren für exotischere Systeme wie z. B. Macintosh, z-Series (S/390 Mainframes), Apple II oder Spektrum.

In der Anfangszeit waren die Emulatoren zum Teil sehr wichtig, so liefen Oracle und der ADSM-Client (zur Datensicherung) nur im iBCS2-Emulator – eine Linux-Version war nicht verfügbar. Heute ist bei vielen Software-Produkten Linux (für Intel) eine der ersten unterstützten Plattformen. In der Gegenrichtung entwickelten viele Unix-Hersteller Produkte oder Strategien, um Linux-Programme auf ihren Plattformen ausführbar zu machen oder die Portierung zumindest zu vereinfachen.

13.2 Der BIOS-Emulator dosemu

Der dosemu (*http://dosemu.sf.net/*) ist kein DOS-Emulator, wie der Name andeutet, sondern ein PC- und BIOS[1]-Emulator. Es werden Teile der Hardware und die wichtigsten Funktionen des BIOS emuliert. Daher ist zum Betrieb eine DOS-Lizenz notwendig. Das (echte) DOS läuft dann innerhalb eines Benutzerprozesses unter Linux. Es kann praktisch jede DOS-Version verwendet werden, sei es FreeDOS, MS-DOS, Novell-DOS (früher als DR-DOS bekannt) oder PC-DOS von IBM.

`dosemu` verwendet zur Emulation die Kernel-Funktion `vm86()`. Damit ist es möglich, verschiedene CPU-Befehle oder Speicherzugriffe abzufangen und die gewünschte Umgebung bereitzustellen. Die Kompatibilität zu einem echten DOS-System ist in vielen Bereichen erstaunlich hoch.

Seit der Freigabe der Version 1.0 erfolgt die weitere Entwicklung unter der als instabil deklarierten Version 1.1 – die Nummerierung folgt hier dem Beispiel des Linux-Kernels. Für viele Anwendungszwecke sollten Sie mit der stabilen Version auskommen, eine Binärversion ist in den meisten Distributionen enthalten.

13.2.1 Installation des Emulators

Sollten Sie eine neue Version des `dosemu` benötigen, so können Sie diese von der Webseite herunterladen. Anschließend müssen die Quellen auf dem Linux-Rechner ausgepackt und dann das Programm kompiliert und installiert werden. Zunächst sollten Sie den Emulator mit dem Befehl `default-configure` konfigurieren und dann kompilieren. Listing 13.1 zeigt dafür ein Beispiel.

```
(linux):~# cd /usr/src
(linux):/usr/src# tar zxvf /tmp/dosemu-1.1.3.tar.gz
(linux):/usr/src# cd dosemu-1.1.3
(linux):/usr/src/dosemu-1.1.3# ./default-configure
(linux):/usr/src/dosemu-1.1.3# make
```

Listing 13.1 Konfigurieren des `dosemu`

Das Skript `configure-default` erkennt automatisch, ob auf dem System das X-Window-System (mit Header-Dateien und Bibliotheken) installiert ist, und erstellt dann auch einen unter X lauffähigen Emulator. Weitere Einstellungen, die zur Übersetzungszeit durchgeführt werden müssen, werden auf Standardwerte gesetzt. Tabelle 13.1 zeigt eine Auswahl der möglichen Konfigurationsoptionen.

1. BIOS = Basic-Input/Output-System

Option	Bedeutung
--enable-novm86plus	Die neue Variante zur Umschaltung in den vm86-Modus wird nicht verwendet.
--with-x=[yes|no]	Mit/ohne X-Unterstützung
--enable-nomitshm	Kein Shared-Memory beim X-Zugriff einsetzen
--enable-dodebug	Debugging aktivieren
--enable-nosbemu	Soundblaster-Emulation nicht aktivieren
--enable-linkstatic	Statisch linken
--enable-runasroot	Installation als setuid-Programm

Tabelle 13.1 Optionen für das configure*-Skript des* dosemu

Jegliche weitere Konfiguration des Emulators kann durch die Konfigurationsdatei /etc/dosemu.conf bzw. ~/.dosemurc erfolgen, so dass es nicht notwendig ist, in der Datei Makefile-Änderungen vorzunehmen.

13.2.2 Konfiguration des Emulators

Nach der Installation müssen nun noch die Konfigurationsdateien für den Emulator angepasst werden. Die Konfiguration des DOS-Emulators kann jeder Benutzer selbst durchführen, indem er eine Datei .dosemurc in seinem Home-Verzeichnis erstellt. Der Systemadministrator kann eine systemweite Konfiguration für alle Benutzer vornehmen. Dazu dient die Datei /etc/dosemu.conf. Die Syntax beider Dateien ist identisch, als Beispiel kann die Datei ./etc/dosemu.conf aus der Quelldistribution des DOS-Emulators dienen.

Die Konfiguration sollte in mehreren Schritten erfolgen, so dass man nach jedem Schritt die Lauffähigkeit der Änderungen überprüfen kann. Benötigt wird eine DOS-Boot-Diskette, erstellt mit FORMAT A: /S, ohne die Dateien CONFIG.SYS und AUTOEXEC.BAT. Auf diese Diskette kopiert man alle Programme und Treiber aus dem Verzeichnis ./src/commands.

```
$_rawkeyboard = (0)      # kein RAW-Keyboard
$_layout = "de-latin1"   # Deutsche Tastaturbelegung
$_keybint = (on)         # Emulieren von Interrupts
```

Listing 13.2 Tastaturkonfiguration für dosemu

Zunächst benutzt man eine möglichst einfache Konfiguration. Die Wahrscheinlichkeit, dass dabei Probleme auftreten, ist dann relativ gering. Folgende Einstellungen sollten in der Konfigurationsdatei /etc/dosemu.conf im ersten Schritt angepasst werden:

- Die deutsche Tastaturbelegung muss für den Start auf der Konsole eingerichtet werden (siehe Listing 13.2). Dabei sollte nicht der sehr hardwarenahe Zugriff im `raw`-Modus benutzt werden, weil dieser dazu führen kann, dass nach einem Absturz des Emulators keine Tastatureingaben erfolgen können. Eine Anmeldung über eine Netzwerkverbindung oder ein serielles Terminal funktioniert aber auch dann noch.
- Zunächst sollte nur eine einfache, unvollständige Emulation der Bildschirmausgabe eingerichtet werden (siehe Listing 13.3). Eine inkompatible Einstellung kann zu einem Absturz des Emulators führen, nach dem keine Bildschirmausgaben mehr zu lesen sind. Das System insgesamt ist dann aber nicht abgestürzt, sondern nur die Bildschirmausgabe, so dass das System z. B. über eine Netzwerkverbindung oder ein serielles Terminal noch geordnet heruntergefahren und dann neu gestartet werden kann.
- Die einzigen Geräte, die vom Emulator direkt angesprochen werden können, sind die Diskettenlaufwerke. Diese Einträge sind an die eigene Rechnerausstattung anzupassen (Listing 13.4).

```
$_video = "vga"          # VGA-Bildschirm
$_console = (0)          # Keinen Konsolen-Zugriff
$_graphics = (0)         # Kein Grafikmodus
```

Listing 13.3 Tastaturkonfiguration für `dosemu`

```
$_floppy_a ="threeinch" # oder "fiveinch" or leer
$_floppy_b = ""         # dito for B:
```

Listing 13.4 Konfiguration der Diskettenlaufwerke

Nun kann man versuchen, den DOS-Emulator zum ersten Mal mit der DOS-Boot-Diskette zu starten. Dabei werden noch keine weiteren Partitionen, Festplatten oder unter Linux angehängte Partitionen zur Verfügung gestellt. Auch der Zugriff auf Drucker, Hardware-Erweiterungen oder Netzwerke ist nicht möglich.

Im nächsten Schritt kann man dann auf die normalen DOS-Partitionen zugreifen. Der bevorzugte Weg dazu ist die Verwendung des Redirektors `LREDIR.EXE` oder des Treibers `EMUFS.SYS`. Beide Möglichkeiten haben dieselbe Funktionalität, Laufwerke können sowohl in der Datei `CONFIG.SYS` als auch später in der Datei `AUTOEXEC.BAT` oder von der Kommandozeile aus zugeordnet werden. DOS-Programmen gegenüber erscheint das Laufwerk wie ein Netzwerklaufwerk.

Der Aufruf des Treibers `EMUFS.SYS` hat die Syntax `DEVICE=EMUFS.SYS` *`Linux-Pfad`* `[r]`. Damit wird dem nächsten freien Laufwerksbuchstaben der angegebene Pfad unter Linux zugeordnet. Wenn zusätzlich der Parameter `r` angegeben

wurde, so wird das Laufwerk mit einem Schreibschutz versehen. Ein Beispiel finden Sie in Listing 13.5.

```
DEVICE=A:\EMUFS.SYS /mount/dos/c
DEVICE=A:\EMUFS.SYS / r
```

Listing 13.5 Virtuelle Laufwerke im dosemu

Das Programm LREDIR.EXE (Listing 13.6) erfüllt denselben Zweck und kann auch von der Kommandozeile aus benutzt werden. Die Befehle in Listing 13.6 würden dasselbe Ergebnis wie im Beispiel Listing 13.5 liefern.

```
A:LREDIR C: LINUX\MOUNT\DOS\C
A:LREDIR D: LINUX\
```

Listing 13.6 Virtuelle Laufwerke mit LREDIR

Es können auch Laufwerkszuordnungen wieder gelöscht werden, indem der Befehl LREDIR DEL *Laufwerk* eingegeben wird. Der Befehl LREDIR ohne Parameter gibt eine Übersicht über die Laufwerkszuordnungen aus.

Diese Möglichkeit des Zugriffs ist immer dann sinnvoll, wenn auf die entsprechenden Partitionen auch von Linux aus zugegriffen wird. Sind die Partitionen bisher nicht angehängt, so ist dies ein geeigneter Zeitpunkt, das zu tun.

Sollten die Partitionen unter Linux nicht verfügbar sein, weil sie mit einem »Festplatten-Verdoppler« wie Stacker oder DoubleSpace behandelt wurden, so bleibt als einzige Möglichkeiten der direkte Zugriff auf eine einzelne Partition. Ein Beispiel dafür finden Sie in Listing 13.7.

```
$_hdimage = "/dev/hda1"     # für den Zugriff
                            # auf die Partition /dev/hda1
$_hdimage = "/dev/hda2:ro" # für den Lesezugriff
```

Listing 13.7 Zugriff auf Partitionen

Greifen Sie niemals mit dosemu direkt auf unter Linux angehängte Partitionen zu! Andernfalls kann es zu erheblichen Fehlern in der Dateisystemstruktur kommen und Daten können verloren gehen.

Es ist sinnvoll, wenn unter dem dosemu die Laufwerksbuchstaben denen entsprechen, die auch unter echtem DOS verwendet werden. Als sinnvoll hat sich die Verwendung einer Boot-Diskette erwiesen, deren Image auf der Linux-Platte abgelegt wird. Die Konfiguration erfolgt mit dem Schlüsselwort bootdisk. Diesem virtuellen Laufwerk ist dann der Laufwerksbuchstabe A: zugeordnet. Durch das Programm BOOTOFF kann aber dieses Laufwerk auf die mit dem ersten floppy festgelegte Konfiguration umgeschaltet werden; das Boot-Image wird vom Diskettenlaufwerk überdeckt. Der umgekehrte Weg, also das Über-

decken des Diskettenlaufwerks mit dem Boot-Image, ist mit `BOOTON` möglich. Programme wie `MSD`, die die Startdateien analysieren wollen, benötigen den Zugriff auf das Boot-Image.

Für die Darstellung von Grafiken haben Sie zwei Möglichkeiten: Entweder verwenden Sie unter einer X-Oberfläche `xdosemu` oder Sie gestatten dem Emulator direkten Zugriff auf die Grafikarte. Aus Sicherheitsgründen, der besseren Stabilität des Gesamtsystems wegen und weil X in den meisten Fällen bereits verwendet wird, bietet sich `xdosemu` an.

Wenn Sie DOS-Gafikprogramme auch auf der Konsole verwenden wollen, dann müssen Sie dies gesondert einrichten. Dazu ist es notwendig, dass der Emulator Zugriffsrechte erhält, die nur der Systemadministrator hat. Das erfolgt in der Standardinstallation durch das `s`-Bit. Dadurch wird der Emulator mit den Rechten des Systemadministrators (`setuid-root`) gestartet, jedoch wird für alle Zugriffe, die nur normale Benutzerrechte erfordern, auf die `uid` des Benutzers zurückgeschaltet, der den Emulator gestartet hat.

Wichtig ist außerdem, dass ein eventueller Shadow des VGA-BIOS ausgeschaltet wird. In der Datei `etc/dosemu.conf` sind mit dem Schlüsselwort `video` eine Reihe von Beispielen für verschiedene Hardware-Konfigurationen aufgeführt. Die meisten modernen Karten werden nicht mehr unterstützt, so dass Sie die X-Oberfläche nutzen müssen.

Nun kann weitere Hardware, wie z. B. die Maus oder der Zugriff auf die seriellen Schnittstellen, konfiguriert werden. Für alle Einstellungen findet man bereits ausführlich kommentierte Beispiele in der Konfigurationsdatei.

13.2.3 Überlegungen zur Systemsicherheit

Der Emulator wird, wenn er viele sehr hardwarenahe Zugriffe durchführen muss, als `setuid`-Programm installiert. Damit laufen viele Zugriffe mit den Rechten des Systemadministrators `root` ab. Dies wird in einem echten, vernetzten Mehr-Benutzersystem nicht gern gesehen. Hier sollte man überlegen, ob sich direkte Hardwarezugriffe durch eine entsprechende Konfiguration des Emulators nicht völlig unterbinden lassen. Distributoren sollten eine minimale, lauffähige, aber sichere Konfiguration vorgeben.

Der Vorschlag ist, jedem Benutzer ein eigenes `hdimage` anzulegen und gegebenenfalls via Redirector auf ein Home-Verzeichnis zugreifen zu lassen. Damit sind Benutzer voneinander unabhängig, der Systemverwalter kann allerdings DOS-Programme nicht mehr zentral installieren.

13.2.4 Unterstützung bei der Fehlersuche

Der Emulator bietet die Möglichkeit, verschiedene Informationen zur Fehlersuche auszugeben. Dazu kann zum einen in der Konfigurationsdatei die entsprechende Ausgabe mit dem Schlüsselwort `debug` eingeschaltet werden. Eine andere Möglichkeit ist die Angabe als Parameter. Im folgenden Beispiel werden alle Meldungen ausgegeben und in die Datei `dos.log` geschrieben. Diese Datei kann bei der Suche nach Fehlern im Emulator oder in der Konfigurationsdatei eine große Hilfe sein. Die Erstellung von Debug-Ausgaben kann auch bei laufendem Emulator mit dem Programm `DOSDBG` ein- oder ausgeschaltet werden. Diese Dateien können in sehr kurzer Zeit erschreckend groß werden, wenn Sie wie in Listing 13.8 alle Ausgaben aktivieren.

```
(linux):~$ dos -D+a -o dos.log
```

Listing 13.8 Debug-Ausgabe beim Starten des `dosemu`

13.2.5 Benutzung

Als Benutzer startet man den DOS-Emulator mit dem Befehl `dos`. Danach wird in der virtuellen Maschine DOS gestartet. Nach dem Abarbeiten der Startdateien kann wie unter DOS gewohnt gearbeitet werden. Dennoch sind einige Dinge zu beachten:

- Programme, die auf ein installiertes CD-ROM-Laufwerk über die MSCDEX-Schnittstelle zugreifen, können verwendet werden, wenn man die CD nicht unter Linux anhängt, sondern den `dosemu`-Treiber `cdrom.sys` lädt. Dazu ist es notwendig, einen symbolischen Link `/dev/cdrom` auf das betreffende CD-ROM-Gerät anzulegen. Nähere Informationen dazu finden Sie in der Datei `README.CDROM`.
- Netzwerkzugriffe sind über einen virtuellen Paket-Treiber möglich. Das gilt sowohl für Zugriffe auf ein TCP/IP- als auch auf ein Novell-Netz mit IPX als Protokoll. Da der Linux-Kernel auch das IPX-Protokoll unterstützt, kann der `dosemu` dieses Protokoll direkt verwenden. Genauere Informationen zur Konfiguration des Emulators im Netz finden Sie im Abschnitt 13.2.7, »Netzwerke«.
- Wenn der DOS-Emulator auf einer virtuellen Konsole gestartet und die Einstellung `rawkeyboard` benutzt wird, so wird die virtuelle Konsole wie unter X mit der Tastenkombination Alt+Strg+Fn gewechselt.

13.2.6 Technik und interne Abläufe

Mit dem DOS-Emulator steht ein Programm zur Verfügung, das die Hardware und das BIOS eines IBM-kompatiblen PCs emuliert. Diese Emulation ist so leistungsfähig, dass jedes handelsübliche DOS im Emulator gestartet werden kann. Um diese Kompatibilität zu erreichen, wurden von den Programmierern eine Reihe von speziellen Funktionen der Intel-Prozessoren ab 386 und des Linux-Systems verwendet.

Grundsätzliches zur Emulation

Bei einem Multitasking-Betriebssystem darf ein Benutzerprozess nur unter ganz bestimmten Voraussetzungen die Hardware direkt ansprechen. Unter DOS hingegen benutzt fast jedes Programm nicht nur die Programmierschnittstelle über den Interrupt `21h`, sondern auch die Möglichkeit, auf die Hardware direkt zuzugreifen. Das gilt sowohl für den Zugriff auf den Bildschirmspeicher als auch für die Programmierung der sonstigen Hardware, wie z. B. der seriellen Schnittstellen.

Daher wird durch den DOS-Emulator der Prozessor in den so genannten virtuellen 8086-Modus versetzt. In diesem Modus löst jede unter einem Multitasking-System unzulässige Operation einen Interrupt aus, so dass das Betriebssystem den DOS-Emulator anstößt, um die betreffende Funktion zu emulieren. Privilegierte Befehle sind z. B. Zugriffe auf IO-Ports und das Auslösen von Interrupts.

Speicher im Emulator

Der Emulator kann für das virtuelle System jede unter DOS bekannte Art von Speicher, also EMS (Expanded Memory), XMS (Extended Memory), UMB (Upper Memory Blocks), HMA (High Memory Area) und den Zugriff über DPMI (DOS Protected Mode Interface) emulieren. Da dieser Speicher im virtuellen Speicher des Linux-Systems emuliert wird, kann mehr Speicher, als tatsächlich verfügbar ist, unter dem Emulator benutzt werden. Dies geht natürlich auf Kosten der Geschwindigkeit, da dieser Speicher nur virtuell durch Swapping zur Verfügung steht.

Der unterhalb von 1 Mbyte liegende Speicher wird wie unter DOS üblich benutzt. Dazu wird der Emulator oberhalb der 1 Mbyte-Adresse in den Speicher geladen. Bei Verwendung des `a.out`-Binärformats wurde dies durch eine Bibliothek erreicht, die an eine »hohe« Adresse geladen wurde. Mit dem ELF-Binärformat ist dieser Trick nicht mehr notwendig.

Es ist auch möglich, den physischen Speicherbereich unter 1 Mbyte direkt in den Adressraum des DOS-Emulators einzublenden. Dann ist es möglich, aus dem Datenbereich einer Erweiterungskarte Daten auszulesen oder in diesem Bereich abzulegen. Diese Speicherzuordnung kann mit dem Schlüsselwort `hardware_ram` konfiguriert werden.

Für alle hier erwähnten Speicherarten kann eine genauere Konfiguration erfolgen. Beispiele dafür finden Sie in der Konfigurationsdatei. Wenn Sie EMS verwenden, dann müssen Sie in der CONFIG.SYS den Treiber EMS.SYS aus dem Verzeichnis ./commands laden.

Eingabegeräte für den Emulator

Als Eingabegeräte stehen dem Benutzer wie unter DOS die Tastatur und die Maus zur Verfügung. Die Tastatur kann auf verschiedene Arten benutzt werden, wobei eine Reihe von Besonderheiten sowohl bei Unix als auch bei DOS zu beachten sind.

Die beste Emulation der Tastatur (auf der Konsole) wird aktiviert, indem in der Datei /etc/dosemu.conf der rawkeyboard-Modus eingestellt wird. Dadurch können DOS-Programme direkt auf die an der Konsole angeschlossene Tastatur zugreifen. Unter X werden die Tastatur-Events durch den Emulator in die entsprechenden Scan-Codes der Tastatur übersetzt. Der DOS-Emulator kann auch über ein Netzwerk oder auf einem seriellen Terminal benutzt werden. Hier ist es dann wie unter X nicht möglich, die Tastatur im Raw-Modus zu benutzen, so dass auch hier eine Umsetzung erfolgen muss.

Die Maus kann ebenfalls in verschiedenen Modi benutzt werden:

- Unter X sollte der interne Maustreiber verwendet werden, der wie unter Windows oder OS/2 dafür sorgt, dass X-Mausereignisse in die entsprechenden DOS-Mausbewegungen übersetzt werden.
- Auf der Konsole kann entweder auch der interne Maustreiber benutzt oder die entsprechende serielle Schnittstelle für den Emulator freigeschaltet und dann ein normaler DOS-Treiber verwendet werden. Da diese Treiber häufig sehr empfindlich auf Timing-Probleme reagieren, sollte man falls erforderlich die in der HowTo erwähnten Treiber verwenden.
- Bei der Verwendung des Emulators über ein Netzwerk oder ein serielles Terminal kann keine Maus benutzt werden.

Ausgabe

Der DOS-Emulator kann eine Reihe von Ausgabegeräten benutzen. Das beginnt mit den verschiedenen Möglichkeiten der Bildschirmausgabe:

- Der DOS-Emulator kann auf der Konsole benutzt werden und dann sowohl Text- als auch Grafikmodi verwenden, sofern Ihre Grafikkarte unterstützt wird.

- Unter X kann derzeit nur der Textmodus benutzt werden, die Maus jedoch arbeitet »seamless«[2] für DOS-Programme.
- Der DOS-Emulator kann sowohl im Netz als auch über ein serielles Terminal verwendet werden. Hier wird eine spezielle Terminal-Ansteuerung (`slang`) benutzt, so dass auch über langsame Verbindungen ein vernünftiges Arbeiten möglich ist.

Ein weiteres Ausgabegerät ist der Drucker. In der Datei `/etc/dosemu.conf` können bis zu drei Drucker konfiguriert werden. Dabei kann entweder die Ausgabe in einer Datei gespeichert oder direkt an ein Programm weitergeleitet werden. Eine Möglichkeit ist hier die Verwendung von `lpr`, so dass Druckausgaben durch das Linux-Spool-System verwaltet werden. Dort ist dann mit GhostScript auch die Ausgabe von PostScript-Dateien auf nicht PostScript-fähige Drucker möglich. Aufgrund der beschränkten Speichergröße ist dies unter DOS selbst praktisch unmöglich. Es werden Druckausgaben verarbeitet, die über den DOS- oder BIOS-Drucker erstellt wurden. Das trifft für die meisten Programme zu. Unter Windows 3.0 gibt man als Druckerschnittstelle `LPT1.DOS` oder `LPT1.OS2` an.

Disketten und Festplatten

Der Emulator entfaltet erst dann seine volle Leistungsfähigkeit, wenn der Zugriff auf die gespeicherten Daten und installierten Programme genauso erfolgen kann wie unter direkt gestartetem DOS. Dabei sollten sowohl alle Partitionen verfügbar sein als auch die Zuordnung zu Laufwerksbuchstaben im Emulator den Laufwerksbuchstaben unter DOS entsprechen. Der Zugriff auf Disketten und Festplatten ist auf verschiedene Weisen möglich:

- Der Treiber `EMUFS` und das Programm `LREDIR` sind in der Lage, ein DOS-Laufwerk zu erstellen, das von DOS wie ein Netzwerklaufwerk behandelt wird. Dazu sind keine Zugriffe außerhalb des Linux-Dateisystems notwendig. Mit diesem Verfahren kann der DOS-Emulator auf alle unter Linux anhängbaren Partitionen zugreifen. Das ermöglicht auch den Lesezugriff auf HPFS-Partitionen von OS/2.
- Der direkte Zugriff auf eine einzelne Partition (Direct Partition Access, DPA) simuliert eine Festplatte, die nur aus dieser einen Partition besteht. Partitionen, die auch mit DPA angesprochen werden, dürfen auf keinen Fall gleichzeitig unter Linux angehängt sein, die Dateisysteme könnten beschädigt werden!
- Sowohl Disketten als auch Festplatten können durch eine entsprechend große Datei im Linux-Dateisystem simuliert werden. Man spricht dann von einem Disk-Image, das z. B. mittels `dd` erstellt werden kann. Eine sinnvolle Verwen-

2. Die Maus sendet, wenn der Mauscursor im XDos-Fenster ist, die Mausereignisse an das DOS-Programm.

dung ist ein Image für den Start des DOS-Betriebssystems, so dass keine Diskette mehr eingelegt werden muss. Dadurch wird der Start erheblich beschleunigt.

- Die Diskettenlaufwerke können direkt verwendet werden.
- CD-ROM-Laufwerke können, sofern sie von Linux unterstützt werden, auf zwei Weisen verwendet werden:
  - Die CD kann unter Linux angehängt und dann über den Netzwerk-Redirektor (`EMUFS` bzw. `LREDIR`) angesprochen werden.
  - Es existiert ein DOS-Treiber, der innerhalb des DOS-Emulators ein mit `MSCDEX` ansprechbares CD-ROM-Laufwerk simuliert.

13.2.7 Netzwerke

Mit dem DOS-Emulator ist auch der Zugriff auf Netzwerke möglich. Dazu ist je nach verwendetem Protokoll eine unterschiedliche Vorgehensweise notwendig, die im Folgenden genauer erläutert werden soll.

dosemu im TCP/IP-Netz

Linux verfügt über eine komplette Ausstattung mit Netzprogrammen wie `ftp`, `telnet` und anderen. Daher erscheint es zunächst nicht sinnvoll, diese Programme unter dem DOS-Emulator zu betreiben. Es gibt jedoch eine Reihe von Funktionen, die ohne einen TCP/IP-Protokoll-Stack im Emulator nur schwer zu realisieren wären.

Im DOS-Emulator steht eine komplette Packet-Schnittstelle zur Verfügung. DOS-basierte Netzprogramme (Client und Server) lassen sich so auf einem Rechner implementieren und testen. Dabei steht ein eigenes (virtuelles) Subnetz zur Verfügung (siehe Abbildung 13.1). Damit muss ein Entwickler nicht mehr über mehrere Rechner verfügen und hat keine Konsistenzprobleme, da alle Programme auf einer gemeinsam genutzten Platte gespeichert werden können.

Für diese Funktion müssen mehrere Protokoll-Stacks auf dem Linux-Rechner implementiert werden. Durch den Einsatz einer oder zweier weiterer Netzwerkkarten können jedoch nur wenige Stacks zusätzlich verwaltet werden. Daher wird ein virtuelles Netzwerk implementiert, in dem die Pakete zu den Emulatoren geroutet werden. Den Emulatoren wird ein eigenes Subnetz zugeordnet, das dann in das reale Netz geroutet wird. Der administrative Aufwand ist hier nicht zu unterschätzen, jedoch können DOS-Programme dann problemlos alle Ressourcen im TCP/IP-Netz nutzen.

Zunächst soll den Emulatoren ein eigenes Subnetz zugeordnet werden. Für ein internes Netz, dessen Daten nicht nach außen (ins Internet) geroutet werden sollen, stehen eine Reihe von Netznummern zur Verfügung. Aus diesen in [rfc1918]

festgelegten Netzen sollte ein Class-C-Netz gewählt werden. In diesem Beispiel wird das Netz 192.168.xxx.0 verwendet. Der lokale Rechner hat die Adresse aaa.bbb.ccc.ddd.

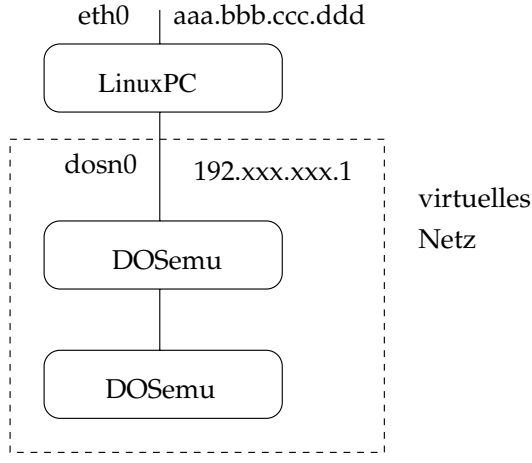


Abbildung 13.1 Netzschema für TCP/IP im DOSemulator

Auf dem lokalen Rechner muss nun das Routing zu den virtuellen DOS-Rechnern aktiviert werden. Zunächst muss ein neues (virtuelles) Netzwerkgerät implementiert werden, über das die Daten zu den DOS-Emulatoren geroutet werden. Im Verzeichnis `./src/dosext/net/v-net` befindet sich ein Kernel-Modul, das das entsprechende Gerät implementiert. Nach dem Übersetzen des Moduls mit `make` und dem Laden des Moduls mit dem Programm `insmod` muss das entsprechende Interface mittels `ifconfig` konfiguriert werden und mit `route` die zugehörige Route eingerichtet werden (Listing 13.9).

```
ifconfig dsn0 192.xxx.xxx.1 broadcast 192.xxx.xxx.255 \
         netmask 255.255.255.0
route add -net 192.xxx.xxx.0 dsn0
```

Listing 13.9 Konfiguration des virtuellen Netzwerk-Interface

Dieses Module ordnet jedem `tty` eine virtuelle Ethernet-Adresse zu, die z. B. für die Zuordnung einer Internetadresse mittels `bootp` verwendet werden kann. Problematisch ist diese Zuordnung dann, wenn mehrere Emulatoren, z. B. unter X, aus demselben Fenster gestartet wurden.

Die virtuelle Ethernet-Adresse besteht aus einem fixen und einem variablen Teil. Die Adresse hat die Form `64:62:`*`Variable`*`:78:78:78`. Die Variable errechnet sich gemäß Tabelle 13.2 aus dem `tty`, von dem der Emulator aus gestartet wurde.

tty	drittes Adressfeld
tty*n*	*n*
ttyp*n*	0xC0 + *n*
ttyS*n*	0x40 + *n*

Tabelle 13.2 Virtuelle Ethernet-Adressen

Wenn Sie mit dem Emulator nicht nur auf Ressourcen des eigenen Rechners zugreifen wollen, müssen Sie auf dem anderen Rechner ebenfalls die Routing-Tabellen anpassen. Dort können Sie für das den Emulatoren zugeordnete virtuelle Netz eine statische Route zu Ihrem Rechner eintragen (Listing 13.10).

```
route add -net 192.xxx.xxx.0 gw aaa.bbb.ccc.ddd
```

Listing 13.10 Routen zum virtuellen DOS-Rechner

Innerhalb des Emulators können dann die bekannten Programme, die auf einem Packet-Treiber aufsetzen, verwendet werden. Sinnvoll kann hier die Verwendung eines Multiplexers wie PKTMUX sein, genauso wie die Verwendung von SOSS oder XFS.

dosemu im IPX-Netz

Der dosemu kann auf zwei verschiedene Weisen Zugang zu einem Novell-Netz mit IPX-Protokoll erlangen. Novell verwendet drei verschiedene Frame-Typen (Ethernet_II, Ethernet_802.2 und Ethernet_802.3). Die Einsatzbereiche und Möglichkeiten, Linux-Rechner einzubinden, sind unterschiedlich, so dass Sie mit Ihrem Netzwerkadministrator Kontakt aufnehmen sollten.

- Ethernet_II wird von Novell verwendet, wenn der Server das IP-Protokoll unterstützt. Auch das IPX-Protokoll kann mit diesem Frame-Typ genutzt werden. Dies macht den Anschluss eines Linux-Rechners relativ einfach.
- Der Frame-Typ Ethernet_802.3 wurde von Novell verwendet, bevor das VLM-System bzw. NetWare 4.x eingeführt wurde. Bei diesem Frame-Typ werden keine IP-Pakete geroutet.
- Heute empfiehlt Novell die Verwendung von Ethernet_802.2. Dies ist auch der Standard für neu eingerichtete Server oder Netze. Dieser Frame-Typ wird vom dosemu-Packet-Treiber nicht unterstützt, so dass Sie hier entweder einen anderen Frame-Typ im Netz einrichten oder das Linux-IPX verwenden müssen.

Die Konfiguration von Linux-IPX erfordert nur wenige Schritte und hat sich in der Praxis durch die deutlich höhere Geschwindigkeit und die Unterstützung aller Frame-Typen besser bewährt. Zunächst muss der Kernel das IPX-Protokoll unterstützen. Wird beim Booten von Linux keine entsprechende Meldung aus-

gegeben, so müssen Sie den Kernel neu konfigurieren und kompilieren. Dabei muss die Frage nach »The IPX protocol« mit »y« beantwortet werden. Ebenfalls notwendig sind die Programme, die im Verzeichnis `./ipxutils` in den Quellen des Emulators zu finden sind. Mit dem Programm `ipx_interface` können Sie auf Ihrer Netzkarte das IPX-Protokoll aktivieren (Listing 13.11).

```
ipx_interface add -p eth0 802.2
```

Listing 13.11 Erzeugen eines IPX-Interface unter Linux

Ersetzen Sie im obigen Beispiel das Gerät und den Frame-Typ durch das bei Ihnen verwendete Gerät bzw. den entsprechenden Frame-Typ (`802.2`, `802.3` oder `EtherII`). Als letzten Schritt tragen Sie in der Datei `dosemu.conf` die Zeile `$_ipxsupport = (on)` ein. Damit ist die Konfiguration auf der Linux-Seite abgeschlossen. Wenn Sie den Emulator gestartet haben, starten Sie nur die Netware-Shell `NETX.EXE` bzw. `VLM.EXE`. Sie benötigen kein `LSL`, `IPXODI` oder andere Programme.

Bei der zweiten Möglichkeit, Anschluss an ein IPX-Netz zu finden, nutzen Sie den virtuellen Packet-Treiber des Emulators. Dazu ist kein Kernel-Support (außer dem allgemeinen TCP/IP) notwendig. Im Emulator müssen Sie dann die Novell-Programme aus Listing 13.12 starten.

```
LSL
PDETHER
IPXODI
VLM         oder NETX
```

Listing 13.12 Starten der Packet-Treiber im `dosemu`

Das Programm `PDETHER` ist ein ODI-Treiber, der auf einem Packet-Treiber aufsetzt. Damit ist es möglich, über den internen Packet-Treiber des DOS-Emulators IPX-Pakete zu übertragen. Dieser Weg funktioniert nur für die Frame-Typen Ethernet II und 802.3. Bei Letzterem muss in der Konfiguration die Zeile `pktdriver novell_hack` aktiviert werden. Für die Verwendung eines ODI-Treibers muss die Datei `NET.CFG` entsprechend angepasst werden.

13.2.8 Weitere Entwicklungen

Vom großen Erfolg von Linux als freiem Betriebssystem und vom Bedarf nach einem frei verfügbaren DOS inspiriert, hat sich eine Gruppe zusammengefunden, die an dem `FreeDOS`-Projekt (*http://www.freedos.org/*) arbeitet. Damit ist ein freies DOS verfügbar, das auch jetzt noch gepflegt wird, nachdem Microsoft die aktive Entwicklung von DOS eingestellt hat. Einige Distributionen enthalten auch ein FreeDOS, so dass unmittelbar nach der Installation des `dosemu` eine vollständige DOS-Umgebung bereitsteht.

dosemu verwendet Interna der Intel-Umgebung und emuliert keine CPU, so dass er nicht auf anderen Hardware-Plattformen verwendet werden kann. Der Emulator Bochs (*http://www.bochs.org/*) stellt auch auf anderen Prozessoren eine (virtuelle) Intel-CPU bereit. Die Performance ist relativ langsam, da alle CPU-Befehle emuliert werden müssen.

Wiederum nur für Intel-Plattformen existiert plex86 (*http://www.plex86.org/*), dies ist eine freie Implementierung eines virtuellen PCs unter Linux. Ein Kernel-Modul ist erforderlich, dafür ist die Geschwindigkeit gut, insbesondere im Vergleich zu Bochs.

13.3 Der Windows-Emulator wine

Der Windows-Emulator wine (Wine Is Not an Emulator oder WINdows Emulator, *http://www.winehq.com/*) setzt Aufrufe des MS-Windows[3]-API (Application Programming Interface) in äquivalente X-Aufrufe um. Damit ist es möglich, dass Windows-Programme unter X ablaufen.

Die Entwicklung des Windows-Emulators wine begann, als Sun die Auslieferung seines Windows-Emulators Wabi ankündigte (die Weiterentwicklung scheint schon lange eingestellt zu sein). Einige erfahrene Linux-Programmierer begannen unmittelbar mit der Entwicklung des Emulators. Bereits die ersten Versionen von wine wurden auf die frei verfügbaren *BSD-Systeme portiert. Die erste und wichtigste Applikation, die unter wine lauffähig war, war das altbekannte Solitär aus Windows 3.1.

Der Emulator soll einmal eine komplette Windows-Emulation werden. Derzeit werden jedoch noch einige DLLs (Dynamic Link Libraries) aus der Windows-3.1-Umgebung benutzt. Neben der Kompatibilität mit dieser Version wird auch die Win32-Schnittstelle implementiert, damit sind auch Programme für Windows 95/98, NT und Windows 2000 unter wine lauffähig. Ein weiteres Ziel ist die Entwicklung einer Bibliothek (libwine), mit der Windows-Programme einfach nach Unix und X portiert werden können.

Das Programm wine ist noch kein vollständiger Windows-Emulator. Die Autoren implementieren Schritt für Schritt alle Windows-APIs. Solange noch nicht alle Windows-DLLs vollständig und fehlerfrei implementiert sind, kann per Kommandozeilenoption das Laden der Original-DLL verlangt werden. Ein wichtiger Meilenstein ist die Lauffähigkeit der MS-Office-Programme. Noch laufen nicht alle Windows-Programme, auf absehbare Zeit wird das auch noch so bleiben. Auf der anderen Seite sind viele Programme recht gut benutzbar und einige kommerzielle Software wurde bereits mit Hilfe der Wine-Library »portiert«.

3. Wenn im Folgenden von Windows die Rede ist, dann gilt dies für Microsoft Windows. X ist eine Kurzform für das X-Window-System.

Der Emulator `wine` liegt nicht unter der GPL, sondern wird unter der GNU Lesser General Public License (LGPL) vertrieben. Damit möchte man einerseits die Portierung kommerzieller Programme nach Linux ermöglichen, auf der anderen Seite aber verhindern, dass einzelne Unternehmen proprietäre Versionen von Wine ableiten. Ältere Versionen standen unter dem liberaleren BSD- oder X-Copyright, nach reiflicher Überlegung haben die Entwickler die Lizenz geändert – ein Teil der Entwickler führt allerdings unter dem Namen `rewind` die Entwicklung unter der alten Lizenz weiter (*http://rewind.sf.net/*).

13.3.1 Kompilation und Installation

Auch von `wine` existiert keine Binärdistribution, daher muss zur Übersetzung der Quelltexte der C-Compiler installiert sein. Die Installation von `wine` erfordert ebenfalls, dass das X-Window-System mit den zugehörigen Bibliotheken und Include-Dateien installiert ist. Die Installation von `wine` ist genauso einfach wie die der GNU-Programme und beschränkt sich auf `./configure` und `make`.

Zurzeit wird parallel zum Emulator-Programm `wine` auch die Kompatibilitätsbibliothek `libwine` erzeugt. Es bleibt zu hoffen, dass parallel zur Entwicklung des Emulators auch an der Bibliothek gearbeitet werden kann und damit das Portieren von Windows-Programmen nach Unix auf beliebigen Hardware-Plattformen einfacher wird. Nach der Übersetzung wird `wine` mit dem Kommando `make install` installiert, in der Standardeinstellung unter `/usr/local`.

13.3.2 Konfiguration des Emulators

Vor dem ersten Start des Emulators muss dieser noch entsprechend konfiguriert werden. Als Standard wird die Datei `wine.conf` im Verzeichnis `/usr/local/etc` verwendet. Jeder einzelne Benutzer kann den Emulator nach eigenen Bedürfnissen mit Hilfe der Datei `~/.winerc` konfigurieren. Beide Dateien sind identisch aufgebaut. Es existiert auch eine Beispielkonfiguration mit dem Namen `wine.ini` im `wine`-Verzeichnis. Für die Emulation aktueller Windows-Versionen muss noch die (emulierte) Registry mit einigen Standardeinträgen gefüllt werden, dazu können Sie das `wine`-Programm `regapi` verwenden. Die aktuell notwendigen Registry-Einträge finden Sie in der Datei `winedefault.reg`. Sie können zur Übersetzung und Einrichtung auch das Skript `tools/wineinstall` verwenden, das Sie durch den Prozess begleitet.

Zum Start der ersten Programme sind die Zuordnung von Laufwerksbuchstaben zu Linux-Verzeichnissen und der Suchpfad nach Programmen und DLLs wichtig. Dazu enthält die Datei `wine.conf`, deren Syntax an die der ini-Dateien von Windows erinnert, verschiedene Abschnitte:

```
[Drive]
Path=/home/dos
Type=hd
Label=DOS-PLATTE
```

Listing 13.13 Definition von virtuellen Festplatten für `wine`

- Die Zuordnung von Laufwerksbuchstaben zu Verzeichnissen im Dateisystem bzw. zu Geräten wie zum Beispiel Diskettenlaufwerken erfolgt in je einem Abschnitt `Drive` je Laufwerk. Dabei können sowohl Geräte als auch Verzeichnisse des Linux-Verzeichnisbaums angegeben werden.
- Der Suchpfad nach Programmen oder DLLs und andere Pfade werden im Abschnitt `wine` konfiguriert. Die Laufwerke sind dabei gemäß der in den `Drive`-Abschnitten getroffenen Regeln zugeordnet.
- Die Konfigurationsdatei enthält noch weitere Abschnitte zur Auswahl von Fonts (`fonts`), der seriellen und parallelen Schnittstellen (`serialports` und `parallelports`) und der Steuerung von Ausgaben zur Fehlersuche (`spy`). Zunächst können diese Abschnitte unverändert bleiben. Mehr hierzu finden Sie in der Datei `./documentation/fonts` in den `wine`-Quellen.

```
Windows=c:\windows
System=c:\windows\system
Temp=c:\tmp
Path=c:\windows;c:\windows\system;C:\bin\win
```

Listing 13.14 Definition der wichtigsten Windows-Verzeichnisse

Damit ist die Konfiguration zunächst abgeschlossen und das erste Windows-Programm kann gestartet werden. Eines der beliebtesten Windows-Programme ist Solitär, eine Patience. Viel Vergnügen beim Spielen!

```
(linux):~$ wine sol
```

Listing 13.15 Aufruf des Windows-Solitär-Programms

13.3.3 Die Technik der Emulation

Zur Emulation des Windows-API unter Linux und X sind eine Reihe von Problemen zu lösen. Hier sollen die Konzepte, die jeweils die Lösung bestimmen, kurz erläutert werden. Für einen vertieften Einblick steht Ihnen wie bei fast allen Programmen der Quellcode zur Verfügung.

Zunächst müssen die Windows-Programme zur Ausführung in den Speicher geladen werden. Da Linux und *BSD ein vollkommen anderes internes Format für Programme (nämlich ELF bzw. `a.out`) statt des Windows-PE-Formats verwenden, mussten die dafür notwendigen Routinen neu entwickelt werden. Diese

Routinen laden sowohl die Programme als auch die verwendeten DLLs in den Speicher. All dies geschieht im User-Space, so dass hier, anders als beim iBCS2-Emulator, keine Veränderung des Kernels notwendig ist. Sie können jedoch mit dem `binfmt_misc`-Modul einen Handler für Windows-Programme einrichten, so dass diese automatisch mit Hilfe von `wine` gestartet werden.

Die einzelnen Segmente eines Windows-Programms werden in den Speicher geladen und ein Selektor, der auf dieses Segment zeigt, wird erzeugt. Windows-Programme können (und müssen) die Deskriptortabellen der Intel-Prozessoren i386 und höher (oder der kompatiblen Prozessoren) manipulieren. Dazu war es notwendig, im Kernel einen neuen System-Call einzuführen, mit dessen Hilfe man die lokale Deskriptortabelle (LDT) verändern kann. Ein normaler Prozess darf dies nicht direkt tun, ohne die Integrität des gesamten Systems zu gefährden.

Außerdem müssen sowohl das Windows-Programm als auch der Emulator auf die Datenbereiche wie den globalen oder lokalen Windows-Heap zugreifen können. Da Windows-3.x-Programme im 16-Bit-Modus und der Emulator im 32-Bit-Modus läuft, ist hier eine geeignete Adressumsetzung notwendig. Die Adressen der entsprechenden Speicherbereiche lassen sich nicht direkt ineinander umrechnen. Damit können Windows-Programme nicht jede beliebige Adresse ansprechen. Auch sind unter Windows scheinbar zusammenhängende Adressbereiche im Emulator zerstückelt.

Der 16-Bit-Aufruf der Windows-Funktionen aus dem Windows-Programm heraus wird durch ein Call-Gate (eine Funktion, die das gesamte Handling zentral erledigt) in einen 32-Bit-Aufruf der entsprechenden Funktion des Emulators umgesetzt (auch für die 32-Bit-Funktionen gibt es entsprechende Umsetzungen). In diesen Funktionen werden die Parameter in ein geeignetes Format für X-Aufrufe gebracht und die entsprechenden X-Funktionen aufgerufen. Zur Übergabe von Parametern und für die Rückgabe von Ergebnissen sind verschiedene Dinge zu beachten:

- Bei Funktionen, die Ergebnisse in Prozessorregistern zurückliefern, müssen die Register am Ende der Funktion korrekt gesetzt werden.
- Callback-Funktionen im Anwendungsprogramm, die von Windows aufgerufen werden, müssen ermöglicht werden.

Innerhalb von `wine` werden die X-Fenster selbst verwaltet, so dass einerseits die Benutzung von Windows-Programmen unter `wine` analog zu der unter dem echten Windows ist. Andererseits kommt es zu Problemen mit virtuellen Window-Managern, die mehrere virtuelle X-Desktops zur Verfügung stellen. Dort ist das Fenster eines `wine`-Programms stets auf dem aktuellen Schirm zu sehen. Derzeit ist es möglich, den Emulator mit dem Parameter `-desktop` zu starten, so dass der X-Window-Manager ein Fenster mit einem virtuellen Windows-Desktop verwaltet (Listing 13.16).

```
(linux):~$ wine -desktop 640x480 sol
```

Listing 13.16 Starten von wine *mit der Option* -desktop

Wenn Ihnen ein Fenster, in dem sich die Windows-Programme tummeln, nicht zusagt, dann hilft Ihnen möglicherweise die Option -managed weiter (siehe Listing 13.17). Hier wird die Fensterdekoration vom Window-Manager durchgeführt. Da jedoch unter Windows dieses Verfahren nicht bekannt ist, gibt es bei allen Optionen einige Probleme.

```
(linux):~$ wine -managed sol
```

Listing 13.17 Starten von wine *mit der Option* -managed

Diese Art der Bedienung ist ein deutlicher Bruch in der Benutzung des X-Window-Systems, aber eine bessere Lösung ist augenblicklich nicht in Sicht. Möglich wäre die Entwicklung eines Windows-kompatiblen Window-Managers. Auf meinem System verwende ich den fvwm95, der die Oberfläche von Windows 95 nachahmt, zusammen mit der Option -desktop.

Weitere Aspekte, die bei der Emulation eine Rolle spielen und vom wine-Projekt implementiert wurden, sind:

- Zugriffe auf das Netzwerk müssen umgesetzt werden.
- Windows unterscheidet bei Dateinamen nicht zwischen Groß- und Kleinschreibung, Linux hingegen schon. Daher ist auch hier eine Anpassung notwendig.
- Windows-Programme bestehen sehr häufig aus mehreren Threads und mehrere Windows-Programme können in einer wine-Sitzung parallel ausgeführt werden. Verschiedene Funktionen müssen synchronisiert werden und zentrale Datenstrukturen für die einzelnen Windows-Programme müssen konsistent bleiben. Zu diesem Zweck wurde der wine-Server entwickelt, der intern diese Aufgaben übernimmt.
- Fehlersuche im Emulator und in den Windows-Programmen ist schwierig, daher wurde der wine-Debugger entwickelt.

Die Entwicklung von wine zielt nicht nur auf die Erstellung eines Windows-Emulators für Linux und *BSD-Systeme, sondern auch auf andere Betriebssysteme, die nicht auf Intel-Prozessoren laufen. Dazu könnte später bochs, ein CPU-Emulator für Intel-CPUs, verwendet werden.

Eine weitere Funktion von wine ist die Verwendung als Bibliothek zur Portierung von Windows-Programmen auf beliebige Unix/X-Systeme. Mit Hilfe dieser Bibliothek wurden bereits verschiedene, auch kommerzielle Programme portiert.

13.3.4 Weitere Entwicklung

Auch nach vielen Jahren der Entwicklung sind zur Vervollständigung des Emulators noch viele Features zu implementieren. Dabei wird man immer etwas hinter der Entwicklung des »echten« Windows hinterherhinken. Ein weiteres Problem ist, dass viele Windows-APIs nicht vollständig oder gar fehlerhaft dokumentiert sind. Damit steht vor der Implementierung einer Funktion häufig die Recherche, was diese eigentlich genau tun soll – dabei kann sich das zwischen den verschiedenen emulierten Windows-Versionen auch noch unterscheiden.

Bei der bisherigen Entwicklung verfolgte man zwei Strategien: die Implementierung der APIs, eine Gruppe von Funktionen nach der anderen und die Implementierung bzw. Korrektur von Funktionen, bis eine bestimmte Anwendung oder ein bestimmtes Spiel funktionsfähig ist. Beide Vorgehensweisen haben ihre Vor- und Nacheile. Aktuelle `wine`-Versionen sind in der Lage, verschiedene Office-Pakete auszuführen, auch die Verwendung von Lotus Notes ist möglich (und sinnvoll, weil es immer noch keinen Linux-Client gibt).

Derzeit wird eine Testsuite implementiert, um implementierte Funktionen zu überprüfen und damit die Funktionalität von `wine` für möglichst viele Applikationen sicherzustellen. Parallel dazu implementieren verschiedene Firmen APIs für Programme, die sie portieren, bzw. entwickeln `wine` für ihre Kunden weiter. Es bleibt zu hoffen, dass der Wechsel von der X11-Lizenz zur LGPL die Gemeinde nicht zu sehr splittert, so dass die Entwicklung auch in Zukunft weitergeht.

13.4 Linux auf einem Mainframe

Linux läuft auf sehr vielen verschiedenen Hardware-Architekturen – von der Armbanduhr bis zum Mainframe. Eine interessante Variante ist der Betrieb von Linux/390 auf einem PC mit Hilfe des Mainframe-Emulators `hercules` (*http://www.conmicro.cx/hercules*). Das ist zwar langsam, aber es funktioniert. Das Kompilieren von Programmen kostet viel Zeit, dennoch ist es eine Möglichkeit, Programme auf einen Großrechner zu portieren.

`hercules` bietet entweder den Zugriff auf eine einfache Konsole oder Sie können sich über ein virtuelles Netzwerk an die virtuelle Maschine anmelden. Dazu benötigen Sie den TUN/TAP-Treiber im Kernel. Wenn Sie diesen als Modul kompiliert haben, dann können Sie das Laden mit dem Eintrag in Listing 13.18 automatisieren.

```
alias tun0 tun
```

Listing 13.18 Konfiguration des `TUN` *Moduls in der Datei* `modules.conf`

Damit vom virtuellen Rechner aus auch andere Rechner im LAN oder Internet erreichbar sind, muss der Wirtsrechner IP-Forwarding erlauben. Forwarding können Sie mit dem Befehl aus Listing 13.19 einschalten. Dort finden Sie auch ein Beispiel für die Einrichtung von Proxy-ARP, so dass Sie kein virtuelles Transfernetz benötigen. Ersetzen Sie die MAC-Adresse durch diejenige des Gastrechners.

```
(linux):~# echo 1 > /proc/sys/net/ipv4/ip_forward
(linux):~# arp -sn hercules 00:50:18:00:0F:FE
```

Listing 13.19 Netzwerkkonfiguration für `hercules`

Auf Großrechnern sind verschiedene Dinge ganz anders, als man es von PCs gewohnt ist. Festplatten heißen nicht Platten oder Disks, sondern DASDs (Direct Access Storage Device). Alle Devices werden über einen Kanal (Channel) angeschlossen und erhalten zur Bezeichnung eine vierstellige Nummer, bestehend aus der Kanal- und einer Device-Nummer. Auch die Device-Typen werden mit einer vierstelligen Nummer gekennzeichnet.

Diese Nummern finden wir in der Konfigurationsdatei `hercules.cnf` wieder, die aus dem aktuellen Verzeichnis gelesen wird. In der ersten Spalte lesen Sie die Device-Nummern, in der zweiten Spalte die Device-Typen. Die wichtigsten Parameter sehen Sie in Listing 13.20, dort sind allerdings nur diejenigen Parameter aufgeführt, die wir anpassen müssen. Wenn Sie `devfs` verwenden, so müssen Sie das Gerät `/dev/tun0` durch `/dev/misc/net/tun` ersetzen.

```
# Dev   Typ     Datei/Konfiguration
# Lochkartenleser
000C    3505    bf/kernel.debian bf/parmfile.debian bf/initrd.debian
autopad eof
# Bandlaufwerk
0181    3480    /tapes/debian.tdf
# Platten
0191    3380    linux.191
0192    3380    linux.192
# virtuelles Netzwerk
#                       device    MTU  Hercules IP  Host IP      Netmask
0A00    3088 CTCI      /dev/tun0 2000 192.168.30.220 192.168.30.202
255.255.255.255
0A01    3088 CTCI      /dev/tun0 2000 192.168.30.220 192.168.30.202
255.255.255.255
```

Listing 13.20 Auszug aus der `hercules`*-Konfiguration*

Bevor wir das erste Mal starten, müssen wir zunächst die emulierten Festplatten als Dateien anlegen. Das erfolgt mit dem Befehl `dasdinit` (Listing 13.21).

```
(linux):~# dasdinit -bz2 linux.191 3380 LNX191 400
(linux):~# dasdinit -bz2 linux.192 3380 LNX192 100
```

Listing 13.21 Initialisieren der emulierten DASDs

Für die ersten Versuche benötigen wird nun nur noch ein Boot-Medium. Großrechner können von DASDs, Bandlaufwerken (Tapes) oder Lochkarten-Lesern (Reader) booten. Bei den Linux/390-Distributionen finden Sie in der Regel sowohl einen Kernel für Tapes als auch für den Lochkarten-Leser (Reader). Die Installation unter `hercules` ist derjenigen unter z/VM sehr ähnlich – auch unter z/VM (für Virtual Machine) wird ein Teil der Hardware nur simuliert.

Auf einem bootfähigen Band benötigen Sie drei Dateien: den Kernel, die Kommandozeile des Kernels und die initiale RAM-Disk. Für `hercules` kopieren Sie die Dateien in das Verzeichnis `/tapes` und erstellen eine TDF-Datei (Tape Description File, Listing 13.22). Im Beispiel verwende ich die Debian/390-Distribution.

```
@TDF
/tapes/kernel.debian  FIXED RECSIZE 1024
TM
/tapes/parmfile.debian    TEXT
TM
/tapes/initrd.debian  FIXED RECSIZE 1024
TM
TM
EOT
```

Listing 13.22 Die Bandbeschreibung `debian.tdf`

Nun können wir das erste Mal den simulierten Rechner starten. Dazu rufen Sie `hercules` auf und starten das Betriebssystem mit einem IPL (Initial Program Load). Zum Booten vom Bandlaufwerk (Device 0181 im Beispiel) geben Sie im Emulator `ipl 181` ein.

In der `hercules`-Konsole müssen Sie zunächst das Netzwerk konfigurieren, danach können Sie sich mittels `telnet` anmelden und die Installation fortsetzen (`ssh` benötigt sehr lange, um die Session-Keys zu generieren). Anschließend werden Sie durch die Installation geführt, fast wie vom PC gewohnt. Befehle an den Emulator geben Sie direkt ein, Befehlen für das Gastbetriebssystem müssen Sie einen Punkt (».«) voranstellen.

Neben dem Spaß, einen Großrechner zu Hause zu haben und Erfahrungen mit einer doch sehr anderen Plattform zu sammeln, verspricht Linux auf einem Mainframe jede Menge »Hack-Value«.

14 Linux in einer vernetzten Umgebung

The take-up of the network protocol TCP/IPv9 has been phenomenal over the last few years. Gone are the days when there were just a few million hosts, and the network was understood.

rfc1606, J. Onions

14.1 TCP/IP und das Internet

Häufig wird Linux im privaten Bereich als Lernsystem eingesetzt. Aufgrund seiner Stabilität und Leistungsfähigkeit erobert Linux aber auch den professionellen und kommerziellen Bereich. Dort ist die Zusammenarbeit mit anderen Rechnerarchitekturen und Betriebssystemen besonders wichtig. Auch im privaten Bereich wird Linux oft mit einem älteren Rechner vernetzt, der dann z. B. als Terminal dienen kann.

Ein Multiuser- und Multitasking-Betriebssystem wie Linux bietet in einem Netzwerk viele Vorteile. Viele Ressourcen, wie z. B. ein Modem, können von mehreren Personen genutzt werden. Dazu ist nur die Anmeldung auf dem Rechner erforderlich, an dem das Modem angeschlossen ist. Nebenbei kann dieser Rechner aber auch für eine Reihe von anderen Aufgaben, z. B. als Datei- und Drucker-Server, verwendet werden.

In den folgenden Kapiteln werden einige Möglichkeiten vorgestellt, die die Vernetzung mit Linux bietet, dabei wird auch auf die Vernetzung mit MS-Windows (speziell Windows for Workgroups) und Windows NT eingegangen. Daneben werden Sicherheitsüberlegungen einen breiten Raum einnehmen, da mit der Installation eines Netzes der Rechner angreifbarer wird.

Linux verwendet, wie andere Unix-Systeme auch, TCP/IP (Transmission Control Protocol/Internet Protocol). Darunter versteht man eine Sammlung von Protokollen, die zur Kommunikation zwischen verschiedenen Rechnern verwendet werden. Dabei handelt es sich um die Standard-Internet-Protokolle.

Die Spezifikationen der Protokolle werden in so genannten RFCs (Request for Comment) festgelegt, die im Internet frei verfügbar sind. Diese RFCs unterteilen sich in Vorschläge für das weitere Vorgehen oder Aufsätze über Probleme in der Netzimplementierung oder -verwaltung, geplante Standards (»Draft Standards«) und anerkannte »offizielle« Standards. RFCs sind kein Standard internationaler oder nationaler Organisationen wie ANSI, DIN oder ISO, sondern eine Absprache der Internetbenutzer.

Durch die freie Verfügbarkeit der RFCs ist eine Implementierung jedermann möglich. Dies ist einer der Gründe für die weite Verbreitung der Protokolle und die gute Interoperabilität verschiedener Implementierungen. Die RFCs werden von erfahrenen Internetbenutzern und Systemverwaltern entwickelt.

RFCs können prinzipiell von jedem interessierten Internetteilnehmer erstellt werden. Die Definition eines stabilen und sicheren Protokolls ist jedoch relativ schwierig, so dass nur die wenigsten Benutzer den Prozess der öffentlichen Diskussion durchstehen dürften. Die IETF (Internet Engineering Task Force) erarbeitet viele der RFCs. Das Verfahren zur Erlangung eines Internetstandards ist ebenfalls in einem RFC festgelegt ([rfc1311] und [rfc2026]).

Bei intensiver Beschäftigung mit Unix-Netzen, dem Internet oder einem Intranet ist die Lektüre der entsprechenden RFCs unerlässlich. Eine Liste der wichtigsten RFCs finden Sie im RFC-Verzeichnis in Anhang D, »Verzeichnis der wichtigsten RFCs«, außerdem wird im Folgenden stets auf die entsprechenden RFCs verwiesen. Die RFCs finden Sie z. B. unter *ftp://ftp.nic.de/pub/doc/rfc* oder *ftp://ftp.gwdg.de/pub/rfc*.

Aufgrund der Komplexität von Netzen und der dort eingesetzten Software kann dieses Buch nicht alle Aspekte beleuchten. Viele weitere Informationen finden Sie in den HowTo-Dokumenten des Linux Documentation Project, z. B. zu PPP, News oder Mail, und den FAQs der Newsgruppen, die sich mit diesen Themen befassen. Viele der unter Linux verwendeten Programme sind nicht Linux-spezifisch und werden daher nicht in den Linux-Newsgroups diskutiert, sondern in denjenigen der Programme oder Protokolle. Daneben finden Sie im Literaturverzeichnis in Anhang C, »Literaturverzeichnis« eine Reihe von lesenswerten Büchern zu diesem Themenkomplex.

Die aktuellen Windows-Versionen enthalten Clients für den Zugriff mittels `telnet` und `ftp`, für praktisch alle anderen Protokolle stehen auch frei verfügbare Clients zur Verfügung. Der Einsatz von Windows als Internet-Server erscheint nicht ratsam, aber als Client ist Windows vielfach eine Alternative zu Linux oder Unix. In den neuesten Versionen der verbreiteten PC-Betriebssysteme sind in der Regel alle notwendigen Internet-Clients enthalten.

14.2 Schichten in der Netzwerk-Software

Die TCP/IP-Software besteht aus einer Reihe von Schichten, die aufeinander aufbauen. Dabei basiert jede Schicht auf einer klaren Schnittstelle zu den darunter liegenden Schichten und kann deren Dienstleistungen in Anspruch nehmen, ohne zu wissen, mit welchen Mitteln diese erbracht werden. Dadurch ist es z. B. für Anwendungsprogramme vollkommen unerheblich, über welches Medium die Daten übertragen werden. Die Trennung der Schichten sorgt für eine klare Schnittstelle und vereinfacht die Implementierung.

Im 7-Schichten-Referenzmodell OSI (Open Systems Interconnection) der ISO, siehe Abbildung 14.1, werden sieben Schichten definiert. Die Aufgaben und Funktionen dieser sieben Schichten finden sich (etwas anders strukturiert) auch in den Schichten der TCP/IP-Protokolle wieder. Durch die andere Struktur der TCP/IP-Protokolle ist keine direkte Abbildung zwischen beiden Modellen möglich.

Application	Anwendung	Level 7
Presentation	Darstellung	Level 6
Session	Kommunikation	Level 5
Transport	Transport	Level 4
Network	Vermittlung	Level 3
Link	Sicherung	Level 2
Physical	Bitübertragung	Level 1

Abbildung 14.1 Das OSI-Schichtenmodell

Einen groben Überblick über die Schichten der TCP/IP-Protokolle liefert Abbildung 14.2. Die zum Vergleich angegebenen OSI-Schichten entsprechen in ihrer Funktion in etwa den Schichten in den TCP/IP-Protokollen. Im folgenden Kapitel werden wir uns von unten nach oben durch die einzelnen Schichten durcharbeiten, um am Ende einen Überblick über die Funktionsweise eines TCP/IP-Netzes zu bekommen.

Application level (OSI: Schichten 57)	telnet	ftp	Mail	DNS	tftp	NFS
Transmission level (OSISchicht 4)	TCP		UDP			
Internet level (OSISchicht 3)	IP & ICMP					
Network level (OSISchichten 1+2)	Ethernet		Token Ring		SLIP	

Abbildung 14.2 TCP/IP-Protokollschichten

14.3 Netzwerk-Hardware

Zum Anschluss an andere Rechner, sowohl lokale als auch entfernte, stehen verschiedene Möglichkeiten zur Verfügung. Zum einen ist die Einbindung in ein lokales Netz (LAN, Lokal Area Network) mittels Ethernet, ArcNet oder Token Ring möglich. Für die Anbindung an entfernte Rechner oder Netze steht PPP (Point-to-Point-Protocol) zur Verfügung. Weitere Informationen finden Sie auch in den entsprechenden HowTo-Dokumenten.

Im folgenden Kapitel geht es um die Einbindung eines Linux-Rechners in ein lokales Netz, die Anbindung über ein Modem oder mittels ISDN wird in Kapitel 26, »Internetzugang über Wählverbindungen« besprochen. Im ersten Schritt müssen die Rechner durch ein geeignetes Kabel miteinander verbunden werden. Heute hat sich weitgehend Ethernet (CSMA/CD[1]) durchgesetzt, wobei drei verschiedene Kabeltypen möglich sind:

- Thick Ethernet (RJ-45-Kabel) wird heute kaum noch verwendet. Die technische Bezeichnung ist 10Base5.
- Thin Ethernet (Coax-Kabel) wird in kleineren Installationen gern verwendet und hat einen Netto-Durchsatz von 10 MBit/sec. Der Ethernet-Strang wird in Form eines Busses verlegt und muss an beiden Enden mit einem Abschlusswiderstand (Terminator) versehen werden. Die einzelnen Rechner werden mit einem T-Stück an den Bus angeschlossen. In Abbildung 14.3 finden Sie eine schematische Darstellung der korrekten und beliebten fehlerhaften Anschlüsse.

  Als Kabeltyp wird RG-58 eingesetzt, die Stecker sind unter dem Begriff BNC bekannt. Die technische Bezeichnung ist 10Base2. Bei der Unterbrechung des Busses ist am gesamten Kabelstrang mit Problemen zu rechnen, daher wird dieser Kabeltyp in der Regel nur in überschaubaren Installationen eingesetzt. Außerdem ist die Kabellänge (ohne Repeater) auf etwa 180 Meter beschränkt.
- Twisted Pair (10BaseT) wird oft in größeren Installationen verwendet und erlaubt bei geeigneter Hardware bis zu 100 MBit/sec Datendurchsatz. Allerdings wird, zusätzlich zu den Netzwerkkarten in den Rechnern, ein zentraler Konzentrator oder Hub benötigt, so dass dieser Kabeltyp im privaten Bereich und in kleinen Installationen praktisch keine Rolle spielt. Die Verkabelung erfolgt sternförmig, so dass von jedem Rechner ein eigenes Kabel zum Konzentrator verlegt werden muss. Die maximale Kabellänge beträgt hier ca. 100 Meter. Wenn man nicht wirklich mit der letzten Mark rechnen muss, ist diese Netztopologie zurzeit die beste Empfehlung.

1. Carrier Sense Multiple Access, Collision Detect: Jeder Teilnehmer darf sofort senden, Kollisionen werden erkannt und die Übertragung wird erneut durchgeführt.

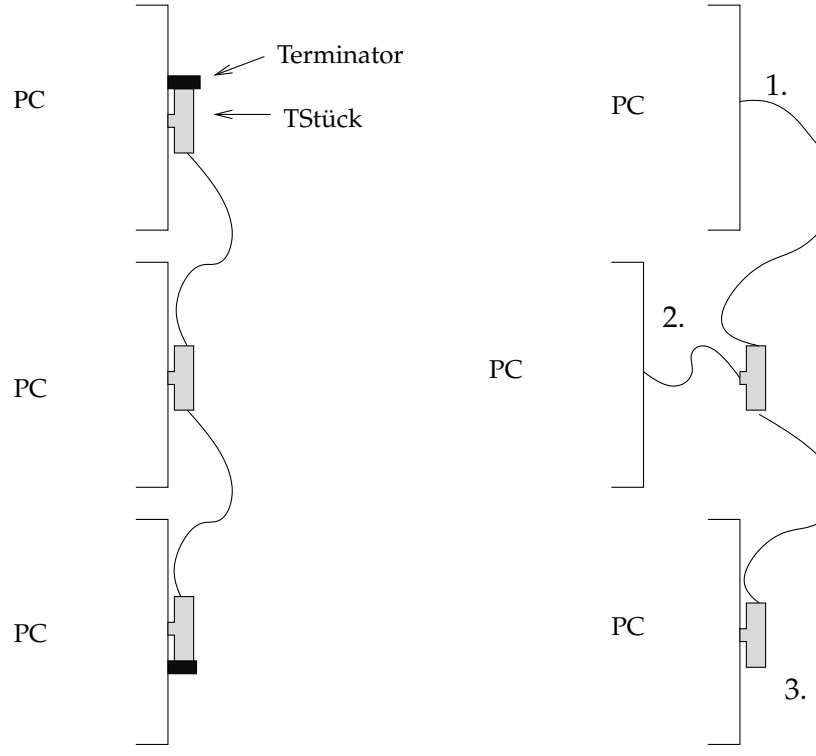


Abbildung 14.3 Verkabelung mit Thin Ethernet – richtig und falsch

- In größeren Installationen, in denen es auf hohen Durchsatz und hohe Verfügbarkeit ankommt, findet man auch FDDI; für den privaten Bereich hat dieser Typ keine Bedeutung.

Eine Ethernet-Karte ist im einfachsten Fall für exakt einen Verbindungstyp konstruiert. Es gibt auch Kombikarten, die an mehreren Kabeltypen betrieben werden können. Wenn absehbar ist, dass Sie in der nächsten Zeit auf einen anderen Kabeltyp wechseln, kann sich diese Anschaffung lohnen. Andernfalls kommen Sie normalerweise mit einer einfachen Karte aus. Für praktisch alle verbreiteten Ethernet-Karten der verschiedenen Hersteller finden Sie im Kernel einen passenden Treiber. Mehr dazu finden Sie in der Ethernet-HowTo bzw. für Token Ring in der Token Ring-HowTo.

Ein Problem bei der Verkabelung mit Thin Ethernet ist, dass ein fehlerhafter Anschluss am Netz oder ein etwas zu langer Kabelstrang unerklärliche Phänomene auf anderen Rechnern verursachen kann. Daher ist es bei Problemen sinnvoll, zunächst das Netz in kleinere Teile zu unterteilen und so zu versuchen, den Fehler einzugrenzen. Beliebte Fehler beim Anschluss sind (siehe auch Abbildung 14.3):

1. Das Ende des Netzwerkkabels wird nicht mit einem T-Stück an den Rechner angeschlossen, sondern direkt mit der Netzwerkkarte verbunden.
2. Ein Rechner wird nicht direkt am T-Stück angeschlossen, sondern mit einem kurzen Kabel mit dem Netz verbunden.
3. Das Netzwerkkabel ist nicht mit einem Endwiderstand versehen, es ist nicht terminiert. Ohne Terminator kommt es zu Störungen auf dem Netz.

Die größten Probleme in der Praxis sind die begrenzte Kabellänge und damit die Anzahl der anzuschließenden Rechner und die mit der Anzahl der Rechner steigende Netzlast. Je höher die Netzlast ist, desto größer ist die Wahrscheinlichkeit, dass eine Kollision auftritt und beide sendenden Rechner eine kurze Zeit warten und die Übertragung erneut versuchen müssen. Die Wartezeit wird in gewissen Grenzen zufällig gewählt, damit nicht dieselben Rechner wieder eine Kollision verursachen. Außerdem wird die Wartezeit bei einem erneuten Versuch größer, damit das Netz entlastet wird.

Bei der Verwendung von Token Ring können keine Kollisionen auftreten. Im Token Ring wird ein spezielles Symbol, das so genannte Token, generiert. Es signalisiert, dass der betreffende Rechner sendeberechtigt ist. Der Rechner nimmt das Token vom Netz, sendet sein Paket und schickt das Token wieder auf die Reise. Es ist nur die Station sendeberechtigt, die das Token besitzt. Wenn das Token z. B. durch Ausfall eines Rechners vom Netz verschwindet, so wird von einem der verbleibenden Rechner ein neues Token generiert.

Ist die Länge des Netzwerkkabels zu groß geworden, treten die unterschiedlichsten, nicht reproduzierbaren Probleme auf. In solchen Fällen kann es sinnvoll sein, das Netz testweise zu verkleinern. Verschwinden die Probleme, so muss über eine Trennung des Netzes durch eine Bridge nachgedacht werden. In vielen Netzinstallationen stehen Messgeräte für die Kabellänge und andere wichtige Größen zur Verfügung, oft muss man aber auch ohne diese Geräte auskommen.

Im einfachsten Fall kann ein Ethernet durch den Einsatz eines *Repeater* verlängert werden. Dieses Gerät leitet alle Netzpakete des einen Strangs auf alle anderen Kabel weiter. Damit umgeht man (bis zu einem gewissen Grad, denn die Anzahl der Repeater in einem Netz ist ebenfalls begrenzt) die maximale Kabellänge. Es erfolgt keine Lasttrennung, so dass die Anzahl der Kollisionen nicht sinkt. Daher werden heute nur noch selten Repeater eingesetzt.

Sinnvoller ist der Einsatz einer *Bridge*, die anhand der Hardware-Adressen der Netzwerkkarten entscheidet, auf welchen Strang die Pakete weitergeleitet werden müssen. Linux kann genauso als Bridge eingesetzt werden wie PC-Bridge, ein DOS-Programm. Es gibt allerdings auch Hardware-Bridges zu kaufen. Ein *Switch* ist eine verbesserte Bridge, die mehrere Netze parallel miteinander verbinden kann. Es können mehrere Pakete gleichzeitig durch den Switch übertragen werden.

Router dienen ebenfalls zur Trennung von Netzen und arbeiten auf der nächsthöheren Protokollebene, also z. B. mit IP-Paketen. Neben der Trennung der Segmente können Pakete zwischen verschiedenen Anschlussarten (Token-Ring, Ethernet oder SLIP) umgesetzt werden. In praktisch allen großen Netzen findet man Router, die die Fernverbindungen steuern können. Router arbeiten auf der Ebene der Protokolle und nicht mehr wie Repeater, Bridges und Switches auf der Hardware-Ebene. Daher müssen die eingesetzten Router alle notwendigen Protokolle (wie IPX, Appletalk oder SNA) beherrschen. In der Anfangszeit des Internets wurden Router auch als Gateways bezeichnet.

14.4 Netzwerkbezogene Kernel-Konfiguration

Nach der Installation der Netzwerkkarte und dem Anschluss an das Netz muss der Kernel für den Netzbetrieb konfiguriert und der passende Hardwaretreiber eingebunden werden. Mit der Option `CONFIG_NET` wird allgemein die Netzwerkfähigkeit von Linux aktiviert. Im Weiteren können die verfügbaren Protokolle und Optionen weiter bestimmt werden.

Sie sollten nur die Treiber und Protokolle aktivieren, die Sie auch tatsächlich verwenden möchten, um nicht einen unnötig großen Kernel zu erhalten. Fast alle Treiber können auch als Module übersetzt werden, die dann zur Laufzeit in den Kernel geladen werden. Damit wird der entsprechende Speicher nur dann belegt, wenn der Treiber auch benötigt wird. Dies kann z. B. für temporäre Verbindungen mittels SLIP oder PPP interessant sein. Weitere Informationen zu Kernel-Modulen finden Sie im Abschnitt 4.3, »Kernel-Module«.

Zu jeder Konfigurationsoption existiert ein recht ausführlicher Hilfetext. Dieser gibt auch bei neueren Optionen detailliert Auskunft über Auswirkungen dieser Option, möglicherweise benötigte Programme und deren Bezugsquellen sowie weitere Dokumentationen, wie z. B. HowTo-Dokumente.

14.4.1 Networking-Optionen

TCP/IP networking
: ermöglicht die Verwendung von TCP/IP. Diese Option muss für den Betrieb von Linux im Netz unbedingt aktiviert werden. Auch wenn Sie nicht an ein Netzwerk angeschlossen sind, sollten Sie diese Option anwählen. Einige Programme (z. B. `syslog` oder `lpd`) benötigen Netzwerkfunktionen.

Network firewalls
: Diese Frage sollten Sie nur bejahen, wenn Sie Linux als Firewall verwenden wollen. Sie sollten dann später auch die Frage nach `IP: firewalling` mit Ja beantworten. In der Regel werden Sie hier mit Nein antworten. Mehr Informationen finden Sie in der Firewall-HowTo.

Network aliasing
: Wenn Ihr Rechner mehrere Netzwerkadressen gleichzeitig bedienen und dabei unterschiedliche Funktionen wahrnehmen soll, dann antworten Sie mit Ja. Diese Funktion wird z. B. von einigen `http`-Servern verwendet, um Web-Hosting für mehrere Domains auf einem Rechner anbieten zu können. Auch hier folgt später eine Frage nach den zu bearbeitenden Protokollen. Mehr Informationen zu diesem Thema finden Sie in den Mini-HowTos zu »Virtual Web« und »Virtual `wuftp`«.

IP forwarding/gatewaying
: IP forwarding/IP gatewaying oder Routing ist normalerweise nicht aktiviert. Ein Internetrechner darf keine Pakete zwischen verschiedenen Netzen weiterleiten, es sei denn, er ist speziell als Gateway oder Router konfiguriert. Wenn Sie Ihren Rechner als SLIP- oder PPP-Server verwenden wollen, so müssen Sie diese Option aktivieren und entsprechende Routen setzen. Alternativ können Sie hier mit Nein antworten und den Datentransfer einem Firewall-Paket überlassen.

IP multicasting
: Das Internetprotokoll kennt zunächst nur zwei Möglichkeiten der Adressierung: Das Ansprechen eines speziellen Rechners bzw. das Ansprechen aller Rechner im lokalen Netz (Broadcast). Mit Multicasting können mehrere Rechner gezielt angesprochen werden. Näheres dazu finden Sie in [rfc1112], [rfc1301] und [rfc1458].

IP accounting
: IP accounting ermöglicht die benutzer- und paketweise Abrechnung der Netzbelastung. Dies ist sinnvoll, wenn Sie gegen Entgelt einen Internetzugang anbieten oder diesen firmenintern abrechnen müssen. Dazu benötigen Sie spezielle Programme. Die Bezugsquellen finden Sie im Hilfetext zu dieser Option.

IP firewalling
: IP firewalling dient zur Abschirmung eines lokalen Netzes gegen Angriffe aus fremden Netzen über das Internet. Weitere Informationen zu diesem Themenkomplex finden Sie in der Firewall-HowTo und in [Zwicky2000]. Zum Einsatz von Linux als Firewall benötigen Sie noch einige Utilities, die Sie z. B. bei `ftp.tis.com` finden.

Nun folgen einige Einstellungen, die in der Regel nicht verändert werden sollten. Dennoch gibt es Situationen, in denen die hier vorgestellten Konfigurationsoptionen notwendig sind.

PC/TCP compatibility mode
: um mit fehlerhaften TCP/IP-Implementierungen für MS-DOS-Rechner kommunizieren zu können.

Reverse ARP
(RARPRARPReverse ARP) wird in [rfc0903] definiert und erfüllt einen ähnlichen Zweck wie BOOTPBOOTP (siehe auch Kapitel 24, »Dynamische IP-Konfiguration«. Diese Funktion benötigen Sie für manche Rechner, wenn diese über das Netz booten sollen.

Assume subnets are local
dient zur automatischen Bestimmung der MTU (Maximal Transfer Unit). Für lokale Netze kann die MTU größer gewählt werden als für SLIP-Verbindungen, ohne dass Performance-Verluste im interaktiven Betrieb zu befürchten sind. Die MTU kann außerdem für einzelne Interfaces oder Routen gesetzt werden, um einen optimalen Datendurchsatz zu erreichen.

IP: Disable Path MTU Discovery (normally enabled)
Je nach verwendetem Netzanschluss können unterschiedlich große MTUs (Maximal Transfer Units) zu einem höheren Netzdurchsatz führen. Zur automatischen Erkennung der besten MTU gibt es ein Verfahren, das normalerweise verwendet wird.

Disable NAGLE algorithm (normally enabled)
Der Nagle-Algorithmus ist eine Verfahren, das versucht, die Übertragung von sehr kleinen Paketen (z. B. einzelne Tastendrücken bei `telnet`) über langsame Leitungen zu verhindern, indem das Paket kurze Zeit zurückgehalten und eventuell mit einem zweiten Paket gemeinsam versendet wird.

IP: Drop source routed frames
Üblicherweise wird im Internet von den Routern der zu verwendende Weg bestimmt. Beim Source-Routing bestimmt der Absender, welchen Weg das Paket nehmen soll. Dies kann zu Sicherheitsproblemen führen und ist von der IP-Spezifikation nicht gefordert. In der Regel wird man hier mit Nein antworten.

IP: Allow large windows (not recommended if < 16Mb of memory)
Wenn Ihr Rechner über 16 Mbyte oder mehr Hauptspeicher verfügt, kann diese Option den Netzwerkdurchsatz verbessern.

The IPX protocol
wird von Novell Servern und Personal Netware verwendet. Neben der Möglichkeit, dieses Protokoll zu routen und vom `dosemu` aus zu verwenden, kann Linux auch als NetWare-Server oder -Client eingesetzt werden. Mehr Informationen finden Sie im Abschnitt 21.2, »Linux als NetWare-Client und -Server«.

Appletalk DDP
ist eine Möglichkeit, Apple-Computer mittels Ethernet anzubinden. Apple nennt dieses Protokoll EtherTalk. Diese Option ist nur sinnvoll, wenn in Ihrem Netz auch Apple-Rechner verwendet werden.

Amateur Radio AX.25 Level 2
dient zur Kommunikation über Amateurfunk.

14.4.2 Network device support

Hier werden die entsprechenden Hardware-Treiber aktiviert. Sie sollten nur den zu Ihrer Karte gehörenden Treiber in den Kernel einkompilieren, um Speicher zu sparen und Probleme beim Auto-Probing zu vermeiden. Fast alle Treiber können auch als Modul kompiliert werden, so dass selten benutzte Treiber erst dann geladen werden, wenn sie benötigt werden. Wenn Sie mehrere Netzwerkkarten in einem Rechner installieren wollen, um diesen Rechner z. B. als Router einzusetzen, müssen Sie die Adressen und Interrupts dieser Karten entweder in den Kernel einkompilieren (`Space.c`) oder mittels einer Kernel-Kommandozeile übergeben.

Network device support?

Soll überhaupt ein zusätzlicher Treiber aktiviert werden? Wenn der Rechner nur das loopback nutzen soll, so sind keine zusätzlichen Treiber notwendig. Das `loopback`-Device dient zum Ansprechen des eigenen Rechners, auch wenn keine Netzwerkkarte installiert ist.

Dummy net driver support
: Der Dummy-Treiber kann verwendet werden, wenn der Host normalerweise nicht in einem Netz integriert ist, aber über eine SLIP- oder PPP-Verbindung temporär an einen Provider angeschlossen wird. Dabei bekommt der Rechner eine IP-Adresse zugewiesen, die dann über `ifconfig` dem SLIP- oder PPP-Interface `sl0` oder `ppp0` zugewiesen wird. Ist diese Verbindung unterbrochen, so kann der Rechner sich selbst nicht über diese IP-Adresse erreichen, was oft zu Problemen führt. Daher kann man dem Dummy-Interface `dummy` (fest im Kernel einkompiliert bzw. `dummy0`, wenn der Treiber als Modul geladen wird), das dann ähnlich wie das Loopback-Device funktioniert, diese Adresse zuweisen.

SLIP (serial line) support
: Wollen Sie SLIP (Serial Line IP) verwenden? Mehr Informationen zu SLIP (und PPP) finden Sie in Kapitel 26, »Internetzugang über Wählverbindungen«, sowie in der SLIP-HowTo.

PPP (point-to-point) support
: Wollen Sie PPP (Point-To-Point Protocol) verwenden? Mehr Informationen finden Sie im Abschnitt 26.2, »Point-to-Point Protocol« und in der PPP-HowTo.

PLIP (parallel port) support
: Wollen Sie PLIP (Parallel Line IP) verwenden? Dieses Protokoll ist wesentlich langsamer als ein lokales Ethernet, funktioniert nur für begrenzte Entfernungen zwischen zwei Rechnern und benötigt ein spezielles Parallelkabel.

Anschließend müssen Sie die verwendete Netzwerkkarte bestimmen, damit der entsprechende Treiber in den Kernel eingebunden werden kann. Eine aktuelle

Liste der unterstützten Karten finden Sie in der Ethernet-HowTo. Dort finden Sie auch Kaufempfehlungen (leider auf den amerikanischen Markt bezogen) und Warnungen vor bestimmten Karten.

Nach der Übersetzung des Kernels und dem Neustart des Rechners sollte die Netzwerkkarte vom Kernel erkannt und initialisiert werden (Ausgabe der IO-Adresse, des verwendeten Interrupts, gegebenenfalls des DMA-Kanals und der Hardware-Adresse der Karte). Andernfalls ist kein Zugriff auf das Netz möglich. Die Meldungen finden Sie meist in der Datei `/var/log/messages` oder Sie können sie mit dem Befehl `dmesg` ausgeben lassen.

Bei Problemen mit dem Auto-Probing können die IO-Adresse und der Interrupt der Karte mit einer Kernel-Kommandozeile (z. B. `eth0=0x300,11`) festgelegt oder in der Datei `drivers/net/Space.cSpace.c` eingetragen werden. Sind in einem Rechner mehrere Ethernet-Karten installiert, so müssen die entsprechenden Parameter entweder in die Datei `Space.c` eingetragen oder als Kernel-Kommandozeile übergeben werden. Damit hat der Systemadministrator die volle Kontrolle darüber, welches Kernel-Device zu welcher Karte gehört.

14.4.3 Aktivierung der Netzwerkgeräte

Nun muss dem Rechner eine Internetadresse (und ein Host-Name) zugewiesen werden und der Anschluss zum Netz aktiviert werden. Damit können Rechner im gleichen Subnetz angesprochen werden. Der Anschluss ans Internet erfolgt in der Regel über einen Router (auch Gateway genannt), der Datenpakete an entfernte Rechner, die nicht im lokalen Netz liegen, weiterleitet. Dazu müssen Routen festgelegt werden, die angeben, wie welcher Rechner oder welches Netz erreicht werden kann.

Im lokalen Netz bekommen Sie die entsprechenden Einträge vom Netzwerkverwalter zugewiesen. Wenn Sie über einen Provider ans Internet angebunden sind, erhalten Sie die entsprechenden Daten dort. Sie sollten auf keinen Fall einfach irgendwelche Adressen eintragen, ohne die entsprechenden Verantwortlichen zu fragen. Wenn Sie selbst ein lokales Netz aufbauen möchten, dann können Sie Adressen aus [rfc1918] verwenden.

Bei jedem Unix-Rechner hat die IP-Adresse `127.0.0.1` eine besondere Bedeutung: Sie dient zum Ansprechen des eigenen Rechners, des `localhost`. Dazu wird das Interface `lo` (für loopback) verwendet. Damit ist es möglich, auch mit nur einem Rechner Netzwerkprogramme zu entwickeln und zu testen. Bei der Verwendung von TCP/IP-Protokollen wird dieses Interface immer aktiviert. Das Loopback-Interface kann z. B. vom Portmapper, dem Syslog-Dämon oder dem Print-System verwendet werden.

Die Initialisierung der Netzwerk-Interfaces erfolgt mit dem Programm `ifconfig`. Ohne Parameter aufgerufen, wird unter Linux der aktuelle Status der Netzwerk-

schnittstellen angezeigt. Wird als Parameter der Name eines Device angegeben, so werden nur die Informationen zu diesem Gerät angezeigt. Auf anderen Systemen kann der Status aller IP-Interfaces mit dem Kommando `ifconfig -a` abgefragt werden.

Listing 14.1 zeigt die Konfiguration der `loopback`-Schnittstelle. Diese dient zum direkten Ansprechen des lokalen Rechners. Zu diesem Zweck ist die Adresse `127.0.0.1` reserviert. Die Adresse `127.0.0.0` beschreibt das ebenfalls reservierte, zugehörige Netz. Auf jedem TCP/IP-Rechner muss diese Schnittstelle konfiguriert werden, da sie im normalen Betrieb verwendet wird.

```
# loopback-Interface aktivieren
/sbin/ifconfig lo 127.0.0.1
# Route setzen
/sbin/route add -net 127.0.0.0
```

Listing 14.1 Initialisierung des Loopback-Interface

Das Listing 14.2 zeigt die Konfiguration einer Ethernet-Karte. Wird der Rechner an einen Token Ring angeschlossen, so ist `tr0` statt `eth0` anzugeben[2]. Dabei wird nicht nur das Interface aktiviert, sondern auch die Erreichbarkeit anderer Rechner sichergestellt, indem die entsprechenden Routen eingerichtet werden. Die Metrik gibt die »Kosten« für diese Route an, so dass im Internet eine günstige Route gewählt werden kann.

```
# Ethernet-Interface aktivieren
/sbin/ifconfig eth0 ${IPADDR} broadcast ${BROADCAST} \
               netmask ${NETMASK}
# Routen festlegen
/sbin/route add -net ${NETWORK} netmask ${NETMASK}
# Standard-Route über das Gateway
/sbin/route add default gw ${GATEWAY} metric 1
```

Listing 14.2 Initialisierung des Ethernet-Interface

Im Listing 14.2 sind nicht die IP-Adressen angegeben, sondern nur Shell-Variablen. Diese Werte werden Ihnen entweder von Ihrem Provider oder Netzverwalter vorgegeben oder zentral vom Network Information Center vergeben. Für Deutschland ist DE-NIC zuständig, so dass Anfragen nach IP-Adressen relativ »lokal« bearbeitet werden können. Durch die zentrale Adresszuordnung ist die weltweite Eindeutigkeit dieser Adressen sichergestellt. Bei nur temporären Verbindungen zum Internet mittels PPP können diese Adressen bei jedem Anruf auch dynamisch vom Provider vergeben werden. Bei PPP brauchen Sie sich dabei um nichts zu kümmern, da die PPP-Dämonen diese Funktion bereits enthalten.

2. Unter SunOS oder Solaris heißt die Ethernet-Schnittstelle `le0`, unter AIX `et0` bzw. `en0`, bei manchen Systemen auch `ed0` oder `wd0`.

Die Broadcast-Adresse informiert darüber an, wie alle Rechner im gleichen Subnetz erreicht werden können. Die Subnetzmaske (*Netmask*) gibt im Gegenzug an, wie festgestellt werden kann, ob ein Rechner im lokalen Netz ist oder nicht. Eine ausführlichere Erläuterung finden Sie im Abschnitt 14.4.5, »Netmask- und Broadcast-Adresse«.

Nach diesen Einstellungen sollten Sie sowohl Ihren eigenen als auch andere im Netz angeschlossene Rechner erreichen können. Der erste Test erfolgt dabei üblicherweise mit `ping`, wie Sie es im Listing 14.3, sehen. Dabei können (und sollten) Sie die Loopback-Adresse, die eigene IP-Adresse und andere Rechner im Netz ansprechen. Sollten Sie hier Probleme haben, so prüfen Sie anhand der Kernel-Meldungen, ob Ihre Netzwerkkarte erkannt wurde. Verwenden Sie zunächst im ersten Test die IP-Adressen, um Probleme mit der Namensauflösung durch die Datei `/etc/hosts` oder den Nameserver auszuschließen. In einem zweiten Test können Sie dieselben Prüfungen mit den Rechnernamen durchführen.

Das Programm `ping` benutzt eine Funktion (Echo) des ICMP (Internet Control Message Protocol, [rfc0792] und [rfc0950]). Hier wird der andere Rechner aufgefordert, das empfangene Paket zurückzusenden. Die Durchlaufzeiten sollten sich für das Loopback-Device im Rahmen von einer Millisekunde, im lokalen Netz innerhalb weniger Millisekunden bewegen. Verlorengegangene Pakete deuten auf Probleme in der Netzinstallation hin, z. B. Kabelbrüche oder Ähnliches. Laufzeiten zu Rechnern im Internet können wesentlich größer sein. Hier muss auch mit dem Verlust von Paketen gerechnet werden. Diese Verluste werden allerdings in der Regel von den Anwendungen erkannt, und die Pakete werden erneut gesendet. Beachten Sie, dass manche Firewalls keine ICMP-Pakete passieren lassen und Sie daher keine Rückmeldung bekommen.

Wie bei IP-Paketen selbst wird auch hier keine vollständige und sichere Übertragung gefordert, sondern nur ein »best effort«. Es kann sich also kein System auf die Zuverlässigkeit von ICMP-Nachrichten und deren Zustellung verlassen. Viele Funktionen des Protokolls können den Netzverkehr empfindlich beeinträchtigen, daher darf nur der Benutzer `root` ICMP-Pakete erzeugen. Aus diesem Grund ist das Programm `ping` setuid.

```
(linux):~$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=1.6 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=1.6 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=1.6 ms
^C
--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 packets received, 0round-trip
min/avg/max = 1.6/1.6/1.6 ms
```

Listing 14.3 Das Programm `ping` zum Test der TCP/IP-Konfiguration

Wenn Sie Probleme mit dem Netzzugriff haben, prüfen Sie die korrekte Initialisierung Ihres Kernel-Interface mittels `ifconfig` und die gesetzten Routen mittels `route` (nur unter Linux) oder `netstat -r`. Da möglicherweise kein Zugriff auf einen Nameserver besteht, sollten Sie zusätzlich die Option `-n` angeben, damit nur die IP-Adressen ausgegeben werden und nicht die zugeordneten Namen. Als weiteren Test können Sie sich mit `ifconfig` bzw. `netstat` anzeigen lassen, ob überhaupt Pakete übertragen werden.

14.4.4 Vergabe von IP-Adressen

IP-Adressen bestehen aus 32 Bit und werden üblicherweise in der »dotted-decimal-notation« (d. h. je Byte eine Zahl, getrennt durch Punkte) angegeben. Die Adressen werden in einen Netzwerk- und einen Host-Teil unterteilt. Dabei stehen verschieden große Netze zur Verfügung. Eine Übersicht finden Sie in Tabelle 14.1. Außerdem gibt es Adressen für spezielle (experimentelle) Dienste, wie z. B. Multicasting, siehe auch [rfc1112].

Klasse	Adressbereich	Anzahl Rechner je Netz	Default Netmask
A	1.0.0.0 – 127.0.0.0	etwa 1,6 Millionen	255.0.0.0
B	128.0.0.0 – 191.255.0.0	65024	255.255.0.0
C	192.0.0.0 – 223.255.255.0	254	255.255.255.0
D	224.0.0.0 – 239.255.255.0	Multicast-Adressen	
E	240.0.0.0 – 255.255.255.255	reserviert	

Tabelle 14.1 IP-Adressbereiche je Netzklasse

Die ersten Bits in der IP-Adresse bestimmen die Zugehörigkeit zu einer Netzklasse. Der Rest der Adresse unterteilt sich dann je nach Netzklasse in eine Netznummer und eine Host-Nummer. Abbildung 14.4 zeigt dies für die zurzeit verwendeten Adressklassen.

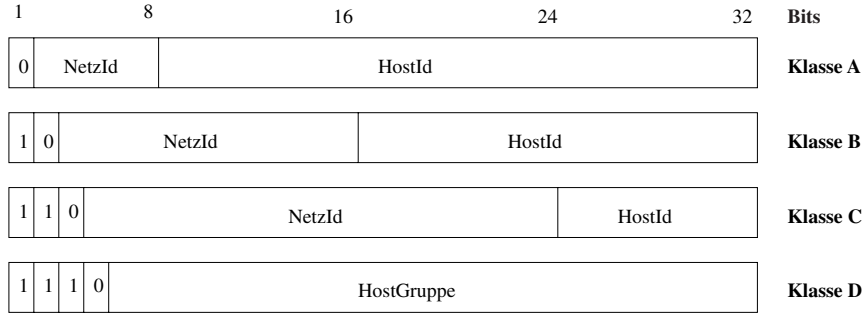


Abbildung 14.4 Aufbau der Subnetzadressen

In Tabelle 14.2 finden Sie einige Beispiele. Die Netzadressen sind durch Fettdruck hervorgehoben. In der Regel sollten an einem Ethernet-Strang nur Rechner mit einer Netznummer betrieben werden. Genauso sollten Rechner mit gleicher Netznummer nicht auf verschiedenen Seiten eines Routers stehen.

Klasse	IP-Adresse	Bit-Darstellung
A	`10.0.0.32`	*00001010*.00000000.00000000.00100000
B	`128.14.58.60`	*10000000.00001110*.00111010.00111100
C	`192.9.150.202`	*11000000.00001001.10010110*.11001010

Tabelle 14.2 Beispiele für IP-Adressen aus verschiedenen Netzklassen

Sie können die IP-Adresse nicht nur als Gruppe von vier Bytes, sondern auch als eine Integerzahl angeben, z. B. als `telnet 2130706433` statt `127.0.0.1`. Diese Zahl errechnet sich bei der IP-Adresse *a.b.c.d* durch $a*256^3+b*256^2+c*256+d$. Die Eingabe der IP-Adresse mit Hilfe dieser Formel kann auch gestandene Systemadministratoren verwirren.

Soll Ihr internes Netz nicht (ständig) mit dem Internet verbunden werden, so ist es oft nicht möglich, für alle Rechner eine offizielle IP-Adresse zu erhalten, da der verfügbare Adressraum beschränkt und durch das rasante Wachstum des Internets beinahe ausgeschöpft ist. Dies wird sich mit dem Einsatz von IPv6, das den Adressraum deutlich erweitert, wieder ändern.

Für solche Fälle gibt es durch [rfc1918] reservierte Nummern, die im Internet nicht geroutet werden (siehe Tabelle 14.3). Diese Nummern können praktisch bedenkenlos in privaten Netzen verwendet werden, selbst wenn diese später mit dem Internet verbunden werden sollen. Auch für diesen Fall findet man in [rfc1918] einige Tipps, so dass später nicht alle Rechner eine neue IP-Adresse bekommen müssen. In [rfc1627] werden demgegenüber die Probleme aufgezeigt, die man sich mit internen Adressen einhandeln kann.

Netzgröße	reservierte IP-Adressen
Klasse A	10.0.0.0 - 10.255.255.255
Klasse B	172.16.0.0 - 172.31.255.255
Klasse C	192.168.0.0 - 192.168.255.255

Tabelle 14.3 Reservierte Adressbereiche

In den folgenden Kapiteln werden stets für den internen Gebrauch reservierte IP-Adressen verwendet. Die verwendeten Domain- und Host-Namen sind ebenfalls nicht offiziell registriert. Dies soll auch den beispielhaften Charakter der vorgestellten Konfigurationen verdeutlichen.

Die notwendigen IP-Adressen können im entsprechenden `rc`-Skript oder einer Konfigurationsdatei eingetragen werden. Bei der Red-Hat-Distribution werden die IP-Adressen in der Datei `/etc/sysconfig/network` abgelegt, die SuSE-Distribution speichert diese Informationen in der Datei `/etc/rc.config`.

Die IP-Adresse kann man aber auch dynamisch beim Booten durch eine BOOTP- oder DHCP-Abfrage ([rfc0952] oder [rfc2131]) erhalten. Dazu muss im lokalen Netz ein BOOTP- oder DHCP-Server oder ein entsprechendes Gateway konfiguriert sein, der bzw. das zur Hardware-Adresse die entsprechende IP-Adresse bestimmt und an den Client weitergibt. Lesenswerte RFCs zu diesem Komplex sind [rfc2132] und [rfc1542]. Eine Erweiterung des BOOTP-Protokolls ist mit DHCP (Dynamic Host Configuration Protocol) in [rfc2132] definiert. Mehr Informationen zur Einrichtung und zum Betrieb eines BOOTP- oder DHCP-Servers finden Sie in Kapitel 24, »Dynamische IP-Konfiguration«. Da der im Internet (mit IPv4) zur Verfügung stehende Adressraum langsam zur Neige geht, beginnt derzeit die Implementierung von IPng (IP next Generation; IPv6), wodurch der Adressraum wesentlich erweitert wird. Die IP-Adressen werden durch die Umstellung deutlich verlängert, so dass erheblich mehr Rechner eindeutig adressiert werden können.

Es existieren bereits eine ganze Reihe von RFCs, die IPv6 beschreiben. Praktisch alle Hersteller sind dabei, IPv6 in ihren Produkten zu implementieren. Bei der Spezifikation hatten die Entwickler das Ziel, dass die alte und die neue IP-Version möglichst einfach miteinander kommunizieren können. Daher besteht die Hoffnung, dass der Umstieg nicht allzu schmerzhaft wird.

Auch unter Linux ist IPv6 seit der Kernel-Version 2.2 verfügbar. Mehr Informationen dazu finden Sie in der IPv6-HowTo, in der Datei `net/ipv6/README` in den Kernel-Quellen oder auf den allgemein im Internet verfügbaren Seiten zu diesem Thema. Noch ist dieses Thema für die meisten Anwender nicht interessant, es wird aber vermutlich nicht mehr lange dauern, bis die ersten Dienste unter IPv6 angeboten werden.

14.4.5 Netmask- und Broadcast-Adresse

Das Internet Protocol kennt eine Unterscheidung zwischen lokalen Rechnern, die im selben Subnetz installiert sind, und entfernten Rechnern. Ein Rechner ist im gleichen Subnetz, wenn die Adresse dieses Rechners auf allen Bits, die durch die Netmask angegeben sind, mit der Broadcast-Adresse übereinstimmt.

In diesem Fall ist die Zugehörigkeit zum selben Netz auch für den Benutzer noch direkt zu sehen. Wenn Sie Subnetting (siehe auch [rfc0940] und [rfc0950]) verwenden, müssen Sie sich hierfür die Bitmasken ansehen. Subnetting verwenden Sie, um ein zugeteiltes Netz intern weiter zu unterteilen und die Subnetze durch einen Router zu trennen. Diese Trennung ist aus Gründen der Betriebs-

sicherheit des Netzes ab einer gewissen Größe anzustreben. Ein weiterer Grund für das Subnetting kann sein, dass Ihr Provider Ihnen nur 32 oder 64 offizielle IP-Adressen zugewiesen hat.

Die Grenze zwischen Netz- und Host-Adresse muss beim Subnetting nicht an einer Byte-Grenze liegen. Der Netzwerkadministrator ist also in der Unterteilung frei. Leider werden dabei oft knappe IP-Adressen verschenkt (je eine Netzadresse und eine Broadcast-Adresse für jedes Subnetz), so dass hier überlegt werden sollte, eventuell private Adressen gemäß [rfc1918] einzusetzen.

Das Gegenstück dazu ist das Supernetting (siehe [rfc1518] und [rfc1519]). Durch das exponentielle Wachstum des Internets wurden die Routing-Tabellen der Router schnell größer, da für jedes Class-C-Netz eine eigene Route eingetragen werden musste. Später hat man einzelnen Providern ein ganzes Bündel aufeinander folgender Netze zugeteilt, für die nur noch ein Tabelleneintrag je Provider notwendig ist. Jeder Provider spaltet dann intern diesen Nummernbereich für seine Kunden weiter auf.

Auch Firmen und anderen Organisationen werden zwei oder mehr zusammengehörende Klasse-C-Netze zugewiesen, wenn ein Netz nicht ausreicht. Diese Struktur kann dann mittels Subnetting beinahe beliebig weiter unterteilt werden, ohne dass die anderen Netzteilnehmer spezielle Maßnahmen ergreifen müssen.

14.4.6 Alles über ifconfig

Das Programm `ifconfig` dient zur Verwaltung und Aktivierung der entsprechenden Kernel-Netzwerk-Interfaces. Das erste Interface heißt `eth0` für Ethernet-Karten und `tr0` für Token-Ring-Karten. Das zweite Interface heißt `eth1` bzw. `tr1`. Der Kernel selbst erkennt nur das erste Interface automatisch, das zweite muss entweder durch eine Anpassung in der Datei `drivers/net/Space.c` oder durch eine Kernel-Kommandozeile angegeben werden.

Mit `ifconfig` können Interfaces gestoppt (`down`) oder gestartet (`up`) werden. Wird `ifconfig` ohne Parameter aufgerufen, dann gibt es den Status der Schnittstellen aus. Wird als Parameter der Name einer Schnittstelle angegeben, so wird nur der Status dieser Schnittstelle angezeigt. Außerdem können eine Reihe von zusätzlichen Einstellungen durchgeführt werden:

- `arp` bestimmt, ob ARP (Address Resolution Protocol) von diesem Interface verwendet wird. Soll ARP nicht verwendet werden, so kann vor der Option ein Minuszeichen (`-`) angegeben werden. Wenn Sie Ethernet verwenden, dann müssen Sie ARP verwenden, sonst ist Ihr Rechner nicht im Netz erreichbar. Dieses Protokoll wird im gleichnamigen Abschnitt 14.5, »Das Address Resolution Protocol (ARP)« ausführlich erläutert.

- Die Maximum Transfer Unit (MTU) ist die maximale Größe eines IP-Pakets, das über dieses Device übertragen wird. Für serielle Verbindungen sollte ein relativ kleiner Wert vorgesehen werden, da dann die Antwortzeiten bei interaktiven Programmen besser sind, es werden z. B. weniger Bytes für ein Zeichen bei `telnet` übertragen. Für die Dateiübertragung mit `ftp` ist jedoch eine große MTU besser, da dann der Overhead, also die für Header und Prüfsummen anstelle von Nutzdaten verwendete Anzahl an Bytes, sinkt.

  Die MTU ist durch das verwendete Medium bestimmt. Beim Einsatz von Brücken zwischen unterschiedlichen Netzstrukturen muss stets die kleinste MTU aller Netze verwendet werden. Router können IP-Pakete mittels Fragmentierung auch über Medien mit einer kleineren Paketgröße übertragen.
- `ifconfig` erkennt anhand der angegebenen IP-Adresse, um welche Netzklasse es sich handelt, und bestimmt die entsprechende Netmask automatisch. Zu Zwecken des Subnetting kann nach dem Parameter `netmask` eine eigene Netmask vorgegeben werden.
- Der Standardwert für die `broadcast`-Adresse wird automatisch bestimmt, kann aber ebenfalls verändert werden, um Subnetting betreiben zu können. Mit dem Parameter `-broadcast` kann die Verwendung von Broadcasts auf diesem Interface deaktiviert werden.
- `pointopoint`, gefolgt von der IP-Adresse der Gegenstelle, konfiguriert das angegebene Interface für SLIP, PPP oder PLIP.
- Mit dem Parameter `hw` kann die Hardware-(Ethernet-)Adresse des Adapters verändert werden, sofern der entsprechende Treiber das unterstützt. Dies kann in speziellen Netzkonfigurationen sinnvoll sein, wenn der Partner auf Hardware-Adressen konfiguriert werden muss.
- Als letzter Parameter kann dem Interface eine IP-Adresse zugewiesen werden. Ist dies der Fall, so wird das Interface automatisch aktiviert (`up` wird ausgeführt).

Listing 14.4 zeigt beispielhaft die Ausgabe von `ifconfig` auf einem Rechner mit aktivem Token-Ring-Anschluss (und natürlich `loopback`). Für jedes Interface werden verschiedene Daten angezeigt: Hardware-Adresse, vergebene IP-Adresse (inklusive Netmask und Broadcast-Adresse), der Status der Schnittstelle und die MTU. In den folgenden zwei Zeilen wird die Anzahl der empfangenen (RX) und gesendeten (TX) Pakete angegeben. Dabei wird auch die Anzahl der Fehler, der ausgelassenen Pakete und der Überläufe angegeben. In der Regel sollten diese Zahlen, verglichen mit der Anzahl der insgesamt übertragenen Pakete, für einen normal belasteten Rechner sehr klein sein.

```
(linux):~$ /sbin/ifconfig
lo   Link encap:Local Loopback
     inet addr:127.0.0.1 Bcast:127.255.255.255 Mask:255.0.0.0
     UP BROADCAST LOOPBACK RUNNING  MTU:2000  Metric:1
```

```
      RX packets:0 errors:0 dropped:0 overruns
      TX packets:50 errors:0 dropped:0 overruns:0

tr0   Link encap:UNSPEC  HWaddr 10-00-5A-D2-62-F8
      inet addr:192.168.3.5 Bcast:192.168.3.255 Mask:255.255.255.0
      UP BROADCAST RUNNING  MTU:2000  Metric:1
      RX packets:10723 errors:0 dropped:0 overruns:0
      TX packets:5456 errors:0 dropped:0 overruns:0
      Interrupt:3 Base address:0xa20
```

Listing 14.4 Ausgabe des Programms `ifconfig` *für Loopback- und Token-Ring-Interface*

Die meisten Distributionen bringen Skripten mit, die das Starten bzw. Stoppen der Netzwerkgeräte übernehmen. Unter Red Hat können Sie mit dem Befehl `/etc/init.d/network stop` das Netz anhalten und mit dem Befehl `/etc/init.d/network start` wieder starten. Das ist viel einfacher als manuelles Vorgehen.

14.4.7 Alles über route

Das Programm `route` dient zur Festlegung von statischen Routen, die die Erreichbarkeit von anderen Rechnern im Internet sicherstellen. Wie oben bereits erklärt, ist die direkte Verbindung von Rechnern mit Ethernet nur bis zu einer bestimmten Netzgröße sinnvoll. Danach muss eine Trennung in logisch zusammengehörende Teilnetze vorgenommen werden.

Die Verbindung zwischen den einzelnen Teilnetzen wird durch Router (früher auch als Gateways bezeichnet) hergestellt. Ein Router arbeitet auf Protokollebene (z. B. IP, IPX oder SNA) und kann aufgrund der in den Protokollen enthaltenen Informationen eine geeignete Verbindung herstellen. Im Folgenden betrachten wir nur das Routing von Internet-Protokollen, also IP-Routing.

Es wird mit dem Programm `route` festgelegt, welche Rechner oder Netze über welches Interface erreichbar sind bzw. welche IP-Adresse als Standard-Gateway verwendet wird. Routen können, sofern das Programm `routed` oder `gated` gestartet ist, mit verschiedenen Routing-Protokollen verändert werden. Im privaten, lokalen Netz ist dies meist nicht notwendig bzw. sogar eher störend.

Eine weitere Möglichkeit, wie Routen dynamisch verändert werden können, sind ICMP-Redirect-Nachrichten. Eine ausführliche Beschreibung dieses Protokolls finden Sie im [rfc0792].

Der Befehl `route` ohne Parameter gibt (nur unter Linux) die Kernel-Routing-Tabellen aus. Der Parameter `-n` bewirkt die Ausgabe von IP-Adressen anstelle der Host-Namen, so dass kein Nameserver befragt werden muss. Besser, weil auf praktisch allen Unix-Systemen verfügbar, ist `netstat -r`. Auch dieser Befehl gibt die aktuellen Routing-Tabellen des Kernels lesbar aus. Diese Informationen stellt der Kernel in der Datei `/proc/net/route` bereit.

```
(linux):~$ route -n
Kernel routing table
Destination Gateway     Genmask       Flags MSS  Window Use Iface
192.168.3.0 *           255.255.255.0 U     1872 0     5537 tr0
127.0.0.0   *           255.0.0.0     U     1872 0       41 lo
default     192.168.3.1 *             UG    1872 0        0 tr0
```

Listing 14.5 Ausgabe des Programms `route`

Nach der Aktivierung eines Interface mit `ifconfig` können mit `route` die entsprechenden Routen zu Rechnern oder Netzen konfiguriert werden. Eine Route wird mit dem Parameter `add` hinzugefügt. Gilt die Route für einen Rechner, so folgt die entsprechende Adresse dem Schlüsselwort `-host`. Ist es die Route zu einem Netz, so wird `-net` angegeben.

Auf jedem TCP/IP-Rechner kann der eigene Rechner mit der dafür vorgesehenen IP-Adresse `127.0.0.1` bzw. dem Namen `localhost` angesprochen werden. Das entsprechende Netz `127.0.0.0` wird oft als `localnet` bezeichnet. Für den Zugriff auf die `loopback`-Schnittstelle muss diese mit `ifconfig` aktiviert und die entsprechende Route eingerichtet sein. Das geschieht mit dem Befehl aus Listing 14.6.

```
route add -net 127.0.0.0
```

Listing 14.6 Hinzufügen der Route für das `localnet`

Das Programm `route` verwendet hier automatisch die Netmask `255.0.0.0`, da es die Adresse einem Class-A-Netz zuordnet. Sie können mit dem Parameter `netmask` eine andere Netmask angeben. Außerdem wird ebenfalls automatisch die Netzwerkschnittstelle `lo` verwendet. Mit dem Parameter `dev` kann eine Schnittstelle angegeben werden, falls der automatisch bestimmte Wert nicht korrekt ist.

Das nächste Beispiel (siehe Listing 14.7) zeigt die Einrichtung einer Route zu einem Netz über die Ethernet-Schnittstelle. Auf die Angabe der Netmask könnte verzichtet werden, da die verwendete Adresse eine Class-C-Adresse ist.

```
route add -net 192.168.3.0 netmask 255.255.255.0 dev eth0
```

Listing 14.7 Hinzufügen einer Route für das lokale Ethernet

Mit dem Parameter `default` kann eine Standardroute angelegt werden, die verwendet wird, wenn kein anderer Routing-Eintrag passt. Wenn Ihr Netz mit dem Internet verbunden ist, tragen Sie hier z. B. den Router zum Internet ein. Eine Anwendung dafür finden Sie im Listing 14.8.

```
route add default gw router-3.jochen.org
```

Listing 14.8 Hinzufügen einer Default-Route

Listing 14.8 wurde statt der IP-Adresse ein Host-Name angegeben. Das ist so lange möglich, wie der Name ohne Zugriff auf Rechner außerhalb des eigenen Subnetzes aufgelöst werden kann. Dies ist der Fall, wenn der Name des Gateway in der `/etc/hosts` steht oder der Nameserver sich im gleichen Subnetz befindet. In der Regel ist es eine eher schlechte Idee, einen Namen anstelle der IP-Adresse zu verwenden.

Eine Route kann, z. B. bei Problemen oder beim Ende einer Wählverbindung, aus der Routing-Tabelle mit dem Parameter `del` gelöscht werden. Die Befehle in Listing 14.9 entfernen die Routing-Einträge der im letzten Beispiel verwendeten SLIP-Verbindung.

```
route del -net 192.168.6.0
route del slipsrv
```

Listing 14.9 Löschen von Routen

14.4.8 Das Programm netstat

Das Programm `netstat` ist eine weitere Möglichkeit, über den aktuellen Zustand des Netzes informiert zu werden. Dazu stehen eine Reihe von Optionen bereit, die verschiedene Informationen anzeigen. Dabei vereinigt `netstat` eine Reihe von Utilities in einem Programm. Die verschiedenen Funktionen werden über Optionen angewählt.

- Die Option `-a` erzeugt eine Liste aller aktiven Internet Sockets, also TCP, UDP, RAW und UNIX Sockets. Dabei werden auch alle Ports ausgegeben, auf denen nur gelauscht wird (Listening). Mit diesem Befehl kann man sich über die bestehenden Verbindungen und deren Status informieren.
- `-i` zeigt eine Statistik der Netz-Devices an. Dabei werden gesendete und empfangene Pakete sowie aufgetretene Fehler, Drops oder Overruns ausgewiesen. Eine entsprechende Ausgabe sehen Sie in Listing 14.10. In der letzten Spalte finden Sie eine Reihe von Flags, die die Konfiguration des Interface genauer beschreiben. Diese Flags sind in Tabelle 14.4 aufgeführt.
- `-c` erzeugt die angeforderte Liste jede Sekunde neu, so dass Veränderungen festgestellt werden können. Das Programm muss mit Strg+C beendet werden.
- `-n` verhindert die Ausgabe von Host-Name und Service, so dass nur die numerischen Werte angezeigt werden. Damit wird der »Reverse-Lookup« im Nameserver ausgelassen, was zu langen Timeouts führen kann und die Netzlast erhöht.
- `-l` zeigt nur die Ports, auf denen ein Prozess auf Verbindungen wartet.
- `-o` Anzeige des Timer-Status.

- `-p` gibt zusätzlich die Prozess-ID und den Namen des Prozesses aus.
- `-r` ist unter Linux äquivalent zu `route`, es werden die Kernel-Routing-Tabellen angezeigt. Unter anderen Unix-Systemen, wo `route` nicht zur Anzeige der Routing-Tabellen des Kernels verwendet werden kann, muss stets `netstat -r` verwendet werden.
- `-s` Ausgabe der Statistiken aus `/proc/net/snmp`.
- `-t` Informationen über TCP-Sockets anzeigen.
- `-u` Informationen über UDP-Sockets anzeigen.
- `-w` Informationen über RAW-Sockets anzeigen.
- `-x` Informationen über UNIX-Sockets anzeigen.

```
(linux):~> netstat -i
Kernel Interface table
Iface  MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP
TX-OVR Flags
lo    2000   0     0      0      0      0    89      0      0
0 BLRU
tr0   2000   0 10840      0      0      0  1000      0      0
0 BRU
```

Listing 14.10 Ausgabe des Befehls `netstat -i`

Flag	Beschreibung
`A`	Das Interface empfängt alle Multicast-Adressen.
`B`	Broadcasts sind erlaubt.
`D`	Debugging ist eingeschaltet.
`L`	Das ist ein Loopback-Interface.
`M`	Alle Pakete werden empfangen (Promisc-Mode).
`N`	Es werden keine Trailer erzeugt.
`O`	Dieses Interface benutzt kein ARP.
`P`	Das Interface ist eine Point-to-Point-Verbindung.
`R`	Das Interface ist in Betrieb (`running`).
`U`	Das Interface ist `up`.

Tabelle 14.4 Flags bei `netstat -i`

Solange Ihr Rechner nicht vernetzt ist, können Sie, sofern an der Konsole kein Benutzer aktiv ist, den Rechner bedenkenlos herunterfahren. Ist an Ihren Rechner ein serielles Terminal angeschlossen, so sollten Sie vor jedem `shutdown` z. B. mit `w` und `ps aux` nachsehen, ob dort noch ein Benutzer aktiv ist. Ist Ihr Rechner jedoch als Netzwerk-Server im Einsatz, so müssen Sie auch auf andere Netzverbindungen Rücksicht nehmen. Der Befehl `netstat -a` zeigt alle aktiven Verbindungen an.

Gelegentlich werden Sie sich fragen, welches Programm einen bestimmten Port belegt. Das Kommando `netstat -lpn` hilft Ihnen weiter. Wenn Sie noch eine ältere Version von `netstat` verwenden, können Sie ersatzweise auch `lsof` verwenden.

14.5 Das Address Resolution Protocol (ARP)

Auf der Ethernet-Ebene gibt es zwei Typen von Verbindungen zwischen Rechnern: Pakete können direkt an einen Rechner, d. h. an die Hardware-Adresse (MAC-Adresse oder Media Access Control) seiner Ethernet-Karte geschickt werden, oder an alle Rechner im lokalen Netz (Broadcast). Wenn ein Rechner direkt angesprochen werden soll, muss die Ethernet-Adresse[3] bekannt sein. Für den Transport im lokalen Netz wird jedes IP-Paket mit einem zusätzlichen Header versehen, in dem u. a. die Hardware-Adresse des Senders und Empfängers eingetragen werden.

Zunächst ist jedoch nur die IP-Adresse des Empfängers verfügbar. Mit Hilfe des Address Resolution Protocol (ARP) wird zur IP-Adresse die passende MAC-Adresse beschafft. Der Sender schickt ein Broadcast an alle lokalen Rechner und sucht die zur gewünschten IP-Adresse gehörende Ethernet-Adresse. Das zugrunde liegende *Address Resolution Protocol* ist im [rfc0826] definiert. Die Antwort (die Zuordnung von der IP- zur Hardware-Adresse) vom gewünschten Empfänger wird im ARP-Cache gespeichert, so dass diese Abfrage nicht ständig wiederholt werden muss. Damit beim Austausch einer Karte dieser Rechner auch angesprochen werden kann, verfallen die Einträge im ARP-Cache nach kurzer Zeit und werden dann neu erstellt.

Listing 14.11 zeigt die Aufzeichnung eines `ping` vom Rechner `jupiter` zum Rechner `typhon`. In der ersten Zeile fragt das System `jupiter` per Broadcast, welcher Rechner denn `typhon` sei. Die zweite `arp`-Zeile stellt die entsprechende Antwort dar: Der Rechner `typhon` meldet sich mit seiner Hardware-Adresse. Die restlichen Pakete sind der `ping` und seine Antworten. Es ist keine neue oder weitere `arp`-Anfrage nötig. Für die Gegenrichtung kennt ja der Rechner `typhon` die IP- und Ethernet-Adresse des Rechners `jupiter`.

```
(linux):# tcpdump -t -N -i eth0
tcpdump: listening on eth0
arp who-has typhon tell jupiter
arp reply typhon is-at 8:0:5a:3b:aa:30
jupiter > typhon: icmp: echo request
typhon > jupiter: icmp: echo reply
```

3. Einige Treiber unterstützen den Promiscous Mode, in dem alle Pakete vom Ethernet gelesen werden. Dies ist für die Netzwerküberwachung wichtig, stellt aber auch ein Sicherheitsproblem dar.

```
jupiter > typhon: icmp: echo request
typhon > jupiter: icmp: echo reply
```

Listing 14.11 Ein ARP-Request, mit `tcpdump` *aufgezeichnet*

Mit dem Programm `arp` können Sie diesen ARP-Cache anzeigen und verändern. Dies kann nützlich sein, wenn zwei Rechner (mit verschiedenen Hardware-Adressen) meinen, dieselbe IP-Adresse benutzen zu können. Solche Fehler sind mit anderen Mitteln kaum zu diagnostizieren. ARP-Cache-Einträge werden mit der Option `-a` angezeigt. Einträge können mit der Option `-d` gelöscht und mit der Option `-s` hinzugefügt werden.

```
(linux):~# arp -a
Address     HW type         HW address         Flags Mask
192.168.3.161 10Mbps Ethernet 00:00:83:21:D1:FA  C     *
192.168.3.150 10Mbps Ethernet 10:00:5A:B3:01:E9  C     *
```

Listing 14.12 Das Programm `arp`

Eine weitere Anwendung für das Programm `arp` ist das Routen von IP-Paketen, z. B. zu einem Rechner, der mittels SLIP oder PPP angeschlossen ist. Der lokale Rechner nimmt Pakete für den entfernten Rechner an und leitet diese über die Punkt-zu-Punkt-Verbindung weiter. Damit ist es nicht notwendig, ein Subnetz abzuspalten, wodurch man kostbare IP-Adressen verschenkt.

Beim zwischen Ethernet und PPP routenden Rechner wird der ARP-Cache so verändert, dass er seine eigene Ethernet-Adresse als die auch für den PPP-Rechner geltende verbreitet. Dazu dient der Parameter `pub` beim Aufruf von `arp`. In Listing 14.13 wird für den Rechner ein Proxy-ARP für die eigene Ethernet-Adresse (00:00:54:34:76:23) eingerichtet.

```
arp -s 00:00:54:34:76:23 pub
```

Listing 14.13 Einrichten eines Proxy-ARP

Sie können mit Linux auch Proxy-ARP für ganze Subnetze einrichten, was manchmal sinnvoll sein kann. Wenn Sie für mehrere einzelne Rechner Proxy-ARP verwenden wollen, so können Sie die Host-Namen und zugehörigen Hardware-Adressen in eine Datei, z. B. `/etc/ethers`, eintragen und den Namen der Datei nach der Option `-f` angeben.

Wenn Sie besonders hohe Sicherheitsansprüche haben, dann können Sie ARP auch komplett abschalten und für jeden Rechner die entsprechenden Einträge statisch in den ARP-Cache aufnehmen. Dann kann nur der Rechner mit der passenden Hardware-Adresse eine IP-Adresse übernehmen. Man handelt sich damit allerdings eine Menge zusätzlichen Aufwand bei der Konfiguration und im täglichen Betrieb ein.

14.6 Advanced Routing

Die bisher beschriebenen Befehle reichen aus, um ein Netzwerk unter praktisch jedem Unix-System zu aktivieren. Linux bietet jedoch einige erweiterte Optionen, die mit diesen Programmen nicht eingerichtet werden können. Alternativ können Sie zur Konfiguration Ihres IP-Netzwerks auch den Befehl `ip` aus dem `iproute`-Paket (*ftp://ftp.inr.ac.ru/ip-routing/*) verwenden.

Als ersten Parameter erwartet `ip` das zu verändernde Objekt. Mögliche Werte sind `link` (network device), `address` (protocol (IP or IPv6) address on a device), `neighbour` (ARP or NDISC cache entry), `route` (routing table entry), `rule` (rule in routing policy database), `maddress` (multicast address), `mroute` (multicast routing cache entry) und `tunnel` (tunnel over IP). Als zweiten Parameter geben Sie an, was Sie mit dem Objekt tun wollen.

Mit den bekannten Tools können Sie zum Beispiel keinen GRE-Tunnel einrichten – Sie benötigen neue Tools. In das Programm `ip` ist diese Funktionalität integriert. Auch viele andere, relativ neue Funktionen des »Advanced Routing« sind hier implementiert. Für mehr Informationen schauen Sie auch in die entsprechende HowTo.

Damit ist es möglich, mit einem relativ konsistenten Befehl alle Netzwerkeinstellungen vorzunehmen. Die genaue Syntax des Befehls und viele Beispiele finden Sie in der mitgelieferten Dokumentation (`ip-cref.tex`). Leider existiert keine Manpage, in der man wie gewohnt schnell nachschlagen könnte.

15 TCP/IP-Grundlagen

15.1 Protokolle

Im letzten Kapitel erfolgte die Konfiguration eines Linux-Rechners, damit dieser an einem TCP/IP-basierten Netz teilnehmen kann. In diesem Abschnitt soll eine Einführung in die einzelnen Protokolle und deren Verwendung innerhalb des Netzes erfolgen. Damit sollen Sie in die Lage versetzt werden, die Arbeitsweise eines TCP/IP-Netzes zu verstehen. Wie bereits im letzten Kapitel in Abbildung 14.2 gezeigt, basierten auf der eigentlichen Hardware-Verbindung eine Reihe von Protokollschichten, deren Zusammenspiel hier genauer erklärt wird.

Das Internet basiert auf einer Reihe von Protokollen, die zum geordneten Datenaustausch zwischen Rechnern dienen. Die Protokollnummern sind in der Datei `/etc/protocols` (siehe Listing 15.1) definiert. Es ist äußerst unwahrscheinlich, dass Sie diese Datei jemals verändern müssen.

```
ip      0       IP              # internet protocol, pseudo 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
ipv6    41      IPv6            # IPv6
ipv6-route 43   IPv6-Route      # Routing Header for IPv6
ipv6-frag 44    IPv6-Frag       # Fragment Header for IPv6
ipv6-icmp 58    IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt 59   IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts 60    IPv6-Opts       # Destination Options for IPv6
raw     255     RAW             # RAW IP interface
```

Listing 15.1 Internetprotokolle (`/etc/protocols`)

Im Wesentlichen werden vier Protokolle verwendet, deren Existenz im Kernel Linux auch beim Booten meldet:

- IP, das *Internet Protocol*, dient als Basis für TCP und UDP. Es ist ein verbindungsloses Protokoll, das keine Mechanismen zur Sicherstellung der Datenübertragung enthält. Alles, was von der Implementierung verlangt wird, ist ein »best effort« bei der Zustellung der Pakete. Außerdem kann mit ICMP die Verbindungsqualität verbessert werden. Das Protokoll ist in [rfc0791] definiert.

- ICMP, das *Internet Control Message Protocol*, dient zum Austausch von Nachrichten zwischen Rechnern auf einer sehr niedrigen Ebene. Dieses Protokoll ist unter Linux wie bei vielen anderen Systemen direkt im Kernel selbst implementiert. Die einzelnen Nachrichtentypen des ICMP sind in [rfc0792] definiert. Die wichtigsten sind: ICMP-Echo (wird von `ping` verwendet), ICMP-Redirect (Verändern von Routen) und ICMP-Nachrichten, die die Ablehnung eines Verbindungsaufbaus melden (»connection refused«).

  Sie können das Programm `icmpinfo` verwenden, um sich ICMP-Pakete anzeigen und mitprotokollieren zu lassen. Besonders bei Rechnern in sehr aktiven Netzen werden viele Meldungen ausgegeben. Daher werden die »unwichtigen« Nachrichten nur auf spezielle Anforderung ausgewiesen. Mehr zu diesem Programm lesen Sie in Kapitel 28, »Netzwerkadministration«.
- TCP (*Transmission Control Protocol*, [rfc0793]) ist ein verbindungsorientiertes, verlässliches (reliable) Protokoll, das aber recht langsam ist. Anwendungsprogramme können sich darauf verlassen, dass alle Pakete in der richtigen Reihenfolge übertragen werden. Der Kernel stellt durch Neuübertragung, Bestätigungen und Timeouts die Vollständigkeit der Daten sicher.
- UDP (*User Datagram Protocol*, [rfc0768]) ein verbindungsloses, nicht verlässliches (unreliable) Protokoll, das schneller als TCP ist. Der Geschwindigkeitsvorteil entsteht u. a. dadurch, dass der Verbindungsaufbau, der bei TCP aus drei Schritten besteht, nicht erforderlich ist. Auf der Basis dieses Protokolls können Anwendungen jedoch wieder sichere Protokolle implementieren, obwohl von der Netzwerkschicht keine Garantie für eine vollständige Übertragung gegeben werden kann.

Die Anwendungsdaten werden, je nach Protokoll, mit dem passenden Header versehen. Diese Header sind in den jeweiligen RFCs definiert. Jede zusätzliche Protokollschicht sorgt dafür, dass mehr Verwaltungsdaten übertragen werden, also der Overhead steigt. Erst im letzten Schritt werden die Pakete in das für das Übertragungsmedium passende Format gebracht, also z. B. mit einem Ethernet- oder Token-Ring-Header versehen und dann auf das Übertragungsmedium ausgegeben.

Auf der Basis von TCP und UDP werden weitere (Anwendungs-)Protokolle oder Dienste (in der Datei `/etc/services`, siehe Listing 15.2) definiert, die dann von einzelnen Anwendungen, wie z. B. `telnet` oder `ftp`, verwendet werden. Erst diese Anwendungen machen den Nutzen eines Netzes aus. In der Datei `/etc/services` werden Protokollnamen einem `tcp`- oder `upd`-Port zugeordnet, über den die Kommunikation stattfindet.

Die Datei `/etc/services` (siehe Listing 15.2) enthält in der ersten Spalte den offiziellen Namen des Protokolls. In der zweiten Spalte (durch Tabulatoren oder Leerzeichen getrennt) finden Sie die Nummer des Ports und das verwendete Protokoll. Ein Port kann gleichzeitig für Dienste auf UDP und TCP verwendet

werden. Aufgrund der in `/etc/protocols` festgelegten und im Paket-Header verwendeten Protokollnummern wird der richtige Treiber im Kernel ausgewählt und die Daten werden an das richtige Programm weitergereicht. In weiteren Spalten können Aliasnamen für die Ports definiert werden. Kommentare werden durch eine Raute eingeleitet und reichen bis zum Zeilenende.

```
tcpmux          1/tcp         # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp         # sink null
discard         9/udp         # sink null
```

Listing 15.2 Auszug aus der Datei `/etc/services`

Viele Protokolle sind inklusive ihrer Port-Nummern Internet-weit standardisiert. Die offiziell belegten Ports werden von der Internet Assigned Numbers Authority (IANA, *http://www.iana.org*) verwaltet. Die Port-Nummern sollten nach Möglichkeit eindeutig sein. Daher sollten Programmierer die von ihnen verwendeten Ports bei der Internet Assigned Numbers Authority (IANA) registrieren lassen.

Die Nummern der Ports können im Bereich von 1 bis 65.535 liegen. Dabei sind die Ports von 1 bis 1024 für den Benutzer `root` reserviert. Damit ist sichergestellt, dass am anderen Ende der Verbindung auch ein autorisierter Prozess wartet, z. B. bei einer `telnet`- oder `ftp`-Verbindung, über die auch das Passwort (unverschlüsselt) übertragen wird.

Diese Unterscheidung zwischen `root` und anderen Benutzern des Systems existiert nur bei Unix, so dass man sich darauf nicht in jedem Fall verlassen kann. Selbst bei Unix-Systemen kann man sich dabei nicht sicher sein. Ab der Kernel-Version 2.2 wird Linux so genannte Capabilities kennen. Mit einer davon (`CAP_NET_BIND_SERVICE`) erlangt ein Programm Zugriff auf reservierte Ports, ohne dass der Benutzer `root` seine Finger im Spiel hat.

C-Programme verwenden intern Port-Nummern. Die Port-Namen können mit der Funktion `getservbyname(3)` in die entsprechende Nummer umgewandelt werden. In einem großen Netz kann es sehr aufwändig sein, die `services`-Dateien auf den verschiedenen Rechnern konsistent und aktuell zu halten. In diesen Fällen kann die Datei z. B. mittels NIS (siehe Kapitel 23, »Network Information Service«) verteilt werden.

Die Kommunikation zwischen zwei Rechnern findet über diese Ports statt. Dazu muss an einem Ende ein Programm eine Verbindung über diesen Port aufbauen, auf der anderen Seite muss ein anderes Programm diesen Port überwachen und die notwendigen Aktionen durchführen. Dafür gibt es zwei Möglichkeiten:

- Der Server-Prozess wird (im Hintergrund) gestartet und lauscht auf dem Port (Listening). Damit ist der Server vollständig für den gesamten Verbindungsaufbau zuständig.
- Der Server wird vom `inetd` gestartet, wenn auf dem passenden Port eine Anfrage eintrifft. Die zu beobachtenden Ports und die auszuführenden Optionen und Programme werden in der Datei `/etc/inetd.conf` konfiguriert. Ein Port kann nicht gleichzeitig mit beiden Möglichkeiten beobachtet werden.

Ob auf der anderen Seite der Verbindung ein entsprechender Server läuft, lässt sich bei einer `tcp`-Anwendung feststellen, indem eine `telnet`-Verbindung zu diesem Port aufgebaut wird. So dient z. B. der Service `echo` zur Überprüfung, ob Zeichen in beide Richtungen übertragen werden können. Sie können aber auch Anwendungsprotokolle wie z. B. `smtp` (Simple Mail Transfer Protocol) mittels `telnet` bedienen und so den entsprechenden Server testen.

Zum ersten Test werden oft die Protokolle `echo` und `discard` verwendet. Das Listing 15.3 zeigt eine entsprechende Anwendung. Allerdings können diese Dienste auch für so genannte »Denial of Service«-Angriffe verwendet werden. Aus diesem Grund sind diese Dienste bei vielen Systemen nicht verfügbar – da sie normalerweise nicht benötigt werden, auch mit Recht.

```
(linux):~$ telnet localhost echo
Trying 127.0.0.1...
Connected to localhost.
Escape character is `^]'.
Das ist ein Test
Das ist ein Test
^]
telnet> quit
Connection closed.
```

Listing 15.3 `telnet` *als Test-Tool für TCP-basierte Dienste*

Wenn Sie einen Überblick über die aktuell gestarteten Dienste auf Ihrem System benötigen, dann können Sie dazu den Befehl `netstat -a` verwenden. Es werden alle bestehenden Verbindungen angezeigt sowie alle Ports, auf denen ein Dienst angeboten wird.

15.2 Der inetd-Server

Auf einem Server müssen eine Reihe von Ports überwacht werden, damit Anfragen der Clients korrekt beantwortet werden. Sollte nun für jeden Port ein eigener Hintergrundprozess (Dämon) gestartet werden, würde schnell das Systemlimit an Prozessen erreicht werden. Daher wird beim Systemstart nur der `inetd`-Ser-

ver gestartet. In der Datei /etc/inetd.conf wird konfiguriert, welche Ports zu überwachen und welche Aktionen auszuführen sind, wenn eine Anfrage eintrifft (siehe Listing 15.4).

```
serv soc_typ prot flags  user server_path    args
echo  stream  tcp  nowait root internal
ftp   stream  tcp  nowait root /usr/sbin/tcpd /usr/sbin/wu.ftpd
talk  dgram   udp  wait   root /usr/sbin/tcpd /usr/sbin/in.ntalkd
```

Listing 15.4 Auszug aus der Datei /etc/inetd.conf

Kommentare beginnen mit einer Raute und reichen bis zum Zeilenende. Für jeden Service aus der Datei /etc/services kann eine Zeile aufgenommen werden, sofern der entsprechende Dämon den Start durch inetd unterstützt. In der ersten Spalte steht der Name des Anwendungsprotokolls. Hier sollten Sie den offiziellen Namen aus der Datei /etc/services verwenden.

In der zweiten Spalte wird der Typ des Socket angegeben. Erlaubte Werte sind stream, dgram für Datagramme, raw, rdm (Reliably Delivered Message) oder seqpacket (Sequenced Packet Socket). Wenn Sie einen neuen Service in diese Datei aufnehmen wollen, weil Sie z. B. einen neuen Dämon installieren, finden Sie den entsprechenden Eintrag in der Regel in der Dokumentation des entsprechenden Programms.

Die dritte Spalte enthält ein gültiges Protokoll aus der Datei /etc/protocols. Für TCP- und UDP-Dienste wird hier tcp bzw. udp eingetragen. Wenn Sie auch RPC-Dienste mit inetd verwalten wollen, können diese mit rpc/tcp oder rpc/udp eingetragen werden (Genaueres dazu finden Sie in der Manpage zu inetd(8)).

In der nächsten Spalte wird angegeben, was der inetd-Server tut, wenn eine Verbindung aufgebaut wird. Die Unterscheidung zwischen wait und nowait ist nur für Datagramm-Sockets interessant, für alle anderen muss hier nowait eingetragen werden. Wenn eine Verbindung aufgebaut wird (und der entsprechende Dämon das unterstützt), kann der entsprechende Socket wieder durch inetd überwacht und möglicherweise weitere Anfragen bearbeitet werden. Dazu muss nowait eingetragen werden. Bei Diensten, die nicht multi-threaded sind, muss wait eingetragen werden. Wenn Sie die maximale Anzahl an Verbindungen, die innerhalb von 60 Sekunden bearbeitet werden, einschränken wollen, können Sie diese nach einem Punkt (.) angeben.

Die fünfte Spalte enthält den Benutzernamen, unter dessen Kennung der Dienst ausgeführt wird. Wenn Sie auch die Gruppe ändern wollen, so können Sie diese, durch einen Punkt getrennt, angeben. Viele Dämonen müssen mit root-Rechten gestartet werden. Bei einzelnen Diensten (wie z. B. finger) kann es aus Sicherheitsüberlegungen sinnvoll sein, den Dienst als nobody oder unter einer anderen Kennung zu starten.

Als nächster Eintrag folgen jetzt der Name des zu startenden Programms und die eventuell zu übergebenden Parameter. Die Parameter beginnen mit `argv[0]`, also dem Programmnamen selbst. Ältere Unix-Systeme kennen hier eine Beschränkung der Parameter auf fünf Einträge, so dass möglicherweise erst ein Skript gestartet werden muss, das den Server mit den passenden Parametern aufruft.

Der `inetd`-Server stellt einige einfache Dienste (`echo`, `discard`, `chargen`, `daytime` und `time`) intern bereit. Dort wird als Programm `internal` eingetragen.

Nach einer Änderung an der Datei `inetd.conf` muss der Dämon diese neu lesen. Dazu kann er entweder beendet und neu gestartet werden oder man sendet ihm das Signal `SIGHUP`. Das geschieht z. B. mit dem Befehl `kill -HUP $(cat /var/run/syslogd.pid)` oder `kill -HUP inetd`.

Im Allgemeinen wird der `inetd`-Server jede Verbindung zulassen; für die Autorisierung ist die jeweilige Anwendung zuständig. Viele Anwendungen kennen aber keine derartigen Verfahren, so dass ein Rechner, der Internet-Services anbietet, relativ ungeschützt ist. In der Regel wird daher ein lokales Netz durch eine Firewall, die auch unter Linux installiert werden kann, vom Internet abgetrennt. Die Wartung einer Firewall erfordert viel Zeit und Know-how und ist für den privaten Anwender in der Regel zu aufwändig. Wenn Sie sich dennoch mit der Konfiguration einer Firewall beschäftigen wollen, dann werfen Sie einen Blick in die Firewall-HowTo.

15.3 Der TCP-Wrapper (tcpd)

Einen gewissen Schutz implementiert der TCP-Wrapper (`tcpwrap` bzw. `tcpd`). Dieses Programm wird anstelle des richtigen Dämons gestartet. Es prüft anhand der Dateien `/etc/hosts.allow` und `/etc/hosts.deny`, ob die Verbindung zulässig ist und startet den richtigen Anwendungsserver. Damit ist es möglich, bestimmte Protokolle für bestimmte Rechner oder Netze freizugeben oder zu sperren. Dabei verlässt man sich allerdings darauf, dass ein Rechner auch tatsächlich die IP-Adresse hat, unter der er die Verbindung aufbauen will. Das Vortäuschen einer anderen IP-Adresse nennt man auch IP-Spoofing. Der TCP-Wrapper versucht, durch gezieltes Nachforschen mögliche Spoofing-Attacken auszuschließen. Weitere Informationen zu diesen Themen finden Sie in der Manpage `tcpd(8)`.

Das Format der Einträge in den Dateien `hosts.deny` bzw. `hosts.allow` ist in der Manpage `hosts_access(5)` beschrieben. Zu jedem Protokoll und zu jedem Host bzw. jeder Domain oder jedem IP-Adressbereich lässt sich festlegen, ob ein Service erlaubt oder verboten ist und welche Aktionen auszuführen sind. Es können Log-Einträge (mit `syslog`) geschrieben werden, »sichere« Programme

aufgerufen werden, die die angeforderte Funktion durchführen, oder gemäß [rfc1413] bzw. [rfc1413] kann man versuchen, den Benutzernamen auf dem entfernten Rechner zu erhalten (siehe Abschnitt 15.4, »Der ident-Dämon«). Die folgenden Beispiele benutzen die erweiterte Syntax, die in der Manpage hosts_options(5) erklärt wird.

Listing 15.5 zeigt zunächst ein Beispiel für das Freischalten des telnet-Dienstes. Dabei wird zwischen internen Verbindungen, die unmittelbar durchgeführt werden, und externen Verbindungen unterschieden. Interne Verbindungen (aus dem Netz 192.168. bzw. der Domain jochen.org) werden ohne weitere Aktionen erlaubt. Bei externen Verbindungen wird mit syslog eine Log-Datei geschrieben. Wo sich die Einträge wiederfinden, ist in der Datei /etc/syslog.conf (siehe auch Abschnitt 4.4.4, »System-Logs«) festgelegt.

```
 "ftp" und "telnet" von ausserhalb mitloggen...
in.ftpd: 192.168.
in.telnetd: .jochen.org
in.ftpd: ALL@ALL: severity local0.notice
in.telnetd: ALL@ALL: severity local0.notice
```

Listing 15.5 Beispiel für die Datei /etc/hosts.allow

Anstelle der Netznummern kann auch ein Pattern für den Domainnamen verwendet werden. Netznummern werden mit einem Punkt abgeschlossen, Domainnamen müssen mit einem Punkt beginnen. Wenn Sie das Network Information Service (NIS) verwenden, dann können Sie mit @ eine Netgroup angeben. Das Matching kann auch durch eine Netznummer- und Netmask-Kombination angegeben werden. So steht der Eintrag 192.168.72.0/255.255.254.0 für jede Adresse zwischen 192.168.72.0 und 192.168.73.255.

Der Eintrag ALL gilt für alle Rechner, die bisher noch nicht von einem Eintrag erfasst wurden. Das Schlüsselwort LOCAL steht für lokale Rechner. Diese werden daran erkannt, dass kein Punkt im Host-Name auftaucht, der einen Domainnamen abtrennen würde.

Listing 15.6 zeigt einfache Möglichkeiten, bestimmte Verbindungen von einigen oder allen Rechnern oder Netzen zu verbieten.

```
 Allen weiteren Hosts (ausser den in /etc/hosts.allow)
# TFTP verbieten
in.tftpd: ALL
in.sshdd: 192.168.31.155
ALL EXCEPT in.fingerd: rtfm.mit.edu, .com
```

Listing 15.6 Beispiel für die Datei /etc/hosts.deny

Neben dem Erstellen von Log-Dateien kann auch ein beliebiger Shell-Befehl abgesetzt werden. Damit lassen sich weitere Informationen beschaffen oder z. B. der Systemadministrator kann per Mail informiert werden. Dies kann sinnvoll sein, wenn man mehr Informationen über den entfernten Rechner und die Benutzer dort erhalten möchte. Es ist allerdings nicht sinnvoll, auf einen Zugriff mit dem Programm `finger` mit einem `finger` in der Gegenrichtung zu reagieren. Wenn das beide Administratoren tun, dann sind beide Rechner bis in alle Ewigkeit beschäftigt.

```
in.tftpd: ALL: spawm (/usr/sbin/safe_finger -l @/usr/bin/mail -s
```

Listing 15.7 Shell-Befehle von `tcpd` *starten*

Listing 15.7 zeigt eine Möglichkeit, Shell-Befehle zu starten. Sie sollten entweder absolute Pfade verwenden oder die `PATH`-Variable entsprechend setzen. `tcpd` wartet auf das Ende des Befehls. Wenn Sie die Verbindung schon vorher zulassen wollen, starten Sie den Shell-Befehl im Hintergrund, indem Sie das kaufmännische Und (`&`) an den Befehl anhängen. In dem Shell-Befehl können Sie einige Variablen verwenden, die vom `tcpd` durch die aktuellen Werte ersetzt werden. Eine Übersicht über die Variablen finden Sie in Tabelle 15.1. Beachten Sie, dass nicht immer alle Informationen vorliegen, so kennen z. B. DOS-Rechner keine Benutzerkennungen.

Variable	Ersetzung
`%a`	Adresse des entfernten Rechners
`%c`	*User@Host*, *Host-Name* oder Adresse
`%h`	Host-Name des Rechners (oder nur die Adresse)
`%d`	Name des zu startenden Dämons
`%p`	Prozess-ID des Dämons
`%u`	Benutzername (oder `unknown`)
`%%`	Ein einzelnes `%`-Zeichen

Tabelle 15.1 Variablensubstitution durch `tcpd`

15.4 Der ident-Dämon

Viele Protokolle und `tcpd` bieten keine Möglichkeit, die Benutzeridentität zu überprüfen. Oft ist dies nicht möglich, da z. B. DOS-PCs keine Benutzerkennungen haben. Das `ident`-Protokoll [rfc1413] ermöglicht es jedoch, unter bestimmten Voraussetzungen die Benutzerkennung, unter der eine Verbindung aufgebaut wurde, festzustellen. Dazu muss auf dem Rechner, der die Verbindung aufgebaut hat, der `identd`-Dämon installiert sein, was nur selten der Fall ist.

Die Informationen, die dieses Protokoll liefert, betreffen nur bestehende Verbindungen zwischen beiden Rechnern. Es ist also nicht möglich, etwas über Verbindungen zu anderen (fremden) Systemen zu erfahren. Im besten Fall erhalten Sie die Benutzerkennung der Person, die die Verbindung aufgebaut hat. Im schlimmsten Fall erhalten Sie völlig sinnlose Informationen. Daher sollten Sie diese Daten nicht zur Authentifizierung einsetzen, sondern z. B. in Log-Dateien aufnehmen und mit der gebotenen Vorsicht betrachten.

Zunächst scheint der Betrieb eines `ident`-Servers elementaren Sicherheitsüberlegungen zu widersprechen. Besonders im Internet sollte ein Rechner nur die Dienste und Informationen bereitstellen, die unbedingt notwendig sind. Gerade das `ident`-Protokoll liefert aber Informationen, die man oft anderen nicht zugänglich machen möchte (z. B. Namen von Benutzerkennungen).

Wenn aber ein Benutzer auf Ihrem System in andere Rechner einbricht oder anderweitig deren Betrieb stört, kann der andere Systemadministrator diesem Benutzer gezielt Verbindungen verweigern. Ohne `ident` würde möglicherweise dem gesamten System der Verbindungsaufbau verweigert, so dass auch andere Benutzer betroffen sind. Auch für Sie ist es einfacher, Angriffe zu beurteilen und abzuwehren, wenn auf dem entfernten Rechner der entsprechende Dämon läuft.

Sie können mit Kommandozeilenoptionen einschränken, welche Daten Ihr Rechner auf Anfrage ausgeben soll. Damit lässt sich die Weitergabe von Benutzerkennungen oder anderen Daten verhindern. In Tabelle 15.2 finden Sie eine Auswahl an Optionen für `in.identd`. Alle Optionen sind in der Manpage zu `in.identd(8)` dokumentiert.

Option	Bedeutung
`-o`	Keine Informationen zum Betriebssystem liefern
`-e`	Bei Fehlern »UNKNOWN-ERROR« melden
`-n`	Numerische User-ID statt Benutzername melden
`-N`	»HIDDEN-USER«, wenn `~/.noident` existiert

Tabelle 15.2 Einige `identd`-Optionen

Die Option `-N` ermöglicht es jedem Benutzer, selbst zu entscheiden, ob Informationen zu seiner Kennung weitergegeben werden sollen. Die Meinungen hierzu sind geteilt. Es gibt `identd`-Programme, die keine Klartextmeldungen, sondern einen Hash-Wert aus Uhrzeit und Benutzerkennung liefern. Das hat den Vorteil, dass keine relevanten Daten weitergegeben werden, der Systemadministrator aber auf Anfrage diese Daten aus den Log-Dateien beschaffen kann. Damit kann dieses Verfahren aber nicht dazu verwendet werden, einzelnen Benutzern eines Rechners den Zugang zu bestimmten Diensten zu verwehren; dazu ist das Protokoll aber auch nicht gedacht. Die Daten sind in jedem Fall nur für den Systemadministrator des Systems, auf dem der `identd`-Prozess läuft, benutzbar.

Der Dämon sollte mit der Option `wait` vom `inetd` gestartet werden. Mehr zu den verschiedenen Modi und Aufrufparametern finden Sie in der Manpage zu `identd(8)`.

15.5 Aufnahme neuer Services

Auf praktisch jedem Unix-System sind die Standarddienste wie `telnet` oder `ftp` installiert. Es werden jedoch ständig neue Protokolle entwickelt, die einzelne Funktionen verbessern oder erst ermöglichen. Für die Arbeit Ihrer Benutzer oder die Systemsicherheit kann es daher sinnvoll sein, wenn Sie auch neuere Dienste installieren, bevor diese Bestandteil Ihrer Distribution geworden sind.

In der Regel werden diese Dienste durch einen neuen Dämon bereitgestellt. Nach der Übersetzung und Installation muss nun der entsprechende Server konfiguriert werden. Bei einigen Dämonen ist es möglich, diesen von `inetd` starten zu lassen oder ihn direkt zu starten. Beide Methoden haben Vor- und Nachteile, die man in jedem Einzelfall abwägen muss:

- Ein zusätzlich gestarteter Dämon belegt einen zusätzlichen (sonst unbenutzten) Prozess-Slot. Wenn schon sehr viele Prozesse auf dem Rechner laufen, kann es dazu kommen, dass keine weiteren Prozesse mehr gestartet werden können.
- Der Start eines Dämons durch `inetd` erfordert das Lesen der Konfigurationsdateien bei jedem Request. Dauert die Initialisierung lange oder benötigt sie viel CPU-Zeit, so ist es sinnvoll, den Dämon direkt zu starten. Ein Beispiel hierfür ist z. B. der `httpd`-Dämon.

Eine neue Anwendung mit einem neuen Protokoll erfordert zur Aktivierung folgende Tätigkeiten:

- Der Name des Services und die entsprechende Nummer müssen in der Datei `/etc/services` eingetragen werden. Betreuen Sie viele Rechner, bedeutet dies eine Menge Arbeit. Alternativ können Sie die Datei `/etc/services` aber auch mittels NIS (siehe Kapitel 23, »Network Information Service«) verteilen.
- Die entsprechenden Dämonen und Programme müssen übersetzt und installiert werden.
- Die neuen Dämonen müssen entweder beim Systemstart (z. B. in dem Skript `/etc/rc.local` oder den entsprechenden `init`-Skripten) aufgerufen oder in die Datei `/etc/inetd.conf` eingetragen werden. Da diese Datei auf unterschiedlichen Rechnern unterschiedliche Formate hat, kann sie leider nicht mittels NIS (Network Information Service) verteilt werden. Zu diesem Zweck könnte man aber ein Paket wie GNU `cfengine` einsetzen.

15.6 Remote Procedure Call

Eine andere verbreitete Kommunikationsmöglichkeit sind *Remote Procedure Calls* (RPC). Dieses Verfahren wurde unter anderem von Sun entwickelt. Die Spezifikation ist in [rfc1057] veröffentlicht. Auf der Basis von RPC wurden Dienste wie z. B. NFS (siehe Kapitel 20, »Network File System (NFS)«) und NIS (siehe Kapitel 23, »Network Information Service«) entwickelt.

Jeder Version eines RPC-Protokolls (z. B. `mount`, `ypbind`), wird eine Versionsnummer zugeordnet. Bei inkompatiblen Änderungen am Protokoll wird die Versionsnummer erhöht. Jeder Client gibt beim Verbindungsaufbau an, welche Version eines Protokolls verwendet werden soll. Viele Server unterstützen mehrere Versionen und verwenden dann das passende Protokoll. Damit ist es möglich, zunächst auf dem Server und dann auf den Clients in einem Netz eine neue Version zu installieren, ohne den Betrieb des Netzes zu stören.

Die Verwaltung der Dienste übernimmt, ähnlich wie `inetd`, der *Portmapper*. Unter Linux heißt das Programm `rpc.portmap`, es ist im Verzeichnis `/usr/sbin` zu finden. Jedes RPC-Programm muss sich beim Start beim Portmapper registrieren, der dem Programm dann einen freien Port zuweist. Damit können praktisch beliebig viele RPC-Dienste laufen, ohne dass deren Port-Nummern bei der IANA registriert sein müssen. Wenn der Portmapper neu gestartet wird, so müssen alle RPC-Server ebenfalls neu gestartet werden, damit sich diese wieder beim Portmapper registrieren können.

Clients, die einen RPC-Dienst nutzen möchten, müssen zunächst den Portmapper nach der verwendeten Port-Nummer fragen und können erst danach mit dem Server direkt Kontakt aufnehmen. Damit sind RPC-Dienste nicht für den Transport über eine Firewall geeignet, da die Port-Nummern nicht im voraus bekannt sind.

In der Datei `/etc/rpc` werden die RPC-Programmnummern einzelnen Anwendungen zugeordnet. Mit dem Programm `rpcinfo` können Sie feststellen, welche RPC-Services ein Rechner anbietet (`rpcinfo -p` *`Rechner`*, siehe Listing 15.8). Wird diese Funktion ohne einen Rechnernamen aufgerufen, so werden die Informationen zum eigenen Rechner angezeigt.

```
(linux):~# rpcinfo -p
   program vers proto   port
   100000    2   tcp    111  portmapper
   100000    2   udp    111  portmapper
   100005    1   udp    668  mountd
   100005    1   tcp    670  mountd
   100003    2   udp   2049  nfs
   100003    2   tcp   2049  nfs
   100004    2   udp    698  ypserv
```

```
100004    1   udp    698  ypserv
100004    2   tcp    701  ypserv
100007    2   udp    709  ypbind
100007    2   tcp    711  ypbind
```

Listing 15.8 RPC-Services eines Rechners

Mit der Option `-u` (für UDP) oder `-t` (für TCP) können Sie gezielt feststellen, ob ein Rechner einen bestimmten Service (mit einer speziellen Version) anbietet (siehe Listing 15.9). Dabei wird die Prozedur 0 des entsprechenden Servers aufgerufen, um die Funktionalität sicherzustellen. Ist der Dienst nicht verfügbar oder nicht mehr funktionsfähig, dann wird eine entsprechende Fehlermeldung ausgegeben.

```
(linux):~# rpcinfo -u localhost 100004 2
program 100004 version 2 ready and waiting
(linux):~# rpcinfo -u localhost 100004
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

Listing 15.9 Das Programm `rpcinfo` mit der Option `-u`

Wenn Sie einen Rechner im lokalen Netz suchen, der einen speziellen Dienst anbietet, verwenden Sie die Option `-b`. Dadurch wird ein Broadcast erzeugt, auf den der oder die Server für diesen Dienst antworten. Ein Beispiel dafür finden Sie in Listing 15.10).

```
(linux):~# rpcinfo -b 100004 2
127.0.0.1 localhost
192.168.30.202.26 jupiter.jochen.org
127.0.0.1 localhost
192.168.30.202.26 jupiter.jochen.org
127.0.0.1 localhost
192.168.30.202.26 jupiter.jochen.org
```

Listing 15.10 Das Programm `rpcinfo` mit der Option `-b`

15.7 Architekturunabhängiges Datenformat

An der Kommunikation im Netz können beinahe beliebige Rechnerarchitekturen teilnehmen. Bei der Entwicklung von Protokollen und Programmen wirft dies eine Reihe von Problemen auf. Dazu gehört z. B. die unterschiedliche Kodierung von Texten in verschiedenen Zeichensätzen (ASCII, ISO-Latin-1 oder EBCDIC). Ein weiteres Problem ist die unterschiedliche interne Darstellung von Integer-Zahlen. Diese müssen vor dem Transport (z. B. als Port-Nummer) vom Programm in die Netzwerk-Order gebracht werden. Dazu dient in der Program-

miersprache C die Funktion `htons(3)`. Ist auf der gerade verwendeten Architektur das interne Datenformat gleich der Netzwerk-Order, dann führt diese Funktion keine Umwandlung durch. In Abbildung 15.1 finden Sie ein Beispiel für die unterschiedliche Darstellung von IP-Adressen auf verschiedenen Rechnerarchitekturen.

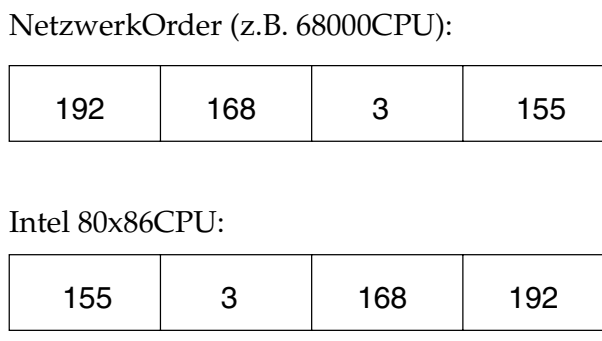


Abbildung 15.1 Interne Darstellung der IP-Adresse `192.168.30.202`

Auch beim Austausch von anderen Daten müssen eventuell Texte umkodiert oder Datenbankdateien umgesetzt werden. Auch für diese Zwecke existieren eine Reihe von Tools, wie `recode` für Textdateien oder spezielle Bibliotheken. Hinweise zur »External Data Representation« finden Sie in [rfc1014].

16 IP-Adressen und Rechnernamen

16.1 Rechnernamen als Hilfsmittel

Bisher haben wir fast ausschließlich numerische IP-Adressen verwendet. Diese werden von den betroffenen Systemen intern verwendet, sind aber für Anwender schlechter zu merken als Namen. Daher wird jedem Rechner zusätzlich zu seiner numerischen IP-Adresse auch ein Name zugeordnet, unter dem er ebenfalls anzusprechen ist. Es ist also eine maschinelle Umsetzung von Namen in IP-Adressen und umgekehrt notwendig.

Der Host-Name eines Unix-Systems wird beim Systemstart mit dem Kommando `hostname` gesetzt. Im Internet gibt es Tausende von Rechnern, die alle eindeutig adressierbar sein müssen. Dazu müssen IP-Adressen und Host-Namen eindeutig sein, es muss aber keine 1-1-Beziehung zwischen diesen herrschen.

Wird nur ein einzelnes Wort als Name verwendet, wie es in der UUCP-Map üblich ist, so ist es schwierig, eindeutige Namen zu finden. Es gibt einfach zu wenige Wörter, die man verwenden möchte, insbesondere sind die Gallier aus Asterix oder die Figuren aus dem Herrn der Ringe bereits vergeben. Ob Computerleute zu ähnliche Bücher lesen?

Einen Ausweg aus diesem Dilemma bietet die Verwendung von Domainnamen. Eine Domain ist z. B. das organisatorisch zusammengehörende Netz einer Firma oder Universität. Der Domainname wird unter Linux mit `dnsdomainname` festgelegt.

Der Namensraum wird durch die Verwendung von Top-Level-Domains (wie `.de`, `.com` oder `.edu`) und Domains (z. B. `jochen.org` oder `example.org`) unterteilt. Die Domainnamen `example.org` und `example.com` werden nicht vergeben und können in Beispielen verwendet werden.

Eine Domain lässt sich weiter in Sub-Domains unterteilen. Die Organisation `example.org` wird hier weiter unterteilt in die Sub-Domains `verkauf.example.org` und `lager.example.org`. Damit muss ein Host-Name nur in der eigenen Domain eindeutig sein, was in der Praxis relativ einfach sicherzustellen ist. Normalerweise vergibt oder verwaltet der Netzwerkverwalter, der auch die IP-Adressen vergibt, die entsprechenden Namen.

Ein Rechnername soll den Benutzern die Arbeit erleichtern. Er sollte daher leicht zu merken und sinnvoll sein. Einige Vorschläge, die bei der Namensvergabe beachtet werden sollten, finden Sie im [rfc1178], eine Zusammenfassung daraus folgt hier. Diese Hinweise spiegeln langjährige Erfahrungen im Internet mit der Namensvergabe wider, werden aber in der Praxis oft genug ignoriert. Verwenden Sie

- keine Rechnertypen oder Betriebssystemnamen, denn diese können sich recht schnell ändern.
- keine »wichtigen« Begriffe, die in wenigen Monaten schon unwichtig oder beleidigend sein könnten.
- keine Tätigkeiten oder Projekte, da sich diese Zuordnungen sehr schnell ändern.
- keine Namen von Personen, da dies für Verwirrung sorgt. Ist nun die Person oder der Rechner gemeint? Außerdem geschieht es in der Praxis häufiger, dass ein Rechner an einen anderen Arbeitsplatz gestellt wird.
- keine langen Namen, die nur schwer zu merken und mühsam zu tippen sind. Oder möchten Sie dem Benutzer *mailto:hein@quetzalcoatl.in.tu-clausthal.de* eine E-Mail schreiben?
- keine Domainnamen oder sehr ähnliche Namen, das könnte unbedarfte Benutzer verwirren.
- keine beleidigenden Namen.
- keine Ziffern am Anfang (oder nur Hexadezimalzeichen).
- keine Sonderzeichen wie Unterstriche (_), Umlaute o.ä.
- für die Namen grundsätzlich Kleinbuchstaben. Groß-/Kleinschreibung wird in der Regel nicht unterschieden, einige Programme tun es dennoch.
- keine zufällig zusammengewürfelten Namen, die praktisch nicht merkbar sind.
- nur Host-Namen, die nicht länger als 63 Zeichen sind. Das ist die Länge, die laut [rfc1123] jeder Internet-Host unterstützen muss.

Was soll man als Host-Namen wählen? Auch hier findet man im [rfc1178] einige Vorschläge, die aber im täglichen Leben nur selten beachtet werden. Diese Ideen sind oft dazu geeignet, die oben genannten Probleme zu umgehen. Hier die wichtigsten Tipps:

- Verwenden Sie seltene Wörter,
- thematisch zusammenhängende Namen: Farben, mystische Figuren o.ä.
- einfach zu schreibende Namen.

Beachten Sie bei der Wahl der Namen, dass diese vermutlich noch in einigen Jahren verwendet werden. Die Änderung eines Rechnernamens ist relativ aufwändig, wenn bereits einiges an Konfigurationsarbeit geleistet wurde. Wählen Sie daher den Namen Ihres Rechners mit Bedacht.

Der vollständige Name (Fully Qualified Domain Name, FQDN) eines (beliebigen) Rechners besteht im Internet aus mehreren, durch Punkte separierten Teilen. Dies sind z. B. bei dem Rechnernamen `apfel.verkauf.example.org` folgende Teile:

- `apfel` ist der Name des Rechners.
- `verkauf.example.org` ist die Domain des Rechners.
- `.org` ist die Top-Level-Domain (TLD) für »Organisationen«. Weitere TLDs sind z. B. `.com` für kommerzielle Firmen, `.edu` für Universitäten oder `.mil` für das Militär. Für deutsche Internetteilnehmer wird in der Regel die Top-Level-Domain `.de` verwendet.
- `verkauf.example.org` ist eine Sub-Domain der Domain `example.org`. Es ist möglich, weitere Sub-Domains einzurichten, um den Namensraum zu erweitern.

Nach langer Diskussion wurden einige neue Top-Level-Domains wie zum Beispiel `.info` und `.biz` eingerichtet. Ob das den immer intensiver werdenden Streit um Domainnamen, der häufig auch vor Gericht ausgetragen wird, in Zukunft entschärfen kann, ist fraglich. Die zweite aktuelle Entwicklung ist ein Standard für Host-Namen, die auch Sonderzeichen wie Umlaute enthalten können. Hier wird noch über die Technik diskutiert.

Durch die Aufteilung in Domains muss ein Rechnername nur in der eigenen (Sub-)Domain eindeutig sein und kann trotzdem im gesamten Internet adressiert werden. Eine Domain erhalten Sie entweder von Ihrem Provider oder durch das Network Information Center (NIC, *http://www.nic.de*). Den Rechnernamen vergeben Sie in Absprache mit Ihrem Netzbetreuer (dem Verantwortlichen für den Nameserver) oder Ihrem Provider.

16.2 Zuordnung von Namen und IP-Adressen in der Datei /etc/hosts

Normale Benutzer werden in der Regel die Rechner mit ihrem Host-Namen ansprechen. Die Rechner untereinander kommunizieren aber mit Hilfe ihrer IP-Adressen. Daher muss es ein (maschinelles) Verfahren geben, mit dem Namen in IP-Adressen und umgekehrt umgewandelt werden können.

Im einfachsten Fall werden die IP-Adressen und die zugeordneten Namen in der Datei `/etc/hosts` (siehe Listing 16.1) aufgeführt. In der ersten Spalte finden Sie die IP-Adresse des Rechners, danach seinen Namen (FQDN, Fully Qualified Domain Name), eventuell gefolgt von Aliasnamen.

Die Verwendung von Aliasnamen ermöglicht es, einen Rechner unter einem Namen zu adressieren, der seine derzeitige Verwendung beschreibt. So können viele `ftp`-Server unter dem Aliasnamen `ftp.`*`Domain`* erreicht werden. Weitere verbreitete Aliasnamen sind `www`, `gopher` oder `news`. Übernimmt nun ein anderer Rechner diese Funktion, so wird nur dieser Aliasname verändert, nicht aber der

eigentliche Rechnername. Die Änderung eines Aliasnamens erfordert nur wenig Aufwand, die Änderung eines Rechnernamens kann dagegen sehr aufwändig sein.

```
# IP-Adresse     Rechnername          Aliasnamen
127.0.0.1        localhost
192.168.30.202    jupiter.jochen.org    jupiter
```

Listing 16.1 Inhalt der Datei `/etc/hosts`

In einem größeren Netz ist die Wartung vieler `hosts`-Dateien sehr aufwändig und führt leicht zu Inkonsistenzen, da auf verschiedenen Rechnern oft unterschiedliche und unvollständige Namenstabellen vorliegen. Die Zuordnung von Namen und Adressen sollte eindeutig sein, damit auf jedem Rechner dieselben Hosts unter denselben Namen ansprechbar sind. Hier ist eine zentrale Verwaltung sinnvoll, um diese Konsistenz zu gewährleisten.

Die Eindeutigkeit von Host-Namen kann durch den *Domain Name Service* (DNS) sichergestellt werden, für den ein so genannter Nameserver eingerichtet wird. Alle Daten werden zentral auf diesem Rechner gepflegt und auf Anforderung über das Netz verteilt.

Eine andere Möglichkeit besteht in der Verwendung von NIS (Network Information Service, Kapitel 23, »Network Information Service«). Die Verwendung von NIS ist jedoch nur für Unix-Netze sinnvoll, da Windows-Clients in der Regel kein NIS beherrschen, jedoch DNS-Anfragen starten und die Antworten verwenden können.

16.3 Domain Name Service

Die Einrichtung eines Nameservers ist aus verschiedenen Gründen sinnvoll. Der administrative Aufwand beim Abgleich der Namenstabellen bei einem größeren Netz lässt sich vermindern; insbesondere bei der Verwendung von DOS- und Windows-Clients ist eine zentrale Administration von `hosts`-Dateien nicht möglich. Bei Dial-up-Verbindungen kann es sinnvoll sein, einen Cache-Server einzurichten. Dadurch werden wiederholte Anfragen an externe Nameserver vermieden. Mehr Informationen zum DNS finden Sie in [rfc1033], [rfc1034] und [rfc1035].

Ein Nameserver ist für einen bestimmten Bereich (eine Zone) zuständig (authoritative). Bei Anfragen, die der Nameserver nicht beantworten kann, wird der übergeordnete Server gefragt, der seinerseits die Anfrage weiterleiten kann. Da auch der übergeordnete Server nicht alle Daten der untergeordneten Server kennt, kann er die Anfrage auch nach unten an den zuständigen Server weiterleiten. Daher ist es beim Anschluss an das Internet wichtig, die Existenz eines eigenen Nameservers, z. B. beim DE-NIC, bekanntzugeben. Dies geschieht beim

Antrag auf eine eigene Domain entweder durch den Provider oder das zuständige Network Information Center.

Aus Gründen der Ausfallsicherheit sollte es für jede Zone mindestens zwei Nameserver geben, einen primären (primary) und mindestens einen Backup-Server (secondary). Auf dem Primary-Server werden die Datenbanken gepflegt, in denen die Informationen über Rechner und deren Adressen gespeichert sind. Der Backup-Server holt in regelmäßigen Abständen die Daten und bietet die Funktion des Nameservers für die Clients. Zu diesem Zweck speichert der Secondary-Server die Daten in einer Datei, die dann als Basis zur Beantwortung der DNS-Anfragen dient.

In vielen Fällen wird der Provider entweder den Nameserver für Sie betreiben oder zumindest einen Secondary-Server bereitstellen. Oft betreiben zwei oder drei Kunden eines Providers wechselseitig einen Secondary-Server.

Sorgen Sie in Ihrem Netz für das Vorhandensein eines Backup-Servers für den DNS. Ist dieser Dienst ausgefallen, erscheint das vielen Benutzern wie ein Komplettausfall des Netzes, daher ist der DNS ein sehr zentraler Dienst in einem TCP/IP-Netz. Aus diesem Grund sind im Protokoll Secondary-Server vorgesehen, die ihre Daten automatisch vom primären Server übertragen. Auf den Clients tragen Sie einfach mehrere Nameserver in die Konfiguration ein.

16.4 Nutzung eines Nameservers mit Linux

In diesem Abschnitt geht es um die Nutzung eines bereits bestehenden Nameservers als Client. Die Konfiguration eines Nameservers ist ausführlich in Kapitel 22, »Konfiguration und Betrieb eines Nameservers« erläutert, dort finden sich auch weitere Hinweise zum Betrieb desselben.

Die Anbindung an einen Nameserver erfolgt in der Datei `/etc/nsswitch.conf`. Diese Datei hat dasselbe Format wie unter Solaris. Listing 16.2 zeigt eine Anwendung.

Der Name `nsswitch` steht für »Name Service Switch«. Die Datei `nsswitch.conf` dient nicht nur zu Konfiguration des DNS-Clients, sondern ermöglicht viele andere Konfigurationen. So können Sie festlegen, in welcher Reihenfolge die verschiedenen Datenquellen ausgewertet werden bzw. welche überhaupt benutzt werden können.

```
passwd:     files nisplus nis
shadow:     files nisplus nis
group:      files nisplus nis
hosts:      files nisplus nis dns
```

Listing 16.2 Die Datei `/etc/nsswitch.conf`

In der ersten Spalte finden Sie den Namen der (lokalen) Konfigurationsdatei, gefolgt von einem Doppelpunkt (`:`). In den folgenden Spalten wird festgelegt, welche Dienste zu befragen sind, wenn aus der entsprechenden Konfigurationsdatei gelesen werden soll. Eine Übersicht über die möglichen Einträge finden Sie in Tabelle 16.1.

Schlüsselwort	Bedeutung
`files`	Die entsprechende lokale Datei lesen
`nisplus`	Die Daten von einem NIS+-Server (NIS Ver. 3) holen
`nis+`	Wie der Eintrag `nisplus`
`nis`	Die Daten von einem NIS-Server (NIS Ver. 2) holen
`yp`	Wie der Eintrag `nis`
`dns`	Den Nameserver befragen
`db`	Aus einer lokalen Datenbank lesen
`ldap`	Einen LDAP-Server befragen
`[NOTFOUND=return]`	Suche beenden, wenn bisher nicht gefunden

Tabelle 16.1 Mögliche Einträge in der Datei `/etc/nsswitch.conf`

Der letzte Eintrag in Tabelle 16.1 ist nur ein Beispiel für eine mögliche Reaktion. Mögliche Status, die wie in der letzten Zeile der Tabelle zur Abfrage verwendet werden können, sind `success`, `notfound`, `unavail` und `tryagain`. Die Abfrage kann mit einem Ausrufezeichen (`!`) negiert werden, außerdem sind mehrere Abfragen möglich. Als Aktionen nach dem Gleichheitszeichen können `return` oder `continue` angegeben werden. Mehr zu diesen Funktionen und den Default-Einträgen finden Sie wieder in der Info-Dokumentation der GNU-`libc`.

Wenn Sie das Modul `libnss-ldap` installiert haben, dann können Sie auch einen LDAP-Server befragen. Genauso lässt sich eine MySQL-Datenbank verwenden. In gewisser Weise ähneln diese Module dem bei PAM angewendeten Verfahren – einfach ein Modul installieren und eventuell einrichten, aber keine Programmänderungen.

Im nächsten Schritt muss in der Datei `/etc/resolv.conf` (siehe Listing 16.3) der zu verwendende Nameserver eingetragen werden. Der Name der Datei ist von der Resolver-Bibliothek (`libresolv`) abgeleitet, die für die Namensauflösung zuständig ist. Auf einigen Unix-Systemen (wie unter Linux mit `libc5`) ist diese Bibliothek integraler Bestandteil der `libc`, auf anderen Systemen und mit der GNU-`libc` Version 2 muss explizit mit der Option `-lresolv` die Resolver-Bibliothek hinzugelinkt werden.

In der Datei `/etc/resolv.conf` sind bis zu drei `nameserver`-Einträge erlaubt. Die Server werden in der angegebenen Reihenfolge angesprochen, so dass der lokale (schnellste oder verlässlichste) Server an erster Stelle stehen sollte. Liefert

der erste keine Antwort, so wird auf die Antwort eines anderen Servers gewartet. Damit ist für den Client der Ausfall eines DNS-Servers praktisch nicht mehr spürbar, abgesehen von einem relativ kurzen Timeout.

```
nameserver 192.168.3.150
nameserver 192.168.30.254
domain verkauf.example.org
search verkauf.example.org example.org
# sortlist sort-list
# options option-list
```

Listing 16.3 Die Datei `/etc/resolv.conf`

Ist in der `resolv.conf`-Datei kein Nameserver eingetragen, so wird der eigene Rechner als Nameserver benutzt. Hier sollte die IP-Adresse anstelle des Host-Namens angegeben werden, um Rekursionen (Nameserver- oder NIS-Anfragen zur Auflösung des Namens des Nameservers ...) oder Dead-Locks zu vermeiden. Der Nameserver sollte nach Möglichkeit die IP-Adresse nicht wechseln, da damit ein größerer Konfigurationsaufwand bei den Clients verbunden ist, wenn kein DHCP eingesetzt wird.

Weiterhin kann in der Datei `resolv.conf` der Domain-Name angegeben werden. Andernfalls wird der Teil nach dem ersten Punkt (`.`) im Host-Namen verwendet. Im Listing 16.3 wird als Domain `verkauf.example.org` verwendet.

Die Search-Liste (Schlüsselwort `search`) gibt an, wie aus einem unvollständig angegebenen Host-Namen (ohne Domain) ein möglicherweise vorhandener Rechnername konstruiert werden kann. Standardmäßig wird nur der Domainname angehängt, nicht wie in älteren Versionen alle durch Punkte getrennten Teile der Domain. Ein unbedachter Eintrag in die Search-Liste kann zu einer Reihe von nicht beantwortbaren DNS-Anfragen an fremde Rechner führen, die nicht gern gesehen werden. Das folgende Beispiel soll den Ablauf verdeutlichen:

In der Beispieldatei `/etc/resolv.conf` (siehe Listing 16.3) ist die Search-Liste als `verkauf.example.org example.org` eingetragen. Ein Benutzer versucht nun, eine Verbindung zum Rechner `typhon` aufzubauen. Zum Auflösen des Namens in eine IP-Adresse (mit Hilfe eines Nameservers, vergleiche die Datei `/etc/host.conf`) wird nach dem Rechnernamen selbst gesucht. Wird er nicht im Nameserver gefunden, wird der String `verkauf.example.org` durch einen Punkt (`.`) an den Rechnernamen angehängt und erneut der Nameserver befragt. Bleibt auch diese Anfrage erfolglos, so wird der nächste Eintrag aus der Search-Liste (im obigen Beispiel `example.org`) versucht.

Ist auch die Top-Level-Domain (z. B. `.de`, `.edu` oder `.com`) Bestandteil der Search-Liste, so würde zu den oben erwähnten Anfragen noch eine weitere kommen, die dann z. B. für den Rechner `Somehost.Somewhere.org` an den Nameser-

ver der (möglicherweise vorhandenen) Domain `org.de` weitergeleitet würde. Derartige Domainnamen werden, genauso wie alle nur zweibuchstabigen Subdomain-Namen, nicht mehr vergeben.

Für einzelne Prozesse können Sie die Konfiguration für `search` und `option` durch die Umgebungsvariablen `LOCALDOMAIN` bzw. `RES_OPTIONS` überschreiben. Weitere Informationen zu den Schlüsselwörtern `sortlist` und `options` finden Sie in der Manpage zu `resolver(5)`.

16.5 Testen eines Nameservers

Um die Anbindung an einen Nameserver oder die Konfiguration desselben zu testen, stehen verschiedene Programme zur Verfügung. Einige der verbreiteten Tools werden im Folgenden vorgestellt.

Vor allem muss der Nameserver im Netz erreichbar sein. Dies können Sie durch einen kurzen Test mit dem Programm `ping` sicherstellen. Es ist nicht notwendig, dass sich der benutzte Nameserver im lokalen Subnetz befindet, daher müssen auch die notwendigen Routen gesetzt sein.

16.5.1 Interaktive DNS-Anfragen mit nslookup

Mit dem Programm `nslookup` können DNS-Anfragen in zwei Modi durchgeführt werden: interaktiv und nicht interaktiv. Im interaktiven Modus können verschiedene Anfragen nacheinander angegeben werden. Im nicht interaktiven Modus werden nur die gewünschten Informationen ausgegeben und das Programm beendet.

Zusätzliche Optionen können mit einem führenden Minuszeichen (-) vor dem Host-Namen oder der Adresse angegeben werden. Außerdem können Standardwerte in der Datei `~/.nslookuprc` abgelegt werden. Ein Beispiel für den nicht interaktiven Aufruf von `nslookup` finden Sie in Listing 16.4.

```
(linux):~$ nslookup ftp
Server:  janus.jochen.org
Address:  192.168.30.254

Name:    jupiter.jochen.org
Address:  jochen.org
Aliases:  ftp.jochen.org
```

Listing 16.4 `nslookup` *im nicht interaktiven Modus*

Der interaktive Modus wird verwendet, wenn entweder kein Parameter oder nach einem Minuszeichen (-) ein Nameserver (Name oder IP-Adresse) angegeben ist. Anschließend können Anfragen gestartet werden, indem ein Host-Name oder eine IP-Adresse eingegeben werden. Mit dem Befehl `set type=any` können alle Informationen über einen Rechner angefordert werden. Ein Beispiel dafür finden Sie im Listing 16.5. Weitere Informationen über `nslookup` und eine Liste aller möglichen Abfrageoptionen finden Sie in der Manpage zu `nslookup(1)`.

```
(linux):~$ nslookup
Default Server:  janus.jochen.org
Address:  192.168.30.254

> ftp
Server:  janus.jochen.org
Address:  192.168.30.254

Name:    jupiter.jochen.org
Address:  jochen.org
Aliases:  ftp.jochen.org

> set type=any
> ftp
Server:  janus.jochen.org
Address:  192.168.30.254

ftp.jochen.org canonical name = jupiter.jochen.org
jochen.org     nameserver = janus.jochen.org
jupiter.jochen.org      internet address = jochen.org
> exit
```

Listing 16.5 `nslookup` *im interaktiven Modus*

Mit dem Programm `nslookup` können Sie ebenfalls sehr einfach einen so genannten Reverse-Lookup ausführen. Darunter versteht man die Frage nach dem Namen, der zu einer bestimmten IP-Adresse gehört. In diesem Fall geben Sie einfach anstelle des Host-Namens die IP-Adresse an.

Das Programm `nslookup` kann verschiedene Nameserver befragen und die gesendeten und empfangenen Pakete analysieren. Da dieses Programm auch unter anderen Systemen, wie z. B. Windows NT, verfügbar ist, ist es zum Test eines Nameservers sehr verbreitet.

16.5.2 Nameserver-Anfragen mit host

Auch das Programm `host` ermöglicht eine Analyse der in einem Nameserver abgelegten Daten. Nützlich ist hier, dass die Daten im Format passend für den Eintrag in die Nameserver-Datenbank ausgegeben werden können (Option `-v`). Mit

der Option `-t` können die gewünschten Einträge selektiert werden (z. B. `ns`, `cname`, `soa`, `hinfo`, `mx` und `any` oder `*` für alle Einträge). Die Bedeutung der Einträge ist in Kapitel 22, »Konfiguration und Betrieb eines Nameservers« genauer erläutert.

Die Option `-a` ist eine Kurzform für `-t any`. Eine genauere Beschreibung aller Einträge finden Sie in der Manpage `host(1)` und im *Bind Operators Guide*, den Sie gemeinsam mit den Quellen zu `bind` erhalten. Das Listing 16.6 zeigt einige Beispiele für die Verwendung von `host`.

```
(linux):~$ host jupiter
jupiter.jochen.org has address jochen.org
(linux):~$ host -v jupiter
Trying domain "jochen.org"
rcode = 0 (Success), ancount=1
jupiter.jochen.org      86400 IN        A      jochen.org
For authoritative answers, see:
jochen.org     86400 IN        NS     janus.jochen.org
Additional information:
janus.jochen.org      86400 IN        A      192.168.30.254
(linux):$ host -a jupiter
Trying domain "jochen.org"
rcode = 0 (Success), ancount=3
jupiter.jochen.org  86400 IN        A      jochen.org
jupiter.jochen.org  86400 IN        HINFO  IBM-PC/AT       UNIX-PC
jupiter.jochen.org  86400 IN        TXT "Test mit Text"
For authoritative answers, see:
jochen.org 86400 IN        NS     janus.jochen.org
```

Listing 16.6 Nameserver-Anfragen mit `host`

16.5.3 Nameserver-Anfragen mit dig

Ein weiteres Tool zur Analyse eines Nameservers ist `dig` (Domain Information Groper). Auch dieses Programm kann interaktiv und im Batch-Betrieb verwendet werden. Viele TCP/IP-Pakete enthalten nur dieses Programm und nicht `nslookup` oder `host`.

Die Optionen und Funktionen von `dig` sind ausführlich in dessen Manpage beschrieben. Beachten Sie, dass Sie bei einem »Reverse-Lookup« zusätzlich zur IP-Adresse die Option `-x` angeben müssen.

```
(linux):~$ dig jochen.org mx
...
;; QUESTION SECTION:
;jochen.org. IN MX
```

```
;; ANSWER SECTION:
jochen.org. 86400 IN MX 10 inbound.jochen.org.criticalpath.net.
...
(linux):~$ dig @uranus.lan-ks.de jochen.org mx
...
;; QUESTION SECTION:
;jochen.org. IN MX

;; ANSWER SECTION:
jochen.org. 86400 IN MX 90 mail.dinoex.sub.de.
jochen.org. 86400 IN MX 10 uranus.lan-ks.de.
jochen.org. 86400 IN MX 80 mail.dinoex.org.
...
```

Listing 16.7 Die Verwendung von `dig`

In der ersten Anfrage sucht `dig` nach einem MX-Record für die Domain `jochen.org` und erhält eine Antwort. In den mit Auslassungszeichen markierten Bereichen gibt `dig` weitere Informationen zur Anfrage, zum befragten Nameserver und zur benötigten Zeit aus. Aus Platzgründen sind diese Teile hier nicht dargestellt.

Die zweite Anfrage spricht mit dem Parameter `@uranus.lan-ks.de` einen anderen Name-Server an – dieser wird in den nächsten Tagen auch der offizielle Nameserver für die Domain sein. Auf diesem Weg kann man auch einen Nameserver, der zunächst nur testweise betrieben wird, in Ruhe prüfen.

16.6 Die Verlässlichkeit von Nameservern

Die Internetprotokolle wurden bisher mit Blick auf die Implementierbarkeit und Funktionalität entwickelt und implementiert. Dies war so lange kein Problem, wie nur wenige und verlässliche Teilnehmer im Internet arbeiteten. In den letzten Jahren hat sich die Anzahl der Internetrechner und -benutzer deutlich erhöht, so dass auch die Zahl der »bösen Buben« darunter zugenommen hat.

Aus diesem Grund sehen sich heute Internetrechner, insbesondere die mit sehr bekannten Namen, vielfältigen Angriffen ausgesetzt. Je nach Protokoll, eingesetzten Programmen und der sonstigen Umgebung gibt es verschiedene Angriffstypen und auch Abwehrmöglichkeiten. Viele Informationen zu diesem Themenkomplex finden Sie in den CERT-Advisories zum Beipiel unter *http://www.cert.org/*.

Ein verbreiteter Angriffstyp ist das Verfälschen von Nameserver-Antworten. Viele Programme (z. B. `rlogin`) lassen Benutzer oder Zugriffe in Abhängigkeit vom Rechnernamen zu. Bei einer eingehenden Verbindung steht aber nur die IP-Adresse des Absenders zur Verfügung, wenn diese nicht bereits durch so ge-

nanntes IP-Spoofing verfälscht ist. Für die Authentifizierung wird ein Reverse-Lookup durchgeführt, der vom zuständigen Nameserver bearbeitet wird. Ein Tool für diese Angriffe ist `dnspoof` aus dem `dsniff`-Paket.

Da der zuständige Nameserver unter der Kontrolle des Angreifers stehen kann, sind hier beliebige Verfälschungen möglich. Abwehrmaßnahmen sind verbesserte Programme, die die DNS-Antworten noch einmal gegenprüfen, neue Protokolle (z. B. `ssh` statt `rsh`) und der Verzicht auf angreifbare Dienste.

In Zukunft wird man hoffentlich mehr Nameserver finden, die ihre Daten mit kryptographischen Verfahren sichern. Erst dann sind Angriff mittels DNS-Spoofing nur noch sehr schwer möglich.

Der Einsatz eines DNS ist auch im Intranet sehr nützlich, da es sehr aufwändig ist, auf allen Rechnern die Host-Tabellen zu pflegen. Der wesentliche Vorteil des DNS ist hier, dass es eine definitive Instanz gibt, deren Daten überall verfügbar sind. Alternativ kann man in einem Unix-Netz auch NIS zum Verteilen der Host-Tabellen einsetzen, man ist dabei aber auf Unix-Systeme beschränkt.

17 Applikationen im Netz

17.1 Anwendungen für TCP/IP

Auf der Basis von TCP/IP wurden eine Reihe von Anwendungen entwickelt, die bei der Arbeit im Netz hilfreich sind. Einige dieser Anwendungen werden im Folgenden vorgestellt. Dabei kann es sich nur um eine kleine Auswahl handeln, die keinerlei Anspruch auf Vollständigkeit erhebt. Es werden auch ständig neue Protokolle entwickelt, implementiert und verbessert, so dass eine definitive Referenz praktisch unmöglich ist.

Viele dieser Applikationen wurden unter BSD entwickelt und dann nach Linux portiert. Zahlreiche Programme findet man daher auf jedem BSD-System und in neuerer Zeit auch auf diversen System-V-Rechnern. Einige der hier vorgestellten Programme sind jedoch in den Standardinstallationen von Linux- und Unix-Systemen nicht enthalten, so dass der Systemadministrator diese bei Bedarf selbst kompilieren und installieren muss. Hinweise zur Installation von Programmen und Netzwerkdiensten finden Sie in Kapitel 9, »Unix-Tools«, und Abschnitt 15.5, »Aufnahme neuer Services«.

Zurzeit werden viele neue Applikationen entwickelt, insbesondere mit einem Web-Frontend oder in das WWW integriert. Damit ist es in vielen Fällen nicht mehr notwendig, auf dem Client spezielle Programme zu installieren, es wird einfach der in der Regel bereits vorhandene Browser verwendet. Die »Installation« beschränkt sich in solchen Fällen auf die Eingabe einer URL. In Zukunft werden Applikationen sowohl für den Ablauf im Browser mittels Java und JavaScript als auch für den Server (als CGI-Programme, in PHP oder Zope) entwickelt. Für die Systemverwaltung werden allerdings die hier vorgestellten Programme auch weiterhin notwendig und nützlich sein.

17.2 Web-Browser einmal anders

Die grafischen Web-Browser wie `netscape`, `mozilla`, `galeon` oder `konqueror` werden sehr häufig zum Browsen verwendet. Ein Schattendasein fristen textbasierte Browser, nicht immer zu Recht. Viele Webseiten sind mit Grafiken überladen, haben demzufolge lange Ladezeiten und sind ohne die Grafiken kaum zu benutzen. Leider machen sich viele Webmaster nicht die Mühe, die Seite mal mit einem Textbrowser zu betrachten, sonst wäre der Anreiz, so genannte `ALT`-Tags einzufügen, sicher größer. Auf diese Art schließt man nicht nur textbasierte Browser, sondern häufig auch Anwender aus, die zum Beispiel auf eine Braille-Zeile angewiesen sind.

17.2.1 Textbrowser

Viele Webseiten sind mit Textbrowsern wie `lynx`, links oder `w3m` einfach und schnell zu bedienen. Gerade bei Seiten, die ohne großen grafischen Schnickschnack auskommen, kann man sich auf das Wesentliche konzentrieren – den Inhalt. Besonders angenehm empfinde ich, dass ich die Finger nicht von der Tastatur nehmen und nach der Maus suchen muss.

Die Browser haben alle eine Option `-dump`, mit der man eine Webseite oder HTML-Datei als Text abspeichern kann. Damit ist es möglich, nur die wesentlichen Informationen in eine Mail zu packen oder als Text bereitzustellen. Mit diesen Browsern lässt sich auch eine einfache Überwachung von Webservern einrichten: gezielt eine URL abfragen und das Ergebnis mit einer Datei vergleichen.

17.2.2 Web- und ftp-Downloads

Nicht nur das Ansehen von Webseiten ist interessant. Manschmal möchte man einen Mirror, also eine direkte Kopie eines Servers, erstellen, manchmal nur eine Seite mit allen dort enthaltenen Bildern oder Dateien kopieren. Zu diesem Zweck kann man `wget` einsetzen. Im einfachsten Fall ruft man `wget` mit einer URL als Parameter auf und findet dann die Datei im aktuellen Verzeichnis wieder.

`wget` bietet aber noch mehr. Mit der Option `-i` (`--input-file`) kann man eine Datei angeben, die die zu holenden URLs enthält. Die Datei kann im HTML-Format gespeichert sein, geben Sie dann die Option `-F` (`--force-html`) an. Wenn diese Datei relative Links enthält, dann lässt sich die Basis dazu mit der Option `-B` (`--base-url`) angeben.

Mit `-r` (`--recursive`) veranlassen Sie `wget`, Links zu folgen, bis die gewünschte Suchtiefe (Standardwert ist 5) erreicht ist. Den Wert für die Suchtiefe ändern Sie mit der Option `-l` (`--level`). Wenn Sie `wget` als Mirror-Programm einsetzen wollen, dann werden Sie vermutlich die Option `-m` (`--mirror`) wählen.

Wollten Sie schon immer mal wieder Ihre Bookmarks aufräumen? Im Laufe der Zeit sammeln sich jede Menge Links an, von denen schon nach kurzer Zeit viele nicht mehr aktuell sind. `wget` unterstützt Sie dabei, wenn Sie die Option `-spider` verwenden (Listing 17.1).

```
(linux):~$ wget --spider --force-html -i bookmarks.html
...
```

Listing 17.1 `wget` *zum Prüfen einer Bookmark-Datei*

`wget` kann mit SSL-Unterstützung kompiliert werden und natürlich Proxies verwenden. Alle weiteren Optionen finden Sie in der Manpage dokumentiert. `wget` erhalten Sie unter *http://www.wget.org*.

17.3 Angemeldete Benutzer im lokalen Netz

Oft besteht ein Netz aus einer Reihe von Rechnern. Sucht man nun einen bestimmten Benutzer, um ihn z. B. mittels `talk` anzusprechen, ist es schwierig und langsam, auf allen Rechnern ein `w` abzusetzen oder für alle möglichen Rechner das Programm `finger` zu starten.

Hat der Administrator auf den Rechnern des lokalen Netzes den `rwhod`-Dämon gestartet, so können Sie mit dem Befehl `rwho` eine Liste der aktiven, angemeldeten Benutzer erhalten (siehe Listing 17.2). Mit dem Befehl `ruptime` (siehe Listing 17.3) erhalten Sie eine Übersicht über die Rechner im lokalen Netz und deren Auslastung. Dabei werden nur die Rechner erfasst, die im gleichen Subnetz liegen, also zur Broadcast-Adresse passen. Es werden nur die Benutzer angezeigt, die nicht länger als eine Stunde »idle« sind. Wenn Sie alle angemeldeten Benutzer sehen möchten, verwenden Sie `rwho -a`.

```
(linux):~$ rwho
hein     jupiter:tty2    Aug  1 13:43 :21
jochen   jupiter:tty3    Aug  1 13:25 :10
jochen   jupiter:tty4    Aug  1 13:22 :55
katrin   typhon:ttyp1    Aug  1 14:30 :47
katrin   typhon:ttyp2    Aug  1 14:30 :41
```

Listing 17.2 Im Netz angemeldete Benutzer

```
(linux):~$ ruptime
karotte     up    22:05,     2 users,  load 0,02, 0,03, 0,04
jupiter     up     0:58,     4 users,  load 0,00, 0,01, 0,00
typhon    down     3:45
```

Listing 17.3 Aktive Rechner im Netz

Oft wird der `rwhod`-Dämon nicht gestartet, da durch den regelmäßigen Austausch der angemeldeten Benutzer und die Auslastung eine recht große Netzlast entsteht. Das gilt besonders in dem Fall, wenn plattenlose Rechner eingesetzt werden. Nach jedem `rwhod`-Broadcast werden von allen Workstations die `rwhod`-Programme über das Netz in den Hauptspeicher geladen und dann die entsprechenden Einträge in die Dateien vorgenommen. Dies geschieht auch wieder über das Netz, so dass in relativ kurzer Zeit ein großes Datenvolumen entsteht.

Von Sun wurden als Alternative zu `rwho` die Programme `rusers` und `rstat` entwickelt. Diese Programme kommunizieren mit den entsprechenden Servern (`ruserd` und `rstatd`), die mit Hilfe von RPC implementiert wurden. Unter Linux heißen die Programme normalerweise `rpc.userd` und `rpc.rstatd`. Damit entsteht die Netzlast nur dann, wenn auch tatsächlich ein Benutzer an den Daten interessiert ist.

Oft ist man aus Datenschutz- oder Sicherheitsgründen daran interessiert, den Zugriff auf die Anmelde- und Arbeitszeiten der Benutzer zu unterbinden. In diesen Fällen wird man auf jeden Fall den `finger`-Dienst und `rwhod` nicht starten.

17.4 Warnungen an entfernte Rechner schicken mit rwall

Auf dem lokalen Rechner können Warnungen oder Nachrichten an alle Benutzer mit dem Befehl `wall` versendet werden. Dies geschieht z. B. beim Aufruf von `shutdown`. Wird ein Server heruntergefahren, so kann es sinnvoll sein, auch die Nutzer anderer Rechner zu informieren. Dies kann mit dem Befehl `rwall` *`Rechner`* erfolgen. Dazu muss jedoch auf dem Zielrechner der `rwalld`-Dämon (`rpc.rwalld`) gestartet sein. Unter Linux ist dies das Programm `/usr/sbin/rpc.rwalld`. Ob und auf welchen Rechnern dieser Service gestartet ist, können Sie mit dem Befehl `rpcinfo -b 100008 1` feststellen.

17.5 Übertragen von Dateien mit rdist

Mit dem Programm `rdist` lassen sich Verzeichnisbäume im Netz abgleichen. Damit ist es möglich, einen Master-Server mit Programmen zu installieren, die dann an weitere Rechner (lokale Fileserver) weitergegeben werden. Dabei werden nur die neuen oder geänderten Dateien übertragen und (eventuell) überflüssige Dateien gelöscht. Die Anmeldung auf dem entfernten Rechner erfolgt mit `rsh`. Es ist jedoch auch möglich, die `ssh` zu verwenden. In diesem Fall müssen Sie entweder das Programm `rsh` durch eine Kopie der `ssh` ersetzen oder `rdist` entsprechend kompilieren.

Das Programm `rdist` kann sowohl mit Parametern und Optionen als auch mit Dateien gesteuert werden. Insbesondere bei komplexeren Aufgaben und damit umfangreicheren Parametern bietet sich die Verwendung von `rdist`-Skripten an. Dabei wird eine Eingabedatei für `rdist` erstellt, in deren erste Zeile das Programm `rdist` als Interpreter eingetragen wird. Dort können auch weitere Parameter angegeben werden.[1] Eine `rdist`-Datei besteht aus Einträgen, die einem der drei folgenden Formate gehorchen:

`Variable` = *`Liste`*
: Eine Variable, die im weiteren Verlauf des Skripts verwendet wird, erhält einen Wert zugewiesen.

1. Auf älteren Betriebssystemen, wie z. B. Ultrix, führt dies möglicherweise zu Problemen.

Source-Dateien -> *Destination Kommandos*
: Die angegebenen *Source-Dateien* sollen auf die Rechner, deren Namen in *Destination* angegeben sind, übertragen werden. Für die Übertragung wird das angegebene *Kommando* verwendet. Die erlaubten Kommandos werden in diesem Abschnitt genauer vorgestellt. Es werden jedoch nur die Dateien übertragen, die auf dem entfernten Rechner älter oder nicht vorhanden sind.

Source-Dateien :: *Timestamp-Datei Kommandos*
: Bei diesem Format werden die zu übertragenden Dateien durch die *Timestamp-Datei* bestimmt.

Innerhalb von `rdist`-Skripten können eine Reihe von Kommandos aufgerufen werden. Damit können Dateien übertragen oder andere Programme gestartet werden.

- `install` kopiert die zu übertragenden Dateien auf den oder die Zielrechner. Dabei können weitere Optionen angegeben werden, die Sie in der Manpage dokumentiert finden.
- Mit dem Kommando `notify` wird `rdist` veranlasst, eine Liste der übertragenen Dateien und eventuellen Fehlermeldungen an den angegebenen Benutzer zu schicken. Enthält der Benutzername kein `@`-Symbol, dann wird der entsprechende Benutzer auf dem Zielrechner verwendet.
- `except` kopiert alle Dateien, die danach nicht aufgeführt wurden. Damit ist es einfach möglich, ein Verzeichnis bis auf wenige Dateien zu übertragen.
- `except_pat` arbeitet ähnlich wie `except`, es können jedoch Muster (Pattern) für Dateinamen angegeben werden. Die Expansion des Pattern erfolgt wie bei der `sh`.
- `special` kann dazu verwendet werden, externe Kommandos für jede übertragene Datei aufzurufen.
- Der Befehl `cmdspecial` wird erst nach dem Übertragen aller betroffenen Dateien einmalig ausgeführt.

In Listing 17.4 kopiert `rdist` die wichtigsten Konfigurationsdateien eines Benutzers auf einen anderen Rechner. Damit ist es relativ einfach möglich, auch auf Rechnern ohne gemeinsames Home-Verzeichnis eine einheitliche Umgebung zu konfigurieren. Dies ist nur ein einfaches Beispiel. Alle Optionen und Möglichkeiten von `rdist` werden ausführlich in der Manpage erläutert, dort finden Sie auch ein ausführlicheres Beispiel.

```
# Auf welche Rechner (und welche Benutzer) übertragen?
HOSTS = ( jupiter.jochen.org injhe@typhon )

# Welche Dateien sind zu übertragen?
FILES = ( ~/.emacs ~/.gnus ~/.vm ~/.project
          ~/.plan ~/.custom.el ~/.fvwmrc ~/.netrc
```

```
          ~/.rhosts  ~/.elm  ~/.tin  ~/.emacs
          ~/emacs-lisp ~/emacs.init ~/insert ~/work )

# Hier passiert's
$FILES -> $HOSTS
        install -R;
```

Listing 17.4 Beispiel für die Verwendung von `rdist`

17.6 Abgleich von Dateien über das Netz

Die Übertragung von Dateien mit `rdist` anstelle von `rcp` kann das Netz spürbar entlasten. Allerdings werden die zu übertragenden Dateien in jedem Fall vollständig kopiert. Gerade bei großen Dateien, in denen nur sehr wenige Änderungen durchgeführt wurden, kann das eine relativ große, unnötige Netzlast verursachen. Die bessere Alternative wäre es, wenn man nur die Änderungen über das Netz übertragen müsste.

Bei zentral verwalteten Programmen wird bei einer neuen Version oft auch ein Patch bereitgestellt, der genau die Änderungen zwischen den Versionen enthält. Dieses Verfahren hat aber den Nachteil, dass die alte und die neue Version auf einem Rechner vorhanden sein müssen – man gewinnt also nichts.

Mit dieser Problemstellung im Hintergrund wurde das Programm `rsync` entwickelt. Das Programm generiert auf beiden Seiten der Verbindung Prüfsummen über die Datei und kann anhand der Übereinstimmungen bzw. Abweichungen genau die geänderten Daten übertragen.

17.7 Gespräche zwischen Benutzern mit talk

Zwei angemeldete Benutzer können sich, auch über ein Netzwerk, mit dem Programm `talk` unterhalten. Dazu muss jeweils auf der Gegenstelle der entsprechende `talk`-Dämon installiert sein. Leider gibt es zwei unterschiedliche und inkompatible Protokolle (`talk` und `ntalk`), so dass eine Kommunikation oft nicht möglich ist.

Nach dem Verbindungsaufbau (siehe Listing 17.5) wird der Bildschirm geteilt. Im obersten Feld wird das angezeigt, was man selbst eingegeben hat. Darunter sind weitere Flächen dargestellt, in denen die Texte der `talk`-Partner angezeigt werden. Bei vier oder mehr Teilnehmern wird daher der Bildschirm unübersichtlich.

Beim Standard-`talk` ist die Kommunikation auf zwei Benutzer beschränkt. Leistungsfähiger und flexibler ist das Programm `ytalk`, das beide `talk`-Protokolle beherrscht und außerdem die Kommunikation mit mehreren Gesprächspartnern

ermöglicht. Dabei können Benutzer mit ytalk die Texte aller anderen ytalk-Benutzer im entsprechenden Bildschirmbereich sehen, aber einen Benutzer, der nur das Standard-Unix-talk benutzt, nur bei direkter Ansprache.

Eine talk-Verbindung wird mit ytalk *user@host* eröffnet. Der entsprechende Benutzer erhält eine Nachricht über eine Einladung zu talk. Gibt er den in der Meldung angegebenen Befehl ein (siehe Listing 17.5), dann wird die Verbindung hergestellt. Alle weiteren Eingaben werden an den Gesprächspartner übertragen, genauso, wie man dessen Eingaben angezeigt bekommt. Die Verbindung wird mit Strg+C beendet.

```
Message from Talk_Daemon@jupiter at 11:56 ...
talk: connection requested by jochen@jupiter.jochen.org
talk: respond with:  talk jochen@jupiter.jochen.org
```

Listing 17.5 Einladung zum Talk

Das Programm ytalk bietet im Vergleich zu Unix-talk eine Reihe von zusätzlichen Möglichkeiten. Diese Funktionen werden über ein Menü aufgerufen, das nach dem Druck auf die Taste Esc erscheint. Bei einer deutschen Tastaturbelegung sendet die Taste Alt ein »Escape« vor der zweiten Taste, so dass diese Funktionen auch direkt mit Alt+Taste aufgerufen werden können. Diese Einstellung der Tastatur lässt sich mit dem Befehl setmetamode konfigurieren.

- Mit der Taste a (add) kann ein neuer Teilnehmer aufgenommen werden.
- Die Taste d (delete) beendet die Verbindung mit einem Partner.
- Weitere Optionen können mit der Taste o verändert werden. Diese Optionen können Sie auch dauerhaft in der Datei ~/.ytalkrc konfigurieren.
- Nützlich ist der Aufruf einer Shell mit der Taste s, so dass Sie Befehle eingeben können, deren Ausgaben an die Gesprächspartner übertragen werden.
- Eine Übersicht über alle aktiven Benutzer erhalten Sie mit der Taste u (user).
- Den Text eines Partners können Sie mit der Taste w (write) in eine Datei kopieren, so dass Sie diesen Text später noch zur Verfügung haben.
- Die Taste q beendet ytalk.

In lokalen Netzen ist die Kommunikation mit talk schnell und bequem. Ist der Kommunikationspartner weiter entfernt, so kommen recht schnell große Wartezeiten zusammen, da jeder Tastendruck einzeln übertragen wird.

17.8 Internet Relay Chat (irc)

Bei der Kommunikation mit `ytalk` wird bei mehreren Partnern schnell der Platz auf dem Bildschirm knapp. Außerdem muss jeder der Gesprächspartner über das Programm `ytalk` verfügen, das aber unter vielen Unix-Varianten nicht standardmäßig vorhanden ist. Ein anderes Protokoll zur Kommunikation mit mehreren Partnern ist Internet Relay Chat (IRC). Im IRC findet man eine große Anzahl von Channels, in denen die Diskussion online stattfindet. Dabei geht es um die unterschiedlichsten Themen.

Das Rückgrat des Systems wird durch eine Reihe von IRC-Servern gebildet, die untereinander die eingegebenen Meldungen austauschen. Diese Rechner sind in der Regel über schnelle Leitungen verbunden, um Wartezeiten möglichst kurz zu halten. Die Server leiten die Meldungen an die angemeldeten Clients weiter, die dann die Interaktion mit den Benutzern übernehmen.

Will oder kann man auf seinem System keinen IRC-Client betreiben, so gibt es die Möglichkeit, über ein `telnet`-Gate auf IRC zuzugreifen. Ausführliche Informationen zu IRC finden Sie in der Dokumentation zum IRC-Client `ircII-*`. Weitere, komfortablere Clients sind `irchat` für Emacs und Zircon für X.

Die neueste Attraktion ist das Einrichten von Chat-Räumen auf seinem Web-Server. Der Vorteil ist, dass hierfür nur ein Web-Browser notwendig ist und kein anderer Client. Andererseits wird hierdurch der Benutzerkreis sehr eingeschränkt, da nicht mehr die weltweite Infrastruktur des IRC verwendet wird. Für viele Firmen ist jedoch gerade dies das schlagende Argument, da sie nur so die alleinige Kontrolle des Chat-Raums ausüben und dort beispielsweise Werbung hinterlegen können.

17.9 Die Versendung von Dateien mit sendfile

In einem Netz mit mehreren Benutzern ist es oft notwendig, Dateien zwischen Benutzern und Rechnern auszutauschen. Wenn nur eine Textdatei zu versenden ist, dann kann das oft bereits mit dem Standard-`mail`-Programm erfolgen: ***`mail Benutzer < Datei`***.

Sind in der Datei Umlaute enthalten oder handelt es sich um eine Binärdatei, dann muss diese vor der Übertragung zunächst in ein Format konvertiert werden, das den Transport als Mail übersteht. Dazu dienen Programme wie `uuencode` oder es kann eine MIME-Kodierung verwendet werden.

Dieses Verfahren ist jedoch recht umständlich, da möglicherweise das Mail-System zusätzlich die Länge der Nachrichten begrenzt. Mit dem Programm `sendfile` (Listing 17.6) können eine oder mehrere Dateien an einen Benutzer versendet werden. Die übertragenen Dateien können dann mit `receive` empfangen werden.

```
(linux):~$ sendfile Datei User@Host
```

Listing 17.6 Versenden einer Datei mit `sendfile`

Dieses Programm verwendet das SAFT-Protokoll (Simple Asynchronous File Transferprotocol), für das bereits ein Port offiziell reserviert wurde. Ein RFC für dieses Protokoll ist in Vorbereitung. Neben der Übertragung von Dateien ist eine Funktion zum Senden von Nachrichten implementiert (`sendmsg`). Damit können Sie kurze Texte auch über das Netz an einen angemeldeten Benutzer senden, ohne sich auf einem anderen Rechner anmelden zu müssen (lokal können Sie `write` verwenden).

`sendfile` kann nur verwendet werden, wenn auf dem Zielrechner der entsprechende Server gestartet ist, was zurzeit nur selten der Fall ist. Außerdem muss der Rechner direkt per TCP/IP erreichbar sein, das kann z. B. durch eine Firewall verhindert werden. Wenn Sie Dateien per Mail verschicken, so muss der Zielrechner nicht online sein, sofern für den Rechner ein entsprechender MX-Eintrag im DNS vorhanden ist.

Der Systemadministrator kann in der Datei `/usr/local/etc/nosendfile` einzelnen Benutzern den Gebrauch von `sendfile` verbieten. Jeder Benutzer kann sich eine Datei `aliases` (unter `/var/spool/sendfile/`*`User`*`/config/`) mit Aliasnamen für Empfänger einrichten. Außerdem kann er die Übertragung von Dateien oder Nachrichten von fremden Rechnern in der Datei `restrictions` untersagen. Weitere Einstellungen lassen sich in der Datei `config` vornehmen, z. B., ob entfernte Benutzer übertragene Dateien wieder löschen dürfen oder wie man bei eingegangenen Dateien benachrichtigt werden möchte.

18 Die Secure Shell ssh

18.1 Sensible Daten in potenziell unsicheren Netzen

Viele Protokolle, die im Zusammenhang mit TCP/IP verwendet werden, sind zunächst mit Blick auf Funktionsfähigkeit und Performance entworfen worden. Auch Aspekte wie Ausfallsicherheit und Redundanzen spielten eine Rolle, aber weniger die Aspekte des Datenschutzes. Heute hat dieses Thema oft mehr Bedeutung, als man dies früher erwartete. Als Beispiel seien die Fernwartung von Rechnern über das Internet oder die Übertragung von Daten über unsichere Netzwerke genannt.

Bei vielen Programmen werden Passwörter im Klartext übertragen. Beispiele hierfür sind z. B. `telnet` oder `ftp`. Wenn eine Verbindung aufgebaut wird, dann kann im lokalen Netz mit einem Packet-Sniffer wie `tcpdump` oder `ethereal` die gesamte Sitzung inklusive der Anmeldung protokolliert werden. Abhilfe können hier Verfahren wie Einmal-Passwörter (etwa S/Key) schaffen, aber die gesamte Verbindung wird weiterhin im Klartext übertragen, also auch weitere Passwörter, wenn beispielsweise das Kommando `su` verwendet wird.

Auf den folgenden Seiten werden wir sehen, dass die Secure Shell (`ssh`) nicht nur sicherer als ältere Protokolle ist, sondern auch wesentlich komfortabler zu benutzen ist. Schon nach kurzer Zeit werden Sie sich fragen, wie Sie bisher ohne `ssh` ausgekommen sind.

Es gibt zwei verbreitete Versionen der `ssh`: einmal das kommerzielle Produkt von SSH Communications Security (*http://www.ssh.com*), dann die `OpenSSH` (*http://www.openssh.com*), die im Rahmen des OpenBSD-Projekts weiterentwickelt wird. OpenSSH basiert auf einer älteren Version der kommerziellen SSH, die unter einer freien Lizenz verfügbar war[1]. Beide Versionen implementieren die Protokolle SSH1 und SSH2 und sind miteinander kompatibel. Für die Beispiele im folgenden Abschnitt habe ich OpenSSH verwendet.

Weitere Informationen finden Sie in den Manpages und auf der oben genannten Webseite. Ein empfehlenswertes Buch über `ssh` ist [Barret2001].

1. Die genaue Historie können Sie auf der Webseite zur OpenSSH nachlesen.

18.2 Kompatibilität mit älteren Anwendungen

Der Begriff »Secure Shell« ist eine Anspielung auf das Programm `rsh`, die Remote Shell (siehe auch Abschnitt 19.5, »Ausführen von Programmen auf entfernten Rechnern«). Auf vielen Unix-Systemen ist dieses Protokoll im Einsatz, es hat aber einige bekannte Sicherheitsschwächen (und findet daher im Kapitel 19, »Obsolete Anwendungen im Netz« Erwähnung). Hier ist nur wichtig, dass `ssh`, `scp` und `slogin` dieselbe Kommandozeilensyntax verstehen wie die entsprechenden r-Tools. Damit kann man die r-Tools erstzen, indem man beispielsweise symbolische Links auf die Programme des ssh-Pakets setzt.

`scp` hat gegenüber `rcp` einen entscheidenden Vorteil: Wenn bei `rcp` die Authentifizierung mittels der Datei `~/.rhosts` nicht gelingt, so wird eine Fehlermeldung ausgegeben und die Datei nicht kopiert. `ssh` wird in diesem Fall nach einem Passwort fragen und nach erfolgter Authentifizierung die Datei kopieren.

Der zugehörige Dienst `sshd` kann in der Datei `/etc/ssh/sshd_config` so konfiguriert werden, dass die `~/.rhosts`-Datei gelesen und akzeptiert wird. Für den Anfang sicher brauchbar, wir werden aber sehen, dass es noch viele weitere Möglichkeiten der Authentifizierung gibt.

In der Standardinstallation wird nur eine Verbindung via SSH-Protokoll versucht. Wenn die Option `FallBackToRsh` für den `ssh`-Client eingeschaltet ist, so wird im Fehlerfall zum `rsh`-Dienst umgeschaltet. In diesem Fall wird er darüber informiert, dass die Sitzung nicht verschlüsselt wird. Verwenden Sie diese Option am besten nur für eine Übergangszeit, bis alle Rechner umgestellt sind.

18.3 Authentifizierung eines Benutzers in der ssh

18.3.1 Authentifizierung mit Passwörtern

In der Standardinstallation ist in der Konfigurationsdatei des ssh-Dämonen (`/etc/ssh/sshd_config`) die Option `PasswordAuthentication` mit »yes« aktiviert. Damit ist es möglich, sich mit Benutzername und Passwort zu authentifizieren.

Man kann auf Passwörter vollständig verzichten, indem eine der im Folgenden vorgestellten Authentifizierungen verwendet wird. Das hat auch den Vorteil, dass man die verschiedenen Passwörter der Benutzer auf den möglicherweise vielen Rechnern vergessen kann – man benötigt diese nie wieder. In diesem Fall kann man im Passwort-Feld in der Datei `/etc/passwd` ein Sternchen (`*`) eintragen – der Benutzer kann sich nicht mehr mittels `login` anmelden, `sshd` benutzt dieses Programm nicht (normalerweise jedenfalls – es gibt die Option `UseLogin` in der Server-Konfiguration).

18.3.2 Authentifizierung mittels ~/.rhosts

Wenn Sie kompatibel zu den r-Tools bleiben müssen, dann können Sie in der Server-Konfiguration den Parameter `IgnoreRhosts` auf `no` und `RhostsAuthentication` auf `yes` setzen. Außerdem muss auf dem Client das Programm `ssh` mit einem s-Bit auf den Benutzer `root` installiert werden, damit ein »sicherer« Quellport verwendet werden kann. In der Regel wird man das heute nicht tun wollen, sondern die Authentifizierung aus dem nächsten Abschnitt wählen.

18.3.3 RhostsRSAAuthentication

Hier wird die Identität der Hosts nicht mit Hilfe der IP-Adresse bestimmt, sondern der zu jedem Host gehörende Public-Key verwendet (`/etc/ssh/ssh_host_key.pub`, `/etc/ssh/ssh_host_dsa_key.pub` oder `/etc/ssh/ssh_host_rsa_key.pub`). Nur wenn dieser passt, d. h. in der Datei `~/.ssh/known_hosts` bzw. `/etc/ssh/ssh_known_hosts` enthalten ist, wird ein passender Eintrag in der Datei `~/.rhosts` akzeptiert.

Um sicherzustellen, dass nicht aus Versehen eine IP-basierte Authentifizierung durch `rsh` erfolgt, kann man die Datei `~/.shosts` verwenden. Der Systemverwalter kann für alle Benutzer die Datei `/etc/ssh/ssh_known_hosts` mit den notwendigen Public-Keys füllen bzw. jeder Benutzer kann die Datei `~/.ssh/known_hosts` selber pflegen (sofern die Serverkonfiguration dies zulässt). Man kann entweder den Public-Key manuell hinzufügen (zum Beispiel mit Hilfe des Programms `ssh-keyscan`) oder loggt sich weschelseitig auf beiden Maschinen ein und lässt die Keys in die Konfiguration aufnehmen.

Sicherheitsbewusste Administratoren veröffentlichen die Public-Keys auf einer gesicherten und signierten Webseite, so dass Anwender prüfen können, ob der vom Rechner gesendete Public-Key tatsächlich der korrekte ist. Ohne diese Prüfung ist eine Man-In-The-Middle-Attacke nicht auszuschließen.

18.3.4 Authentifizierung mit Benutzer-Keys

Bei dieser Version erzeugt jeder Benutzer für sich mit dem Programm `ssh-keygen` ein Key-Paar. Beim Aufruf ohne Parameter wird derzeit ein Key-Paar für die Protokoll-Version SSH1 erzeugt. Sie sollten mit dem Parameter `-t rsa` oder `-t dsa` einen Key für SSH2 erzeugen. Viele Server werden heute mit der Protokollversion SSH2 als Standard installiert. Zu dem Key wird ein Passwort abgefragt, das Sie später für die Authentifizierung benötigen.

Sie können den Key ohne Passwort zu speichern, indem Sie einfach Enter drücken (so werden die Host-Keys erstellt). Damit ist die Anmeldung mit dem Key auch ohne Passwort möglich, das kann allerdings auf dem Zielrechner mit der Option `PermitEmptyPasswords no` verhindert werden.

Als weitere Vorbereitung muss auf dem Zielrechner einmalig die Datei `~/.ssh/authorized_keys` um den betreffenden Public-Key ergänzt werden. Dabei kann sich der Benutzername auf beiden Rechnern unterscheiden, außerdem können in dieser Datei natürlich mehrere Public-Keys enthalten sein. Eine Anwendungsmöglichkeit ist beispielsweise, dass jeder Systemverwalter seinen Public-Key in `/root/.ssh/authorized_keys` hinzufügt und dann mittels `ssh -l root` *`Rechner`* als `root` anmeldet.

Wenn Sie diese Art der Authentifizierung wählen, dann werden Sie bei jeder Verbindung nach Ihrem Passwort für den Key gefragt – nicht besonders komfortabel. Abhilfe verspricht das Programm `ssh-agent`. Dieses läuft im Hintergrund und verwaltet Ihre Keys. Mit dem Programm `ssh-add` laden Sie einen Key in den Agenten und entsperren diesen Key mit Ihrer Passphrase. Anschließend kann jedes Programm, das Zugriff auf den Agenten hat, die Keys dort auslesen und ohne weitere Abfragen verwenden.

Am besten starten Sie den `ssh-agent` in Ihrer X-Session so, dass alle weiteren Programme als Kind-Prozesse gestartet werden. Bei manchen Distributionen ist das standardmäßig der Fall, bei anderen müssen Sie eventuell ein Konstrukt wie in Listing 18.1 verwenden. Wichtig ist, dass alle neu gestarteten Programme die Umgebungsvariablen vom Programm `ssh-agent` erben. Wenn Sie nur einen Window-Manager verwenden, dann starten Sie diesen, ansonsten rufen Sie das Start-Skript Ihres Desktops auf.

```
(linux):~$ cat .xsession
exec ssh-agent /usr/bin/x-session-manager
```

Listing 18.1 Die Verwendung von `ssh-agent`

Im Ergebnis sind in jedem neu gestarteten Programm zwei Umgebungsvariablen gesetzt (Listing 18.2). Als letzten Schritt müssen Sie einmalig für diese Sitzung das Programm `ssh-add` aufrufen, um Ihren Key in den Agenten zu laden. Auch diesen Aufruf finden Sie in Listing 18.2. Den Aufruf von `ssh-agent` können Sie zum Beispiel in den Start-Skripten Ihres Desktops unterbringen.

```
(linux):~$ env | grep SSH
SSH_AUTH_SOCK=/tmp/ssh-XXk3z0Gd/agent.1683
SSH_AGENT_PID=1717
(linux):~$ ssh-add ~/.ssh/id_dsa
Enter passphrase for /home/jochen/.ssh/id_dsa:
Identity added: /home/jochen/.ssh/id_dsa
(/home/jochen/.ssh/id_dsa)
```

Listing 18.2 Umgebungsvariablen für den Zugriff auf `ssh-agent`

18.3.5 Fehleranalyse bei Problemen

Die Einrichtung der `ssh` ist nicht unbedingt einfach und es ist unwahrscheinlich, dass es bei Ihnen auf Anhieb funktioniert. Zwei Befehle werden Ihnen in der Regel weiterhelfen. Starten Sie den `ssh`-Client mit der Option `-v`. Damit erhalten Sie ein Protokoll, in dem Sie nachlesen können, was der Client getan hat.

Auf der Server-Seite rufen Sie `sshd` mit der Option `-d` auf. Dann erhalten Sie ein ähnliches Protokoll aus der Sicht des Servers. Mit beidem und ein wenig Grundverständnis über die Funktionsweise von Public-Key-Verfahren sollte die Lösung der Probleme nicht allzu schwer fallen. Häufig auftretende Probleme sind unter anderem die folgenden:

- Die Berechtigungen für das Home-Verzeichnis bzw. für das Verzeichnis `~/.ssh/` sind unsicher. Vergeben Sie die Rechte so, dass nur der betreffende Benutzer Schreibrechte auf diese Verzeichnisse und Dateien hat. `ssh` hat keine Kenntnis davon, ob sie möglicherweise User Private Groups (Abschnitt 5.8, »Berechtigungen und Gruppen im Einsatz«) verwenden.
- Auf dem Zielrechner ist ein falscher Host-Name des in der Datei `~/.ssh/known_hosts` enthaltenen Schlüssels gespeichert. `ssh` ist da recht pingelig, genauso wie `rsh`.
- Prüfen Sie in den Debug-Ausgaben, welche Keys verwendet werden. Sind das die richtigen und haben Sie das passende Passwort eingegeben?
- Lesen Sie die Meldungen aufmerksam – die Lösung war bisher immer dort zu finden.

18.4 Die Verschlüsselungen von ssh

Für jede Sitzung wird ein so genannter Session-Key generiert, der zur Verschlüsselung verwendet wird. Dieser Schlüssel wird nicht auf der Festplatte gespeichert, sondern existiert nur im Hauptspeicher. Aus Sicherheitsgründen wird dieser Schlüssel regelmäßig verworfen und neu generiert.

Den `sshd`-Prozess startet man sinnvollerweise einmalig beim Systemstart, z. B. in der Datei `rc.local`, da das Generieren der Schlüssel durchaus einige Sekunden dauert, was beim Start durch den `inetd`-Super-Server hinderlich sein kann.

Im Gegenzug wird der Benutzer gewarnt, wenn sich der Host-Key geändert hat. Das kann z. B. bei einem Hardware-Wechsel oder anderen größeren Aktionen notwendig werden. Diese Warnung sollte einen aber normalerweise eher stutzig machen, da sie auf einen Einbruch hindeuten könnte. Ein weiterer Vorteil der `ssh` ist, dass automatisch eine X-Umleitung durchgeführt wird (sofern `ForwardX11` konfiguriert ist). Dabei wird wiederum die gesamte Sitzung ver-

schlüsselt und die Authentifizierung erfolgt mit Hilfe der `ssh` und nicht mit `xhost` oder `xauth`.

Wie Sie sehen, bietet die Secure-Shell einen deutlichen Sicherheitsgewinn und verbessert die Bedienbarkeit des Systems. Aus meiner Sicht gehören diese Programme zur Standardausrüstung eines jeden Unix-Systems, da sie nicht nur für Linux, sondern für fast alle Unix-Systeme verfügbar sind.

18.5 ssh benutzen

Nun aber zum Wesentlichen – wie geht das mit `ssh` eigentlich. Die gute Nachricht ist, wenn Sie bereits mit den r-Tools vertraut sind, dann ändert sich praktisch nichts. Wenn Sie die r-Tools nicht kennen – macht nichts, denn das werden wir uns jetzt ansehen.

18.5.1 Netzwerkanmeldung mit slogin

Eine interaktive Shell erhalten Sie mit dem Befehl `slogin`. Wenn Sie nur den Host-Namen angeben, so wird auf dem Zielrechner derselbe Benutzername verwendet. Mit der Option `-l` können Sie einen anderen Namen angeben (Listing 18.3).

```
(linux):~> slogin typhon
(typhon):~>
...
(typhon):~> exit
Connection to typhon closed.
(linux):~> slogin typhon -l root
Last login: Wed Feb 13 22:22:57 2002 from localhost on pts/1
(typhon):~# id
uid=0(root) gid=0(root) Gruppen=0(root)
(typhon):~# exit
```

Listing 18.3 Anmeldung mit `slogin`

Eine weitere, nicht zu `rlogin` kompatible Notation, ist *`Benutzer@Rechnername`*. Wenn man diese Art der Bezeichnung von Mail gewöhnt ist (und wer ist das heute nicht), dann ist das vermutlich einfacher zu merken. `slogin` ist nichts anderes als ein symbolischer Link auf `ssh`, deshalb machen wir gleich damit weiter.

18.5.2 Kommandos ausführen mit ssh

Wenn bei ssh kein Kommando angegeben wird, dann wird ein Pseudo-Terminal alloziert und auf dem entfernten Rechner eine Login-Shell gestartet. Wird ein Kommando angegeben, so wird normalerweise kein Terminal bereitgestellt, so dass zum Beispiel Programme wie vi nicht verwendet werden können. Da es aber manchmal doch praktisch ist, genau das zu tun, kann man mit der Option -t ein Pseudo-Terminal erhalten. Ich benutze die Funktion in verschiedenen Aliasen als etwas anderes sudo.

```
(linux):~> ssh -- jupiter ls -l -d bin
drwxr-xr-x   2 hein     staff        512 Jul 26 16:22 bin
```

Listing 18.4 Ausführen von Programmen auf entfernten Rechnern mit ssh

Wieder ist ssh kompatibel zu rsh, der remote Shell. Wenn bei rsh die Authentifizierung fehlschlägt, so erhalten Sie die Maldung »permission denied«. ssh fragt nach dem Passwort bzw. Ihrer Passphrase, so dass Sie diese Funktion auch zwischen Rechnern verwenden können, die nicht für eine passwortfreie Anmeldung konfiguriert sind.

18.5.3 Dateien kopieren mit scp

Das Programm scp (Secure Copy) dient zum Kopieren von Dateien zwischen zwei Rechnern, auf denen Sie jeweils eine gültige Benutzerkennung haben müssen. Auf den Rechnernamen des entfernten Rechners folgt ein Doppelpunkt (:) und der Name einer Datei oder des Zielverzeichnisses. Eine andere Benutzerkennung kann angegeben werden, indem vor dem Rechnernamen der Benutzername, gefolgt von einem at-Symbol (@), angefügt wird (siehe Listing 18.5). Die Autorisierung erfolgt über denselben Mechanismus wie bei ssh.

```
scp .netrc jupiter.jochen.org:.
scp .netrc jochen@jupiter.jochen.org:.
scp user1@host1:file1 user2@host2:file2
```

Listing 18.5 Kopieren von Dateien im Netz mit scp

ssh ist aufrufkompatibel mit dem Kommando rcp, so dass Benutzer sich praktisch nicht umgewöhnen müssen. Ein wesentlicher Vorteil gegenüber rcp ist allerdings, dass scp die Daten verschlüsselt überträgt und gegebenenfalls eine Passwortauthentifizierung ermöglicht – rcp gibt in diesem Fall nur »Permission denied« aus.

18.5.4 Sicherer Dateitransfer mit sftp

Gelegentlich ist der Dateitransfer mit `scp` zwar angenehm in Skripten, für Benutzer mag es aber komfortabler sein, mit Befehlen wie `cd` und `lcd` auf dem entfernten bzw. lokalen Rechner in die Verzeichnisse zu wechseln und erst dort den Dateitransfer anzustoßen. Besonders, wenn man sich mit dem Dateinamen oder Verzeichnisnamen nicht sicher ist, dann ist dies sinnvoll[2].

Ein weiterer Vorteil von `sftp` ist, dass die Benutzeroberfläche der von `ftp` nachempfunden wurde – Benutzer, die bisher `ftp` zum Dateitransfer verwenden, können sehr einfach zu einer sicheren Übertragung wechseln.

Ein anderer Benutzername für den Zielrechner kann, wie bei den anderen s-Tools mit der `@`-Notation angegeben werden. Außerdem kann in der Kommandozeile die zu übertragende Datei und der Name der Zieldatei angegeben werden (Listing 18.6).

```
(Linux):~> sftp typhon
Connecting to typhon...
jochen@typhon's password:
sftp> cd tmp
sftp> dir
drwxrwxr-x    3 jochen   users         512 Feb  3 09:47 .
drwxr-xr-x   17 jochen   users        1024 Mar 10 09:11 ..
sftp> put .gnus
Uploading .gnus to /home/jochen/tmp/.gnus
sftp> dir
drwxrwxr-x    3 jochen   users         512 Mar 10 09:11 .
drwxr-xr-x   17 jochen   users        1024 Mar 10 09:11 ..
-rw-r--r--    1 jochen   users        8020 Mar 10 09:12 .gnus
sftp> quit
(Linux):~> sftp typhon:tmp/.gnus .
Connecting to typhon...
jochen@typhon's password:
Fetching /home/jochen/tmp/.gnus to ./.gnus
```

Listing 18.6 Dateiübertragung mit `sftp`

18.6 TCP-Verbindungen tunneln

Eine weitere Funktion von `ssh` ist die Möglichkeit, beliebige TCP-Verbindungen umzuleiten und dabei zu verschlüsseln. Dazu startet man zunächst `ssh` zusätzlich zum Parameter *`Zielrechner`* mit der Option `-L` *`Port`*`:`*`Host`*`:`*`Hostport`*. Da-

2. Alternativ kann man zum Beispiel die programmierbare Vervollständigung seiner Shell anpassen und verwenden – aber auch dann kann es aufgrund der Antwortzeiten unkomfortabel langsam sein.

bei stellt ssh eine Verbindung zwischen dem lokalen Port *Port* und dem Port *Hostport* auf dem Rechner *Host* her. Die Teilstrecke der Verbindung zwischen dem lokalen Rechner und dem *Zielrechner* wird dann verschlüsselt.

In Ihrem Anwendungsprogramm verwenden Sie als Zielrechner und -Port nicht den endgültigen Rechner, sondern den angegebenen Port auf dem lokalen Rechner. Alles andere wird intern von der ssh erledigt.

Die Umleitung von Ports kann in beiden Richtungen erfolgen, die Gegenrichtung zur Option -L ist die Option -R. Da man den sshd-Dienst und den Client mit der Option -p auf einem beliebigen Port starten kann, ist es damit möglich, z. B. Beschränkungen einer Firewall zu umgehen. Wenn Sie also eine Firewall betreiben, seien Sie sich bewusst, dass auf dem Port 80 (www) nicht in jedem Fall ein Web-Server laufen muss – das kann genausogut eine Secure-Shell sein.

18.7 Weitere Optionen in der Datei authorized_keys

Wenn man ssh einsetzt, so gewinnt man einiges an Sicherheit – Benutzer können mittels Public-Key-Verfahren authentifiziert werden und Daten werden nur noch verschlüsselt übertragen. Ein Problem ist, dass der Private Key des Benutzers die Zugangsberechtigung zum Rechner darstellt (eventuell noch mit einem Passwort gesichert). Wenn nun dieser Key entwendet wird, so sind zunächst keine weiteren Schutzmaßnahmen aktiv – der entsprechende Benutzer steht dem Angreifer offen. Aber auch hier bietet ssh eine Hilfe: Mit der Option from="..." kann für einen Key nur der Zugang von bestimmten Host-Namen erlaubt werden. Neben dem Key müssten dann noch die Nameserver bzw. Router kompromittiert werden.

Eine weitere Option zu einem Key ist command="...", damit wird das Kommando, das der Benutzer eventuell angegeben hat, ignoriert und stattdessen das hier konfigurierte Kommando ausgeführt. Damit ist es zum Beispiel möglich, einem Operator nur das Kommando zur Datensicherung zu erlauben, aber sonst nichts. In der Manpage zu ssh finden Sie noch weitere Optionen, mit denen Sie unter anderem X- oder Port-Forwarding verbieten oder einzelne Ports gezielt erlauben können.

18.8 Secure Shell unter Windows

Vielfach haben Unix-Administratoren einen Arbeitsplatzrechner unter Windows, und müssen von diesem aus die Systeme verwalten. Da das in Windows enthaltene telnet-Programm oft zu wünschen übrig lässt, sollte man es alsbald durch einen ssh-Client ersetzen.

Eine Möglichkeit dazu ist die Verwendung von `PuTTY`, das Sie unter *http://www.chiark.greenend.org.uk/~sgtatham/putty/* finden. Alternativ können Sie die Cygwin-Portierung der OpenSSH verwenden, die Sie unter *http://www.networksimplicity.com/openssh/* finden.

19 Obsolete Anwendungen im Netz

19.1 Ausgereift, bewährt, aber nicht mehr auf der Höhe der Zeit

ACHTUNG

Die in diesem Kapitel vorgestellten Programme sind zwar noch weit verbreitet, vom Einsatz im Internet oder in nur begrenzt vertrauenswürdigen Umgebungen ist jedoch dringend abzuraten. Und nebenbei, die Verwendung von `ssh` (siehe Kapitel 18, »Die Secure Shell ssh«) ist nicht nur sicherer sondern auch noch bequemer. Also tun Sie sich und mir einen Gefallen und lesen Sie dieses Kapitel nur, wenn Sie die Tools *wirklich* benötigen.

Vielfach werden Sie keinen direkten Einfluss auf fremde Systeme haben, auf denen zum Beispiel `ssh` noch nicht installiert ist. Dann können Sie mit den Informationen hier immerhin arbeiten, sollten aber außerdem darauf drängen, modernere Tools zumindest parallel zu installieren und zu evaluieren.

Die häufig ins Feld geführten Argumente »Wir haben doch nichts zu verbergen« und »Verschlüsselung kostet zu viel Rechenzeit« sind für die hier vorgestellten Programme irrelevant: Es werden sensitive Daten (zum Beispiel Passwörter) und relativ geringe Datenmengen übertragen.

Sie haben bis hier gelesen? Dann bleibt uns wohl nichts anderes übrig, als uns die betreffenden Programme mal anzusehen. Nun denn ...

19.2 Das telnet-Programm

Mit dem Programm `telnet` (Terminal Emulation over Network) kann eine Anmeldung auf einem entfernten Rechner erfolgen. Das Programm (und das gleichnamige Protokoll) definieren ein »Network Virtual Terminal«, über das Programme bedient oder andere Dienste angesprochen werden können.

Dazu muss auf dem entfernten Rechner der `telnetd`-Dämon gestartet sein. Das geschieht üblicherweise mit dem `inetd`-Super-Server. Außerdem müssen Sie über eine gültige Benutzerkennung (Benutzername und Passwort) auf diesem Rechner verfügen. Neben dieser normalen Anwendung kann `telnet` auch zum Test von anderen TCP-Protokollen dienen. Das `telnet`-Protokoll ist in [rfc0854] definiert.

Im einfachsten Fall lautet der Aufruf `telnet Host-Name`. Dadurch wird eine `telnet`-Verbindung zum angegebenen Rechner hergestellt. Wenn Sie das Programm `telnet` ohne Parameter aufrufen oder die Escape-Sequenz senden (normalerweise Strg-[]), dann gelangen Sie in den Kommandomodus. In diesem Modus können Sie am Prompt (`telnet>`) eine Verbindung öffnen (`open`), schließen (`close`) oder spezielle Zeichen schicken (`send`). Eine ausführliche Hilfe zu allen Befehlen finden Sie in der Manpage zu `telnet(1)`. Eine kurze Übersicht erhalten Sie, wenn Sie im Kommandomodus `help` eingeben. Beachten Sie, dass der Kommandomodus nach jedem Befehl beendet wird, wenn eine Verbindung zu einem anderen System besteht.

HINWEIS

Sie können zu einem anderen Port als dem `telnet`-Port (23) Verbindungen aufbauen, indem Sie den gewünschten Port als Service-Name oder -Nummer nach dem Host-Namen angeben. So können Sie mit dem Befehl `telnet localhost smtp` oder `telnet localhost 25` eine Verbindung zum `smtp`-Port (Simple Mail Transfer Protocol) aufbauen und dann direkt mit dem Mail-Dämon kommunizieren.

Ein Beispiel für einen solchen Test finden Sie in Abschnitt 28.5, »TCP-Dienste prüfen«. Das ist aus meiner Sicht heute der einzige legitime Einsatzzweck für das Programm `telnet`. Und den Dienst `telnetd` sollten Sie in der Datei `/etc/inetd.conf` deaktivieren, vorausgesetzt, Sie möchten `ssh` einsetzen und haben die Programme installiert und aktiviert.

Neben dem Test von einzelnen Services kann `telnet` auch zu einem Test der Verbindung missbraucht werden. Dazu werden in der Regel die Ports `discard` (`9`) und `echo` (`7`) verwendet. Bei `discard` werden alle eingegebenen Zeichen ignoriert, `echo` sendet alle eingegebenen Zeichen (zeilenweise) unverändert zurück.

Das Programm `telnet` liest beim Start die Datei `~/.telnetrc`. In dieser Datei können Sie verschiedene Einstellungen vornehmen, um `telnet` an Ihre Anforderungen anzupassen. Das Listing 19.1 zeigt einige Konfigurationsmöglichkeiten, eine vollständige Darstellung finden Sie in der Manpage zu `telnet`.

```
# Konfiguration des telnet-Programms
localhost toggle binary
typhon      toggle binary
          set interrupt ^K
```

Listing 19.1 Die Datei `~/.telnetrc`

Kommentare in der Datei beginnen mit einer Raute (#) in Spalte 1 und umfassen die ganze Zeile. Leerzeilen werden ignoriert. Gültige Einträge beginnen mit einem Host-Namen in Spalte 1, gefolgt von Optionen und Befehlen. Ein Eintrag kann sich über mehrere Zeilen erstrecken, sofern die Fortsetzungszeilen mit einem Whitespace (Leerzeichen oder Tabulator) beginnen.

In Listing 19.1 wird für die Rechner `localhost` und `typhon` die binäre Übertragung vereinbart. Bei dem Rechner `typhon` wird zusätzlich das Interrupt-Zeichen verändert. Alle Optionen werden bei der Aktivierung des `telnet`-Programms ausgegeben.

Zum `telnet`-Protokoll gibt es verschiedene Erweiterungen, so auch eine Einbindung in Kerberos. Damit kann die Authentifizierung mit Hilfe von Kerberos-Tickets erfolgen und die Verbindung wird verschlüsselt. Mit diesen Erweiterungen ist `telnet` ähnlich sicher wie `ssh`.

19.3 Das File-Transfer-Protokoll (ftp)

Mit dem Programm `ftp` können Dateien zwischen Rechnern ausgetauscht werden. Das zugrunde liegende File Transfer Protocol wird in [rfc0959] definiert. Beim Datenaustausch zwischen zwei Rechnern sind prinzipiell zwei Möglichkeiten zu unterscheiden:

- Sie haben auf dem entfernten Rechner eine Benutzerkennung und können sich entweder mittels `telnet` einloggen oder mit `ftp` Dateien übertragen.[1]
- Sie haben auf dem entfernten Rechner keine Benutzerkennung, aber dort ist ein Anonymous-ftp-Server eingerichtet. Dort können Sie sich als Benutzer `anonymous` oder `ftp` einloggen. Als Passwort wird üblicherweise erwartet, dass Sie Ihre E-Mail-Adresse angeben. Oft können Sie auch die Kurzform *`Name`*`@` verwenden, die vom `ftp`-Server automatisch erkannt wird.

  Anschließend können Sie von diesem Rechner Dateien herunterladen oder in spezielle Verzeichnisse, die meist `incoming` heißen, ablegen. Nachdem der Administrator die Dateien in das `ftp`-Verzeichnis integriert hat, können andere Benutzer auf diese zugreifen. Dieses Verfahren dient im Internet zum Austausch von Programmen oder Dokumentationen. Weitere Informationen hierzu finden Sie in [rfc1635]. Die Installation eines `anonymous-ftp`-Servers wird im Kapitel 25, »Anonymous-ftp-Server« erläutert.

Sie können das `ftp`-Programm entweder mit einem Rechnernamen als Parameter starten, so dass direkt eine Verbindung zu diesem Rechner hergestellt wird. Sie können aber auch `ftp` ohne Parameter starten und landen dann im Komman-

1. Es gibt auch `ftp`-Server für Betriebssysteme, die keine Benutzerkennungen oder `telnet`-Verbindungen kennen. Manchmal ist eine Benutzerverwaltung in den Dämonen implementiert, manchmal hat man aber auch Zugriff auf den gesamten Rechner.

domodus (mit ftp> als Prompt). Mit help erhalten Sie eine Übersicht über die möglichen Kommandos. In Listing 19.2 finden Sie ein kurzes Beispiel für eine ftp-Sitzung.

```
(linux)~> ftp typhon
Connected to typhon.
220 typhon FTP server (Version wu-2.4(1)) ready.
Name (typhon:jochen): jochen
331 Password required for jochen.
Password:
230 User jochen logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bin
200 Type set to I.
ftp> hash
Hash mark printing on (1024 bytes/hash mark).
ftp> prompt
Interactive mode off.
ftp> lcd /tmp
Local directory now /tmp
ftp> get .emacs
200 PORT command successful.
150 Opening BINARY mode data connection for .emacs.
226 Transfer complete.
11557 bytes received in 0.0184 secs (6.1e+02 Kbytes/sec)
ftp> quit
221 Goodbye.
```

Listing 19.2 Übertragen von Dateien mit ftp

Im ersten Schritt wird die Verbindung zum Rechner typhon aufgebaut. Die Anmeldung erfolgt mit Ihrem Benutzernamen (hier jochen) und Passwort, da der angesprochene Rechner kein anonymous-ftp erlaubt. Die Übertragung soll im Binärmodus erfolgen (Eingabe bin), da andernfalls das Protokoll eine Zeichensatzkonvertierung versuchen würde. Dabei werden Archive und andere Binärdateien in der Regel unbrauchbar.

Mit hash wird bei der Datenübertragung der Fortschritt mit Rauten dargestellt. Gerade bei großen Dateien ist das nützlich, man sieht, dass die Übertragung tatsächlich funktioniert. Der Befehl prompt unterdrückt die Nachfragen des ftp-Clients beim später folgenden mget. Auf dem ftp-Server wechselt man mit dem Befehl cd in ein anderes Verzeichnis; das lokale Verzeichnis wird mit lcd verändert. Im letzten Schritt wird die Datei .emacs übertragen und die Verbindung beendet.

Das Programm ftp kann mit der Datei ~/.netrc konfiguriert werden. In dieser Datei können Sie beispielsweise für jeden Host eine Reihe von Befehlen definieren, die nach der (möglicherweise automatisch durchgeführten) Anmeldung ausgeführt werden. Das Listing 19.3 zeigt eine Initialisierungsdatei für ftp. Beim Verbindungsaufbau zum Rechner typhon wird jochen als Benutzername verwendet (Schlüsselwort machine). Bei allen anderen Rechnern wird ein anonymous-ftp versucht. Mit dem Befehl macdef init können Sie Befehle angeben, die nach der Anmeldung automatisch ausgeführt werden sollen. Alle Optionen und Befehle des ftp-Clients sind in der Manpage ftp(1) dokumentiert.

```
# Eintrag für den Rechner typhon
machine typhon login jochen
macdef init
bin
hash

# Default für anonymous-ftp
default login anonymous password jochen.hein@li.org
macdef init
bin
hash
```

Listing 19.3 Beispiel für eine ~/.netrc

Die speziellen Features mancher ftp-Server sind für die Benutzer meist in einer README-Datei dokumentiert. Dort werden spezielle Funktionen, wie das Übertragen ganzer Verzeichnisbäume und das Umpacken von Dateien erklärt, genauso wie der möglicherweise eingerichtete Mail-Server, der Dateien per Mail versendet.

In diesen Dateien oder den bei der Anmeldung angezeigten Meldungen wird oft darauf hingewiesen, dass alle Befehle und Übertragungen mitprotokolliert werden. Wer dies nicht möchte, sollte derartige Server nicht benutzen. Da die Bereitstellung eines ftp-Servers oft als »Service« für die Benutzer verstanden wird, sollte man als Anwender hierfür Verständnis haben und sich entsprechend verhalten.

19.3.1 Probleme bei der Verwendung von ftp

Wenn Sie Probleme haben, sich via ftp anzumelden, sollten Sie prüfen, ob Ihre Login-Shell auf dem Zielrechner in der Datei /etc/shells aufgeführt ist. Es existieren ftp-Dämonen, die die Anmeldung ablehnen, wenn der Benutzer keine gültige Shell hat. Außerdem kann es sein, dass der Benutzer in der Datei /etc/ftpusers eingetragen ist. In dieser Datei sind die Benutzerkennungen aufgeführt, für die ftp-Verbindungen nicht erlaubt sind.

Wenn Sie beim Laden von Dateien aus dem Internet die Meldung »permission denied« erhalten, dann prüfen Sie, ob Sie im lokalen Verzeichnis Schreibrechte haben. Als normaler Benutzer in ein anderes Verzeichnis zu wechseln, in dem man keine Dateien anlegen darf, und dann `ftp` aufzurufen, ist ein häufiger Fehler. Es kann natürlich auch sein, dass Sie die Dateien auf dem Server nicht lesen können, aber das sollte die Ausnahme sein.

`ftp` hat einige Eigenarten, die es für den Betrieb über eine Firewall oder einen IP-Adressumsetzer nicht geeignet erscheinen lassen. Zusätzlich zur Steuerverbindung, die der Client auf den Port 21 (`ftp`) öffnet, wird die Datenübertragung über eine neue Verbindung durchgeführt. Diese Verbindung wird vom `ftp`-Server zum Client geöffnet und geht vom Port `ftp-data` (20) aus. Als Alternative kann der so genannte Passiv-Modus verwendet werden, in dem die Datenübertragung auch über den Steuerkanal stattfindet. Viele Clients, insbesondere Web-Browser, verwenden diese Option als Standard.

19.3.2 Andere ftp-Clients

Neben dem Standard-`ftp`-Programm gibt es viele andere `ftp`-Clients. Ich benutze häufig das leistungsfähigere `ncftp` (*http://www.ncftp.org/*), das eine komfortablere Benutzeroberfläche bietet. Eine oft nützliche Funktion ist das Übertragen von ganzen Verzeichnisbäumen, das hier im Client implementiert ist. Praktisch sind auch das integrierte Adressbuch und die Möglichkeit, Kurznamen für Server zu vergeben und bei einer neuen Verbindung automatisch im ursprünglichen Verzeichnis zu landen.

Aus derselben Quelle können Sie als Shareware einen `ftp`-Server beziehen, der etwas einfacher zu administrieren ist als der Standard-`ftp`-Server und außerdem eine geringere Last auf dem Rechner verursacht. Nützlich für Anwender sind zwei weitere Programme aus diesem Umfeld: `ncftpget` und `ncftpput`. Mit Hilfe dieser Tools können Sie mit Shell-Skripten einfach einen `ftp`-Transfer durchführen, ohne dass Sie eine `~/.netrc`-Datei schreiben müssen.

Auch für X gibt es andere, komfortablere `ftp`-Clients, wie `xftp` oder `mxFtp`. Der Dateimanager von KDE ist ein vollständiger Web-Browser und kann daher auch in `ftp`-Servern verwendet werden. Wenn Sie sich noch an den Norton Commander erinnern, dann setzen Sie vielleicht doch lieber den GNU Midnight Commander (`mc`) ein; auch dieser beherrscht `ftp`.

Außerdem enthält Emacs einen `ftp`-Modus (`ange-ftp`), mit dem Sie innerhalb der gewohnten `dired`-Umgebung bequem Dateien suchen, ansehen, ändern und übertragen können. Geben Sie als Dateinamen einfach */user@host:Dateiname* an.

Wenn Sie keinen Online-Zugang zum Internet haben, sondern nur mittels UUCP Mail und News austauschen, können Sie trotzdem Dateien auf `ftp`-Servern erhalten. Einige Server versenden Dateien auch per E-Mail, die entsprechenden

Anweisungen dazu finden Sie meist in der README-Datei des Servers. Daneben gibt es Server, die kommando- und mailgesteuert beliebige Dateien von beliebigen Rechnern übertragen und dann per Mail (uuencoded) verschicken. Entsprechende Adressen finden Sie in der FAQ *Accessing the Internet by Mail*.

19.4 Die r-Tools

Ein Nachteil des telnet- und ftp-Protokolls ist, dass bei jeder Anmeldung das Passwort unverschlüsselt über das Netz, also potenziell unsichere Verbindungen, übertragen wird.[2] Mit den r-Tools kann dies vermieden werden, allerdings verlässt man sich auf die Zuverlässigkeit der Systemadministratoren. Außerdem existieren diese Tools, die aus der BSD-Welt stammen, nur auf Unix-Systemen. Einige NT-Server haben diese Dienste zwar auch installiert, sie gehören aber nicht zum normalen Lieferumfang.

Eine weitere Einschränkung besteht darin, dass auf dem entfernten Rechner die entsprechenden Dämonen gestartet sein müssen. Auf vielen Systemen ist dies aus Sicherheitsgründen nicht der Fall. Die Sicherheit kann dadurch beeinträchtigt werden, dass ein Benutzer ein Pluszeichen (+) in die Datei ~/.rhosts aufnimmt und damit seine Benutzerkennung den Blicken aller Welt öffnet. Es gibt Dämonen, die derartige Einträge einfach ignorieren. Das Protokoll selbst ist anfällig für IP- und DNS-Spoofing und nur in Umgebungen wirklich sinnvoll, die unter einheitlicher Administration stehen.

Die r-Tools wurden an der Universität von Kalifornien in Berkeley entwickelt, als das TCP/IP-System unter BSD implementiert wurde. Neben den bereits standardisierten Protokollen wie telnet und ftp wurden noch weitere Protokolle entwickelt und einige lokale Unix-Kommandos an die Verwendung im Netz und auf entfernten Rechnern angepasst. Das r steht dabei als Kürzel für remote.

19.5 Ausführen von Programmen auf entfernten Rechnern

Mit dem Programm rsh (Remote Shell, nicht zu verwechseln mit der Restricted Shell, die zum Teil auch rsh genannt wird) können Programme auf entfernten Rechnern gestartet werden. Dazu muss auf dem entfernten Rechner die Datei ~/.rhosts (siehe Listing 19.5) angelegt werden. Jede Zeile dieser Datei enthält einen Rechnernamen. Damit kann sich der Benutzer direkt von dem fremden Rechner aus mit den r-Tools anmelden, ohne ein Passwort angeben zu müssen.

2. Mit dem Tool tcpdump (siehe auch Abschnitt 28.3.1, »IP-Pakete untersuchen«) kann sich der Systemadministrator jedes TCP-Paket im lokalen Netz ansehen. Ähnliche Programme gibt es auch für DOS-Rechner.

Auf manchen Systemen heißt das Programm `rsh` auch `remsh`, um die Namenskollision mit der »Restricted Shell« zu umgehen.

Das Listing 19.4 zeigt ein Beispiel für die Verwendung des Befehls `rsh`. Der erste Parameter (`--`) verhindert, dass die Optionen des Befehls `ls` bereits von `rsh` auf dem lokalen Rechner ausgewertet werden. Dies ist eine Eigenart der GNU-Funktion, die zur Auswertung der Kommandozeilenoptionen und Parameter verwendet wird.

```
(linux):~> rsh -- jupiter ls -l -d bin
drwxr-xr-x   2 hein     staff        512 Jul 26 16:22 bin
```

Listing 19.4 Ausführen von Programmen auf entfernten Rechnern mit `rsh`

Hat ein Benutzer verschiedene Accounts (z. B. `hein` auf dem einen und `jhein` auf dem anderen Rechner), so kann er vor dem Rechnernamen mit der Option `-l` noch eine Benutzerkennung angeben, von der aus ein Einloggen möglich sein soll. Weitere nützliche Optionen sind `-8` für eine 8-Bit-fähige Sitzung und `-E` zum Ausschalten der Sonderbehandlung des Escape-Zeichens (normalerweise die Tilde). Mit der Option `-e` kann ein anderes Zeichen hierfür bestimmt werden.

Einige `rlogind`-Dämonen gestatten die Angabe von `+` in `.rhosts`-Dateien. Dies erlaubt Benutzern mit demselben Benutzernamen die Anmeldung, egal von welchem Rechner aus sie `rlogin` aufrufen. Derartige Einträge sollten Sie aus Sicherheitsgründen unbedingt vermeiden. Außerdem gibt es Dämonen, die diesen Eintrag ignorieren.

In Listing 19.5 finden Sie ein Beispiel für die Datei `~/.rhosts`. Diese Datei sollte nur für den Benutzer selbst lesbar sein, das Verzeichnis nur für diesen Benutzer änderbar. Jede Zeile beschreibt einen Rechner, von dem aus ein `rsh`, `rlogin` oder `rcp` ohne Passwort möglich sein soll. Als Rechnernamen sollten Sie den ersten Eintrag aus der Datei `/etc/hosts` bzw. den Fully Qualified Domain Name (FQDN) angeben. Haben Sie auf den Rechnern unterschiedliche Benutzerkennungen, so können Sie danach einen Benutzernamen eintragen.

```
typhon.jochen.org hein
jupiter.jochen.org jhein
```

Listing 19.5 Die Datei `~/.rhosts`

Der Systemadministrator kann Rechner (eines lokalen Netzes) als gleichberechtigt einrichten, indem er die Namen aller lokalen Rechner in die Datei `/etc/hosts.equiv` (siehe Listing 19.6) jedes Rechners einträgt. Dies macht jedoch nur für Netze Sinn, in denen Rechner und Benutzerkennungen zentral verwaltet werden.

Die Datei /etc/hosts.equiv wird bei Zugriffen mit der Benutzerkennung root nicht verwendet, so dass in diesem Fall zusätzlich eine ~/.rhosts-Datei erforderlich ist. In einem lokalen Netz, das zentral verwaltet werden soll, kann dies eine erhebliche Erleichterung sein, denn es ermöglicht z. B. die Verwendung von rdist (siehe den Abschnitt 17.5, »Übertragen von Dateien mit rdist«).

```
typhon.jochen.org
-janus.jochen.org
hermes.jochen.org    sven
```

Listing 19.6 Die Datei /etc/hosts.equiv

Der erste Eintrag in Listing 19.6 sorgt dafür, dass sich jeder Benutzer (außer root) von typhon.jochen.org aus mit rlogin ohne Passwort anmelden kann. Der zweite Eintrag verlangt für jede Anmeldung von janus.jochen.org ein Passwort, egal, was einzelne Benutzer in ihrer ~/.rhosts-Datei eingetragen haben. Die dritte Zeile ermöglicht es dem Benutzer sven auf hermes.jochen.org, sich ohne Passwort als beliebiger Benutzer (außer root) anzumelden.

Dieses Verfahren basiert darauf, dass anhand der ankommenden IP-Pakete (nämlich der Absenderadresse) auf den korrekten Sender (Rechner) geschlossen werden kann. Dies ist jedoch nicht immer der Fall, da mit dem so genannten IP-Spoofing eine andere Absenderadresse oder mit DNS-Spoofing ein anderer Host-Name angegeben werden kann. Wenn also Sicherheitsüberlegungen eine große Rolle spielen, ist diese Methode sicher nicht zu empfehlen. Das gilt insbesondere dann, wenn der Zugriff auf oder von externen Rechnern erfolgt. Abhilfe bietet der Einsatz der Secure Shell ssh, siehe auch Kapitel 18, »Die Secure Shell ssh«.

Ein Nachteil von rsh ist, dass mit dem Programm kein tty assoziiert ist. Damit können Programme wie vi nicht direkt mit rsh gestartet werden. Dazu ist die Anmeldung, z. B. mit rlogin, notwendig. Wenn Sie bei der Verwendung von rsh Fehlermeldungen wie »$TERM is not defined« (von setterm) oder »stty: standard input: invalid argument« (von stty) erhalten, so müssen Sie Ihre Login-Skripten entsprechend anpassen. In Listing 19.7, finden Sie Anpassungen für die tcsh, in Listing 19.8 entsprechend für die bash. Damit kann verhindert werden, dass die Befehle stty oder setterm aufgerufen werden, obwohl kein tty verfügbar ist.

```
tty -s
if ( $status == 0 )
        stty pass8
endif
```

Listing 19.7 Anpassungen der ~/.login-Datei

```
tty -s
if [ $? = 0 ]; then
        stty pass8
endif
```

Listing 19.8 Anpassungen der ~/.profile-Datei

Basierend auf rsh wurde für X ein Protokoll rstart entwickelt, das den Start von X-Programmen auf anderen Rechnern vereinfacht. Dabei werden sowohl der aktuelle Wert der DISPLAY-Variable als auch die Autorisierung (siehe Abschnitt 7.8.2, »Benutzerbezogenen Zugriff erlauben mit xauth«) für den X-Server übertragen. Dazu ist allerdings der rstartd-Dämon notwendig, der nur auf wenigen Rechnern gestartet ist.

Eine andere Möglichkeit bietet das Programm xon, das auch auf rsh basiert, aber keinen zusätzlichen Dämon erfordert. Bei der Verwendung von xauth zur Authentifizierung von X-Zugriffen muss diese aber separat erfolgen. Ein entsprechende Anwendung finden Sie in Listing 19.9.

```
xauth extract - $DISPLAY | rsh RemoteHost xauth merge -
```

Listing 19.9 Übertragen der X-Authority mittels rsh

Bei der Verwendung von ssh (siehe Kapitel 18, »Die Secure Shell ssh«) ist es nicht mehr erforderlich, xhost oder xauth zu verwenden. Die Display-Umleitung inklusive der Authentifizierung übernimmt hier die Secure-Shell. Dieses Verfahren ist sehr komfortabel und funktioniert sehr stabil. Es ist nicht einfach, mit einem Shell-Skript die korrekte DISPLAY-Variable zu generieren und zu verwalten – das Skript, das ich auf Rechnern ohne ssh verwende, funktioniert nur unter recht engen Rahmenbedingungen.

Das in der rsh verwendete Protokoll wurde nachträglich in [rfc1258] dokumentiert. In diesem Fall dient der RFC nur zur Dokumentation eines bereits implementierten und verwendeten Protokolls.

19.6 Anmeldung auf entfernten Rechnern

Mit dem Programm rlogin (Remote Login) kann auf einem entfernten Rechner eine interaktive Sitzung durchgeführt werden. Gibt man bei dem Befehl rsh keinen Befehl an, so wird automatisch ein Login durchgeführt und damit der Shell ein tty zugeordnet. Gibt man bei rsh ein Programm an, das ein tty benötigt (wie z. B. vi), dann erhält man eine entsprechende Fehlermeldung.

Die Anmeldung mit rlogin anstelle von telnet wird auf Unix-Rechnern verwendet, weil bei entsprechender Konfiguration (wie bei rsh) kein Passwort eingegeben werden muss. Das Programm rlogin fragt als einziges von den r-Tools

nach einem Passwort, falls eine Anmeldung ohne Passwort nicht möglich ist. Alle anderen r-Tools geben unter diesen Bedingungen nur die Fehlermeldung »permission denied« aus.

Die Überlegungen zur Sicherheit sind hier dieselben wie bei rsh. Dieses Protokoll ist in [rfc1282] definiert.

Wird das Programm rlogin mit dem Namen eines Rechners aufgerufen, so wird automatisch eine Verbindung dorthin aufgebaut. Sie können also beispielsweise für jeden Zielrechner einen symbolischen Link auf das Programm rlogin anlegen. Ich selbst definiere in meinen Anmeldeskripten für jeden häufig verwendeten Rechner einen Aliasnamen.

19.7 Übertragung von Dateien

Das Programm rcp (remote Copy) dient zum Kopieren von Dateien zwischen zwei Rechnern, auf denen Sie jeweils eine gültige Benutzerkennung haben müssen. Auf den Rechnernamen des entfernten Rechners folgt ein Doppelpunkt (:) und der Name einer Datei oder des Zielverzeichnisses. Eine andere Benutzerkennung kann angegeben werden, indem vor dem Rechnernamen der Benutzername, gefolgt von einem at-Symbol (@), angefügt wird (siehe Listing 19.10). Die Autorisierung erfolgt über denselben Mechanismus wie bei rsh bzw. rlogin.

```
rcp .netrc jupiter.jochen.org:.
rcp .netrc jochen@jupiter.jochen.org:.
rcp user1@host1:file1 user2@host2:file2
```

Listing 19.10 Kopieren von Dateien im Netz mit rcp

Auf älteren Unix-Systemen, auf denen der Befehl cp nicht in der Lage ist, Unterverzeichnisse rekursiv zu kopieren, kann stattdessen der Befehl rcp -r verwendet werden. Wenn die Meldung »permission denied« ausgegeben wird, dann kann das verschiedene Ursachen haben, die anhand der Fehlermeldung nicht zu unterscheiden sind. Prüfen Sie zunächst, ob man sich überhaupt per rsh auf dem entfernten Rechner ohne Passwort anmelden kann. Ich verwende dazu gern den Befehl rsh *host* date. Im nächsten Schritt prüfen Sie, ob Sie die Quelldatei lesen dürfen, anschließend, ob Sie Schreibrechte im Zielverzeichnis haben. Eine von diesen drei Ursachen ist normalerweise der Grund für die oben genannte Meldung.

rcp kopiert stets alle angegebenen Dateien. Für einen Abgleich zwischen Verzeichnissen auf verschiedenen Rechnern kann das Programm rdist (siehe den Abschnitt 17.5, »Übertragen von Dateien mit rdist«) verwendet werden. Dieses Programm löscht Dateien, die auf dem Quell-Rechner nicht mehr vorhanden sind,

und überträgt nur die Dateien, die sich geändert haben. Gerade beim Update größerer Verzeichnisbäume, in denen sich nicht sehr viel geändert hat, ist die Netzlast bei der Verwendung von `rcp` wesentlich größer, als sie es tatsächlich sein müsste.

20 Network File System (NFS)

»No File Security« oder »Nightmare File System«

20.1 Allgemeines

Mit dem Network File System (NFS) können verschiedene Rechner Dateien gemeinsam benutzen. Eine häufige Anwendung ist die gemeinsame Verwendung der Home-Bereiche oder Mail-Verzeichnisse der Benutzer auf mehreren Rechnern. Für Benutzer bedeutet es eine erhebliche Vereinfachung ihrer täglichen Arbeit, wenn sie auf allen Rechnern ihre Daten und Konfigurationsdateien verfügbar haben. Ebenfalls sinnvoll ist die einmalige Installation von Programmen in `/usr/local` oder `/usr`. Hier bieten sich insbesondere die Verzeichnisbäume großer Pakete wie X11 oder TeX an. Damit wird auch der administrative Aufwand bei Updates wesentlich verringert.

Der Rechner, auf dem die Verzeichnisse lokal verfügbar sind, wird als NFS-Server bezeichnet, der Rechner, der Daten eines entfernten Rechners mitbenutzt, als NFS-Client. Für den Anwender ist es praktisch[1] unerheblich, ob Dateien lokal oder entfernt gespeichert werden, da der entfernte Verzeichnisbaum in den lokalen Verzeichnisbaum integriert wird. Vom NFS-Server exportierte Verzeichnisbäume werden auch als NFS-Volumes bezeichnet.

Das Network File System ist auf der Basis von Remote Procedure Calls (RPCs, siehe auch Abschnitt 15.6, »Remote Procedure Call«) implementiert. Das Protokoll wurde von Sun entwickelt und ist in [rfc3010] veröffentlicht. NFS ist das Standardprotokoll, wenn Verzeichnisse von verschiedenen Unix-Systemen gemeinsam benutzt werden sollen. Im PC-Bereich sind außerdem Novell Netware (IPX, siehe Abschnitt 21.2, »Linux als NetWare-Client und -Server«) und Server Message Block (SMB, CIFS – Common Internet Filesystem, LAN-Manager, Windows, siehe Abschnitt 21.4, »Linux als SMB-Client« und Abschnitt 21.5, »SMB-Server unter Linux«) verbreitet.

Lange Zeit war die NFS-Performance unter Linux schlechter als unter vergleichbaren Systemen. Außerdem wurde lange nur NFSv2 unterstützt und als Transport konnte nur UDP verwendet werden. Heute ist im Kernel auch NFSv3 implementiert, es wird TCP als Transport unterstützt und Sun hat eine NFSv4-Implementation für Linux veröffentlicht.

Heute werden Sie einen NFS-Server meist mit NFSv3 einsetzen. Auf den folgenden Seiten gehen wir auch auf Sicherheitsaspekte beim Einsatz von NFS ein.

1. NFS verhält sich unter manchen Bedingungen anders, als es der POSIX-Standard vorschreibt, besonders auffällig ist das bei Named Pipes.

Weitere Informationen zu NFS finden Sie in der NFS-HowTo, den Manpages der beteiligten Programme, auf der Webseite *http://nfs.sourceforge.net* und in [Stern2001] und [Callaghan2000].

20.2 Linux als NFS-Client

Häufig existieren in lokalen Netzen bereits NFS-Server für die Home-Verzeichnisse der Benutzer oder z. B. zur gemeinsamen Nutzung von CD-Laufwerken. Insbesondere seit viele Betriebssystemhersteller keine gedruckte Dokumentation zu ihren Systemen mehr liefern, ist dies oft notwendig. In diesen Fällen wird man Linux zunächst als NFS-Client konfigurieren. Folgende Bedingungen müssen dafür erfüllt sein:

- Das Netz muss konfiguriert sein, d. h., ein `ping` vom NFS-Client zum NFS-Server und umgekehrt muss möglich sein. Aufgrund der Verwendung von Remote Procedure Calls (RPC) funktioniert NFS nicht oder nur unter großen Schwierigkeiten über eine Firewall oder einen Packet-Filter.
- Das Dateisystem NFS muss im Kernel einkompiliert sein oder als Modul geladen werden. Dies können Sie prüfen, indem Sie sich die Ausgabe des Befehls `cat /proc/filesystems` ansehen. Dort muss `nfs` auftauchen. Andernfalls müssen Sie entweder das Modul `nfs.o` laden oder den Kernel konfigurieren und kompilieren. Wenn Sie `kmod` verwenden, dann wird das Modul automatisch geladen, wenn es benötigt wird.
- Das RPC-System muss auf dem Client und dem Server gestartet sein. Die beim Portmapper registrierten Services können Sie sich mit dem Befehl `rpcinfo -p` *`Rechner`* anzeigen lassen.
- Ein anderer Rechner im Netz muss Daten für diesen Rechner exportieren. Welche Verzeichnisse exportiert werden, können Sie mit dem Befehl `showmount` -e *nfs-server* feststellen.
- Achten Sie darauf, dass die Rechner einheitliche `hosts`-Dateien oder besser DNS verwenden, da die Freigabe von Volumes an Namen (bzw. IP-Adressen) erfolgt. Bei der Freigabe an Rechnernamen verlassen Sie sich auf die Integrität des DNS.

Mit dem Befehl `mount -t nfs` *`nfs-server:NFS-Volume`* `/mnt` kann das vom Server *`nfs-server`* exportierte NFS-Volume unter `/mnt` gemountet werden. Der Typ (`-t nfs`) muss nicht angegeben werden, da das Programm `mount` an dem Doppelpunkt (`:`) erkennt, dass es sich hier um ein NFS-Volume handelt.

Beim Anhängen der Verzeichnisse können einige Optionen angegeben werden, die für die Sicherheit im Netz sinnvoll sind oder beim Ausfall des NFS-Servers weiteres Arbeiten zulassen. Diese Optionen können, wie bei lokalen Dateisyste-

men auch, in der Datei `/etc/fstab` (siehe Listing 20.1) angegeben werden. Die folgende Aufzählung enthält nur die wichtigsten Optionen. Eine vollständige Beschreibung aller Optionen finden Sie in der Manpage `nfs(5)`.

```
nfs-server:/home   /home      nfs rw,auto          0  0
nfs-server:/cdrom  /cdrom     nfs ro,noauto,user   0  0
```

Listing 20.1 Auszug aus der Datei `/etc/fstab`

- Die Option `rsize` bestimmt die Blockgröße für Lese-Zugriffe. Der Standardwert sind derzeit 1024 Byte, andere Systeme unterstützen Blockgrößen von 2048, 4096 und 8192 Byte. Bei Performance-Problemen kann es notwendig (und sinnvoll) sein, diese Größen zu verändern. Oft verwendet man eine Blockgröße von 8192 Byte. Dabei wird allerdings relativ viel Kernel-Speicher benötigt, der Rechner sollte also über einen ausreichend großen Speicher verfügen.
- Die Option `wsize` bestimmt die Blockgröße für Schreibzugriffe. Analog gilt das für die Option `rsize` Gesagte.
- Mit der Option `timeo` wird die Zeit in Zehntelsekunden bis zu einem Timeout und damit einer Neuübertragung der Anfrage eingestellt. Der Standardwert ist 7 (entspricht 0,7 Sekunden). Dies ist ein »Minor-Timeout«.
- Bei der Option `retrans` (Default 3) wird die Anzahl der Neuübertragungen nach einem »Minor-Timeout«, aber vor einem »Major-Timeout« angegeben.
- Es kann bestimmt werden, wie der NFS-Client sich bei einem »Major-Timeout« verhält:
  - `hard` gibt die Meldung »server not responding« aus und wiederholt die Versuche unendlich oft. Diese Einstellung ist der Standardwert. Der NFS-Client setzt seine Arbeit fort, sobald der NFS-Server wieder verfügbar ist.
  - Bei der Option `soft` wird bei einem »Major-Timeout« ein IO-Fehler an das Programm gemeldet.
  - Die Option `intr` erlaubt bei `hard` angehängten Volumes das Unterbrechen der Anfrage mit [Strg]-[C]. Das Programm erhält den Fehlercode `EINTR`.

20.3 Linux als NFS-Server

Ein Linux-Rechner kann auch als NFS-Server eingesetzt werden. Ein NFS-Server stellt anderen Rechnern Plattenplatz zur Verfügung, der gemeinsam benutzt werden kann. Beim Einsatz von Linux als NFS-Server müssen einige Voraussetzungen erfüllt sein:

- Das Netz muss konfiguriert sein (`ping` vom Server zum Client und umgekehrt).
- Der Portmapper `rpc.portmap` muss gestartet werden.

- Die Dämonen `rpc.mountd` und `rpc.nfsd` müssen gestartet werden.
- Bei der Verwendung von PC/NFS muss der Dämon `rpc.pcnfsd` gestartet werden. Dieser Dämon ist für die Authentifizierung und die Bereitstellung von Druckerdiensten notwendig und wird bei der Einbindung von DOS und Windows-PCs erforderlich.
- Das NFS-Protokoll geht davon aus, dass auf beiden Rechnern die `uid`- und `gid`-Nummern identisch sind. Dies lässt sich durch identische `/etc/passwd`-Dateien oder mittels NIS (siehe Kapitel 23, »Network Information Service«) erreichen. Ist beides nicht möglich (oder sinnvoll), so kann der `rpc.ugidd` verwendet werden.

Welche Verzeichnisbäume anderen Rechnern zur Verfügung gestellt werden, wird in der Datei `/etc/exports` bestimmt. Dort kann der Zugriff auf einzelne Rechner oder Netze eingeschränkt oder nur Lesezugriff gestattet werden. Dazu dienen verschiedene Optionen. Die wichtigsten davon sind:

- Volumes können schreibgeschützt (`ro`, Default) oder beschreibbar (`rw`) exportiert werden.
- Die Option `root_sqash` lenkt die Benutzer-ID `0` auf den Benutzer `nobody` (uid -2) um. Der Standardwert ist `no_root_squash`.
- Die Benutzer-IDs von Server und Client sind gleich `map_identy` (Default) oder werden durch den `rpc.ugidd` zugeordnet (`map_daemon`).

Eine vollständige Liste der möglichen Optionen finden Sie in der Manpage zu `exports(5)` und `nfs(5)`. Listing 20.2 enthält ein Beispiel für die Datei `/etc/exports`. Das Volume `/home` wird an alle Rechner der Domain exportiert. Das Volume `/opt` wird nur an `nfsclient` exportiert. Zugriffe durch den Benutzer `root` werden auf den Benutzer `nobody` umgelenkt. Das Volume `/projekt` kann von allen projektbezogenen Rechnern, deren Name mit `proj` beginnt, verwendet werden. Das Volume `/pub` kann weltweit gemountet werden.

```
/opt        nfsclient(ro,root_squash)
/home       *.jochen.org(rw,root_squash)
/projekt    proj*.jochen.org(rw)
/pub        (ro,insecure,root_squash)
```

Listing 20.2 Auszug aus der Datei `/etc/exports`

Nach Änderungen an der Datei `/etc/exports` muss die Datei `/var/lib/nfs/xtab` aktualisiert werden. Die Aktualisierung erfolgt mit dem Befehl `exportfs` mit der Option `-r` (reexport) oder `-a` (alle Volumes aus der Datei `/etc/exports`). Die Datei `/var/lib/nfs/xtab` wird vom Programm `rpc.mountd` verwendet.

Den Zugriff auf den NFS-Server können Sie mit Hilfe des TCP-Wrappers (Abschnitt 15.3, »Der TCP-Wrapper (`tcpd`)«) weiter einschränken. Dazu können Sie

die zugriffsberechtigten Rechner in die Datei `/etc/hosts.allow` eintragen und für alle anderen den Zugriff auf die Protokolle `portmap`, `lockd`, `mountd`, `rquotad` und `statd` in der Datei `/etc/hosts.deny` unterbinden.

20.4 Strategien zum Einsatz von NFS

Die großen Vorteile von NFS liegen darin, dass Dateien, egal auf welchem Rechner sie tatsächlich gespeichert sind, sich dem Anwender so darstellen, als wären sie lokal verfügbar. Diesen Komfort erkauft man sich mit einigen Problemen (siehe auch den Abschnitt 20.5, »Probleme beim Einsatz von NFS« weiter unten). Eines der größten Probleme ist die Abhängigkeit der Anwendungen und Anwender von der Verfügbarkeit es NFS-Servers. Wenn Sie Verzeichnisse von verschiedenen Systemen aus gemeinsam verwenden wollen, dann sollten Sie darauf achten, dass das Verzeichnis auf allen Systemen unter demselben Namen verfügbar ist. Andernfalls würde man unbedarfte Anwender nur unnötig verwirren – und genau das möchte man ja vermeiden.

Beliebte Verzeichnisse für NFS sind die Home-Verzeichnisse der Benutzer, für einen Pool von Rechnern gemeinsam installierte Programme unter `/usr/local` und andere Verzeichnisse, die gemeinsam benutzt werden können. Ein wesentlicher Vorteil ist, dass Programme nur einmal installiert werden müssen und dann allen Anwendern und Rechnern zur Verfügung stehen. Wenn Sie allerdings verschiedene Rechnersysteme und Betriebssysteme verwenden, dann müssen Sie darauf achten, wie Sie die Verzeichnisse strukturieren. In diesem Fall kann Ihnen zum Beispiel der Automounter weiterhelfen.

20.5 Probleme beim Einsatz von NFS

NFS ist auf der Basis von Remote Procedure Calls (Sun-RPC) implementiert. Dabei wird in der Regel ein zustandsloses Protokoll (UDP) verwendet. Das bedeutet, dass der Server keine Vorgeschichte zu aktuellen Anfragen kennt. Damit sind Caching-Funktionen nur sehr begrenzt zu verwirklichen. Ein Vorteil von NFS ist, dass der Client sicher sein kann, dass die Daten auf der Server-Platte gespeichert sind, wenn der entsprechende Aufruf beendet ist. Mit dem Einsatz von TCP als Transport-Protokoll und dem Einsatz von NFSv3 wird sich die Situation etwas verbessern.

Die gemeinsame Nutzung von Dateien ist nur dann sinnvoll möglich, wenn die Benutzernamen und zugehörigen Benutzernummern (User-IDs) auf den beteiligten Rechnern miteinander übereinstimmen. Wenn Sie erstmals NFS aktivieren, werden Sie feststellen, dass auf verschiedenen Rechnern Benutzer teilweise un-

terschiedliche UIDs haben oder dass eine UID zu verschiedenen Benutzern gehört. Vor dem Einsatz von NFS sollten Sie daher die Passwortdateien bereinigen (und dabei die Dateien den richtigen Benutzern zuordnen).

Erstellen Sie zunächst eine Liste der Dateien, die einem Benutzer gehören, z. B. mit `find / -uid` *`UserID`*. Dann passen Sie die Passwortdatei an und übereignen der neuen Kennung diese Dateien mit `chown`. Die manuelle Synchronisation der Passwortdateien ist aufwändig und fehleranfällig. Sie sollten daher über den Einsatz von NIS nachdenken (siehe auch Kapitel 23, »Network Information Service«). Die Verwendung des `uigdd`-Programms, das anhand der Namen die UserIDs zur Laufzeit abgleicht, ist nur als Notlösung zu betrachten.

Beim Austausch von Daten mittels NFS sollten die internen Uhren der beteiligten Rechner relativ genau synchronisiert werden, um nur schwer nachvollziehbare Probleme zu vermeiden. Beim Anlegen von Dateien erhalten diese den Zeitstempel vom Server. Beim (eventuell direkt folgenden) Lesen dieser Datei wird der Zeitstempel gemäß der lokalen Zeit des Clients interpretiert. Unterscheiden sich beide Zeiten, so kommen Programme wie `make` durcheinander.

Neben diesen Problemen ist es auch für Benutzer weniger verwirrend, wenn sich die internen Uhren nicht unterscheiden. Da insbesondere PC-Uhren sehr ungenau gehen, müssen Sie besondere Maßnahmen ergreifen. Relative Abweichungen können Sie lokal mit dem Programm `adjtime` korrigieren. Zur Synchronisation der Uhren im Netz können Sie verschiedene Methoden einsetzen, die alle ihre Berechtigung haben:

- Sie können einen Rechner mit einer DCF77-Funkuhr versehen und diesen Rechner als Zeitserver verwenden. Dafür ist auf dem Server `ntp` zu installieren.
- Wenn im Netz ein Zeitserver existiert, können Sie die Zeit von dort mit dem Programm `ntp` holen. Wenn Sie damit leben können, dass die Zeitserver unter fremder Kontrolle stehen, dann können Sie die Zeit aus dem Internet von verschiedenen Servern beziehen. Im Zweifel sollten Sie Ihren Provider fragen, ob er den Service anbietet. Einige RFCs (z. B. [rfc1305] und [rfc2030]) beschreiben entsprechende Protokolle zur Zeitsynchronisation.
- Ist Ihr Rechner nur sporadisch angebunden, so können Sie das Programm `chrony` verwenden. Wie bei `ntp` wird die Zeit durch eine Geschwindigkeitsänderung der Uhr langsam und ohne Sprünge angepasst – sofern die Abweichung zur Referenzzeit nicht zu groß ist.
- Sie können die Programme `netdate` oder `rdate` verwenden, um die Zeit von einem oder allen Rechnern im Netz zu erhalten. Der Nachteil dieses Verfahrens ist, dass die Zeit in Sprüngen verstellt wird. Dies kann etliche Programme wie `cron` oder Bildschirmschoner verwirren.

Auf Dauer die beste Lösung ist aus meiner Sicht der Einsatz von `ntp` oder `chrony`. Denn nach der Installation und Konfiguration ist die Zeit dauerhaft korrekt und es sind keine negativen Einflüsse auf andere Programme zu befürchten.

NFS ist bei der Verwaltung von Daten- und Programmdateien für den Benutzer fast vollkommen transparent. Anders sieht es bei Gerätedateien und Named Pipes aus. Hier soll eine Kommunikation mit dem Kernel oder einem Anwendungsprogramm auf der anderen Seite der Pipe aufgebaut werden. Beides ist über NFS kaum zu realisieren.

Sie sollten darauf achten, dass der NFS-Server im selben Subnetz und Netzsegment wie die Clients angeschlossen ist. Andernfalls wird durch die doch recht hohe Netzbelastung der Router stark beansprucht und Brücken können keine Lasttrennung zwischen den einzelnen Segmenten durchführen. Beides dient auf Dauer nicht dem sicheren Betrieb des Netzes und damit der Zufriedenheit der Benutzer.

Zurzeit wird das SMB-Protokoll zum *Common Internet File System* (CIFS) weiterentwickelt. An der Entwicklung ist neben Microsoft auch das Samba-Team beteiligt. Die Entwickler versprechen sich von CIFS eine bessere Performance und eine bessere Semantik als bei NFS.

Vielfach wird NFS als großes Sicherheitsproblem betrachtet. Das liegt zum einen an der Verwendung von Remote Procedure Calls, zum anderen an der Implementierung des Protokolls. Verschiedene Dinge sollten Sie vor der Verwendung von NFS (das auch als »Nightmare File System« oder »No File Security« bezeichnet wird) beachten:

- Exportieren Sie nur an bekannte Rechner.
- Exportieren Sie möglichst nur Read-Only.
- Beobachten Sie die Log-Files des Systems.

Sun hat eine Weiterentwicklung des NFS, das so genannte Secure-NFS, entwickelt. Dieses basiert auf einer Erweiterung der RPC-Schnittstelle, den Secure-RPCs. Andere Hersteller haben diese Technologie lizenziert, aber bisher sind nur Implementationen für wenige Plattformen verfügbar. Damit scheidet der Einsatz in heterogenen Netzen derzeit noch aus.

Sun hat die Spezifikation der nächsten NFS-Version (NFSv4, [rfc3010]) freigegeben und eine erste Implementation für Linux unterstützt. Die aktuelle Testversion finden Sie unter *http://www.citi.umich.edu/projects/nfsv4/*. Derzeit sind die Programme noch nicht für den produktiven Einsatz geeignet, aber es ist schön zu sehen, dass die Verfügbarkeit unter Linux für Sun wichtig ist. Mehr Informationen zu NFSv4 finden Sie unter *http://www.nfsv4.org/*.

20.6 Der Automounter amd

In einem lokalen Netz, in dem viele Verzeichnisse zwischen verschiedenen Rechnern mittels NFS hin- und hergemountet werden, ist es oft nicht sinnvoll, dies beim Start des Rechners zu tun. Zum einen belegt jedes gemountete Volume einen von 256 Superblöcken im Linux-Kernel, zum anderen muss für das Einhängen der entsprechende Server verfügbar sein. Ist der NFS-Server im laufenden Betrieb nicht mehr erreichbar, so muss, je nach verwendeten Daten, mit Einschränkungen im Betrieb gerechnet werden. Außerdem werden die `fstab`-Dateien immer länger und müssen bei jeder Änderung auf allen Rechnern angepasst werden.

Diese Probleme bildeten den Hintergrund für die Entwicklung des *Automounters*. Hier werden die entsprechenden Verzeichnisse erst dann in den Verzeichnisbaum eingehängt, wenn sie tatsächlich benötigt werden. Werden die Daten eine Zeit lang nicht mehr benutzt, so wird das Volume wieder abgehängt. Dabei ist der Automounter so flexibel, dass beim Ausfall eines Servers ein anderer Rechner für diesen einspringen kann.

Durch die flexible (und deshalb etwas komplexere) Syntax kann eine Konfigurationsdatei für alle Rechner des lokalen Netzes verwendet werden. Das vermindert deutlich den Administrationsaufwand, insbesondere wenn diese Datei mittels NIS (siehe Kapitel 23, »Network Information Service«) verteilt wird.

20.6.1 Interna des Automounters

Im lokalen Netz erhält jedes exportierte Volume einen eindeutigen Namen. Unter diesem Namen können alle Rechner im lokalen Netz dieses Volume ansprechen. Damit ist es möglich, auf allen Rechnern dasselbe Home-Verzeichnis zu haben. Genauso werden oft die Mail-Verzeichnisse gemountet.

Der Automounter kann aber auch abhängig von der verwendeten Rechnerarchitektur oder dem Betriebssystem unterschiedliche Volumes mounten, die denselben Zweck erfüllen (z. B. der TeX-Pfad für verschiedene Rechnerarchitekturen).

Daneben können aber auch lokale Geräte wie z. B. ein CD-ROM-Laufwerk oder die Diskettenlaufwerke gemountet werden. Dadurch ist es möglich, auf das Programm `usermount` zu verzichten.

20.6.2 Voraussetzungen für den Einsatz des Automounters

Der Automounter ist intern als NFS-Server implementiert. Daher sind wie bei einem NFS-Server oder Client der RPC-Portmapper und die anderen NFS-Dämonen notwendig. Außerdem muss natürlich auch der `amd` gestartet sein. Die Quellen des `amd` enthalten ein Beispielskript (`./text/amd.start.ex`), das den

Dämon mit den richtigen Parametern aufruft. Das Programm amd sollte heute bei praktisch allen Distributionen enthalten sein.

Außerdem müssen die entsprechenden Volumes konfiguriert werden. Dazu dient z. B. die Datei /etc/mount.amd, deren Namen amd als Parameter übergeben werden muss. Das Listing 20.3 zeigt eine einfache Konfiguration des Automounters.

```
/defaults opts:=intr

fd0     host==${host};type:=ufs;\
        dev:=/dev/fd0;\
        opts:=type=msdos,gid=6,uid=0,umask=002,noexec,nodev

dos     host==${host};type:=ufs;dev:=/dev/hdc1;\
        opts:=type=msdos,gid=120,uid=0,umask=002,noexec

cdrom   host==cdserver;type:=ufs;dev:=/dev/sr0;\
        opts:=type=iso9660,ro \
        host!=${host};type:=nfs;rhost:=cdserver;\
        rfs:=/mount/cdrom;opts:=ro,rsize=4096,wsize=4096
```

Listing 20.3 Die Map-Datei für den Automounter amd

Die Volumes fd0 und dos werden nur für den lokalen Gebrauch bereitgestellt. Das Volume cdrom wird auf dem Rechner lokal angehängt. Auf allen anderen Systemen wird das entsprechende Volume des CD-Servers via NFS angesprochen. amd bietet noch viele weitere Funktionen und Konfigurationsoptionen, die in der Info-Dokumentation erläutert werden.

Größere Installationen werden die Automount-Maps mit Hilfe von NIS oder LDAP verwalten. Auf diese Art und Weise kann der Administrator die Konfiguration zentral verwalten und alle Clients greifen stets auf die aktuelle Version zu.

Zum Start enthält das Paket am-utils die Datei cookbook.txt, in der ein »Kochrezept« zur Konfiguration angegeben ist. Für komplexere Umgebungen werden Sie nicht um die Lektüre der mitgelieferten Dokumentation herumkommen.

20.6.3 Das Kernel-autofs unter Linux

Neben dem amd, der sich als User-Level-NFS-Server darstellt, hat Linux seit der Version 2.0 einen internen Automounter implementiert. Zum Einsatz des internen autofs muss das Dateisystem autofs in den Kernel einkompiliert sein oder als Modul zur Verfügung stehen. Außerdem benötigen Sie die Programme aus dem autofs-Paket, das Sie unter *ftp.kernel.org:/pub/linux/daemos/autofs* finden.

Zum Start des `autofs` muss das Programm `automount` gestartet werden. Moderne Distributionen haben hierfür in der Regel entsprechende Start-Stop-Skripte. Wenn Sie `automount` direkt aufrufen müssen, erwartet der Aufruf folgende Parameter:

`automount` *Optionen Verzeichnis Map-Typ*[*,Map-Format*] *Map-Name Map-Optionen*.

Bei einem Aufruf können mehrere Verzeichnisse und Maps angegeben werden, die dann alle von `automount` verwaltet werden.

`Optionen`
: `-p --pid-file, -t --timeout`

`Verzeichnis`
: `automount` überwacht das angegebene Verzeichnis und hängt bei Bedarf die in der Map beschriebenen Volumes ein. Das Verzeichnis muss beim Aufruf von `automount` bereits existieren.

`Map-Typ,Map-Format`
: Maps für `automount` können in verschiedenen Formen gespeichert sein. Hier wird der Typ der Speicherung angegeben. Mögliche Eingaben sind: `file`, `program`, `yp`, `nisplus` oder `hesiod`. Durch ein Komma abgetrennt, kann das Format der Map angegeben werden. Hier wird derzeit nur `sun` unterstützt.

`Map-Name`
: Der Name der Map; bei den Typen `file` und `program` ist das der Dateiname der Map bzw. des Programms. Bei den anderen Typen ist es der Name der Datenbank (also der NIS- bzw. NIS+-Map).

`Map-Optionen`
: Alle weiteren Optionen werden, nachdem das Minuszeichen entfernt wurde, als `-o`-Option an das Programm `mount` übergeben. Es können beliebig viele Optionen angegeben werden.

Bei der Red Hat-Distribution wird der Aufruf von `automount` per Skript aus der Datei `/etc/auto.master` erstellt. Die Kommentare werden entfernt und aus den restlichen Zeilen wird eine Kommandozeile für `automount` generiert. Ein Beispiel für die Datei `auto.master` finden Sie in Listing 20.4. Die Verzeichnisse `/misc` und `/mount` werden mit `automount` verwaltet, für jedes Verzeichnis ist eine eigene Map zuständig. Das Red Hat-Skript erkennt, ob ein Dateiname angegeben ist. In diesem Fall ist die Map vom Typ `file`. Hat die Datei das executable-Bit gesetzt, dann ist die Map vom Typ `program`. Andernfalls wird angenommen, dass die Map vom Typ `yp` ist, also aus den Network Information Service stammt.

```
# Format of this file:
# mountpoint map options
/misc   /etc/auto.misc  --timeout 60
/mount  /etc/auto.mount --timeout 60
```

Listing 20.4 Die Datei `auto.master`

Änderungen an der Datei auto.master werden erst nach einem Neustart von automount aktiv. Änderungen an den Map-Dateien werden beim nächsten Mounten automatisch berücksichtigt. Die Map-Dateien für automount haben folgendes Format:

Schlüssel [*-Optionen*] *Ort*

Der Schlüssel ist der Pfad unterhalb des in der Datei auto.master angegebenen Mount-Point. Sie können nach einem Minuszeichen Optionen angeben, die an das Programm mount als Parameter zur -o-Option weitergereicht werden. Geben Sie mit dem Schlüsselwort -fstype einen Dateisystemtyp an, so wird dieser als Parameter zur Option -t verwendet.

Als Fundort für die Volumes können folgende Einträge verwendet werden: *Host*:*NFS-Volume* für NFS-Verzeichnisse, :*Device* für lokale Geräte und ://*Host*/*Share* für Windows-Shares oder mittels samba exportierte Verzeichnisse. Das Listing 20.5 zeigt, wie auf meinem System die Wechselmedien Disketten und CD-ROMs sowie die DOS-Partition verwaltet werden.

```
# Schlüssel [ -mount-Optionen ] Ort
cdrom      -fstype=iso9660,ro              :/dev/scd0
floppy     -fstype=auto                    :/dev/fd0
dos        -fstype=vfat,gid=100,umask=002  :/dev/sda1
```

Listing 20.5 Die Datei /etc/auto.mount

Das Listing 20.6 zeigt außerdem den Zugriff auf entfernte Verzeichnisse mittels NFS oder Samba. Das dynamische Einhängen dieser Verzeichnisse sorgt dafür, dass der Rechner bei Netzwerkproblemen nur dann hängt, wenn das NFS-Dateisystem aktuell verwendet wird. automount ermöglicht einen stabilen und komfortablen Betrieb beim Mounten von NFS-Volumes, wie es auch amd tut.

```
# Schlüssel [ -mount-Optionen ] Ort
kernel     -ro                  ftp.kernel.org:/pub/linux
dokument   -fstype=smbfs        ://dokument/dokument
```

Listing 20.6 Die Datei auto.mount

Wenn das Programm automount das Signal USR1 (10) erhält, wird versucht, die eingehängten Volumes wieder zu lösen, das Programm läuft aber weiter. Bei dem Signal USR2 wird ebenfalls versucht, alle Volumes freizugeben. Wenn dies gelingt, beendet sich automount. Dateisysteme, die aktuell verwendet werden (d. h. busy sind), können nicht freigegeben werden.

Modifizierte Maps werden bei der nächsten Verwendung automatisch neu geladen. Es ist daher nicht notwendig, automount zu diesem Zweck ein Signal zu

schicken. Unter Red Hat muss nach der Veränderung der Datei `/etc/auto.master` der Dämon neu gestartet werden, da diese Datei bereits vom Start-Stop-Skript interpretiert wird und nicht von `automount`.

`autofs` ist relativ neu und immer noch in der Entwicklung. Daher sind verschiedene Features, die man vielleicht vom Sun `automount` her kennt, noch nicht implementiert.

20.6.4 amd oder automount?

Die Entscheidung für einen der beiden Dienste ist, je nach Situation, nicht ganz einfach. Zunächst sollten Sie sich fragen, ob Sie die Funktionen überhaupt brauchen. Wenn Sie keine Volumes von anderen NFS-Servern verwenden, dann können Sie zunächst auf beide Dämonen verzichten. Auf der anderen Seite können Sie beide Programme dazu verwenden, Disketten oder CD-ROMs zu mounten, wenn diese benötigt werden (siehe Listing 20.3 und Listing 20.5); nur für diesen Zweck ist es praktisch egal, welchen Dienst Sie verwenden.

Wenn Sie in einem größeren Netz arbeiten und dort viele NFS-Volumes verwalten, dann ist der Einsatz von `amd` oder `autofs` wirklich sinnvoll. Wenn Sie Features benötigen, die ein Dienst enthält, der andere aber nicht, dann ist die Wahl einfach. Haben beide Programme die Funktionen, die Sie benötigen, dann müssen Sie andere Kriterien heranziehen.

Wenn Sie `amd` auf anderen Plattformen einsetzen, dann verwenden Sie ihn am besten auch unter Linux. Die einheitliche Umgebung spricht dafür. Wenn Sie bisher mit `automount` gearbeitet haben und Sie mit den Einschränkungen von `autofs` leben können, dann sollten Sie dabei bleiben. Beachten Sie, dass Maps derzeit nicht vollständig zwischen Sun und Linux kompatibel sind.

In Zukunft könnte `amd` auch unter Linux das `autofs` unterstützen, wie das bereits für Solaris der Fall ist. Dann hätte man die technischen Vorteile des `autofs` mit der Stabilität, Flexibilität und Verbreitung des `amd` kombiniert.

20.7 Andere verteilte Dateisysteme

NFS ist im Unix-Umfeld das verbreitetste verteilte Dateisystem. Daneben gibt es aber noch einige andere, recht interessante Dateisysteme.

CODA
: CODA ist ein Ersatz für NFS, der im Linux-Kernel implementiert ist. Wesentliche Vorteile sind die Möglichkeit zur Server-Replikation und der Betrieb ohne Verbindung zum CODA-Server, zum Beispiel mit einem Laptop. Mehr zu

CODA finden Sie unter *http://www.coda.cs.cmu.edu/*. Leider ist CODA nicht besonders weit verbreitet, es ist für Linux, FreeBSD und NetBSD und Windows verfügbar.

InterMezzo

Ein relativ neues, noch experimentelles Dateisystem ist InterMezzo. InterMezzo kann zur Replikation von Daten und auf Laptops ohne Verbindung zum Server eingesetzt werden. Mehr Informationen finden Sie unter *http://www.inter-mezzo.org/*.

OpenAFS

OpenAFS ist aus dem Andrew File System der Carnegie Mellon University entstanden und wurde von IBM übernommen und schließlich freigegeben. Sie erhalten OpenAFS auf der Webseite *http://www.openafs.org/*.

Bei OpenAFS können die Daten auf vielen verschiedenen Servern gespeichert sein. Die AFS-Clients nutzen die Dienste der Server und ermöglichen den transparenten Zugriff auf die Daten. Eine AFS-Zelle ist eine organisatorische Einheit, in der der Administrator die Verteilung der Daten unabhängig von anderen Zellen organisieren kann. Aus den Zellen lässt sich ein globaler Namespace generieren, so dass der Zugriff auf über Zellengrenzen möglich ist.

Für weiträumig verteilte Dateisysteme ist es sinnvoll, häufig benötigte Dateien lokal zu cachen. Außerdem kann OpenAFS lokale Repliken der Dateien verwalten.

AFS ist nicht im Linux-Kernel enthalten, sondern muss vom Systemadministrator lokal installiert werden.

OpenGFS

OpenGFS ist entstanden, als Sistina seine GFS-Implementation nicht mehr frei zur Verfügung stellte. Die Webseite ist *http://www.opengfs.org/*.

OpenGFS ermöglicht den parallelen Zugriff von mehreren Servern auf einen gemeinsam genutzten Speicherbereich. Ein Einsatz ist z. B. eine Webserverfarm, die denselben Datenbestand verwenden soll. Auch wenn ein Server ausfällt, so haben die anderen weiterhin Zugriff auf die Daten.

21 Linux im heterogenen Netz

21.1 Andere Netzwerkprotokolle als TCP/IP

Die bisher angesprochenen, auf TCP/IP basierenden Protokolle und Anwendungen sind Standard für Unix-Systeme aller Hersteller. In Unix-Netzen kommt man in der Regel mit diesen Protokollen aus. Anders ist es, wenn ein Unix-System in ein bestehendes lokales Netz aus PCs und Macintosh-Systemen eingebunden werden soll. Oft herrscht dort ein regelrechter Wildwuchs an Protokollen und jeder Rechner soll nach Möglichkeit ohne großen Aufwand mit jedem anderen direkt kommunizieren können.

Neben dem Protokoll TCP/IP spricht Linux eine Reihe von weiteren Sprachen, die in lokalen Netzen weit verbreitet sind. Damit erweist sich Linux ein weiteres Mal als preiswerte und leistungsfähige Lösung zur Verbindung verschiedener Netzwelten. Vergleichbare kommerzielle Pakete sind für kleinere Firmen oder Privatanwender oft unerschwinglich teuer.

Ein in PC-Netzen weit verbreitetes Protokoll ist das IPX-Protokoll (Internet Packet Exchange) von Novell NetWare. Als Server werden in der Regel gut ausgestattete PCs verwendet. Die Geschwindigkeit ist teilweise etwas höher als bei TCP/IP, daher wird dieses Protokoll gern in reinen DOS/Windows-Netzen eingesetzt. Es gibt Clients für DOS, Windows, OS/2 und Macintosh entweder direkt beim Betriebssystem oder von Novell.

Novell bietet auch ein TCP/IP-Paket für seine Server an, aber die Funktionalität eines Unix-Hosts wird dabei nicht erreicht. Darüber hinaus ist dieses Paket, das auch NFS enthält, relativ teuer. Mit aktuellen Versionen vollzieht auch Novell langsam eine Öffnung in Richtung TCP/IP-basiertes Networking.

Ein anderes Protokoll ist Server Message Block (SMB). Dieses Protokoll wird vom LAN-Manager (von IBM und Microsoft) sowie von Windows NT, Windows for Workgroups und Windows 95/98 verwendet. Unter Linux wird das Paket `samba` als Server verwendet, das auch auf vielen anderen Unix-Plattformen eingesetzt werden kann. Als Client kann das `smbfs`-Dateisystem zum Zugriff auf Windows-Shares verwendet werden. Freigegebene Drucker können mit dem Programm `smbclient` aus dem `samba`-Paket angesprochen werden.

Für die Kommunikation mit Macintosh-Rechnern von Apple wird das Apple Talk-Protokoll eingesetzt. Auch hier gibt es eine Implementierung unter Linux. Damit ist Linux ein sehr offenes und kommunikatives Betriebssystem. Das liegt auch daran, dass praktisch der gesamte Quellcode verfügbar ist und jeder entsprechende Änderungen und Anpassungen vornehmen kann.

21.2 Linux als NetWare-Client und -Server

In lokalen PC-Netzen ist Novell NetWare weit verbreitet. Die Implementation ist stabil und teilweise etwas schneller als die TCP/IP-Protokolle. Vor einigen Jahren, als PCs noch wesentlich weniger leistungsfähig waren, war dies oft die einzig sinnvolle Vernetzungsmöglichkeit. Heute gewinnt, auch aufgrund der Internet- und Intranetstrategien, TCP/IP immer mehr an Bedeutung.

Als unterste Protokollschicht wird IPX (Internet Packet Exchange) verwendet. Dieses Protokoll ist dokumentiert und bereits seit einiger Zeit im Linux-Kernel implementiert. Daher können Sie Linux u. a. als IPX-Router verwenden oder im `dosemu` auf NetWare Services zugreifen.

Die NetWare-Services, also Datei- und Druckdienste, basieren jedoch auf dem auf IPX aufsetzenden Protokoll NCP (NetWare Core Protocol), das von Novell nicht offiziell dokumentiert war. Die Dokumentation steht jetzt (zumindest teilweise) zur Verfügung, so dass Linux als Client und als Server in NetWare-Netzen eingesetzt werden kann. Dabei wird nur die Bindery von NetWare 3.x emuliert. Ein Zugriff auf die NetWare Directory Services (NDS) ist mit der von Novell zur Verfügung gestellten Implementation möglich.

Zunächst muss das IPX-Protokoll in den Kernel einkompiliert werden. Die Implementation wurde u. a. von Caldera durchgeführt. Nach einem Reboot meldet der Kernel die Existenz dieses Protokolls mit Meldungen wie in Listing 21.1.

```
Swansea University Computer Society IPX 0.34 for NET3.035
IPX Portions Copyright (c) 1995 Caldera, Inc.
```

Listing 21.1 Kernel-Meldungen beim Laden der IPX-Protokolle

Anschließend muss, ähnlich wie bei TCP/IP-Netzen auch, die Schnittstelle konfiguriert werden. Dies geschieht zurzeit noch mit einem eigenständigen Programm. Diese Funktionen könnten in der Zukunft in das Programm `ifconfig` integriert werden. Sie können die Schnittstelle automatisch konfigurieren lassen, indem Sie den Befehl `ipx_configure --auto_primary=on --auto_interface=on` verwenden. Damit wird automatisch ein primäres IPX-Interface erzeugt, wenn ein IPX-Paket eintrifft.

Eine automatische Erkennung ist nur dann möglich, wenn Sie Linux nur als Client im Netz betreiben wollen. Andernfalls benötigen Sie eine interne IPX-Netzwerknummer, die von Ihrem Netzwerkverwalter vergeben wird. Wenn Sie im Netz Rechner mit Windows 95/98 als Betriebssystem haben, so sollten Sie ebenfalls auf eine automatische Erkennung verzichten. Diese Rechner senden aufgrund eines Fehlers Pakete mit einem anderen Frame-Typ als sonst im Netz üblich, was einen sicheren Betrieb mit der automatischen Erkennung verhindert.

Wenn die automatische Erstellung nicht möglich ist, können Sie mit dem Befehl `ipx_interface` das Interface manuell erstellen. Dabei müssen Sie das Netzwerkgerät (z. B. `eth0`), den Frame-Typ und die Netzwerknummer angeben. Den Frame-Typ und die IPX-Netzwerknummer erfahren Sie von Ihrem Netzwerkadministrator. Tabelle 21.1 enthält eine Übersicht über die geläufigen Frame-Typen und deren übliche Verwendung.

Frame-Typ	Verwendung
802.2	Ethernet 802.2
802.3	Ethernet 802.3
ether_ii	Das Ethernet-II-Protokoll
802.2TR	Token-Ring Encapsulation

Tabelle 21.1 IPX-Frame-Typen und ihre Verwendung

Heute werden die meisten (neuen) Netze mit dem Frame-Typ 802.3 betrieben. Dies ist auch der Standardwert für viele Produkte. In bestehenden Installationen werden aber auch noch die Typen 802.2 und Ethernet-II verwendet. Wenn Sie ein Token-Ring-Netz haben, dann müssen Sie den Frame-Typ 802.2TR einsetzen.

Der Befehl in Listing 21.2 erstellt ein IPX-Interface für den Frame-Typ 802.2. Die Netzwerknummer ist 2A. Wenn Sie eine Null (`0`) angeben, dann versucht der Kernel, die Netzwerknummer aus den ankommenden IPX-Paketen auszulesen. Beachten Sie, dass Sie bei Kollisionen oder Fehlern in der Vergabe der IPX-Netzwerknummern den Netzbetrieb empfindlich stören können.

```
(linux):~# ipx_interface add -p eth0 802.2 2A
```

Listing 21.2 Erstellen eines IPX-Interface unter Linux

Informationen über den aktuellen Status des IPX-Subsystems finden Sie im `/proc`-Dateisystem in den Dateien `ipx_interface` (siehe Listing 21.3), `ipx` (siehe Listing 21.4) und `ipx_route` (siehe Listing 21.5) im Verzeichnis `/proc/net`.

Die Datei `/proc/net/ipx_interface` enthält für jedes konfigurierte Interface eine Zeile. Das Listing 21.3 stammt von einem Linux-Rechner, der sowohl NetWare-Server (via `mars_nwe`) als auch NetWare-Client via `ncpfs` ist.

```
Network    Node_Address   Primary  Device     Frame_Type
00000030   10005AD2813C   No       tr0        802.2TR
10000106   000000000001   Yes      Internal   None
```

Listing 21.3 Die Datei `/proc/net/ipx_interface`

Die externe Netzwerkschnittstelle `tr0`, eine Token-Ring-Netzwerkkarte, ist an das Netzwerk `30` angeschlossen; es wird der Frame-Typ `802.2TR` verwendet. Die Node-Adresse ist die Hardware-Adresse der Netzwerkkarte, die vom Kernel automatisch eingesetzt wird. Das Interface ist nicht primär.

Jeder NetWare-Server hat ein eigenes (internes) Netzwerk. Die hierfür vergebene Nummer muss eindeutig sein und sollte daher zentral verwaltet werden. Wenn Sie sich nicht sicher sind, fragen Sie Ihren Netzwerkverwalter. Der Server hat in diesem Netz die Node-Adresse `1`. Im Listing 21.3 ist das interne Interface auch das primäre.

Das Listing 21.4 zeigt ein Beispiel für die Datei `/proc/net/ipx`. Sobald Ihr Rechner am IPX-Netz teilnimmt, füllt sich diese Datei automatisch.

```
Local_Address  Remote_Address Tx_Queue  Rx_Queue  State Uid
10000106:4000  Not_Connected  00000000  00000000  07    000
10000106:0452  Not_Connected  00000000  00000000  07    000
10000106:0453  Not_Connected  00000000  00000000  07    000
10000106:4001  Not_Connected  00000000  00000000  07    000
10000106:4002  Not_Connected  00000000  00000000  07    000
10000106:0451  Not_Connected  00000000  00000000  07    000
10000106:4003  Not_Connected  00000000  00000000  07    000
10000106:4004  Not_Connected  00000000  00000000  07    000
```

Listing 21.4 Die Datei `/proc/net/ipx`

Im Listing 21.5 finden Sie einen Auszug aus der Datei `/proc/net/ipx_route`. In dieser Datei wird die Routing-Tabelle für IPX-Pakete aufbereitet dargestellt. Für jedes erreichbare Netz wird angegeben, über welches Netz geroutet wird und welcher Rechner als Router fungiert. Da alle IPX-Server in der Regel auch als Router fungieren und diese Informationen regelmäßig auf dem Netz veröffentlichen, füllt sich diese Tabelle recht schnell. Wenn Sie viele NetWare-Server haben, dann wird diese (Pseudo-)Datei viele Einträge enthalten.

```
Network    Router_Net   Router_Node
10000411   00000030     00001D17C1F9
25031995   00000030     00805F189164
...
00000030   Directly     Connected
10000106   Directly     Connected
```

Listing 21.5 Die Datei `/proc/net/ipx_route`

Nach Gebrauch kann das Interface mit dem Programm `ipx_interface` wieder gelöscht werden (siehe Listing 21.6). Auch hierbei muss wieder der Frame-Typ angegeben werden. Normalerweise wird das Interface von den Startup-Skripten

Ihrer Distribution eingerichtet und beim Shutdown wieder entfernt. Wenn Sie jedoch die verschiedenen Funktionen testen, kann es sinnvoll sein, diese Funktionen von Hand aufzurufen.

```
(linux):~# ipx_interface del eth0 802.2
```

Listing 21.6 Löschen eines IPX-Interface

Neuere Distributionen sind bereits auf die Verwendung als NetWare-Client (und -Server) vorbereitet, so dass die entsprechenden Start-Stop-Skripten bereits zur Verfügung stehen. In den folgenden Abschnitten geht es um die Konfiguration der entsprechenden Anwendungen.

21.2.1 Linux als NetWare-Client

Mit dem Paket `ncpfs` existiert eine Implementierung des NCP, die als Modul in den Kernel geladen werden kann. Wie der Name bereits andeutet, ist es (unter anderem) ein Dateisystem, das auf dem NetWare Core Protocol basiert. Daneben werden auch Druckdienste angeboten und genutzt.

Der Quellcode des `ncpfs` selbst ist im Standard-Kernel enthalten und kann bei der Kernel-Konfiguration angewählt werden. Zusätzlich benötigen Sie das Programm `ncpmount`, mit dem Sie von NetWare-Servern exportierte Volumes in ein Linux-System einhängen können. Da sich die Novell-Volumes deutlich von den unter Linux sonst üblichen Dateisystemen unterscheiden, wurde diese Funktion bisher nicht in das normale `mount`-Programm integriert.

Das `ncpfs`-Paket (*ftp://platan.vc.cvut.cz/pub/linux/ncpfs*) wurde von Volker Lendecke entwickelt, der auch das `smbfs` zum Zugriff auf Windows-Shares implementiert hat. Mehr zu diesem Paket finden Sie im Abschnitt 21.4, »Linux als SMB-Client«.

In der aktuellen Implementierung können neben Platten auch Drucker an NetWare-Servern von Linux mit genutzt werden. Das entsprechende Dateisystem (`ncpfs`) ist in den Standard-Kernel integriert. Sie müssen nur das IPX-Protokoll und dieses Dateisystem bei der Kernel-Konfiguration anwählen.

Nachdem die Netzwerk-Interfaces, wie im letzten Abschnitt beschrieben, aktiviert sind, können Sie mit `slist` eine Liste der aktiven NetWare-Server erhalten. Wenn dieser Befehl eine Fehlermeldung liefert, überprüfen Sie die Kernel- und Netzwerkkonfiguration. Überprüfen Sie die Boot-Meldungen des Kernels auf die Existenz des IPX-Protokolls (mit dem Befehl `dmesg`; oft werden diese Meldungen auch in der Datei `/var/log/kernel` gespeichert). Zum Schluss werfen Sie einen Blick in die Dateien `/proc/net/ipx*`.

Anschließend können Sie mit dem Befehl `ncpmount` (siehe Listing 21.7) ein Volume eines NetWare-Servers mounten. Dazu benötigen Sie den Namen des Servers (siehe die Ausgabe von `slist`), eine gültige Benutzerkennung auf diesem Server (z. B. `guest`) und eventuell das zugehörige Passwort. Wenn kein Passwort notwendig ist, geben Sie die Option `-n` an, so dass keine Benutzerinteraktion notwendig ist. Ist ein Passwort erforderlich, können Sie dieses nach der Option `-P` angeben (es ist dann allerdings in der Prozessliste für alle Benutzer lesbar), oder Sie werden interaktiv danach gefragt.

```
(linux):~# ncpmount -S Server /mnt -U guest -n
```

Listing 21.7 Einhängen eines NetWare-Volume unter Linux

Praktisch alle Programme des `ncpfs`-Pakets benötigen die Informationen zur Anmeldung an einen NetWare-Server. Daher sind die Optionen `-S`, gefolgt von einem Server-Namen, und `-U`, gefolgt von einem Benutzernamen, in praktisch allen Programmen des `ncpfs`-Pakets implementiert.

Sie können einige Einstellungen sowie die NetWare-Benutzerkennung und das NetWare-Passwort in der Datei `~/.nwclient` hinterlegen (siehe Listing 21.8). Diese Datei darf aus Sicherheitsgründen nur für den Eigentümer lesbar sein. Dies wird von den Programmen des `ncpfs`-Pakets überprüft.

Ein Eintrag in der Datei besteht aus zwei oder drei Teilen, dem Namen des Servers, der Benutzerkennung und eventuell dem Passwort. Kommentarzeilen beginnen mit einer Raute (`#`) in der ersten Spalte. Das Format von Servername und Benutzerkennung orientiert sich dabei an den von NetWare gewohnten Konventionen.

```
# Der "preferred Server"
NW311/HEIN
# Ein Gast-Zugang mit Passwort
FS-DEMO/GUEST GUEST
# Ein Zugang ohne Passwort
FS-DEMO/GAST -
```

Listing 21.8 Ein Beispiel für die Datei `~/.nwclient`

Der erste Eintrag in der Datei hat eine spezielle Bedeutung. Er beschreibt den »preferred Server«. Wenn bei einem Aufruf eines Programms des `ncpfs`-Pakets kein Server-Name angegeben wird, dann wird dieser Eintrag verwendet. Beim Aufruf `nwuserlist` meldet sich das Programm beim Server `NW311` als NetWare-Benutzer `HEIN` an und fragt nach dem Passwort.

Beim zweiten Eintrag ist als Passwort `GUEST` eingetragen. Damit kann sich das Programm ohne Interaktion am NetWare-Server anmelden. Bei dem Benutzer im dritten Eintrag ist im NetWare-Server kein Passwort eingetragen, dies wird durch das Minuszeichen (`-`) dargestellt. In diesem Paket sind neben vielen ande-

ren von NetWare bekannten Tools auch Programme (`pqlist` und `nprint`) enthalten, mit denen von Linux aus auf an das Novell-Netz angeschlossene Drucker zugegriffen werden kann. Es ist auch ein einfacher Print-Server (`pserver`) enthalten. Hilfreich für viele Administratoren kann der Zugriff auf Objekte aus der Bindery mit Hilfe von Kommandozeilen-Tools sein. Die Novell-Programme sind zwar bunt und relativ benutzerfreundlich, aber nicht für Massenänderungen oder Skriptsteuerung geeignet.

Neben dem Zugriff auf Volumes und Drucker ist in `ncpfs` auch ein PAM-Modul enthalten, so dass auch Linux-Benutzer gegen einen Novell-Server authentifiziert werden können. Die kommerzielle Implementation der NDS von Novell ermöglicht auch ein Single-Sign-On. Die Zukunft von IPX und Netware/NDS ist zurzeit eher trübe – die meisten Netze verwenden heute TCP/IP und andere Verzeichnis-Dienste (wie zum Beispiel LDAP).

21.2.2 Linux als NetWare-Server

Neben der Implementierung eines NetWare-Clients stehen zwei Implementierungen eines NetWare-Servers zur Verfügung (`mars_new`, *http://www.compu-art.de/download/mars_nwe.html*) und Linware, *http://knihovny.cvut.cz/ftp/pub/linux/lwared-0.95/*). Beide Server bieten zurzeit die Möglichkeit, Volumes für NetWare-Clients zu exportieren. Beide Server werden scheinbar nicht mehr weiterentwickelt, in der Regel setzt man heute `samba` als Server für Windows-Clients ein.

21.3 Linux als SMB-Client und -Server

Mit der Implementierung des SMB-Protokolls (Server Message Block) in Windows for Workgroups und dessen weiterer Verbreitung wurde es Benutzern sehr leicht gemacht, ihre Rechner zu vernetzen. Man benötigt nicht mehr als eine Netzwerkkarte für jeden Rechner und kann loslegen. Besonders für Privatanwender und kleine Firmen ist es günstig, dass kein spezieller Server angeschafft werden muss, um Festplatten und Drucker gemeinsam benutzen zu können.

Beim intensiven Einsatz kehrt sich die Situation jedoch um. Wird ein Rechner von vielen Benutzern als File- und Print-Server verwendet, so ist ein interaktives Arbeiten an diesem Rechner nicht mehr sinnvoll möglich. Da z. B. Windows für Workgroups nicht als Server-Betriebssystem konzipiert ist, hat man Probleme bei der Wartung des Rechners, der jetzt möglicherweise in einem abgeschlossenen Raum steht.

Das SMB-Protokoll wird von vielen Produkten verwendet. Bekannt sind neben der Windows-Familie der LAN-Manager (von Microsoft und IBM) und Pathworks (von DEC). Es wurde im Laufe der Zeit zum Common Internet File System (CIFS, *http://www.cifs.org/*) weiterentwickelt.

21.4 Linux als SMB-Client

Existiert in Ihrem Netz bereits eine Arbeitsgruppe oder Domäne, so können Sie mit Linux an dieser Gruppe teilnehmen. Das hierfür notwendige `smbfs`-Dateisystem ist im Kernel implementiert. Sie benötigen zusätzlich das Programm `smbmount`, das die Einbindung der exportierten Volumes in das Linux-Dateisystem vornimmt. Wenn Sie den `kerneld` verwenden, um dieses Dateisystem zu laden, müssen Sie den Aufruf zum Laden des Moduls im Programm `smbmount` selbst deaktivieren.

Für die Einrichtung des `smbfs` benötigen Sie `root`-Rechte. Auch ist dieses Dateisystem nicht auf andere Systeme portierbar. Dafür werden die Shares in den normalen Linux-Verzeichnisbaum integriert. Für den Benutzer ist es praktisch unerheblich, ob die Dateien lokal, auf einem NFS-Server oder einem SMB-Server gespeichert sind.

Das Paket `samba` enthält einen SMB-Client (`smbclient`), der ähnlich wie das Programm `ftp` bedient wird. Dieses Programm kann ohne spezielle Rechte verwendet werden; Hilfe erhalten Sie mit dem Befehl `help` innerhalb des Programms. Auf anderen Unix-Systemen ist dies eine einfache Möglichkeit, auf SMB-Shares zuzugreifen.

Die Syntax der Share-Namen ist genauso, wie unter Windows gewohnt: erst zwei Backslashes, dann der Rechnername und, durch einen weiteren Backslash getrennt, der Share-Name. Beachten Sie jedoch, dass Sie die Shell-Sonderzeichen (Backslash) entwerten müssen (siehe Listing 21.9). Alternativ können Sie den Backlash durch einen normalen Schrägstrich ersetzen.

```
(linux):~> smbclient -L jupiter
(linux):~> smbclient '\\jupiter\jochen'
```

Listing 21.9 Die Verwendung von `smbclient`

Mit der Option `-L` erhalten Sie eine Liste der auf diesem Server verfügbaren Shares (Platten und Drucker). Zugriff auf diese Shares erhalten Sie, wenn Sie wie im zweiten Beispiel den Namen des Share angeben.

Diese Methode ist für praktisch jedes Unix-System verfügbar. Unter Linux jedoch kann ein SMB-Share direkt in den Verzeichnisbaum eingefügt werden. Dazu muss das `smbfs` in den Kernel einkompiliert sein oder als Modul geladen

werden und das Programm `smbmount` installiert sein. Ein Beispiel für das Einhängen eines Share finden Sie in Listing 21.10.

```
(linux):~> /usr/sbin/smbmount //jupiter/jochen mnt -C
```

Listing 21.10 Einhängen eines SMB-Share

Beim Programm `smbmount` werden als Trennzeichen innerhalb des Share-Namens normale Schrägstriche anstelle der Backslashes verwendet. Damit braucht der Benutzer nicht an Quotes oder andere Sonderzeichen zu denken. Als erster Parameter wird der Share-Name erwartet, als zweiter der Mount-Point. Wenn das Programm `smbmount` `setuid-root` installiert ist, dann kann jeder Benutzer dieses Programm verwenden.

Das Programm `smbmount` kennt eine Reihe von Optionen, mit denen das Verhalten des Programms beeinflusst werden kann. Wenn Sie Probleme mit der Anmeldung haben, dann können Sie die Option `-C` versuchen, mit der verhindert wird, dass das Passwort zunächst in Großbuchstaben übersetzt wird, bevor es an den Server übertragen wird.

21.5 SMB-Server unter Linux

Mit `samba` steht ein leistungsfähiger SMB-Server für Unix-Systeme zur Verfügung. Das Programm kann ohne Probleme auch unter Linux übersetzt und installiert werden.

Sie können die Dämonen `smbd` und `nmbd` entweder direkt (z. B. in `rc.local`) oder mit Hilfe des `inetd`-Servers starten. Nehmen Sie dazu die folgenden zwei Zeilen, angepasst an die im `Makefile` angegebenen Pfade, in die Datei `/etc/inetd.conf` auf (siehe Listing 21.11) und starten Sie den `inetd`-Server neu. Dies ist zum Test sicher sinnvoll, da für jede neue Verbindung ein neuer `samba`-Prozess gestartet wird, der die Konfiguration neu liest.

```
netbios-ssn stream tcp nowait root /usr/local/bin/smbd smbd
netbios-ns dgram udp wait root /usr/local/bin/nmbd nmbd
```

Listing 21.11 Neue Einträge in der Datei `/etc/inetd.conf`

Anschließend muss der Server mit der Datei `smb.conf` konfiguriert werden. Sie können Verzeichnisse und Drucker zur Verwendung durch Clients freigeben. Im Verzeichnis `./example` finden Sie einige Beispiele für die Konfiguration des Servers. Zum Schluss sollten Sie die Syntax der Konfigurationsdatei mit dem Programm `testparm` überprüfen.

Viele Distributionen enthalten dieses Paket bereits vorkonfiguriert. Aus Geschwindigkeitsgründen werden die Dämonen in der Regel nicht vom `inetd` gestartet, sondern in den Startup-Skripten.

In der Praxis werden Sie vermutlich zunächst die Home-Bereiche Ihrer Benutzer exportieren. Dazu benötigen Sie den im Listing 21.12 gezeigten Eintrag in der Datei `smb.conf`. Einen ersten Test können Sie mit dem Programm `smbclient` lokal unter Unix durchführen. Die möglichen Einstellungen sind in der Manpage zu `smb.conf` ausführlich dokumentiert.

```
comment = Home Directories
browseable = no
read only = no
create mode = 0750
```

Listing 21.12 Exportieren der Home-Verzeichnisse

Einige Clients senden das Passwort stets in Großbuchstaben zum Server. Unter Unix ist Groß- und Kleinschreibung in Passwörtern aber signifikant. `samba` wandelt das Passwort in Kleinbuchstaben um und versucht die Anmeldung. Enthält das Passwort einen oder mehrere Großbuchstaben, so schlägt der Versuch fehl. In der Datei `smb.conf` können Sie im Abschnitt `[global]` mit `password level =` *N* eintragen, dass alle Kombinationen mit bis zu *N* Großbuchstaben im Passwort versucht werden sollen. Dies kann allerdings bei größeren Werten für *N* viel CPU-Zeit verbrauchen.

`samba` lässt sich außerdem als Druck-Server für SMB-Netze einsetzen. Dazu müssen Sie die Sektion `[printers]` in die Datei `smb.conf` aufnehmen. Sie müssen nicht jeden einzelnen Drucker definieren, `samba` exportiert alle Drucker und Aliasnamen aus der Datei `/etc/printcap`. Im Listing 21.13 darf ein Gast nicht drucken und in der Browse-Liste werden die Drucker nicht angezeigt.

```
comment = Alle in der /etc/printcap definierten Drucker
path = /var/spool/samba
browseable = no
printable = yes
public = no
writable = no
create mode = 0700
```

Listing 21.13 Druckerdefinition für `samba`

Sie können aber auch private Drucker für einzelne Benutzer definieren. Ein Beispiel dafür finden Sie in den `samba`-Quellen.

Wenn Sie Windows for Workgroups verwenden, so sollten Sie darauf achten, dass bei der Anmeldung die Box »Passwort speichern« nicht aktiviert ist. Andernfalls kann sich jeder, der Zugang zu Ihrem PC hat, unter Ihrem Namen ohne

Passwort beim Server anmelden. Ich halte diese »Komfortfunktion« für ein Sicherheitsproblem, aber die Anschaffung von Produkten, die ein sicheres Single-Logon im Netzwerk ermöglichen, ist oft nicht durchsetzbar.

Weitere sehr ausführliche Dokumentationen zu samba finden Sie im Verzeichnis ./docs. Die meisten Distributionen speichern diese Dateien auch unter /usr/doc. Viele Fragen werden außerdem in der samba-FAQ beantwortet. Es existieren zudem Mailing-Listen, an denen Sie teilnehmen können. Neben den hier kurz vorgestellten Funktionen kann samba auch als WINS-Server oder -Client verwendet werden oder als Master-Browser für die Domäne. Sie wissen nicht, was das ist? Seien Sie froh darüber. Wenn Sie es dennoch wissen möchten, werfen Sie einen Blick in das Dokumentationsverzeichnis.

21.6 Grafische Konfiguration von samba mit swat

Wenn Sie samba nicht mit Hilfe eines Editors konfigurieren wollen, dann können Sie das Programm swat (Samba Web Administration Tool) verwenden. Das Programm wird normalerweise vom inetd Superserver aus gestartet, dazu muss in der Datei /etc/inetd.conf der Eintrag aus Listing 21.14, vorgenommen werden. Der Port swat ist in der Datei /etc/services als Portnummer 901 festgelegt.

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

Listing 21.14 Der Aufruf von swat *in der Datei* /etc/inetd.conf

Nach dem Neustart des inetd können Sie mit einem beliebigen Web-Browser auf swat zugreifen. Als URL verwenden Sie auf dem lokalen Rechner *http://localhost:901/*. Wenn Sie den Zugriff nicht mit Hilfe des TCP-Wrappers einschränken (z. B. auf Rechner aus dem lokalen Netz), dann können Sie mit dem URL *http://servername:901/* von jedem Rechner im Netz aus auf die Konfiguration zugreifen. Der Zugriff ist natürlich mit Passwort geschützt, verwenden Sie z. B. den Benutzernamen root mit dem zugehörigen Passwort. Die Abbildung 21.1 zeigt swat im Einsatz.

Sie können mit swat jede Option in der Datei smb.conf anpassen, zu jeder Option ist der passende Eintrag in der Manpage nur einen Mausklick entfernt. Besonders wenn Sie samba das erste Mal einsetzen, kann Ihnen dieses Tool die Arbeit sehr erleichtern.

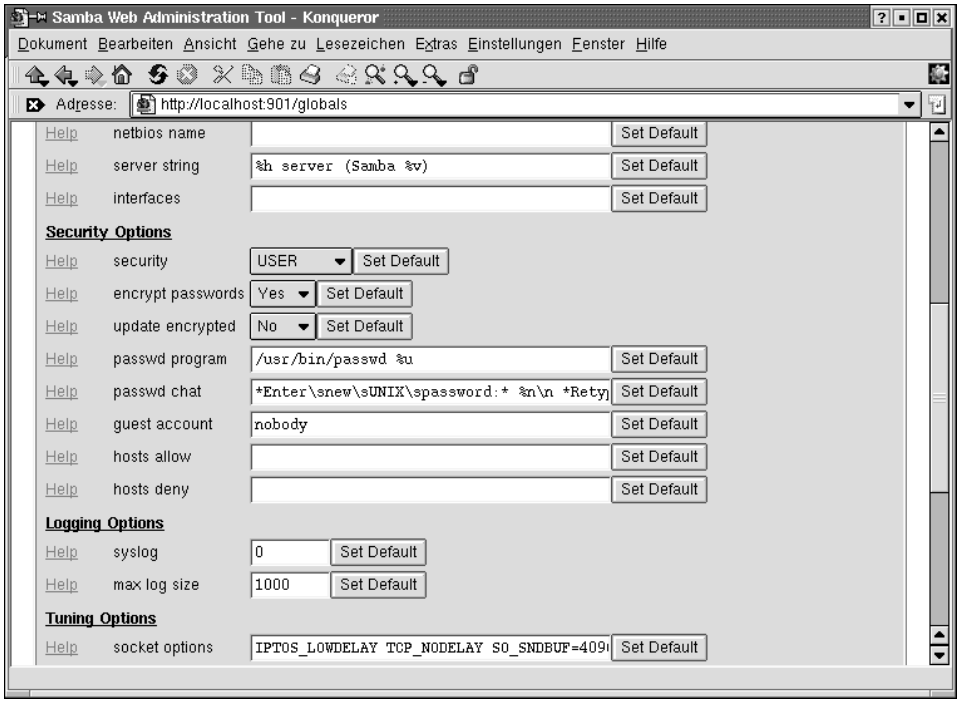


Abbildung 21.1 Das Konfigurationstool `swat` *im Einsatz*

Ich persönlich verwende lieber einen Editor zur Konfiguration, dennoch ist das Stöbern in `swat` und der zugehörigen Dokumentation sehr interessant. Leider generiert `swat` die Konfigurationsdatei `smb.conf` neu, so dass Kommentare gelöscht werden. Wenn Sie also bereits eine eigene, ausgefeilte und kommentierte Konfiguration haben, so sollten Sie diese zumindest sichern, vielleicht verwenden Sie ja bereits ein Werkzeug wie RCS oder CVS.

22 Konfiguration und Betrieb eines Nameservers

22.1 Gründe für den Betrieb eines Nameservers

Der *Domain Name Service* (DNS) ist in [rfc1034], [rfc1035] und [rfc1183] definiert. Er dient dazu, im gesamten Internet eine aktuelle und flexible Datenbasis zu verwalten, die zur Umwandlung von Rechnernamen in IP-Adressen und umgekehrt dient. In der Anfangszeit des Internets wurde an zentraler Stelle die Datei `HOSTS.TXT` gepflegt, die dann von den Administratoren auf die lokalen Rechner kopiert wurde.

Dieses Verfahren ist jedoch nur für relativ kleine Netze praktikabel, so dass ein neuer Service, nämlich der Domain Name Service (DNS), entwickelt wurde. Heute ist die Client-Seite dieses Services Bestandteil aller verbreiteten TCP/IP-Implementationen. Viele Betriebssysteme werden bereits mit den passenden Server-Programmen geliefert oder diese sind separat erhältlich. Das Internet wäre ohne DNS heute gar nicht denkbar, da die Strukturen viel zu groß, komplex und dynamisch geworden sind. DNS ist heute vermutlich die größte, real existierende und funktionierende verteilte Datenbank. Auch lokale Netze ab einer gewissen Größe profitieren vom Einsatz eines Nameservers, auch wenn dieses Netz nicht mit dem Internet verbunden ist. Der Verwaltungsaufwand, der durch viele (verschiedene) `hosts`-Dateien entsteht, und Probleme durch inkonsistente Einträge werden vermieden.

Unter Linux und vielen anderen Unix-Systemen kommt als Nameserver das Paket BIND (Berkeley Internet Name Domain) zum Einsatz. Von dieser Implementierung gibt es auch eine Windows-Portierung.

Ausführliche Informationen zur Konfiguration und zum stabilen Betrieb eines Nameservers finden Sie in den Quellen zu BIND (*http://www.isc.org/products/BIND/*). Weitere Tipps und Tricks sowie Hinweise zu häufig gemachten Fehlern finden Sie in [rfc1536] und [rfc1912]. Das Programm, das die Datenbasis verwaltet und Anfragen der Clients beantwortet, heißt `named`. Auf der Client-Seite wird die Resolver-Bibliothek verwendet, die mit dem Nameserver kommuniziert.

22.2 Das Konzept des Domain Name Service

Die Namensverwaltung im Internet ist hierarchisch aufgebaut. Ganz oben in der Hierarchie stehen die so genannten Root-Nameserver. Diese Rechner sind zunächst für die Auflösung aller Namen zuständig. Bei diesen Rechnern wird ein-

getragen, dass für einzelne Zonen jeweils ein anderer Nameserver zuständig (authority) ist. Man sagt, dass diese Zone delegiert wurde.

Eine Domain, wie man sie aus dem Host-Namen kennt, ist nicht in jedem Fall mit einer DNS-Zone identisch. Ein Beispiel für verschiedene Zonen sehen Sie in Abbildung 22.1.

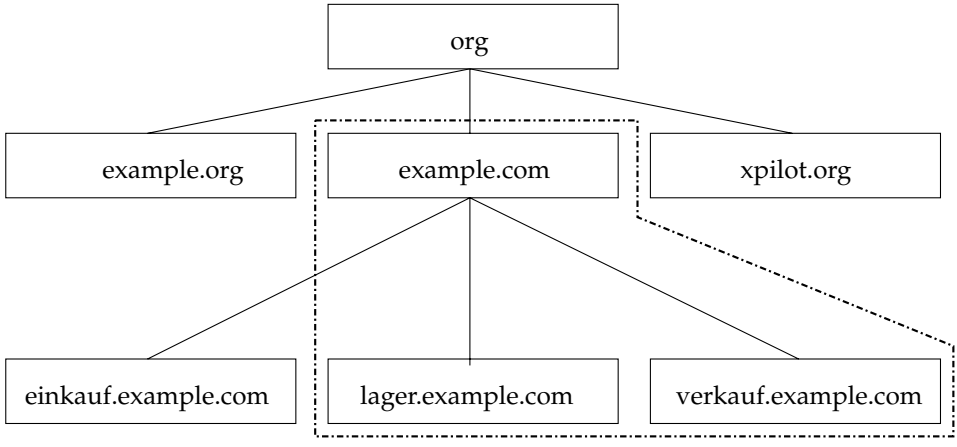


Abbildung 22.1 Beispiele für DNS-Zonen

Die oberste Zone, die hier nicht näher betrachtet wird, ist `.org`. Hier wurden die Zonen `example.org`, `example.com` und `xpilot.org` an andere Nameserver delegiert. In unserer Beispiel-Domain betreibt die Abteilung Einkauf einen eigenen Nameserver. Daher ist die Sub-Domain `einkauf.example.com` nicht Bestandteil der Zone `example.com`. Diese Zone ist in Abbildung 22.1 durch die gestrichelte Linie markiert.

Eine Zone wird technisch durch einen `NS`-Datensatz im übergeordneten Nameserver bestimmt. Eine Zone kann identisch mit einer Domain sein, es können aber auch für Sub-Domains weitere Nameserver als zuständig definiert sein, die dann wieder eine eigene Zone verwalten. Diese Trennung ergibt sich oft aus organisatorischen Gegebenheiten, seien es andere Verantwortlichkeiten für die Netzverwaltung oder die räumliche Trennung zwischen einzelnen Abteilungen oder Zweigstellen.

Anwender verwenden in der Regel nur Rechnernamen und keine IP-Adressen, weil gut gewählte Namen leichter zu merken sind. Daher ist der Ausfall des Nameservers oft gleichbedeutend mit dem Ausfall des Netzes. In jeder Zone sollten daher mehrere Nameserver aktiv sein, wobei einer der primäre Nameserver (primary name server) und alle anderen sekundäre Nameserver (secondary name server) sein sollten. Es ist sinnvoll, diese Rechner auf verschiedene Netzsegmente zu verteilen, damit auch beim Ausfall eines Routers oder Hubs immer noch

ein Nameserver erreichbar ist. Im Internet sollten der primary und der secondary Nameserver in verschiedenen Netzen liegen und über verschiedene Provider erreichbar sein. Weitere Informationen zur Auswahl eines secondary Nameservers finden Sie in [rfc2182].

Ein Secondary-Server ist praktisch wartungsfrei, da er seine Daten automatisch vom Primary-Server holt. Sie sollten nur gelegentlich einen Blick in die Log-Dateien werfen.

22.3 Auswahl eines DNS-Servers

Auch beim DNS-Server haben Sie die Wahl unter verschiedenen Programmen. Das wohl am häufigsten eingesetzte Programm ist BIND, das wir im Folgenden genauer betrachten werden. BIND ist vergleichsweise alt, gut gepflegt und sehr stabil. Besonders für größere Server wird es gerne eingesetzt.

Am anderen Ende der Skala finden Sie `dnrd`. Dies ist ein kleiner und handlicher Nameserver für kleine Netze. Das Besondere an ihm ist, dass er die Datei `/etc/hosts` liest und daraus dynamisch die benötigten Zonen generiert. Mit der Option `-s` können Sie Nameserver angeben, die als Forwarder verwendet werden sollen. Damit ist `dnrd` eine interessante Alternative für per Wählverbindung ans Internet angebundene Netze.

Ein weiterer Nameserver ist `djbdns`, der von Prof. Bernstein mit Blick auf Sicherheitsaspekte entwickelt wurde. Der Einsatz dieses Programms gilt als Geschmackssache. Wie auch immer, in den folgenden Abschnitten beschäftigen wir uns mit BIND.

22.4 Allgemeines zur Konfiguration eines Nameservers

Die Datei `/etc/resolv.conf` kann im Vergleich zu einem einfachen DNS-Client unverändert bleiben. In diesem Fall ist der Server auch sein eigener Client, was in den meisten Fällen gewollt sein dürfte (das ist auch der Standardwert, falls keine `/etc/resolv.conf` existiert). Es ist aber auch möglich, einen anderen Nameserver zu verwenden, als denjenigen auf dem lokalen Rechner.

Ab der Version 8 des BIND erfolgt die Konfiguration mit Hilfe der Datei `/etc/named.conf`. Sollten Sie bereits eine lauffähige Konfiguration einer älteren BIND-Version haben, dann können Sie die Datei `/etc/named.boot` mit Hilfe des Skripts `named-bootconf.sh` in das neue Format konvertieren. Anschließend sollten Sie aber die erstellte Datei prüfen und eventuell weitere Anpassungen vornehmen.

Ein Beispiel für einen primären Nameserver finden Sie in Listing 22.1. In dieser Datei werden die Domain und andere globale Daten, wie z. B. das Arbeitsverzeichnis des `named`, festgelegt. Außerdem werden die Namen der zugrunde liegenden Dateien bestimmt.

Nach dem Ändern der Datenbasis muss der `named` die Daten neu lesen. Dazu kann man dem Dämon das Signal `SIGHUP` schicken (z. B. mit `kill -HUP $(cat /var/run/named.pid)`) oder das Kommando `rndc reload` verwenden. Mit dem Kommando `rndc restart` kann der Dämon beendet und neu gestartet werden (siehe Abschnitt 22.8, »Steuerung des `named`-Prozesses«).

Ein weiteres wichtiges Signal ist `SIGINT`, das den Dämon veranlasst, seine Datenbank in die Datei `/var/tmp/named_dump.db` auszugeben. Dies ist für die Fehlersuche manchmal notwendig. Zur Steuerung des Dämons existieren noch diverse weitere Signale, die in der Manpage dokumentiert sind.

22.5 Primary Nameserver

Es existieren eine Reihe von Nameserver-Typen. In jeder Zone muss es einen Primary Nameserver geben, der die maßgeblichen (authoritative) Daten für die Zone verwaltet. Aus Gründen der Ausfallsicherheit sollte es in einer Zone mindestens einen weiteren Nameserver geben, der dann als Secondary Server betrieben wird. Die notwendigen Konfigurationen für einen Secondary Server finden Sie im Abschnitt 22.5.5, »Secondary-Nameserver«. Für Rechner oder Netze, die nur temporär, z. B. mittels PPP, mit dem Internet verbunden sind, kann es sinnvoll sein, einen Caching-Only-Nameserver einzurichten.

Der Betrieb eines Nameservers erfordert sowohl zusätzliche Rechenleistung und mehr Speicher auf dem entsprechenden Rechner als auch zunächst einen höheren Administrationsaufwand im Vergleich zum einfachen Editieren der Datei `/etc/hosts`. Im Wesentlichen gibt es zwei (zwingende) Gründe dafür, einen eigenen Nameserver zu installieren:

- Sie beantragen eine neue Domain. Besprechen Sie mit Ihrem Provider, wer den Nameserver für diese Domain betreibt. Wollen Sie den Nameserver selbst betreiben, so muss Ihr Provider dies bei sich entsprechend konfigurieren. Oft wird Ihr Provider den Nameserver zunächst für Sie betreiben.
- Sie wollen innerhalb einer bereits bestehenden Domain für einen organisatorischen Bereich einen neuen Nameserver einrichten. Dies müssen Sie mit Ihrem Netzverwalter besprechen, sonst wird Ihr neuer Server nicht als autorisiert anerkannt.

Ein eigener Nameserver hat aber auch Vorteile, egal ob Sie an das Internet angeschlossen sind oder nicht:

- In einem lokalen Netz sind auf jedem Rechner alle anderen Rechnernamen immer bekannt.
- Die Datenbasis ist immer auf allen Rechnern aktuell und konsistent. Die andernfalls auf den verschiedenen Rechnern notwendigen `hosts`-Dateien veralten schnell und widersprechen sich möglicherweise.
- Die Namensvergabe und IP-Adresszuordnung ist konsistent.
- Wenn Sie den Nameserver selbst betreiben, können Sie die Daten ändern, ohne Ihren Provider einschalten zu müssen.

Im Listing 22.1 finden Sie ein Beispiel für die Datei `/etc/named.conf` auf einem primären Nameserver. Mit `directory` im Abschnitt `options` wird das Verzeichnis bestimmt, in dem die Dateien dieses Nameservers abgelegt werden. Ein Rechner kann für mehrere Zonen der primäre und/oder sekundäre Nameserver sein. Für jede Zone ist ein Abschnitt `zone` einzufügen und die entsprechenden Daten müssen verfügbar sein.

```
# Konfig-File für den Name-Server
options {
        directory "/etc/named";
};

# Root Name-Server
zone "." {
        type hint;
        file "named.root";
};

zone "jochen.org" {
        type master;
        file "named.hosts";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "named.local";
};

zone "168.192.in-addr.arpa" {
        type master;
        file "named.rev";
};
```

Listing 22.1 Die Datei `/etc/named.conf` für einen primären Nameserver

Jeder Nameserver sollte einen Cache der Anfragen und Antworten anlegen. Dies wird mit dem Schlüsselwort `cache` bewirkt. Beim Start des Nameservers werden alle noch korrekten Daten aus `named.root` erneut in den Cache geladen. Ein Nameserver wird zum Cache-Only-Server, wenn in der Datei `named.boot` keine `primary` oder `secondary` Schlüsselwörter vorhanden sind.

Der Nameserver muss wissen, wo er den maßgeblichen Name-Server für die Root-Domain des Netzwerks findet. Diese Information ist in der Datei `named.root` gespeichert. In vielen Dokumentationen heißt diese Datei auch `root.cache`, was aber deren Verwendung nicht beschreibt. Diese Datei liegt, wie alle anderen Datendateien auch, im Standard Resource Record Format oder auch im Masterfile Format vor. Eine ausführliche Beschreibung dieses Formats finden Sie im BIND-Operators Guide, eine Einführung in den folgenden Abschnitten.

Der Nameserver im Listing 22.1 ist primärer (`primary`) Nameserver für die Zone `jochen.org`. Die Datenbasis befindet sich in der Datei `named.hosts` im Verzeichnis `/var/named`. Außerdem ist er der primäre Server für die Zonen `0.0.127.in-addr.arpa` und `168.192.in-addr.arpa`, die zum »Reverse-Lookup« dienen. Mehr Informationen zum Reverse-Lookup finden Sie im Abschnitt 22.5.4, »Die Datei `named.rev`«.

Im nächsten Schritt müssen die Daten der einzelnen Rechner in die entsprechenden Dateien, deren Namen in der Datei `/etc/named.boot` festgelegt wurden, eingetragen werden. Das Format dieser Dateien und die wichtigsten Einträge werden auf den folgenden Seiten genauer erläutert.

22.5.1 Die Datei named.hosts

In der Datei `named.hosts` werden die Daten für alle Rechner in der Zone gespeichert. Das Format und die einzelnen Einträge werden im folgenden Abschnitt beschrieben, da sie auch in den anderen Dateien zur Konfiguration eines Nameservers verwendet werden. In diese Datei tragen Sie alle Hosts der Zone ein, für die Sie autorisierter Nameserver sind. Im Listing 22.2 finden Sie ein Beispiel.

```
@       IN    SOA  janus.jochen.org. hostmaster.jochen.org. (
                   1997040101      ; Serial
                   28800           ; Refresh     8 hours
                   7200            ; Retry       2 hours
                   604800          ; Expire      7 days
                   86400 )         ; Minimum TTL 1 day
        IN     MX      10 hermes
        IN     NS      janus.jochen.org.

karotte         IN      A           192.168.3.150
mais            IN      A           192.168.3.151
kohlrabi        IN      A           192.168.3.152
```

```
bohne           IN      A           192.168.3.154
erbse           IN      A           192.168.3.155
                HINFO   IBM-PC/AT   UNIX-PC
                IN      TXT         "Test mit Text"
```

Listing 22.2 Die Datei `named.hosts`

Wenn Sie auf Ihrem Nameserver viele Zonen eingerichtet haben, dann werden Sie den Dateinamen sicher sprechend wählen, also den Namen der Zone verwenden. Wenn Sie sehr viele Zonen haben oder Nameserver für verschiedene Kunden betreiben, dann kann es sinnvoll sein, die Zonen in verschiedene Verzeichnisse zu sortieren.

22.5.2 Das Masterfile-Format

Eine Datendatei für den Nameserver besteht aus einzelnen Sätzen, Resource Records (RR) genannt. Das Format ist in [rfc1035] definiert und wird Masterfile-Format genannt. Ein Datensatz besteht aus den in Listing 22.3 dargestellten Teilen.

```
{name}  {ttl}  addr-class  Record Type  Record Specific data
```

Listing 22.3 Masterfile-Format

Im Feld `name` finden Sie den zum Eintrag gehörenden Namen des Rechners oder der Domain. Das Feld `ttl` enthält die Time To Live, die bestimmt, wie lange diese Daten in der Datenbank gespeichert werden. Wird kein Wert angegeben, so ist der Standardwert der im `SOA`-Satz angegebene (siehe Abschnitt 22.5.2.1 »Start Of Authority (`SOA`)«). Außerdem kann der Standardwert für die TTL mit dem Schlüsselwort `$TTL` am Anfang der Datei eingestellt werden. Im Internet wird als Adressklasse `IN` verwendet. Das vierte Feld enthält die Satzart, danach folgen die für die Satzart notwendigen Daten.

In den Dateien im Masterfile-Format haben eine Reihe von Zeichen eine besondere Bedeutung:

- Der Punkt (`.`) steht für die aktuelle Domain, für die die Daten gelten sollen.
- Das at-Zeichen (`@`) steht für das *Origin*, das ist die Zone, für die der Nameserver zuständig ist.
- Die Zeichen `. .` im Feld *Name* stehen für die Null-Domain.
- Die Zeichenfolge `\X`, wobei `X` eines der speziellen Zeichen ist, maskiert diese Sonderbedeutung und steht für das Zeichen selbst.
- Die Zeichenfolge `\DDD` mit `D` als Ziffern steht für das Zeichen mit dem ASCII-Code `DDD`.

- Ein Datensatz kann sich über mehrere Zeilen erstrecken, wenn die Daten in runde Klammern `()` eingeschlossen werden.
- Ein Semikolon (`;`) leitet einen Kommentar ein, der bis zum Zeilenende geht.
- Der Stern (`*`) wird im Feld *Name* bei einigen RR-Typen (z. B. `MX`) speziell interpretiert. Diese spezielle Bedeutung gilt nicht im Datenbereich.

Start Of Authority (SOA)

Ein SOA-Satz (Start Of Authority) markiert den Beginn einer Zone. Der Name (`@`, das Origin) steht für den Namen der Zone. Hier werden die grundsätzlichen Informationen für eine Zone festgelegt. Das Origin ist der Name des Rechners, der die Daten bereitstellt. Ein Beispiel für einen SOA-Satz finden Sie in Listing 22.4. Es gibt höchstens einen SOA-Satz für eine Zone, für die Zone existiert ein `NS`-Satz in der übergeordneten Zone.

Die verantwortliche Person ist der Betreiber des Nameservers, der bei Problemen und Fragen angesprochen werden sollte. Hier wird die E-Mail-Adresse dieser Person angegeben. Dabei wird das at-Zeichen (`@`) durch einen Punkt ersetzt. Oft wird hierfür ein Mail-Aliasname wie *mailto:hostmaster* oder *mailto:dnsadmin* verwendet.

Als weitere Information wird eine Seriennummer verwaltet, die bei jeder Änderung erhöht werden sollte. Sinnvoll ist die Verwendung des Datums und einer Versionsnummer, falls an einem Tag mehrere Änderungen vorgenommen werden. Am 26. Februar 1996 wäre dann eine mögliche Seriennummer `1996022601`. Anhand dieser Seriennummer erkennt ein Secondary-Server, dass neue Daten vorliegen und er einen Zonentransfer durchführen muss. Wenn Sie die Seriennummer versehentlich zu hoch gesetzt haben, dann ist es nicht einfach, dies wieder zu ändern – denn die Secondary-Server würden sich ja nicht mehr synchronisieren.

Der für `Refresh` angegebene Wert bestimmt die Anzahl Sekunden, nach denen die Slave-Server die Daten erneut prüfen und eventuell transportieren sollen. `Retry` gibt an, wie lange auf einen Zonentransfer gewartet wird, bevor ein Fehlschlag angenommen wird. Expire bestimmt, wie lange Daten maximal als gültig anerkannt werden. Die Minimum-TTL gibt den Standardwert für die Time-To-Live (`TTL`) der einzelnen Sätze an.

Vorschläge für die Wahl dieser Werte finden Sie in [rfc1912]. Lange Time-outs und Expire-Zeiten sorgen für wenig Netzlast beim Update der Daten, aber für längere Wartezeiten, bis Änderungen auf die sekundären Server übertragen wurden. Oft setzt man die Zeiten herab, wenn sich eine größere Änderung ankündigt. Nachdem alle sekundären Nameserver die Daten übernommen haben, kann man zu den alten Werten zurückkehren.

```
name {ttl} addr-class SOA Origin    Verantwortlicher
@       IN    SOA  janus.jochen.org. hostmaster.jochen.org. (
                   1997040101   ; Serial
                   28800    ; Refresh     8 hours
                   7200     ; Retry       2 hours
                   604800   ; Expire      7 days
                   86400 ) ; Minimum TTL 1 day
```

Listing 22.4 Ein SOA-Satz (Start Of Authority)

Beachten Sie, dass der `named` an alle Namen, die nicht mit einem Punkt (`.`) abgeschlossen sind, das Origin anhängt. Da dies so ist, ist ein vergessener oder überflüssiger Punkt einer der häufigsten Fehler bei der DNS-Konfiguration.

Nameserver (NS)

Für jeden Nameserver in der Zone muss hier ein Eintrag existieren. Mit diesem Eintrag wird eine neue Zone erzeugt und delegiert. Der erste Eintrag ist der Name der Zone, die bearbeitet wird, der letzte Eintrag ist der Name des verantwortlichen Nameservers. Jede Zone sollte mindestens zwei Nameserver (je einen primären und sekundären) besitzen, damit beim Ausfall eines Rechners der Name-Service weiter funktioniert. Diese Rechner sollten außerdem in verschiedenen Subnetzen bzw. im Internet bei verschiedenen Providern angebunden sein. Damit ist auch ein Router- oder Leitungsausfall zu überstehen.

Für jede Zone gibt es zwei, hoffentlich zueinander passende Definitionen für Nameserver: in der übergeordneten Zone, in der die neu definierte Zone delegiert wird, und in der Zone selbst. Nur diese beiden Einträge gemeinsam machen eine vollständig und korrekt delegierte Zone aus.

Eine beliebte Fehlerkonstellation ist, dass ein Nameserver betrieben wird, der nicht autorisiert ist (für den also im übergeordneten Nameserver (Parent-Name-Server) kein `NS`-Record existiert), oder ein Rechner als (sekundärer) Nameserver eingetragen ist, obwohl dort kein `named` läuft. Wenn ein System im `NS`-Record eingetragen, aber selbst nicht zuständig ist (also keinen `primary`- oder `secondary`-Eintrag in der `named.boot` hat), so spricht man von einer »Lame Delegation«.

In den Quellen von BIND sind verschiedene Skripten enthalten, die die Zonen nach Fehlern und Inkonsistenzen durchsuchen. Leider sind diese Skripten nur selten bei Linux-Distributionen installiert. Bei kommerziellen Systemen sind sie in der Regel auch nicht vorhanden. Sie sollten in jedem Fall den Quellcode von BIND installieren, die mitgelieferte Dokumentation lesen und sich die Skripten ansehen. Außerdem gibt es externe Pakete wie `dnslint`, die ebenfalls erweiterte Prüfungen durchführen.

```
{name}    {ttl}    addr-class    NS    Name servers name
                   IN            NS    janus.jochen.org.
```

Listing 22.5 Nameserver-Record

Address (A)

Ein A-Satz enthält für jeden Rechner die zugehörige IP-Adresse. Für Rechner mit mehreren IP-Adressen (multihomed Hosts), ist je IP-Adresse ein Satz erforderlich. Weitere Informationen, die in den im Folgenden erläuterten Satzarten gespeichert werden, sind für den Betrieb eines Nameservers nicht erforderlich und werden oft auch nicht gepflegt. Die einzige Ausnahme sind die `PTR`-Sätze, die für den »Reverse-Lookup« erforderlich sind.

Für jede IP-Adresse, die in der Zone belegt ist, sollte genau ein `A`-Record existieren. Für Rechner, die mehrere IP-Adressen haben, sind also mehrere `A`-Records aufzunehmen. Dabei kann, wie im Listing 22.6 gezeigt, derselbe Name verwendet werden. Es ist in manchen Fällen sinnvoll, einen zusätzlichen Namen einzurichten, der z. B. das betreffende Interface beschreibt.

```
{name}     {ttl}    addr-class    A    address
jupiter             IN            A    192.168.30.202
typhon              IN            A    192.168.31.151
                    IN            A    10.0.0.78
```

Listing 22.6 Address-Satz im DNS

Die Namen in der ersten Spalte werden hier ohne Domain angegeben. Da der Name auch nicht durch einen Punkt abgeschlossen wurde, wird vom Nameserver automatisch die Zone angehängt. Daher ist der Rechner unter seinem vollen Namen (FQDN) im Nameserver eingetragen. Der Client, d. h. die Resolver-Bibliothek, durchsucht den Nameserver nach einem Namen zunächst mit der angehängten Domain, so dass Benutzer im lokalen Netz diese in der Regel nicht angeben müssen.

Für IPv6-Adressen verwenden Sie den Adresstyp `A6`. In älteren Versionen der IPv6-RFCs wurde der Adresstyp `AAAA` verwendet, diesen sollten Sie bei neuen Adressen jedoch nicht mehr einsetzen.

Well Known Services (SRV)

Im DNS ist es möglich, weitere Informationen zu Services, die ein Rechner anbietet, zu speichern. In der Regel werden diese Daten nicht gepflegt, so dass man sich auf diese Angaben nicht verlassen kann. Am besten probiert man den entsprechenden Service einfach aus. Wenn die Verbindung hergestellt werden kann, dann bietet der Rechner den entsprechenden Dienst an, andernfalls nicht. In früheren Versionen wurde statt `SRV` der Resource-Typ `WKS` verwendet.

Canonical Name (CNAME)

Wie bereits in Kapitel 16, »IP-Adressen und Rechnernamen«, erwähnt wurde, sollte ein Rechnername nicht aufgrund der Funktion des Rechners vergeben werden. Dennoch ist es oft sinnvoll, einen Rechner auch unter einem passenden Namen erreichen zu können. Dies findet man z. B. bei `ftp`-Servern, die auch unter dem Alias-Namen `ftp` erreichbar sind. Weitere Dienste, für die häufig Aliasnamen (oder Nicknames) vergeben werden, sind `www`, `pop3`, `smtp` oder `irc`.

Übernimmt ein anderer Rechner diese Funktion, so muss nicht der Name des Rechners, sondern nur der entsprechende Nickname verändert werden. Damit bleibt den Benutzern verborgen, welcher Rechner tatsächlich den Dienst übernommen hat. Wird kein `CNAME` verwendet, sondern muss der Rechner umbenannt werden, so bedeutet das oft einiges an Konfigurationsaufwand und es ist relativ fehleranfällig.

Wird ein Rechner umbenannt, so ist es oft sinnvoll, den alten Namen eine Zeit lang als Nickname zusätzlich zu erlauben. Es sollte jedoch nie ein `MX`-Record (Mail Exchange) auf einen `CNAME` weisen, da ein Mail Transfer Agent dort nicht nachfragen muss (`named` gibt eine entsprechende Warnung aus). In aktuellen BIND-Versionen kann außerdem zu einem `CNAME` keine weitere Information, insbesondere kein `MX`-Record, eingetragen werden.

```
aliases     {ttl}    addr-class    CNAME    Canonical name
jupiter              IN            CNAME    ftp
```

Listing 22.7 Canonical Name

Domain Name Pointer (PTR)

Ein `PTR`-Satz (Domain Name Pointer) erlaubt es z. B. anhand einer IP-Adresse, auf den Host-Name zu schließen. Diese Funktion wird in der `in-addr.arpa`-Domain benutzt, also z. B. in der `named.rev`-Datei. Damit ist es möglich, auch wenn zunächst nur die IP-Adresse eines Rechners oder Gateways bekannt ist, einen Namen anzuzeigen. Diese Funktion wird z. B. von den Programmen `netstat` oder `traceroute` verwendet. Mehr zum Reverse-Lookup finden Sie im Abschnitt 22.5.4, »Die Datei `named.rev`«.

Die `PTR`-Sätze werden für die Funktion `gethostbyaddr(3)`, die für die Umwandlung der IP-Adressen in Namen zuständig ist, benötigt. Jeder Name im `PTR`-Satz wird mit einem Punkt abgeschlossen, damit BIND nicht die Domain `in-addr.arpa` anhängt. Sie sollten alle Netzwerkteilnehmer, die eine IP-Adresse haben, in den Nameserver aufnehmen, damit Sie nicht auf einen Nameserver-Time-Out warten müssen, nur weil kein Nameserver-Eintrag für den Reverse-Lookup verfügbar ist.

```
name    {ttl}   addr-class    PTR   real name
155.31          IN            PTR   jupiter.jochen.org.
151.31          IN            PTR   typhon.jochen.org.
```

Listing 22.8 Domain Name Pointer

Bei IPv6 wird als Domain für den Reverse-Lookup `ip6.arpa` verwendet. Die IPv6-Adressen sind dabei im bekannten Format anzugeben. In älteren Applikationen kommt eventuell noch das »Nibble«-Format zum Einsatz, dann müssen Sie die Domain `ip6.inet` verwenden.

Mail Exchange (MX)

Im Internet wird Mail mit Hilfe des Simple Mail Transfer Protocol (SMTP, [rfc2822]) übertragen. Dabei wird zwischen dem sendenden und empfangenden Rechner eine TCP/IP-Verbindung aufgebaut.

Ist ein Rechner zum Zeitpunkt des Versendens der Mail nicht erreichbar (oder der Rechner nicht an das Internet, sondern mit UUCP angeschlossen), so kann hier ein anderer, verfügbarer Rechner eingetragen werden, der die Mail annimmt. Ist der Rechner dann wieder erreichbar, so wird die Mail weitergereicht. Niedrigere Werte für die Präferenz entsprechen geringeren »Kosten«.

Diese Funktion wird oft eingesetzt, damit nur auf einem oder zwei Rechnern im Netz Mail angenommen werden muss. Damit konzentriert man die Konfigurationsarbeit und die Verfügbarkeitsprobleme auf wenige Rechner. Sie sollten sich mit dem Administrator des Rechners abstimmen, wenn Sie einen `MX`-Satz auf einen fremden Rechner zeigen lassen. Andernfalls wird dieser mit unerwarteter Mail bombardiert und diese geht möglicherweise verloren.

```
name           {ttl} addr-class MX preference mail exchange
jupiter.jochen.org. IN          MX 0          hermes
```

Listing 22.9 Mail-Exchanger

Ein weiterer Vorteil eines Mail-Exchangers ist, dass die Mail nicht auf Hunderten oder Tausenden von Rechnern weltweit gesammelt wird und möglicherweise unmittelbar nach dem Neustart des Mail-Servers diesen wieder zum Absturz bringt. Wird aufgelaufene Mail zentral gesammelt, dann liefert nur dieser Rechner massenweise Mail aus und im Problemfall kann man mit dem Administrator dieses Rechners einen langsameren Feed vereinbaren.

Prinzipiell ist es möglich, für alle Rechner einer Domain einen zentralen Mail-Hub vorzusehen. Dazu kann man den Eintrag `*` als `MX` verwenden. Beachten Sie, dass das nicht unbedingt den Effekt hat, den Sie erwarten: Der `MX` gilt für alle möglichen Namen, nicht nur für diejenigen, die in der `named.data`-Datei eingetragen sind. In den meisten Fällen ist es besser, den betreffenden Einträgen explizit einen `MX`-Eintrag zuzuordnen.

Mail-Exchanger dürfen nicht auf einen CNAME verweisen. named gibt hier eine entsprechende Warnung aus. Wenn Sie einen zusätzlichen Namen auf Ihrem Mail-Hub annehmen wollen, so müssen Sie bei sendmail mit Hilfe des Cw-Eintrags dafür sorgen, dass dieser Name als lokal erkannt wird. Andernfalls gibt sendmail die Fehlermeldung »local configuration error: mail loops back to myself« aus.

Text (TXT)

Als zusätzliche Information kann zu jedem Rechner ein beliebiger Text abgelegt werden. Außer, dass dieser Text in Anführungszeichen (") eingeschlossen werden muss, bestehen keine weiteren Einschränkungen. Dieser Typ ist nur selten gepflegt, Netzwerkverwalter bevorzugen es in der Regel, diese Informationen mit Hilfe von SNMP direkt vom betreffenden System auszulesen.

```
name                {ttl}   addr-class   TXT   string
jupiter.jochen.org.         IN           TXT   "Ein Text"
```

Listing 22.10 Text

Responsible Person (RP)

Neben dem Verantwortlichen für den Nameserver kann zu jedem weiteren Eintrag ein Ansprechpartner für diesen Rechner hinterlegt werden. Hier wird wieder das at-Zeichen (@) in der Mail-Adresse durch einen Punkt ersetzt.

Verwalten Sie Ihre Rechner jedoch mit SNMP (Simple Network Management Protocol), so tragen Sie diese Information besser in die Management Information Base (MIB) ein.

```
owner  {ttl} addr-class RP mbox-domain-name TXT-domain-name
jupiter      IN RP jochen.jochen.org. sysadmins.jochen.org.
sysadmins.jochen.org. IN TXT "Tel: 4711"
```

Listing 22.11 Responsible Person

22.5.3 Die Datei named.local

Die Datei named.local (siehe Listing 22.12), die sich im Verzeichnis /var/named befindet, enthält die zum eigenen Rechner bzw. zum jeweiligen localhost gehörenden Daten. Die Datei besteht nur aus wenigen Einträgen, die wichtig sind, wenn Benutzer in ihren ~/.rhosts-Dateien auch den Eintrag localhost benutzen wollen und/oder der Rechner localhost nicht in der Datei /etc/hosts enthalten ist.

```
@ IN SOA janus.jochen.org. hostmaster.jochen.org. (
                          1986012101 ; Serial
                          3600    ; Refresh
                          300     ; Retry
                          3600000 ; Expire
                          14400 )  ; Minimum
  IN NS  janus.jochen.org.
0 IN PTR loopback.
1 IN PTR localhost.
```

Listing 22.12 Die Datei `named.local`

22.5.4 Die Datei named.rev

Für den normalen DNS-Lookup wird die DNS-Domain verwendet. Mit deren Hilfe wird der zuständige Nameserver ermittelt und dann befragt. Für den Reverse-Lookup wurde eine spezielle Zone generiert, die `in-addr.arpa`. In diesem vom DNS verwalteten Baum sind die IP-Adressen nach den zugehörigen Subnetzen strukturiert. Bei der Delegation der Zonen werden die Bytes der Netzadresse in umgekehrter Reihenfolge verwendet.

In Listing 22.1 sind zwei Zonen eingetragen. Die Zone `0.0.127.in-addr.arpa` wird in der Datei `named.local` verwaltet und repräsentiert das `localnet`, die Rechner mit den Adressen `127.0.0.`*x*. Die Zone `168.192.in-addr.arpa` steht für die IP-Adressen aus dem reservierten Bereich `192.168.`*x*`.`*y*.

In der Datei `named.rev` wird zu einer IP-Adresse der zugehörige Rechnername vermerkt. Damit ist es relativ schnell und einfach möglich, einen Namen zu einer IP-Adresse zu finden. Diese Funktion nennt man auch Reverse-Lookup. Die Datei `named.rev` liegt im Masterfile-Format vor. Ein Beispiel finden Sie in Listing 22.13. In der ersten Spalte sind die Teile aus der IP-Adresse angegeben, die in der Zonendefinition nicht enthalten sind. Auch hier wird wieder die umgekehrte Reihenfolge verwendet.

Beachten Sie, dass Sie jeden Host-Namen mit einem Punkt abschließen müssen. Andernfalls würde stets die Domain `in-addr.arpa` angehängt. Achten Sie außerdem auf die Konsistenz der Dateien `named.data` und `named.rev`. Dabei kann Ihnen ein Tool wie `dnslint` helfen.

```
@ IN  SOA janus.jochen.org. hostmaster.jochen.org. (
                              1986020501 ; Serial
                              10800   ; Refresh 3 hours
                              3600    ; Retry   1 hour
                              3600000 ; Expire  1000 hours
                              86400 ) ; Minimum 24 hours
        IN      NS      janus.jochen.org.
```

```
0.0     IN      PTR     jochen.org.
        IN      A       255.255.255.0
151.31  IN      PTR     typhon.jochen.org.
155.31  IN      PTR     jupiter.jochen.org.
```

Listing 22.13 Die Datei `named.rev`

Bei IPv6 wird für den Reverse Lookup die Domain `ip6.arpa` verwendet, als Resource-Record wird weiterhin der Typ PTR verwendet. Noch ist IPv6 nicht weit verbreitet, aber das wird sich ändern.

22.5.5 Secondary-Nameserver

Um die Ausfallsicherheit zu erhöhen, sollten in jeder Zone mindestens zwei Nameserver betrieben werden. Beim Ausfall eines Rechners kann der zweite Nameserver immer noch alle Anfragen beantworten. Damit die Daten stets konsistent sind, darf für jede Zone nur ein (autorisierter) Master-Server existieren.

Daneben können ein oder mehrere sekundäre Nameserver konfiguriert werden. Ein Secondary Nameserver holt seine Daten vom primären Server. Ist dieser nicht verfügbar, so werden die in Sicherheitskopien (`*.bak`) gespeicherten Daten verwendet. Ist die Time-to-Live der Daten abgelaufen, so wird versucht, diese Daten erneut vom Nameserver zu erhalten. In Listing 22.14 finden Sie ein Beispiel für die Datei `/etc/named.boot` eines sekundären Nameservers.

```
# conf file excerpt for a secondary name server
zone "jochen.org" {
        type slave;
        masters { 192.168.30.254; };
        file "named.bak";
};

zone "168.192.in-addr.arpa" {
        type slave;
        masters { 192.168.30.254; };
        file "named.rev.bak";
};
```

Listing 22.14 Die Datei `/etc/named.conf` *für einen sekundären Nameserver*

Anhand der in den `SOA`-Sätzen der Nameserver-Daten eingetragenen Seriennummern kann ein sekundärer Nameserver feststellen, ob er noch die aktuellen Daten einer Zone besitzt. Andernfalls kann eine Neuübertragung veranlasst werden. Wenn Sie in den primären Daten eine kleinere Seriennummer vergeben, wird der sekundäre Nameserver entsprechende Meldungen ausgeben. Sie soll-

ten daher die System-Log-Einträge aller Nameserver im Auge behalten, um derartige Fehler möglichst auszuschließen. Ansonsten sind Secondary-Server vollkommen wartungsfrei.

22.6 Slave-Nameserver

Ein Slave-Server verwaltet keine zonenbezogenen Daten, sondern nur einen Cache und leitet Anfragen, die nicht aus dem Cache beantwortet werden können, an festgelegte Nameserver (`forwarders`) weiter. Damit kann dieser Rechner einen guten Cache aufbauen, von dem alle Nutzer profitieren können.

Prinzipiell kann jeder Nameserver für die Verwendung von `forward` konfiguriert werden. Dies ist zum Beispiel sinnvoll, wenn der Nameserver nicht direkt an das Internet angeschlossen ist. Wird als Forwarder der Router oder eine Firewall eingetragen, so kann dieser bzw. diese die Anfragen weiterleiten. Ein weiterer Vorteil ist die Ausbildung eines großen Cache, so dass insgesamt weniger Anfragen durch andere Nameserver beantwortet werden müssen.

Forwarder befragen die eingetragenen Server zusätzlich, z. B. zu den bekannten Root-Nameservern. Bei einem Slave-Server werden nur die Forwarder befragt. Können diese keine Antwort liefern, so kann der Host-Name nicht aufgelöst werden. Da diese Rechner die Daten möglicherweise selbst beschaffen müssen, sollten Sie die Adressen mehrfach in der Datei `/etc/named.boot` aufführen. Ein Beispiel dafür finden Sie in Listing 22.15.

```
#    boot file for slave name server
options {
        directory "/etc/named";
        forward only;
# Alle unbekannten Anfragen an den DNS der uranus weitergeben
        forwarders {
                194.45.71.65;
                194.45.71.65;
                194.45.71.65;
        };
};
zone "." {
        type hint;
        file "named.root";
};
```

Listing 22.15 Die Datei `/etc/named.boot` für einen Slave-Nameserver

Mit der Option `forward-only` wird dafür gesorgt, dass nur die angegebenen Forwarder befragt werden. Wenn Sie eine Firewall haben, so kann es sein, dass Sie diese als Forwarder eintragen müssen, wenn die Nameserver keinen Zugang

zum Internet haben. Als Vorteil kommt hinzu, dass wiederum die Forwarder einen guten Cache ausbilden können. In alten Versionen von named wurde slave anstelle der Option forward only verwendet.

Sie sollten an erster Stelle den schnellsten (und stabilsten) Nameserver eintragen. Da es möglich (und sogar wahrscheinlich) ist, dass dieser Rechner die Daten zunächst selbst beschaffen muss, sollten Sie den Eintrag für diesen Rechner zwischen die weiteren Forwarder ein- oder zweimal erneut einstreuen.

22.7 Weitere Optionen in der Datei named.boot

Das Verhalten von named kann durch einige weitere Einträge beeinflusst werden. Die wichtigsten finden Sie hier erläutert, eine vollständige Dokumentation entnehmen Sie bitte dem BIND Administrators Reference Manual, das den BIND-Quellen beiliegt.

option forward only
: Es werden nur die angegebenen Forwarder befragt. Der früher verwendete Eintrag slave sollte nicht mehr verwendet werden. In der Version 8 von BIND hieß die Option forward-only.

logging
: Protokollierung kann sehr flexibel konfiguriert werden. Neben normalen Logs können Sie zur Fehlersuche auch alle Anfragen protokollieren. Logs können mittels syslog oder in Dateien geschrieben werden. Wenn Sie besondere Wünsche haben, werfen Sie einen Blick in die BIND-Dokumentation.

option recursion no
: Normalerweise beschafft ein Nameserver die gewünschten Daten für den Client, selbst wenn er nicht direkt zuständig ist. Das hat den Vorteil der guten Cache-Bildung, würde aber bei den Root-Nameservern, von denen alle Zonen delegiert werden, zu einer sehr großen Belastung führen. Diese Server liefern nur einen Verweis auf den zuständigen Rechner und nicht die gewünschten Daten. Diesen Eintrag sollten Sie nicht verwenden.

allow-*Option*
: BIND9 gestattet es, den Zugriff auf verschiedene Funktionen einzuschränken. Mit der Option allow-transfer kann der Zugriff auf die gesamten Zonendaten mittels Zonentransfer auf Rechner aus den angegebenen Netzen beschränkt werden. Damit kann man eine einfache Form der Zugriffskontrolle implementieren. Besonders nützlich ist das nicht, da immer noch die Möglichkeit besteht, die zugeordneten IP-Adressen mittels Reverse-Lookup zu durchsuchen. Diese Option können Sie auch je Zone angeben, dann wird der Standardwert überschrieben. Weitere Optionen zur Zugriffskontrolle finden Sie in der BIND-Dokumentation erläutert.

`include`
Sie können die Datei `named.boot` aufteilen und die einzelnen Teile mittels `include` einlesen. Das kann interessant sein, wenn Sie viele Zonen verwalten und diese möglicherweise von verschiedenen Personen betreut werden.

`bogus`
Es kann vorkommen, dass fremde Nameserver fehlerhafte Daten liefern. Mit der Option `bogus` lässt sich verhindern, dass ein derartig verseuchter Nameserver befragt wird.

`max-cache-size`
Ein großer Nameserver kann eine beachtliche Last auf eine Maschine bringen. Mit der Option `max-cache-size` können Sie die Größe des DNS-Cache beschränken. Eine weitere Option, um den Speicherbedarf einzuschränken, ist `recursive-clients`. Hier können Sie die Anzahl der rekursiven Anfragen beschränken, die für Clients durchgeführt werden.

`check-names master|slave|response warn|fail|ignore`
`named` kann die Namen in den Zonen auf Gültigkeit prüfen. Alte Versionen von BIND ignorierten fehlerhafte Namen stillschweigend, bei aktuellen Versionen kann man wählen, ob `named` eine Warnung ausgeben (`warn`), einen Fehler melden (`fail`) oder den Fehler wie gewohnt ignorieren soll (`ignore`). Die Einstellung kann getrennt für primäre und sekundäre Zonen erfolgen.

BIND kennt noch eine ganze Reihe weiterer Optionen. Für viele Fälle werden Sie diese nicht benötigen. Sollten Sie dennoch etwas vermissen, werfen Sie einen Blick in die BIND-Dokumentation.

22.8 Steuerung des named-Prozesses

Das Programm `named`, das den Nameservice bereitstellt, läuft in der Regel als Dämon im Hintergrund. Die Steuerung erfolgt über Signale, die in der Manpage zu `named(8)` dokumentiert sind. Als Vereinfachung existiert das Programm `rndc`, mit dem die wichtigsten Signale versendet werden können. Mögliche Parameter sind:

`stop`
Stoppen des `named`.

`status`
Zeigt die wichtigsten Server-Parameter an, insbesondere die Anzahl der Zonen und ob der Server aktiv ist.

`dumpdb`
`named` schreibt die aktuelle Datenbank und den Cache in die Datei `named_dump.db`. Diese Funktion ist zur Fehleranalyse manchmal recht nützlich.

`reload [Zone]`
: `named` lädt die angegebene Zone neu. Wenn keine Zone angegeben wird, dann liest `named` die Konfiguration und alle Zonen neu ein.

`stats`
: Der Nameserver schreibt seine Statistik in die Datei, die in der Option `statistics-file` angegeben ist. Standardwert ist `named.stats`.

`trace`
: Der Tracelevel wird erhöht und mehr Informationen werden in der Datei `/var/tmp/named.run` protokolliert.

`notrace`
: Der Tracelevel wird um eins verringert und entsprechend weniger Informationen werden protokolliert.

`rndc` kann nicht nur auf dem lokalen Rechner verwendet werden. Mit der Option `-s` kann ein anderer Server und mit `-p` ein anderer Port als der Standardwert 953 angegeben werden. Damit das sicher funktioniert, muss der Server mit der Option `controls` konfiguriert werden. Sie können Zugriffe auf IP-Adressen einschränken oder mittels kryptographischen Schlüsseln authentifizieren. In der Standardinstallation kann der Nameserver nur vom `localhost` aus konfiguriert werden.

22.9 Betrieb eines Nameservers

Die Programme aus dem BIND-Paket sind so stabil, dass im täglichen Betrieb kaum mit Problemen zu rechnen ist. Insbesondere ist der Betrieb eines Secondary-DNS praktisch wartungsfrei. Dennoch sollte man ab und zu einen Blick in die Log-Dateien werfen, insbesondere, wenn man die Daten der Zone geändert hat.

Eine solche Änderung ist nicht sofort auf allen Rechnern bekannt. Die Secondary-Server erkennen nach der Erhöhung der `serial`-Nummer, dass sie die Zone neu laden müssen. Das erfolgt normalerweise einmal je `refresh`-Intervall. Treten hierbei Fehler auf, wird der Zonentransfer alle `retry` Sekunden erneut versucht. Die Anfragen werden vom Secondary-Server aber weiterhin als `authoritative` beantwortet, bis die Zeit `expire` abgelaufen ist.

Das heißt, dass es – auch durch das normale DNS-Caching und die Möglichkeit, einen privaten Secondary-Server für fremde Zonen aufzusetzen – keine Möglichkeit gibt, eine Zonenänderung sofort überall wirksam werden zu lassen. Normalerweise wird man tunlichst versuchen, derartige Situationen zu vermeiden, indem man beispielsweise Dienste temporär auf dem alten und dem neuen Rechner parallel anbietet.

Sollte man eine derart plötzliche Umschaltung nicht vermeiden können, so sollte man vorher die Timing-Werte in der Zone schrittweise vermindern (dabei muss auch die `serial`-Nummer erhöht werden). Zum Umschaltzeitpunkt kann man dann mit einer schnellen Verteilung der Daten rechnen. Das bedeutet allerdings, dass in der Zwischenzeit eine deutlich höhere Netzwerklast nur durch DNS-Anfragen zu verzeichnen sein wird.

Ein weiterer Fall, bei dem Sie vorsichtig sein müssen, ist die Verminderung der `serial`-Nummer in einer Zone. Eigentlich ist das nämlich nicht möglich – und Sie sollten es auch zu vermeiden suchen. Wenn Sie es dennoch tun müssen und *alle* Secondary-Server unter Kontrolle haben, dann können Sie dort die kopierte Zone löschen und die Server neu starten. Im Internet haben Sie normalerweise diesen Zugriff auf die Secondary-Server nicht, so dass Sie hier das in den Quellen zum BIND beschriebene Verfahren (schrittweises und gezieltes Erhöhen der `serial`-Nummer bis zum Überlauf) anwenden müssen.

22.10 Dynamische DNS-Updates

Bisher haben wir DNS-Zonen als statisch betrachtet – das stimmt auch, solange nur der Administrator diese ändert. Bei der Verwendung von DHCP ist die Zuordnung von Namen zu IP-Adressen dynamisch möglich. Nun könnte man für alle möglichen IP-Adressen einen statischen Namen vergeben, aber das hat auch Nachteile. Ein Nachteil ist, dass zunächst alle Namen aufgelöst werden können und der Versuch die Rechner dann wirklich zu erreichen, in einen Timeout laufen kann.

Mit den Schlüsselwörtern `allow-update` oder `update-policy` im `zone`-Statement kann der Administrator das dynamische Update erlauben. Aus Performance-Gründen wird die Zonen-Datei nicht nach jedem Update neu geschrieben, sondern Änderungen werden zunächst in einem Journal protokolliert. Bei einem Neustart wird dann die bestehende Zone geladen und das Journal nachgefahren. In regelmäßigen Abständen wird allerdings die Zone als Datei gespeichert. Wenn Sie die Zone mit einem Editor ändern wollen, so müssen Sie den Nameserver mit `rndc stop` anhalten und das Journal löschen.

22.11 Sicherheit und DNS

DNS ist ein zentraler Dienst im Internet. Bisher hat man sich darauf verlassen, dass schon alles stimmen wird. Programme wie `dnsspoof` beweisen allerdings schon seit geraumer Zeit das Gegenteil. Noch werden die hier vorgestellten Methoden selten eingesetzt, was sich aber hoffentlich ändert.

Bisher kann sich ein Client nie sicher sein, ob die Antwort, die er erhalten hat (eventuell über einen Forwarder oder einen DNS-Cache) tatsächlich vom authoritativen Nameserver stammt oder nicht. Im Rahmen von DNSsec ([rfc2535], [rfc3130]) wurden Methoden entwickelt, dies mit kryptographischen Verfahren sicherzustellen.

Zunächst wird eine Zone wie bekannt erstellt, außerdem wird ein zu dieser Zone gehörendes Keypaar generiert. Die Zone wird mit dem Private Key unterschrieben, so dass Clients anhand des Public Key die Authentität überprüfen können. Ein analoges Verfahren wird bei PGP und GnuPG verwendet.

In Listing 22.16 finden Sie die notwendigen Befehle, um ein Keypaar zu generieren. Die Keys müssen für das Signieren der Zone verfügbar sein, allerdings kann das auf einem Rechner erledigt werden, der nicht der DNS-Server ist.

```
(linux):~# dnssec-keygen -a rsamd5 -b 2048 -n zone jochen.org
Kjochen.org.+001+40716
(linux):~# ls -l
...
-rw-r--r--  1 root  jochen    378 29. Apr 21:02
Kjochen.org.+001+40716.key
-rw-------  1 root  jochen   1697 29. Apr 21:02
Kjochen.org.+001+40716.private
...
(linux):~# cat Kjochen.org.+001+40716.key
jochen.org. IN KEY 256 3 1 AQP7qzWpGTD...
(linux):~# dnssec-makekeyset -a Kjochen.org.+001+40716
keyset-jochen.org.
(linux):~# ls -l
...
-rw-------  1 root  jochen    925 29. Apr 21:05 keyset-jochen.org.
```

Listing 22.16 Generieren eines Keypaars für DNSsec

Der erstellte Public Key wird in die Zone aufgenommen (Listing 22.17).

```
Zone jochen.org
$INCLUDE Kjochen.org.+001+40716.key
```

Listing 22.17 Ergänzung in der DNS-Zone für DNSsec

Mit dem erstellten Keypaar kann die Zone signiert werden (Listing 22.18). Damit ist das Problem noch nicht gelöst, wie der Public Key zum Client kommt. In abgeschlossenen Umgebungen könnte man diesen manuell transportieren und authentisieren, im Internet wird man eine Public-Key-Infrastruktur aufbauen müssen. In dieser PKI wird der Parent der Zone diese signieren und damit diesen Key bestätigen.

```
(linux):~# dnssec-signzone -o jochen.org jochen.org
Kjochen.org.+001+40716
jochen.org.signed
```

Listing 22.18 Signieren der Zone

Beim Signieren der Zone werden zusätzliche NXT- und SIG-Sätze eingefügt, die die Daten authentifizieren. Damit kann ein Client eine Antwort gegen den Public Key prüfen. In Listing 22.19 finden Sie ein Beispiel für eine Signatur. In der Nameserver-Konfiguration verwenden Sie die Datei jochen.org.signed als Zonendefinition.

```
hermes.jochen.org.      86400   IN MX   10 mail.jochen.org.
                        86400   IN MX   80 mail.example.org.
                        86400   IN MX   90 mail2.example.org.
                        86400   SIG     MX 1 3 86400
20020529193717 (
                                        20020429193717 40716
jochen.org.

ksFvd1VtVBjxI+p0RxFWfs4iUCxp0pqhhchr
                                        ...
                                        SvII+aNOpoo51gVy5A== )
```

Listing 22.19 Beispiel aus einer signierten Zone

Der Parent der DNS-Zone signiert das Keyset mit dem Befehl dnssec-signkey. Noch ist das unüblich, das könnte sich aber ändern, wenn die DNS-Registrare diesen Dienst anbieten oder sogar zur Pflicht machen.

Zu DNSsec gehört auch noch die Möglichkeit, Anfragen zu signieren (Transaction Signatures, TSIG, [rfc2845]). BIND9 bietet auch hierfür Unterstützung, primär für Transaktionen zwischen Servern, insbesondere Zonen-Transfers. Das Bind-Handbuch enthält hierfür ein Beispiel, das Sie als Startpunkt verwenden können. Da hier zwischen den Servern ein Shared-Secret verwendet wird, ist es einfach, einen Key zu erzeugen und zu verwenden.

22.12 Weitere Informationen zum DNS

Die wichtigsten Informationen zum Betrieb eines Nameservers finden Sie im »Administrators Reference Manual for BIND«, das in den Quellen zu bind enthalten ist. Empfehlenswert ist darüber hinaus das Buch »Managing DNS and BIND« von [Albitz2001].

23 Network Information Service

23.1 Allgemeines zu NIS

NIS (*Network Information Service*) ist eine Möglichkeit, mehrere Rechner zumindest teilweise zentral zu administrieren. Dabei werden einige Dateien nicht mehr lokal auf jedem Rechner gepflegt, sondern zentral auf dem NIS-Server. NIS war früher unter dem Namen YP (Yellow Pages[1]) bekannt. Die Nutzung von NIS ist immer dann sinnvoll, wenn eine Reihe von Unix-Rechnern unter zentraler Verwaltung stehen sollen (d. h. eine Person oder Abteilung ist für eine Gruppe von Rechnern verantwortlich).

Mit NIS können verschiedene Konfigurationsdateien (siehe Tabelle 23.1) zentral auf einem Server verwaltet und für die Clients bereitgestellt werden. Damit lässt sich der Administrationsaufwand in einem lokalen Netz deutlich verringern. Dies gilt besonders beim Einsatz von NFS (Network File System, siehe Kapitel 20, »Network File System (NFS)«, weil dort zumindest die Dateien `/etc/passwd` und `/etc/group` zwischen den beteiligten Rechnern synchronisiert werden müssen. Neben den Standarddateien aus Tabelle 23.1 können zusätzlich beliebige weitere (eigene) Datenbanken verwaltet werden.

Dateiname	NIS-Maps
`ethers`	`ethers.byname` und `ethers.byaddr`
`hosts`	`hosts.byname` und `hosts.byaddr`
`networks`	`networks.byaddr` und `networks.byname`
`protocols`	`protocols.bynumber` und `protocols.byname`
`rpc`	`rpc.byname` und `rpc.bynumber`
`services`	`services.byname`
`passwd`	`passwd.byname` und `passwd.byuid`
`group`	`group.byname` und `group.bygid`
`netid`	`netid.byname`
`shadow`	`shadow.byname`
`gshadow`	`gshadow.byname`

Tabelle 23.1 Standard-NIS-Maps

Der zentrale Begriff im NIS ist die *NIS-Map*. In einer Map wird eine Datenbank mit einem Schlüssel und zugehörigen Daten verwaltet. Soll eine Datei (z. B. `/etc/passwd`) nach mehreren Schlüsseln (z. B. nach Benutzername und User-ID)

1. Trademark von British Telecom

durchsucht werden können, so werden dementsprechend mehrere Maps angelegt (siehe Tabelle 23.1). Die Daten werden mit einer Hash-Tabelle verwaltet, so dass auch größere Maps sinnvoll mit ausreichender Geschwindigkeit verwendet werden können. In größeren Rechenzentren kommen schnell mehrere tausend Benutzer zusammen.

Die Namen der Programme zu NIS beginnen alle mit dem Präfix `yp`. Dieses Präfix hat sich vor Jahren mit der ersten, damals noch YP genannten, Implementation eingebürgert. Unter Linux gibt es verschiedene Implementationen von NIS. Die Implementierung von Thorsten Kukuk ist die derzeit am besten gepflegte Version, die auch in der GNU-`libc` enthalten ist.

Zu NIS existiert eine lesenswerte HowTo aus dem Linux Documentation Project, ebenfalls von Thorsten Kukuk verfasst. Daneben ist [Stern2001] das Standardwerk für die Administration von NFS und NIS.

23.2 NIS-Dienste als Client nutzen

Für die Nutzung eines NIS-Servers muss zunächst der Portmapper `rpc.portmap` gestartet werden. Wenn Sie NFS nutzen, so ist dies bereits der Fall. Andernfalls können Sie mit dem Befehl `rpcinfo -p` feststellen, welche RPC-Dienste Ihr Rechner anbietet. Mehr Informationen zu Remote Procedure Calls (RPC) finden Sie in Abschnitt 15.6, »Remote Procedure Call«.

Vor dem Start des Programms `ypbind` müssen Sie den Namen der NIS-Domain setzen. Das geschieht mit dem Befehl `domainname` oder `nisdomainname`. Die NIS-Domain hat nichts mit der Domain aus dem DNS zu tun. Sie dient vielmehr zur Unterteilung des Netzes in Gruppen von gemeinsam via NIS verwalteten Rechnern. Daneben wird manchmal der Domainname auch als Passwort für den Zugriff auf den NIS-Server betrachtet, was aber keinerlei Absicherung der Daten darstellt.

Das Programm `ypbind` sucht entweder mittels Broadcast nach einem geeigneten NIS-Server oder liest die Datei `/etc/yp.conf`. In dieser Datei steht der zu verwendende NIS-Server und/oder der NIS-Domainname. Ein Beispiel für diese Datei finden Sie im Listing 23.1. Wenn Sie dort einen Host-Namen eintragen, so muss dieser in der Datei `/etc/hosts` eingetragen und die richtige Suchreihenfolge in der Datei `/etc/host.conf` festgelegt werden, wenn die Namensauflösung auch mittels NIS durchgeführt werden soll.

Mit dem Programm `rpcinfo` können Sie nach einem NIS-Server suchen (`rpcinfo -b ypserv`) oder überprüfen, ob der Server tatsächlich reagiert (`rpcinfo -u` *`NIS-Server`* `ypserv 2`). Analog lassen sich diese Befehle auch verwenden, um nach NIS-Clients, die den Service `ypbind` gestartet haben, zu suchen. Dies kann bei der Fehlersuche hilfreich sein.

```
# Verwenden eines festen NIS-Servers
# und Setzen der Domain:
domain NIS-Domain server NIS-Server
# Setzen der Doamin und Suche nach einen NIS-Server
domain NIS-Domain broadcast
# Festen NIS-Server verwenden
ypserver NIS-Server
```

Listing 23.1 Die Datei `/etc/yp.conf`

Wenn der NIS-Client an einen NIS-Server mit `ypbind` gebunden ist, können einzelne Maps mit dem Befehl `ypcat` angezeigt werden. Für häufig verwendete Maps, wie z. B. `passwd.byname`, sind Aliasnamen definiert (hier `passwd`). Alle bekannten Aliasnamen werden mit dem Befehl `ypcat -x` angezeigt. Sie können mit dem Befehl `ypmatch` in einer Map nach dem Schlüssel suchen. Anwendungsmöglichkeiten hierfür finden Sie in Listing 23.2.

```
(linux):~> 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"
(linux):~> ypcat passwd
dosemu:*:409:10:DOS-Platte:/home/dos:/usr/bin/true
jochen:pNnKshbkY1gvk:405:6:Jochen:/home/jochen:/bin/tcsh
(linux):~> ypmatch jochen passwd
jochen:pNnKshbkY1gvk:405:6:Jochen:/home/jochen:/bin/tcsh
```

Listing 23.2 Beispiele für die Verwendung von NIS

Wenn Sie die Dateien `/etc/passwd` und `/etc/group` vom NIS-Server beziehen möchten, so müssen Sie in den Dateien ein Pluszeichen (+) in der letzten Zeile anfügen. Sie können die Einträge für Passwort, Realname, Home-Verzeichnis und Shell für alle NIS-User überschreiben, indem Sie in dieser Zeile die entsprechenden Werte eintragen. Wollen Sie den Wert für einen einzelnen User ändern, so tragen Sie den Benutzer mit *-User* und den gewünschten Werten in die Datei ein. Einige Beispiele, die am Ende der Passwortdatei eingetragen werden können, finden Sie in Listing 23.3.

```
+jochen::::::/bin/bash
-dosemu
+
```

Listing 23.3 Einträge in der Datei `/etc/passwd`

Ein Benutzername, der mit einem Pluszeichen (+) beginnt, wird mit den eventuell veränderten Werten zur Passwortdatei hinzugefügt. Beginnt der Name mit einem Minuszeichen (-), dann wird diese Benutzerkennung aus der NIS-Map für diesen Rechner entfernt.

Die Passwortdatei ist mit NIS nicht mehr lokal verfügbar, so dass ein Benutzer sein Passwort nicht mehr mit dem Befehl `passwd` ändern kann. Dazu dient nun der Befehl `yppasswd`, der mit dem entsprechenden Dämon (`yppasswdd`) auf dem NIS-Server kommuniziert. Dieser Server verändert die NIS-Datenbank auf dem Server und transportiert diese zu den Slave-Servern. Um die Verwendung von NIS für die Benutzer des Systems transparent zu machen, sollte der Systemadministrator das `passwd`-Programm umbenennen (z. B. in `passwd.old`) und einen (symbolischen) Link von `yppasswd` auf `passwd` erstellen.

Wenn Sie die Datei `/etc/hosts` mittels NIS verteilen wollen, müssen Sie das in die Datei `/etc/host.conf` eintragen. Einen Beispieleintrag finden Sie in Listing Listing 23.4. Beachten Sie, dass Sie in der Datei `/etc/yp.conf` nicht den Namen des NIS-Servers angeben sollten, um Rekursionen (bei der Namensauflösung muss der Name des Nameservers aufgelöst werden ...) zu vermeiden.

```
# Suchreihenfolge nach Host-Namen bestimmen
 möglich sind: hosts, dns und yp siehe auch nsswitch.conf
order hosts, yp
multi on
```

Listing 23.4 Die Datei `/etc/host.conf`

Mit dem Befehl `ypwhich` können Sie feststellen, welcher NIS-Server zurzeit verwendet wird. Wird `ypwhich` ohne Parameter aufgerufen, dann wird die Bindung des lokalen Rechners ausgegeben. Geben Sie als Parameter einen Host-Namen an, dann wird die Bindung dieses Rechners ausgegeben. Mit dem Programm `ypset` können Sie den NIS-Server wechseln.

23.3 NIS-Server

Wenn in Ihrem Netz bisher kein NIS-Server existiert, können Sie einen solchen auch unter Linux einrichten. Dazu benötigen Sie das Programm `ypserv`. In den aktuellen Distributionen sind die entsprechenden Programme in der Regel enthalten. Aktuellere Versionen finden Sie an der in der Info-Box angegebenen Stelle.

Vor dem Start des Servers muss, wie auch beim Client, die NIS-Domain mit dem Befehl `domainname` *`NIS-Domain`* gesetzt werden. Die Datenbanken für diese Domain werden im Verzeichnis `/var/yp/`*`NIS-Domain`* gespeichert. In den Quellen

der NIS-Server finden Sie die Datei `ypMakefile`, die Sie als `Makefile` in das Verzeichnis `/var/yp` kopieren. Die Datenbanken werden erstellt, indem Sie in dieses Verzeichnis wechseln und den Befehl `make` eingeben.

Im `Makefile` können Sie eintragen, welche Dateien des Systems als Master für die NIS-Datenbanken verwendet werden sollen. Teilweise bietet es sich an, die Systemdateien (z. B. `/etc/services`) zu verwenden, manchmal wird man auch eine neue Datei anlegen (z. B. `passwd`). Wenn Sie nicht alle Dateien via NIS verteilen wollen, können Sie die nicht gewünschten Einträge aus dem `Makefile` entfernen oder auskommentieren.

23.4 NIS-Slave-Server

Wenn ein NIS-Server ausfällt oder überlastet ist, so wirkt sich dies auf alle Clients aus. Daher wird man in größeren Installationen in jedem Fall einen oder mehrere Slave-Server vorsehen. Bei jeder Änderung an den NIS-Maps werden diese durch das `Makefile` automatisch an die Slave-Server übertragen. Der Verwaltungsaufwand entsteht also nur bei der erstmaligen Einrichtung.

Sicherheitshalber sollten noch zusätzliche Einträge in die Crontab auf dem Slave-Server aufgenommen werden, damit möglichst alle Änderungen im Master-Server auch dann repliziert werden, wenn der Slave-Server zu diesem Zeitpunkt nicht lief. Ein Beispiel für derartige Einträge finden Sie in Listing 23.5.

```
20 *     * * *    /usr/lib/yp/ypxfr_1perhour
40 6     * * *    /usr/lib/yp/ypxfr_1perday
55 6,18 * * *    /usr/lib/yp/ypxfr_2perday
```

Listing 23.5 Regelmäßige Übertragung der Maps zum Slave-Server

23.5 Tipps zu NIS

Wenn Sie auf dem Server `ypserv` und `ypbind` mit diesem NIS-Server starten, dann ist der Server auch sein eigener Client. Dies ist sinnvoll, wenn der Server sich möglichst wenig von den Clients unterscheiden soll, weil z. B. die lokale `/etc/passwd`-Datei mit anderen Mitteln verteilt wird. Auf der anderen Seite werden viele Clients außerdem als NIS-Slave-Server konfiguriert und an sich selbst gebunden. Damit sind alle NIS-Dienste immer noch verfügbar, auch wenn der Master-Server nicht mehr erreichbar ist.

23.6 Weitere NIS-Anwendungen

NIS kann außer für Systemtabellen auch für andere Anwendungen verwendet werden. So kann beispielsweise der Automounter `amd` so konfiguriert werden, dass er seine Maps von einem NIS-Server bezieht. In diesen Fällen kann man sich eine Menge Arbeit sparen oder zumindest sehr erleichtern.

Eine weitere Anwendung sind *Netgroups*, die in der Datei `/etc/netgroups` angelegt werden. Ein Eintrag beginnt mit dem Namen der Netgroup, gefolgt von Tripeln aus Host-Name, Benutzername und Domainname.

```
allhosts      (uranus,,) (venus,,) (merkur,,) (neptun,,)
mipshosts     (merkur,,) (neptun,,)
i386hosts     (uranus,,) (venus,,)
```

Listing 23.6 Eine einfache `/etc/netgroup`

Eine Anwendung für Netgroups ist die Verwaltung von Benutzern. In der Regel sind neben Rechnern, auf denen Anwender arbeiten, auch Server installiert, zu denen nur die Administratoren Zugang haben sollen. In diesem Fall kann man z. B. eine Netgroup mit den Administratoren einrichten und in der Datei `/etc/passwd` auf dem Server nur diese Anwender zulassen. Listing 23.7 zeigt ein Beispiel dafür.

```
+@admins::::::
+:*::::/etc/NoShell
```

Listing 23.7 Die Verwendung von Netgroups in der Datei `/etc/passwd`

Netgroups sind leistungsfähig, wenn Sie jedoch viele Rechner oder – schlimmer noch – viele Benutzer verwalten müssen, die in den verschiedensten Gruppen enthalten sein sollen, dann wird das unübersichtlich und fehlerträchtig. Aus meiner Erfahrung ist es besser, die Anzahl der Netgroups recht klein zu halten.

23.7 Sicherheitsüberlegungen zu NIS

Innerhalb eines überschaubaren Netzes ist der Einsatz von NIS einfach und bequem. Sie können viele Dateien zentral verwalten und dann automatisch den Clients die aktuellen Daten bereitstellen. Besonders vorteilhaft ist, dass alle Clients einheitliche Informationen zur Verfügung haben.

Aber wie sieht es mit der Sicherheit aus? Eher nicht so gut: Kenner bezeichnen NIS daher auch als »Network Intrusion System«. Zunächst kann sich jeder, der IP-Zugriff auf Ihren NIS-Server hat und die NIS-Domain kennt, dort alle Dateien, beispielsweise mit `ypcat`, abholen. Daher sollten Sie als NIS-Domain nicht die DNS-Domain nehmen. Ein großer Schutz ist das nicht, aber besser als nichts.

Die aktuelle Version des `ypserv` für Linux hat die Möglichkeit, den Zugriff in der Datei `/etc/ypserv.conf` auf bestimmte Clients einzuschränken. Beachten Sie, dass dies keine ultimative Sicherheit darstellt, aber das Leben der Hacker erschwert. Manche ältere YP-Server können mit Hilfe des Portmapper und dessen Access-Control geschützt werden.

Ein zweites großes Problem ist, dass viele NIS-Clients ihren Server mittels Broadcast suchen. Wenn man hier einen (relativ schnellen) Rechner in das Netz einschleust und die Anfragen beantworten lässt, dann kann man den Clients beliebige Daten unterschieben und so Zugriff auf die Clients erhalten. Dies lässt sich verhindern, indem man jedem Client in der Datei `/etc/yp.conf` vorgibt, an welchen Server er sich binden soll.

23.8 In die Zukunft mit NIS+?

Sun hat mit der Implementierung von NIS+ eine Erweiterung des NIS-Konzepts vorgestellt. In der `libc` sind die entsprechenden Routinen bereits enthalten, sie werden aber in der Binärversion nicht einkompiliert. Sie müssen daher die `libc` selbst übersetzen und dabei NYS einbinden. NYS enthält sowohl die Implementierung des traditionellen NIS als auch NIS+.

24 Dynamische IP-Konfiguration

24.1 Nutzen der dynamischen Konfiguration

Bei der Verwendung von TCP/IP muss jedem Rechner bzw. jeder Netzwerkkomponente eine (weltweit) eindeutige IP-Adresse zugeordnet werden. Die Verwaltung der vergebenen Adressen geschieht am besten durch den Netzverantwortlichen. Bei Änderungen an der Adressverteilung oder anderen Anpassungen ist es notwendig, diese Änderungen an allen betroffenen Netzgeräten durchzuführen. Je nach Art der Änderung, z. B. beim Einsatz eines neuen Nameservers, können sehr viele Rechner betroffen sein.

Gerade bei der Verwendung von Windows-Clients ist eine zentrale Wartung mittels `telnet` oder `rlogin` nicht möglich, so dass der Administrator jeden Rechner physisch aufsuchen muss. Um dies zu erreichen, gibt es mehrere Möglichkeiten. Für PPP-Verbindungen können die beiden beteiligten PPP-Programme die zu verwendenden IP-Adressen aushandeln. In lokalen Netzen verwendet man heutzutage das Dynamic Host Configuration Protocol (DHCP).

Ältere Systeme verwenden gelegentlich noch das Bootstrap Protocol (BOOTP). Bei wirklich alten Geräten findet man noch das Reverse Address Resolution Protocol (RARP, [rfc0903]). `bootp` werden wir uns noch genauer ansehen, da es etwas einfacher als DHCP zu konfigurieren ist, RARP wird hier nicht weiter beschrieben.

Ein weiteres Einsatzgebiet von DHCP sind Internetzugänge auf Basis von Cable-Modems oder Virtuelle Private Netze (VPNs). Hier werden von manchen Providern IP-Adressen dynamisch mittels DHCP vergeben. Außerdem kann DHCP sehr nützlich sein, wenn Sie beispielsweise einen Laptop in verschiedenen Netzwerken (zu Hause, in der Firma, bei einem Kunden oder auf einem Kongress oder einer Netzwerkparty) einsetzen. Hier jedes Mal das Netzwerk neu einzurichten, ist aufwändig und fehleranfällig. Eine Alternative kann das Programm `netenv` sein, das beim Booten abfragt, in welcher Umgebung man sich befindet (eine derartige Umgebung kann auch via DHCP ihre Daten erhalten).

Ein Nachteil einer dymanischen Konfiguration sei nicht verschwiegen: Die Fehlersuche im Netz kann schwieriger werden, wenn man alle IP-Adressen dynamisch vergibt und jeder PC bei jedem Booten eine andere Adresse bekommen kann. In diesem Fall ist es auch nicht mehr möglich, IP-basierte Dienste (z. B. Druckdienste) auf diesen Rechnern anzubieten. In diesen Fällen sollten Sie den entsprechenden Rechnern eine statische IP-Adresse zuteilen, diese aber mittels DHCP zugänglich machen.

24.2 Dynamic Host Configuration Protocol

BOOTP ist ein sehr altes Protokoll, der erste RFC wurde 1985 veröffentlicht. Seitdem haben sich viele Dinge verändert: TCP/IP wird in Unternehmen in viel größerem Umfang eingesetzt, die Anzahl der eingesetzten Rechner wächst ebenfalls. Die Anforderungen an eine IP-Konfiguration, die ein Server für die Clients bereitstellen kann, haben sich verändert. Ein Ergebnis war die Entwicklung von DHCP.

DHCP ist eine Weiterentwicklung von BOOTP, es enthält zusätzlich die Möglichkeit, automatisch IP-Adressen aus einem gewissen Bereich zu vergeben und, wenn diese nicht mehr benötigt werden, diese erneut (an einen anderen Rechner) zu vergeben. Dennoch ist DHCP mit BOOTP interoperabel, so dass auch ein BOOTP-Client mit einem DHCP-Server reden kann (und umgekehrt, wenn der DHCP-Server das kann und der Systemverwalter dieses entsprechend eingerichtet hat), siehe [rfc1534].

Die DHCP-Konfiguration in der Datei `/etc/dhcp.conf` besteht aus mehreren Teilen: Der globale Teil, der allgemein gültige Einstellungen festlegt, Optionen für verschiedene Subnetze und schließlich, sofern gewünscht, Parameter für einzelne Clients – ähnlich wie bei BOOTP. In Listing 24.1 finden Sie eine einfache Konfiguration.

```
# Für alle Netzwerke gültige Einstellungen
option domain-name "jochen.org";
option domain-name-servers jupiter.jochen.org, janus.jochen.org;
option routers 192.168.30.254;

option subnet-mask 255.255.255.0;
default-lease-time 3600;
max-lease-time 7200;

subnet 192.168.30.0 netmask 255.255.255.0 {
  range 192.168.30.1 192.168.30.30;
  option broadcast-address 192.168.30.255;
  option routers janus.jochen.org;
}

host hermes {
  hardware ethernet 08:00:5a:3b:aa:30;
  fixed-address hermes.jochen.org;
}

host gswi1164 {
  hardware ethernet 00:10:a4:94:a0:40;
  fixed-address gswi1164.jochen.org;
}
```

Listing 24.1 Eine einfache DHCP-Konfiguration

In meinem lokalen Netz verwende ich die dynamischen Anteile von DHCP nicht, um einfach z. B. mittels `ssh` auf die verschiedenen Rechner zugreifen zu können. Trotzdem behalte ich den Vorteil, mit einer einfachen Konfiguration auf dem DHCP-Server verschiedene IP-Konfigurationen verändern zu können. Besonders in großen Netzen findet man jedoch eine vollständig dynamische Konfiguration.

Der Ablauf einer DHCP-Sitzung sieht etwa wie folgt aus:

- Der Client sendet einen DHCP-Request mittels Broadcast auf das (lokale) Netz. Die einzige Information, die zu diesem Zeitpunkt bekannt ist, ist die Hardware-Adresse der im Client eingebauten Netzwerkkarte.
- Der oder die DHCP-Server empfangen die Anfrage. Anhand der Hardware-Adresse der Netzwerkkarte des Absenders können die notwendigen Informationen aus der Datei `/etc/dhcp.conf` gelesen und an den Client zurückgeschickt werden. Dies erfolgt noch nicht mit der echten IP-Adresse des Clients, sondern auf einer etwas niedrigeren Ebene, da der Client seine eigene IP-Adresse noch nicht kennt.
- Anhand der Antwort des DHCP-Servers kann der Client sein Netzwerk konfigurieren. Dazu gehören Netmask und Broadcast-Adresse, aber auch Nameserver, Log-Host oder Print-Server. Die Antwort erreicht den Client, weil dem Server die Hardware-Adresse des Clients bekannt ist und die Antwort direkt an diese Adresse gesendet wird.

  Wenn der Client mehrere verschiedene Antworten von verschiedenen DHCP-Servern erhalten hat, so wählt er eine der Antworten aus und verwendet diese für seine Konfiguration. Die Server können daher die IP-Konfiguration (Lease) noch nicht als endgültig vergeben markieren. Die Lease besteht aus der Konfiguration und der Laufzeit, nach der eine neue Konfiguration angefordert werden muss.
- Ist auf dem Client-Rechner das Betriebssystem noch nicht geladen, so kann dieses z. B. mittels `tftp` von einem Server geladen werden. Andernfalls wird der Boot-Prozess wie gewohnt fortgesetzt.
- Der Client sendet an den Server, dessen Lease verwendet wurde, eine entsprechende Nachricht. Damit gilt die Lease als belegt und wird bis zum Ablauf der Lease-Dauer nicht wieder vergeben.
- Läuft die Lease ab, so versucht der Client, diese zu erneuern. In der Regel wird der Server die Lease verlängern. Im Fehlerfall muss der Client sich eine vollständig neue Lease besorgen.

Wird ein Client neu gestartet und ist dann kein DHCP-Server verfügbar, so kann der Client eine möglicherweise noch vorhandene und gültige Lease verwenden. Unter Linux gibt es verschiedene DCHP-Clients, die diese Aufgaben wahrnehmen. Verbreitet sind der ISC-DHCP und das Programm `pump`.

Aktuelle Versionen des ISC-DHCP sind in der Lage, zusammen mit einer aktuellen BIND-Version dynamische DNS-Updates durchzuführen. Hier wird vom Client nach der Konfiguration eine Nachricht an den DNS-Server gesendet, um dort die Daten zu aktualisieren. Vorteil ist, dass bei einer dynamischen DNS-Konfiguration nur die Rechner im DNS enthalten sind, die eine aktuelle Lease haben.

Wenn Sie DHCP auf einem Rechner betreiben oder nutzen, auf dem ein Paketfilter aktiv ist, so müssen Sie die Regeln entsprechend anpassen. Sie müssen alle Pakete von der IP-Adresse `0.0.0.0` und UDP-Port 67 zur IP-Adresse `255.255.255.255`, UDP-Port 68 erlauben. Für die Antwort müssen Pakete von der IP-Adresse des DHCP-Servers (Port 67) zum jeweiligen DHCP-Client (Port 68) erlaubt werden. Außerdem müssen eventuell weitergeleitete Pakete auf Port 67 gestattet sein.

24.3 Das bootp-Protokoll

Mit dem Einsatz von BOOTP ist es möglich, viele TCP/IP-Konfigurationen an zentraler Stelle auf dem `bootp`-Server zu verwalten. Die Clients erhalten dann alle notwendigen Angaben beim Start vom `bootp`-Server.

Das `bootp`-Protokoll (definiert in [rfc0951], [rfc1542] und [rfc2132]) ermöglicht es, einem Rechner, abhängig von der Hardware-Adresse der Netzwerkkarte, eine IP-Adresse, die Netmask, die Nameserver u.v.a.m. zuzuordnen. Dies ist beim Booten von plattenlosen Geräten (wie z. B. X-Terminals) und beim Einsatz von TCP/IP unter DOS sinnvoll, um den Konfigurationsaufwand an den DOS-Rechnern gering zu halten. Auch einige Netzwerkcomputer (NCs) verwenden `bootp` zur Konfiguration.

Das `bootp`-Protokoll dient oft dazu, einen Rechner von Grund auf zu starten und zu konfigurieren. In jeder Phase sind nur sehr wenige Informationen verfügbar, die aber ausreichen, um die nächste Stufe zu starten. Es ist möglich, bereits vor dem Laden des Betriebssystems `bootp` zu verwenden, z. B. um den Server zu finden, von dem das Betriebssystem bei einem plattenlosen Gerät geladen werden soll. In diesem Fall muss die Netzwerkkarte mit einem zusätzlichen EPROM, das das `bootp`-Protokoll beherrscht, versehen werden.

Oft wird das Betriebssystem lokal auf der Festplatte des Clients installiert und nur die Netzkonfiguration mit `bootp` durchgeführt. Dadurch vermindert sich nur der Konfigurationsaufwand für die Netzwerkeinstellungen, alle anderen (betriebssystemabhängigen) Dinge müssen weiterhin auf dem gewohnten Weg eingestellt werden. Ein `bootp`-Request und dessen Bearbeitung wird in der Regel etwa wie folgt aussehen:

- Der Client sendet einen bootp-Request mittels Broadcast auf das (lokale) Netz. Die einzige Information, die zu diesem Zeitpunkt bekannt ist, ist die Hardware-Adresse der im Client eingebauten Netzwerkkarte.
- Der oder die bootp-Server empfangen die Anfrage. Anhand der Hardware-Adresse der Netzwerkkarte des Absenders können die notwendigen Informationen aus der Datei /etc/bootptab gelesen und an den Client zurückgeschickt werden. Dies erfolgt noch nicht mit der echten IP-Adresse des Clients, sondern auf einer etwas niedrigeren Ebene, da der Client seine eigene IP-Adresse noch nicht kennt.
- Anhand der Antwort des bootp-Servers kann der Client sein Netzwerk konfigurieren. Dazu gehören Netmask und Broadcast-Adresse, aber auch Nameserver, Log-Host oder Print-Server. Die Antwort erreicht den Client, weil dem Server die Hardware-Adresse des Clients bekannt ist und die Antwort direkt an diese Adresse gesendet wird.
- Ist auf dem Client-Rechner das Betriebssystem noch nicht geladen, so kann dieses z. B. mittels tftp von einem Server geladen werden. Andernfalls wird der Boot-Prozess wie gewohnt fortgesetzt.

Auf dem bootp-Server muss der bootpd-Dämon gestartet werden. Dies kann entweder unter der Kontrolle des inetd-Servers geschehen oder indem der bootpd als Dämon gestartet wird. Der Start durch inetd ist in der Testphase sinnvoll, da bei jedem Neustart des bootpd-Dämons die Datei /etc/bootptab neu gelesen wird. Läuft bootpd als eigenständiger Dämon, so muss er nach jeder Änderung neu gestartet werden.

Bei einer großen bootptab, wenn also viele Clients verwaltet werden sollen, benötigt das Neulesen dieser Datei recht viel CPU-Zeit, so dass der Start als Dämon sinnvoll ist. Tragen Sie in diesem Fall das Programm bootpd z. B. in die Datei rc.local ein. Beim System-V-init können Sie auch ein eigenes Start-Stop-Skript anlegen, mit dem Sie den Dämon auch direkt verwalten können.

Wenn Sie bootpd vom inetd starten lassen möchten, können Sie mit der Option -t festlegen, dass der Dämon mindestens die angegebene Anzahl Minuten läuft. Zum Start durch inetd nehmen Sie auf dem Server die Zeile aus Listing 24.2 in die Datei /etc/inetd.conf auf.

```
bootps  dgram  udp  wait  root  /usr/sbin/tcpd  bootpd
```

Listing 24.2 Neue Einträge in die Datei /etc/inetd.conf

Befindet sich der bootp-Server nicht im gleichen Subnetz wie der Client, so werden die Anfragen nicht über den Router übertragen, da Broadcasts nur im lokalen Netz erfolgen. Daher muss in jedem (Sub-)Netz, in dem bootp-Requests durchgeführt werden und kein bootp-Server existiert, ein Gateway (bootpgw) installiert oder der Router entsprechend konfiguriert werden. Dazu nehmen Sie

die Zeile aus Listing 24.3 in die Datei `/etc/inetd.conf` auf einem dann als Gateway fungierenden Rechner auf. Manche Router haben diese Funktion eingebaut, so dass Sie möglicherweise nur dort den entsprechenden Server eintragen müssen.

```
bootps dgram udp wait root /usr/sbin/tcpd bootpgw
```

Listing 24.3 Ein `bootp`*-Gateway*

Die gesamte weitere Konfiguration erfolgt in der Datei `/etc/bootptab` (Listing 24.4). Die Struktur dieser Datei erinnert an die der Datei `/etc/termcap`, siehe auch Abschnitt 4.6.1, »Die `termcap`-Datenbank«. Die einzelnen Felder werden durch Doppelpunkte (`:`) getrennt, Zeilen können mit einem Backslash (`\`) als letztem Zeichen in der nächsten Zeile fortgesetzt werden.

```
.default:hn:sm=255.255.255.0:dn=jochen.org:\
        :ds=192.168.30.254:

# Referenzeinträge für jedes Subnetz
.subnet31:sm=255.255.255.0:gw=192.168.31.1:tc=.default:
.dosemu  :sm=255.255.255.0:gw=192.168.31.1:tc=.default:

# TCP/IP im DOSEmulator
dosemu-1:tc=.dosemu:ha=646201907878:ip=192.168.31.101:
dosemu-2:tc=.dosemu:ha=646202907878:ip=192.168.31.102:
dosemu-3:tc=.dosemu:ha=646203907878:ip=192.168.31.103:

# reale (DOS-)Rechner
jupiter.jochen.org:tc=.subnet31:ht=ieee802:\
        :ha=10005AD262F8:ip=jochen.org:
```

Listing 24.4 Auszug aus der Datei `/etc/bootptab`

Ein Eintrag (auch Stanza genannt) besteht aus einem Namen und den zugeordneten Werten. Mit dem Schlüsselwort `tc` (Table Continuation) kann auf ein Template verwiesen werden. Diese Verweise sind auch mehrfach möglich, so dass die Definition des Verweises wieder einen Verweis enthalten kann. In Listing 24.4 wird zunächst ein Standardeintrag (`.default`) definiert, der die Domain (`dn`), den Nameserver (`ds`) und die Standard-Netmask (`sm`) enthält. Die im Subnetz gleichbleibenden Einträge werden in einem weiteren Template (`.subnet31`) festgelegt. Der Name eines Template sollte mit einem Punkt (`.`) beginnen.

Schließlich wird für jeden Rechner ein Eintrag erzeugt. Für jeden Rechner unterschiedlich sind die Einträge `ha` (Hardware-Adresse) und `ip` (IP-Adresse), die für die dynamische Zuordnung der IP-Adressen verwendet werden. Der Name eines Eintrags wird als Host-Name an den Client geschickt, falls dies durch die Konfiguration `hn` so eingerichtet wurde.

Beim Einsatz mancher DOS-Clients kann es notwendig sein, zusätzlich den Eintrag `vm=rfc1084` anzugeben. Damit wird diesen Clients die Antwort im Format gemäß [rfc1084] präsentiert, auch wenn sie nach einem anderen Format gefragt haben. Normalerweise wird in dem Format geantwortet, in dem ein Client seine Anfrage gesendet hat.

Für den Einsatz von plattenlosen Geräten, die remote gebootet werden, kann angegeben werden, wo die entsprechenden Boot-Dateien zu finden sind. Darüber hinaus gibt es eine Reihe von weiteren Daten, die an einen Client übertragen werden können. Eine Übersicht über die wichtigsten Einträge enthält die Tabelle 24.1, eine komplette Liste aller Einträge finden Sie in der Manpage `bootptab(5)`.

Eintrag	Bedeutung
ds	Adressen der DNS-Server
gw	Liste der Gateway-Adressen
ha	Hardware-Adresse
hn	Host-Name an den Client übertragen
ht	Hardware-Typ des Clients, siehe z. B. *http://www.iana.org/*
ip	IP-Adresse des Clients
sm	Subnetzmaske des Clients
vm	Vendor Magic Cookie
tc	Referenzeintrag (Template)

Tabelle 24.1 Die wichtigsten Einträge in der Datei `bootptab`

Nach der Konfiguration der Netzwerkprogramme holen sich einige X-Terminals und plattenlose Workstations den Kernel und alle weiteren Programme von einem Boot-Server. In der Regel wird das `tftp`-Protokoll (Trivial File-Transfer Protocol) verwendet, um den Kernel zu laden.

Da dieses Protokoll keine Berechtigungsprüfung kennt, sollte bei der Konfiguration mit besonderer Vorsicht vorgegangen werden. So ist unbedingt in der Datei `/etc/tftptab` einzutragen, welche Dateien übertragen werden dürfen, und der Dämon sollte nur Zugriff auf das Verzeichnis mit dem Boot-Image haben. Außerdem können Sie mit Hilfe des TCP-Wrapper den Zugriff nur für einige bestimmte Rechner gestatten. Näheres dazu finden Sie in der Manpage zu `tftp` und `tftpd`.

Es ist auch möglich, Rechner unter Linux mittels `bootp` zu konfigurieren. Dazu benötigen Sie die entsprechenden Clients (`bootpc`) und müssen die Skripten zur Netzwerkkonfiguration entsprechend anpassen. Beispiele für diese Skripten finden Sie im Paket `bootpc`, das auch ein Programm (`bootptest`) enthält, mit dem Sie die Konfiguration Ihres `bootp`-Servers testen können.

Die derzeit für Linux verfügbaren `bootpd`-Dämonen beherrschen nur einen Teil der DHCP-Funktionen. Es existiert ein freier DHCP-Server, der nun auch auf Linux portiert wurde. Genauso können Sie die IP-Konfiguration Ihrer Linux-Rechner mittels DHCP durchführen. Die Bezugsquelle hierfür ist im Info-Kasten am Ende des Kapitels angegeben.

In einem kommerziellen Netz muss unter anderem auf Ausfallsicherheit geachtet werden. Es ist daher möglich, in einem (Sub-)Netz mehrere `bootp`-Server zu betreiben. Dabei sollten diese allerdings mit denselben `bootptab`s konfiguriert sein. Dies kann durch eine Kopie auf beide Rechner (z. B. mittels `rdist`) oder NFS erreicht werden.

24.4 Variable Netzwerkkonfiguration

Gelegentlich ist der Einsatz von DHCP oder BOOTP nicht möglich oder erwünscht, z. B. in fremden statisch konfigurierten Netzen. Insbesondere wenn man mit seinem Laptop in verschiedenen Netzen arbeitet, so sind die Chancen nicht schlecht, dass mindestens eins davon eine statische Konfiguration verlangt. In diesem Fall ist der Einsatz von `netenv` sinnvoll.

`netenv` stoppt den Boot-Vorgang des Systems und fragt in einem Menü nach der zu verwendenden Netzwerkkonfiguration. Dabei sind sowohl statische als auch dynamische Konfigurationen möglich, also etwa zwei verschiedene vorgegebene Konfigurationen und eine Konfiguration mit DHCP.

Neben der reinen Netzwerkkonfiguration kann `netenv` beliebige Umgebungsvariablen setzen und diese später in Skripten aus der Datei `/etc/netenv/netenv` auslesen. Damit sind Konfigurationen für Drucker, Laptops mit/ohne Docking-Station, die Einbindung von verschiedenen Druckern oder beliebige andere Konfigurationen möglich.

Die `netenv`-Homepage hat die Adresse *http://netlink.sf.net*, dort finden Sie auch eine deutschsprachige Dokumentation.

24.5 Erkenntnisse

Der Einsatz von DHCP stellt in vielen Umgebungen eine echte Erleichterung dar. Anstatt IP-Adressen – zumindest den größten Teil davon – manuell zu verwalten, definiert man auf dem DHCP-Server einen entsprechenden Adressbereich und konfiguriert die Clients entsprechend. In der Regel ist man damit bereits fertig.

Problematisch sind nur Server, die IP-basierte Dienste anbieten. Diese sollten stets unter einem bekannten Namen und möglichst einer festen IP-Adresse verfügbar sein. Auch könnte hier der Ausfall des DHCP-Servers die Verfügbarkeit des Dienstes beeinträchtigen. Für viele Server wird man also weiterhin mit fester Adressvergabe arbeiten.

Einige Nachteile von DHCP sollen nicht verschwiegen werden: Insbesondere die Konfiguration von Anwendungen wird von DHCP nicht unterstützt. Das betrifft zum Beispiel den Zugriff auf Printserver oder Drucker (Abhilfe kann hier z. B. CUPS schaffen) oder verschiedene Proxy-Server (hier könnte man mit transparenten Proxies arbeiten). Auch kann DHCP – zumindest der dynamische Teil – nicht eingesetzt werden, wenn die Clients IP-basierte Dienste anbieten sollen oder zum Beispiel mit den IP-Adressen in der Firewall-Konfiguration verewigt sind.

Auf der anderen Seite wird bei LAN-Parties oder ähnlichen Veranstaltungen sehr gerne DHCP eingesetzt, da es die Arbeit des Netzverantwortlichen sehr vereinfacht. Eine Alternative für LAN-Parties ist die Vergabe von IP-Adressen mit Hilfe von Wäscheklammern [rfc2322], ähnlich wie bei der Frequenzvergabe für Modellflieger.

25 Anonymous-ftp-Server

25.1 Gründe für einen ftp-Server

Auf praktisch jedem vernetzten Unix-System ist der `ftpd`-Dämon installiert. In der Standardversion kann jeder zugelassene Benutzer Dateien vom und zum System übertragen. Ein Benutzer ist dazu berechtigt, wenn er eine gültige Benutzerkennung hat (dazu zählt auch, dass die Login-Shell in der Datei `/etc/shells` eingetragen ist) und nicht in der Datei `/etc/ftpusers` eingetragen ist.

Dies ist für einen Rechner im lokalen Netz durchaus akzeptabel. Die Benutzer, die den Server verwenden sollen, sind bekannt und im Zweifel ist eine neue Benutzerkennung relativ schnell angelegt und dem Anwender bekanntgegeben. In der Praxis sollten Sie in diesem Fall aber besser auf die Verwendung von `ssh` drängen. Die Vorteile sprechen für sich: verschlüsselte Übertragung und Authentifizierung mit Public-Key-Verfahren.

Oft wird es aber so sein, dass auch Benutzer ohne eigene Kennung bestimmte Dateien ansehen oder übertragen müssen. Dazu kann ein so genannter Anonymous-`ftp`-Server installiert werden. Sinnvoll ist ein derartiger Server, wenn man Freeware-Programme verteilen oder der Allgemeinheit Informationen oder Programmkorrekturen oder Ähnliches zur Verfügung stellen will. Oft wird hierfür ein `http`-Server (Web-Server) eingesetzt, das hat aber einige Nachteile: Uploads von Benutzern müssen mit speziellen Konstrukten ermöglicht werden und das Protokoll ist nicht zur Übertragung von größeren Datenmengen entwickelt worden.

Auf der anderen Seite verfügt praktisch jede Firma bereits über einen Web-Server und der wird dann verwendet. Ein weiterer Nachteil von `ftp` ist das Protokoll – es erfordert Kooperation mit den eventuell installierten Firewalls.

In diesem Kapitel werden wir die Konfiguration des `wu-ftpd` kennen lernen. Daneben gibt es noch eine ganze Reihe von `ftp`-Dämonen, die alle ihre Vor- und Nachteile haben. Wenn Sie auf der Suche nach einem `ftp`-Server für spezielle Anforderungen sind, dann sollten Sie sich die anderen Programme ebenfalls ansehen – der ProFTPD (*http://www.proftpd.org/* scheint eine interessante Alternative zu sein.

25.2 Überlegungen zur Konfiguration eines ftp-Servers

Bei der Einrichtung eines `ftp`-Servers sind einige Dinge zu beachten, damit nicht fremde Benutzer vollen Zugriff auf alle Dateien des Rechners erhalten. Man möchte zwar die Daten, um die es geht, mit einfachen Mitteln erreichbar machen, aber nicht alle Sicherheitsaspekte außer Acht lassen.

Es gibt verschiedene `ftp`-Dämonen, die zum Teil unterschiedlich konfiguriert werden. Achten Sie bei der Auswahl Ihres Servers darauf, dass Sie möglichst alle Aktionen der Benutzer mitloggen können, um nachzuvollziehen, was auf Ihrem Rechner passiert. Dies ist eine wichtige Hilfe, um die Sicherheit des Rechners zu gewährleisten und den Missbrauch als Zwischenlager für Raubkopien und ähnliches zu verhindern. Außerdem kann es sinnvoll sein, dass Sie einzelne Befehle verbieten und nützliche Features, wie einen suchbaren Index, einrichten.

Bei der Anmeldung unter dem Benutzernamen `anonymous` oder `ftp` wird in der Regel in ein spezielles Verzeichnis (z. B. `/home/ftp`, oft auch als `~ftp` geschrieben) gewechselt. Dazu wird der Systemaufruf `chroot(2)` verwendet, der dieses Verzeichnis zum neuen Root-Verzeichnis für diesen Prozess (und alle davon erzeugten) erklärt. Dadurch lässt sich verhindern, dass `ftp`-Benutzer, die sich anonym anmelden, Zugriff auf Dateien außerhalb dieses Verzeichnisses erhalten.

Nach der Eingabe des Benutzernamens `ftp` oder `anonymous` wird häufig erwartet, dass die E-Mail-Adresse des Benutzers eingegeben wird. Oft wird eine Anmeldung abgelehnt, wenn die Adresse nicht korrekt ist, z. B. das at-Zeichen (`@`) nicht enthält. Je nach Server wird die Mail-Adresse keiner oder einer wesentlich genaueren Prüfung unterzogen. Viele Server versuchen auch, den Namen des Client-Rechners vom Nameserver zu erfahren (bei `wu-ftpd` ist das eine Konfigurationsoption). Damit kann in den Log-Dateien der Name statt der IP-Adresse notiert werden und man achtet darauf, dass möglichst alle IP-Adressen auch DNS-mäßig verwaltet werden.

Eine andere Möglichkeit, um die Identität des Benutzers genauer zu bestimmen, ist die Verwendung von `ident` (Abschnitt 15.4, »Der `ident`-Dämon«). Dabei kommt es jedoch häufig zu Problemen mit PC-Benutzern, da es auf diesen Systemen keine Benutzerkennungen gibt. Auch sind Datenschutzfragen, die die Herausgabe der Benutzerkennung betreffen, nicht geklärt.

In dem gesamten Verzeichnisbaum unter `~ftp` sollten alle Verzeichnisse und Dateien für die anonymen Benutzer nur im Lesezugriff verfügbar sein. Damit soll verhindert werden, dass z. B. eine Datei `.rhosts` auf dem Rechner abgelegt wird, die einen interaktiven Zugriff ermöglicht. Das Einfachste ist, dass kein Verzeichnis der Benutzerkennung gehört, unter der der `ftp`-Server läuft.

Einige spezielle Verzeichnisse (wie incoming) können vom Schreibschutz ausgenommen werden, damit auch Benutzer ohne Kennung Dateien allgemein zur Verfügung stellen können. Diese Dateien sollten aber erst nach der Prüfung durch den Systemadministrator anderen Benutzern zur Verfügung stehen. Damit können Sie die unberechtigte Verbreitung von Daten über Ihren Rechner hoffentlich verhindern. Der verbreitete wu-ftpd stellt hierfür entsprechende Konfigurationsoptionen bereit.

Bei unberechtigter Benutzung sind Dateien mit dem Namen . . . oder mit Steuerzeichen im Namen sehr beliebt. Daher sollten Sie die Transferstatistik auf ungewöhnliche Namen durchsuchen, um zumindest eine (nachträgliche) Kontrolle ausüben zu können. Wieder bietet der wu-ftpd eine Option, mit der man gültige Dateinamen beschreiben kann, um derartige Probleme zu entschärfen.

Für Benutzer hilfreich sind spezielle Funktionen wie das automatische Zusammenfassen von Dateien eines Verzeichnisses in einem Archiv. Damit kann ein kompletter Verzeichnisbaum mit einem einzelnen Befehl übertragen und dann lokal weiterverwendet werden. Diese Funktion können Sie für einzelne Verzeichnisse wieder ausschalten, indem Sie die Datei .notar in dem Verzeichnis anlegen, was z. B. für das Wurzelverzeichnis sinnvoll ist. In einigen Clients (wie ncftp) ist die Funktion zum Kopieren ganzer Verzeichnishierarchien integriert, bei den meisten Programmen ist das jedoch nicht der Fall.

Eine ebenfalls sinnvolle Funktionalität ist das Wiederaufsetzen einer abgebrochenen Übertragung mit dem ftp-Befehl reget. Leider wird diese Funktion von den meisten Web-Browsern nicht verwendet. Außerdem muss sie vom Server unterstützt werden. Ein weiterer Grund für die Verwendung eines »normalen« ftp-Clients ist die Möglichkeit, dort spezielle Kommandos, wie die Suche im Indexfile oder das Ändern der Permissions, absetzen zu können. Diese Befehle werden mit QUOTE eingeleitet. Je nach ftp-Server werden unterschiedliche Kommandos unterstützt, Hilfe erhalten Sie mit dem Befehl QUOTE HELP.

25.3 Die Installation des wu-ftpd

Wie oben erwähnt, existieren verschiedene ftp-Dämonen. Zahlreiche ftp-Server im Internet verwenden den wu-ftpd, da er viele spezielle Features für einen anonymous-ftp-Server besitzt, ein ausführliches Logging durchführt und bei Sicherheitsproblemen schnell eine korrigierte Version installiert werden kann. Das überzeugende Argument ist hier die Verfügbarkeit des Quellcodes, so dass lokale Anpassungen durchgeführt werden können. Für den Systemadministrator ist die Installation dieses ftp-Dämons etwas aufwändiger als die Verwendung des

standardmäßig zu Unix mitgelieferten[1]. Dennoch ist es sinnvoll, einen Dämon zu verwenden, der im Quellcode existiert, um möglicherweise schnell auf Probleme reagieren zu können. Der `wu-ftpd` kann für fast alle Unix-Plattformen installiert werden, die notwendigen Anpassungen findet das `configure`-Skript selbst heraus. Wenn Sie beispielsweise den Namen der Log-Datei ändern möchten, so ist das mit Hilfe der Option `--with-log-dir=`*Pfad* möglich.

Der neue Dämon wird nach der Installation aktiviert, indem Sie die Datei `/etc/inetd.conf` ändern und den `inetd`-Server mit dem Signal `SIGHUP` neu initialisieren. Dazu dient der Befehl `killall -HUP inetd`. Anschließend steht ein `ftp`-Dämon für die Benutzer dieses Rechners zur Verfügung. Die Einrichtung eines `anonymous-ftp`-Servers wird im nächsten Abschnitt erläutert.

25.4 Administration eines ftp-Servers

Der Betrieb eines `anonymous-ftp`-Servers wirft, mehr noch als andere Services, einige Sicherheitsprobleme auf. Einige davon werden bei der Implementierung des `ftp`-Dämons berücksichtigt. Viele weitere Dinge beeinflussen die Sicherheit Ihres Servers. Bevor Sie den Server anderen Benutzern zur Verfügung stellen, sollten Sie in Ruhe die Manpages und die CERT-Advisories zu diesem Thema lesen.

Unter Linux werden in der Regel Shared-Libraries verwendet, um Programme möglichst klein zu halten. Nach dem `chroot` in das Verzeichnis `~ftp` ist der Zugriff auf die in `/lib` gespeicherten Bibliotheken nicht mehr möglich. Sie müssen also entweder die notwendigen Bibliotheken (`ld.so` und `libc.so`) zusammen mit den benötigten Dateien aus `/etc` (`ld.so.cache`, `group` und `passwd`) in das Verzeichnis `~ftp/lib` kopieren oder statisch gelinkte Programme verwenden, was in der Regel das Einfachste sein dürfte.

Für einen einfachen `ftp`-Server ohne spezielle Features benötigen Sie nur das Programm `ls`. Wollen Sie das automatische Packen von Verzeichnissen und Dateien ermöglichen, benötigen Sie ebenfalls `tar` und `compress` bzw. `gzip`. Diese Programme müssen in das Verzeichnis `~ftp/bin/ftp-exec` kopiert werden. Weitere Programme, die `ftp`-Benutzer mit dem Befehl `site exec` ausführen dürfen, müssen ebenfalls in das Verzeichnis `~ftp/bin/ftp-exec` kopiert werden. Oft findet man hier ein Suchprogramm für den Dateiindex. Seien Sie hier besonders vorsichtig, wenn Sie festlegen, welche Befehle fremde Benutzer auf dem Rechner ausführen dürfen. Insbesondere das Verändern von Berechtigungen mittels `chmod` kann äußerst gefährlich sein.

1. Viele Linux-Distributionen enthalten bereits den `wu-ftpd`, kommerzielle Unix-Systeme hingegen nicht.

Achten Sie bei der Einrichtung des Servers darauf, dass Sie keine Dateien oder Verzeichnisse für fremde Benutzer beschreibbar machen. Die einzige Ausnahme sollte das `incoming`-Verzeichnis sein. Beobachten Sie besonders die Übertragungen in dieses bzw. aus diesem Verzeichnis heraus. Oft werden `ftp`-Server als Zwischenlager für Raubkopien oder anderes illegales Material missbraucht.

Für die Anzeige des Dateieigentümers benötigt das Programm `ls` die Dateien `~ftp/etc/passwd` und `~ftp/etc/group`. Verwenden Sie hier nicht die Systempasswortdatei, sondern eine speziell aufbereitete, in der keine Passwörter gespeichert sind. Wenn Sie es wünschen, dann können Sie auch die Benutzerkennungen (und deren Namen) ändern, um diese nicht weltweit preiszugeben.

Im nächsten Schritt können Sie die speziellen Funktionen des Servers in den Dateien `ftpconversions`, `ftpusers` und `ftpgroups` einrichten. In der Datei `/etc/wu-ftpd/ftpconversions` (oder `/etc/ftpconversions`, je nach Installation) können Sie eintragen, welche Konversionen bei der Übertragung vorgenommen werden können und welche Programme dazu verwendet werden. Jede Zeile dieser Datei besteht aus acht durch Doppelpunkte getrennten Spalten, die eine Umwandlungsmethode beschreiben. Ein Beispiel finden Sie in Listing 25.1.

```
:.Z : :  :/bin/compress -d -c %s:T_REG:O_UNCOMPRESS:UNCOMPRESS
:   : :.Z:/bin/compress -c %s:T_REG:O_COMPRESS:COMPRESS
:.gz: :  :/bin/gzip -cd %s:T_REG:O_UNCOMPRESS:GUNZIP
:   : :.gz:/bin/gzip -9c %s:T_REG:O_COMPRESS:GZIP
:   : :.tar:/bin/tar -cf - %s:T_REG|T_DIR:O_TAR:TAR
:   : :.tgz:/bin/tar -czf - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TGZ
```

Listing 25.1 Die Datei `/etc/ftpconversions`

Die Bedeutung der einzelnen Felder wird in der Tabelle 25.1 erläutert. Beachten Sie, dass derzeit die Präfixoptionen noch nicht implementiert sind. Sie müssen die benötigten Programme (eventuell statisch gelinkt) in das Verzeichnis `~ftp/bin` kopieren, damit diese auch von anonymen Benutzern ausgeführt werden können.

Feld	Beschreibung
1	Zu entfernendes Namenspräfix
2	Zu entfernendes Namenssuffix
3	Anzuhängendes Namenspräfix
4	Anzuhängendes Namenssuffix
5	Das auszuführende Kommando
6	Dateitypen: `T_REG`, `T_DIR` oder `T_ASCII`
7	Optionen: `O_COMPRESS`, `O_UNCOMPRESS` oder `O_TAR`
8	Beschreibung

Tabelle 25.1 Beschreibung der Einträge in der Datei `/etc/ftpconversions`

Die Datei /etc/ftpusers (oder auch /etc/wu-ftpd/ftpusers) enthält die Namen der Benutzer, die sich nicht mit ftp anmelden dürfen. In der Regel werden das die Benutzer root, bin und andere Systemkennungen sein. In der Datei /etc/ftpgroups können verschiedene Benutzergruppen eingerichtet und mit einem Passwort versehen werden.

Die aufwändigste Konfigurationsdatei ist die Datei /etc/ftpaccess (siehe Listing 25.2), in der die wesentlichen Einstellungen zum ftp-Server vorgenommen werden. Ausführliche Beispiele finden Sie im Verzeichnis ./doc/examples des wu-ftpd-Quellcodes und eine aussagekräftige Dokumentation in der Manpage ftpaccess(5).

Der wu-ftpd kennt drei verschiedene Benutzertypen: anonyme, reale und Gast-Benutzer. Anonyme Benutzer sind alle, die sich mit dem Namen ftp oder anonymous anmelden. Die Rechte dieser Benutzergruppe sollten möglichst gering gehalten werden. Außerdem werden häufig Beschränkungen in der Anzahl der gleichzeitigen Anmeldungen vorgenommen, damit der Server und die Internetanbindung nicht überlastet werden. Reale Benutzer besitzen eine normale Unix-Kennung auf dem ftp-Server und die damit verbundenen Rechte.

Interessant ist die Gruppe der Gast-Benutzer. Diese haben eine Unix-Kennung, diese wird aber vom wu-ftpd künstlich eingeschränkt, so dass nicht der Zugriff auf alle Daten des Servers möglich ist. Ein Beispiel hierfür finden Sie in der unten folgenden Beschreibung der Konfigurationsoptionen in der Datei /etc/ftpaccess. Mit dieser Option ist es mit mäßigem Aufwand auch möglich, einzelne Kunden voneinander abzuschotten.

```
class   all   real,guest,anonymous  *

email root@localhost

loginfails 5

readme  README*    login
readme  README*    cwd=*

message /welcome.msg            login
message .message                cwd=*

compress        yes             all
tar             yes             all
chmod           no              guest,anonymous
delete          no              guest,anonymous
overwrite       no              guest,anonymous
rename          no              guest,anonymous
```

```
log transfers anonymous,real inbound,outbound

shutdown /etc/shutmsg

passwd-check rfc822 warn
```

Listing 25.2 Die Konfigurationsdatei `/etc/ftpaccess`

Die Datei `/etc/ftpaccess` (siehe Listing 25.2) enthält eine weitgehende Steuerung, welche Benutzer auf dem Server was tun dürfen. Außerdem kann das Verhalten des Servers in weiten Grenzen beeinflusst werden. Die folgende Liste beschreibt die wichtigsten Einträge der Datei `/etc/ftpaccess`.

`class` *Klasse Typliste Adressmuster ...*
: Benutzer unter `wu-ftpd` lassen sich in verschiedene Klassen einteilen. Diese Einteilung basiert auf dem angegebenen Adressmuster. Das kann entweder ein mit Wildcards versehener Domainname oder eine IP-Adresse sein. Es können mehrere `class`-Einträge erstellt werden; die Benutzer werden zu einer bereits bestehenden Klasse hinzugefügt. Als Eintrag in die `Typliste` sind die Werte `anonymous`, `guest` oder `real` erlaubt.

  Anhand der Einteilung der Benutzer in Klassen kann z. B. die Anzahl der gleichzeitigen Anmeldungen von anonymen Benutzern beschränkt werden (siehe auch die Option `limit`).

`deny` *Adressmuster Nachrichtendatei*
: Benutzer, deren IP-Adresse oder DNS-Name zum Muster `Adressmuster` passt, werden nicht zur Anmeldung zugelassen. Als Nachricht wird der Inhalt der Datei *Nachrichtendatei* angezeigt. Der Eintrag `!nameserved` schließt alle Clients aus, die nicht im Nameserver verwaltet werden. Es können mehrere `deny`-Einträge verwendet werden.

`limit` *Klasse Anzahl Zeiten Nachrichtendatei*
: Die Anzahl der Anmeldungen von Benutzern aus der Klasse *Klasse* (siehe die `class`-Option) wird in den angegebenen *Zeiten* auf die Anzahl *Anzahl* beschränkt. Als Nachricht wird der Inhalt der Datei *Nachrichtendatei* ausgegeben.

`guestgroup` *Gruppe ...*
: Die Mitglieder der Unix-Gruppe *Gruppe* werden vom `wu-ftpd` gesondert behandelt. In der Passwortdatei wird das Home-Verzeichnis speziell formatiert. Der Pfad wird an einer beliebigen Stelle durch die Zeichenfolge `/./` unterbrochen. Dies ist die Aufforderung an den `wu-ftpd`, mittels `chroot()` in das Verzeichnis vor der Zeichenfolge zu wechseln. Damit ist ein Zugriff auf Dateien außerhalb dieses Verzeichnisses nicht möglich. Dieser Pfad muss natürlich mit den benötigten Programmen und Bibliotheken gefüllt werden, wie das auch für den anonymen Bereich gilt. Anschließend wechselt `wu-ftpd` in das Verzeichnis nach der Zeichenfolge `/./`.

Wenn das Home-Verzeichnis, das sich in der Datei `/etc/passwd` befindet, `/home/guests/./hugo` lautet und der Benutzer zu einer durch `guestgroup` verwalteten Gruppe gehört, dann führt `wu-ftpd` zunächst einen `chroot()` in das Verzeichnis `/home/guests` aus. Dateien außerhalb dieses Verzeichnisses können in keinem Fall gelesen werden. Im zweiten Schritt wechselt `wu-ftpd` in das Verzeichnis `hugo`.

`loginfails` *`Anzahl`*
Nach der angegebenen Anzahl von Fehlversuchen wird die Verbindung unterbrochen.

`banner` *`Dateiname`*
Zeigt die Datei *Dateiname* vor dem Login an. Der Dateiname ist relativ zum normalen Root-Verzeichnis des Systems. Für virtuelle `ftp`-Server müssen Sie auch den Eintrag `virtual` beachten.

`email` *`E-Mail-Adresse`*
Setzt die Adresse des `ftp`-Verwalters, die an der Stelle des `%E` Platzhalters eingesetzt wird.

`message` *`Pfad Wann Klasse`* `...`
Häufig will man einen `ftp`-Benutzer auf verschiedene Dateien oder Informationen aufmerksam machen. Mit der Option `message` kann eine Datei zu bestimmten Zeitpunkten angezeigt werden. Dabei wird bei einer Anmeldung jede Datei nur einmal angezeigt. Beachten Sie, dass der Pfad *`Pfad`* relativ zu `~ftp` angegeben werden muss. Der Zeitpunkt *`Wann`* kann entweder `login` oder `cwd=`*`Verzeichnis`* sein. Als Verzeichnis kann ein Sternchen (`*`) angegeben werden. Dann gilt diese Option für alle Verzeichnisse. Die Einstellung lässt sich für verschiedene Benutzerklassen unterschiedlich vornehmen.

In der angezeigten Datei werden verschiedene Zeichenketten durch Variablen ersetzt. Tabelle 25.2 enthält eine Übersicht über die möglichen Ersetzungen.

Format	Beschreibung
%T	Die lokale Uhrzeit in der Form `Thu Nov 15 17:12:42 1998`
%F	Der freie Speicherplatz im aktuellen Verzeichnis
%C	Das aktuelle Verzeichnis
%E	Die E-Mail-Adresse des Verwalters aus der Datei `/etc/ftpaccess`
%R	Der Name des entfernten Rechners (des Clients)
%L	Der lokale Host-Name
%u	Der mittels [rfc1413] ermittelte Benutzername
%U	Der Anmeldename
%M	Maximale Benutzeranzahl dieser Benutzerklasse
%N	Aktuelle Benutzeranzahl in dieser Benutzerklasse

Tabelle 25.2 Ersetzungen bei der Anzeige der `message`*-Dateien*

`readme` *`Pfad Wann Klasse ...`*
Weist den Benutzer beim Login oder Wechseln eines Verzeichnisses auf die Datei *`Pfad`* hin, wenn diese existiert, und gibt den letzten Änderungszeitpunkt an. Für die Optionen *`Wann`* und *`Klasse`* gilt das für `message` Gesagte.

`log commands` *`Typliste`*
Loggt Kommandos der zu den angegebenen Benutzertypen gehörenden Benutzer.

`log transfers` *`Typelist Directions`*
Loggt die Dateitransfers der angegebenen Benutzertypen und die Richtung des Datentransfers.

`virtual` *`address`* `root|banner|logfile` *`Pfad`*
Beschreibt einen virtuellen `ftp`-Server. Für jeden virtuellen Server können drei Optionen eingestellt werden: das root-Verzeichnis, das Login-Banner und die Log-Datei (Listing 25.3). Damit ist es möglich, parallel zu einem virtuellen Web-Server auch `ftp` anzubieten und die Bereiche der einzelnen Kunden strikt voneinander zu trennen. Alle drei Einstellungen sind notwendig, damit ein virtueller Server aktiv wird.

```
virtual 192.168.1.100 root    /home/guests/hugo/ftp
virtual 192.168.1.100 banner  /home/guests/hugo/ftp/welcome
virtual 192.168.1.100 logfile /home/guests/hugo/log/
```

Listing 25.3 Definition eines virtuellen `ftp`*-Servers*

In Verbindung mit einer normalen `chroot`-Umgebung für den Benutzer kann man die Server verschiedener Kunden voneinander trennen, so dass jeder nur Zugriff auf seine Daten hat. Die Konfiguration ist insgesamt nicht einfach und verlangt daher eine gute Planung. Beachten Sie, dass Kunden keinen Schreibzugriff auf die Log-Dateien erhalten, sie könnten andernfalls beliebige Dateien anlegen oder Daten an diese anfügen.

`chmod|delete|overwrite|rename|umask yes|no` *`typelist`*
Erlaubt bzw. schränkt verschiedene Funktionen für die verschiedenen Benutzergruppen (`anonymous`, `guest` und `real`) ein.

`passwd-check none|trivial|rf822 (enforce|warn)`
Betreiber von `ftp`-Servern erwarten oft, dass das Passwort anonymer Benutzer deren Mail-Adresse ist. Das ist sinnvoll, um im Notfall (z. B. Einbruch in den Server) die Anwender informieren zu können. Andererseits kann hier jeder Anwender einen beliebigen Text eingeben. Eine Prüfung der Mail-Adresse auf Gültigkeit ist unmöglich. `wu-ftpd` kann die Adresse nicht prüfen (`none`), eine einfache Prüfung durchführen (`trivial`, die Mail-Adresse muss ein `@`-Zeichen enthalten) oder auf Übereinstimmung mit [rfc2822] prüfen. Je nach Auffassung des Systemverwalters kann eine Warnung ausgegeben oder die Anmeldung verweigert werden.

`path-filter` *typelist msgfile allowed_chars disallow ...*
Ein anonymous-`ftp`-Server kann von Fremden zum Austausch von Raubkopien missbraucht werden. Häufig werden als Namen für Dateien dann Steuerzeichen verwendet oder der Name beginnt mit einem Punkt und ist damit bei einem `ls` nicht sichtbar. Die Einstellung `path-filter` erlaubt die Einschränkung der möglichen Dateinamen abhängig vom Benutzertyp.

```
path-filter anonymous /etc/pathmsg ^[-A-Za-z0-9._]*$ ^\. ^-
```

Listing 25.4 Einschränken auf erlaubte Zeichen im Dateinamen

Der `path-filter` in Listing 25.4 sorgt dafür, dass anonyme Benutzer nur Dateien anlegen dürfen, die Buchstaben, Ziffern und die Zeichen Minus, Punkt und Unterstrich enthalten. Als weitere Einschränkungen dürfen Dateinamen nicht mit einem Punkt oder einem Minuszeichen beginnen. Bei einer Verletzung dieser Regel wird der Inhalt der Datei `/etc/pathmsg` ausgegeben.

`upload` *root dirglob* `yes|no` *owner group mode* `dirs|nodirs`
Erlaubt oder verbietet das Hochladen von Dateien in bestimmte Verzeichnisse. Dabei können der Dateieigentümer, die Gruppe und die Berechtigungen vorgegeben werden. Mit dem Eintrag `dirs` bzw. `nodirs` wird das Anlegen von Unterverzeichnissen erlaubt bzw. verboten.

Mit Hilfe der Datei `/etc/ftphosts` kann der Zugriff auf den `ftp`-Server weiter eingeschränkt werden. Der Zugriff kann für einzelne Kombinationen von Benutzern und Quellrechnern (Namensmuster bzw. Adressmuster) verboten oder erlaubt werden. Als Default ist der Zugriff von überall her erlaubt.

`allow` *Benutzer Adressmuster ...*
Erlaubt den Zugang mit dem Namen *Benutzer*, wenn dieser versucht, sich von einer zu *Adressmuster* passenden IP-Adresse oder Domain aus anzumelden.

`deny` *Benutzer Adressmuster ...*
Verbietet den Zugang mit dem Namen *Benutzer*, wenn dieser versucht, sich von einer zu *Adressmuster* passenden IP-Adresse oder Domain aus anzumelden.

Nützliche Befehle für die Verwaltung des `wu-ftpd` sind `ftpcount` (Anzeige der Auslastung der Benutzerklassen), `ftpwho` (Anzeige der eingeloggten `ftp`-Benutzer) und `ftpshut` (Stoppen des `ftp`-Servers). Diese Befehle dürfen, genauso wie der Befehl `xferstats` zur Anzeige der Transferstatistik, nur vom Benutzer `root` verwendet werden.

Das Verhalten von `ftpd` wird nicht nur mit Hilfe der Konfigurationsdateien beeinflusst, sondern auch durch die Kommandozeilenoptionen beim Aufruf. `ftpd` wird durch `inetd` gestartet, die Parameter sind also in der Datei `/etc/`

`inetd.conf` einzutragen und der `inetd`-Dämon ist neu zu starten. Ältere Unix-Systeme gestatten nur die Angabe von sehr wenigen Parametern in der Datei `/etc/inetd.conf`, so dass Sie eventuell ein Wrapper-Skript starten müssen, das dann den `ftpd` mit den passenden Parametern aufruft. Die wichtigsten Optionen sind:

- `-d` zur Ausgabe von Debug-Informationen.
- `-l` zum Loggen der `ftp`-Anmeldungen im `syslog`.
- `-L` zum Loggen aller Kommandos im `syslog`.
- `-a` zum Verwenden der Datei `/etc/ftpaccess`.
- `-A` zum Ignorieren der Datei `/etc/ftpaccess` (das ist die Standardeinstellung).
- `-i` bzw. `-o` zum Loggen eingehender bzw. ausgehender Dateien in der Datei `xferlog`.
- `-u` *`umask`* zum Setzen der `umask`, das ist nützlich bei privaten Gruppen, wie Red Hat sie verwendet.

Eine einfache Überprüfung, ob die Installation korrekt ist, können Sie mit dem Programm `./bin/ckconfig` durchführen. Dieses Programm weist Sie jedoch nicht auf Sicherheitsprobleme hin, die durch Ihre Konfiguration entstanden sein könnten.

25.5 Nach der Installation

Ist der Server dann konfiguriert und treten im täglichen Betrieb nur noch wenige Probleme auf, sollten Sie dennoch die Log-Dateien genau beobachten. `ftp`-Server können über das Netz angegriffen und manchmal zur Verbreitung von Raubkopien oder für andere illegale Tätigkeiten benutzt werden. Neben der Transferstatistik sind auch die Log-Dateien des Servers eine wichtige Informationsquelle.

Weitere Informationen zum Betrieb und zur Konfiguration eines `ftp`-Servers finden Sie in einigen CERT-Advisories. Auch Pakete wie COPS oder Tripwire, die Systeme nach bekannten Sicherheitslöchern oder unbefugten Veränderungen untersuchen, können Ihnen beim sicheren Betrieb eines `ftp`-Servers weiterhelfen.

Nehmen Sie die Sicherheit eines Servers nicht auf die leichte Schulter. Sie tragen für Ihren Rechner die Verantwortung, insbesondere wenn das System für illegale Zwecke, wie die Verbreitung von Raubkopien, missbraucht wird.

26 Internetzugang über Wählverbindungen

26.1 Allgemeines zu Wählverbindungen

Größere Firmennetze werden in der Regel mit einem Router oder einer Firewall (über einen Provider und mit Hilfe einer Standleitung) an das Internet angeschlossen. Wenn nur ein oder wenige Rechner mit dem Internet verbunden werden sollen, dann bietet sich die Anbindung über eine Modem- oder ISDN-Verbindung an. Dazu stehen unter Linux mehrere Möglichkeiten zur Verfügung. Welche davon Sie einsetzen können, sollen und müssen Sie mit Ihrem Provider besprechen. In der Regel sollte dieser Ihnen auch Beispiele für Konfigurationsdateien oder Skripten geben können. Selbst wenn er keine Unterstützung für Linux liefern kann, sollten Sie einen Blick in die in der Regel verfügbaren Skripten für Windows werfen – diese können Ihnen möglicherweise weiterhelfen.

Beachten Sie, dass Ihr System bei der Online-Verbindung ein beinahe vollwertiger Internetrechner ist. Damit können fremde Benutzer Zugriff auf Ihren Rechner und die dort angebotenen Dienste erlangen. Viele Probleme können Sie recht einfach in den Griff bekommen, andere erfordern einen sehr großen Aufwand. Einige Tipps finden Sie in der folgenden Übersicht:

- Setzen Sie den TCP-Wrapper ein und verhindern Sie Zugriffe von fremden Rechnern über dessen Zugriffsschutzmechanismen. Mehr Informationen zum TCP-Wrapper finden Sie im Abschnitt 15.3, »Der TCP-Wrapper (`tcpd`)«.
- Bieten Sie möglichst wenig Netzwerkdienste an. Untersuchen Sie die Datei `/etc/inetd.conf` auf nicht notwendige Dienste, die Sie dann deaktivieren sollten.
- Starten Sie nur die Dämonen, die Sie wirklich benötigen.
- Binden Sie Dämonen nicht an alle IP-Adressen bzw. Netzwerkschnittstellen, sondern an `localhost` oder Ihre interne IP-Adresse.
- Verwenden Sie gute Passwörter und versehen Sie nicht verwendete Benutzerkennungen mit einem Stern (`*`) im Passwortfeld in der Datei `/etc/passwd` bzw. `/etc/shadow`.
- Beobachten Sie die Log-Dateien Ihres Systems. Hier finden Sie häufig Spuren von misslungenen »Angriffen«. Manchmal ist allerdings auch nur die eigene Gedankenlosigkeit die Ursache für solche Log-Einträge.

- Wenn Sie sich immer noch unwohl fühlen, dann können Sie einige Firewall-Regeln mit `iptables` anlegen. Wenn Sie Ihr System allerdings wie in den anderen Punkten angegeben gesichert haben, dann sind keine Firewall-Regeln notwendig.

Vor der Verwendung von PPP sollten Sie mindestens das Loopback-Device konfiguriert haben. Außerdem müssen Sie den entsprechenden Treiber für PPP in den Kernel einkompilieren oder als Modul laden. Module können mit Hilfe des `kerneld`-Dämons bei Bedarf in den Speicher geladen werden. Werden sie nicht mehr benötigt, so werden die Module wieder entladen.

Dieses Kapitel kann nur eine einfache Einführung in das recht komplexe Thema geben. Die Manpages und HowTo-Dokumente des LDP sollten Sie auf jeden Fall zu Rate ziehen, genauso wie die Lektüre der angegebenen RFCs Ihnen möglicherweise weiterhelfen kann.

26.2 Point-to-Point Protocol

Derzeit wird für die meisten Internetverbindungen über Wählverbindungen das Point-to-Point-Protokoll (PPP) verwendet, das in [rfc1661] beschrieben wird. Dieses Protokoll basiert auf den Erfahrungen, die man mit der Implementierung und dem Einsatz von SLIP gemacht hat. Heutzutage unterstützt praktisch jeder Provider PPP. Über PPP können neben IP auch andere Protokolle wie IPX oder AppleTalk übertragen werden. Weitere Informationen zu PPP finden Sie in der PPP-HowTo des Linux Documentation Project und in der Manpage zu `pppd`.

Für die Verwendung von PPP benötigen Sie den entsprechenden Kernel-Support (einkompiliert oder als Modul). Außerdem muss auf Ihrem System der `pppd`-Dämon, passend zur Kernel-Version, installiert sein.

26.2.1 Dialup-IP mit PPP

Wenn die Modemverbindung zum PPP-Server bereits besteht und dort der PPP-Treiber geladen ist, dann wird die PPP-Verbindung mit dem Befehl in Listing 26.1 erstellt.

```
(linux):~# pppd /dev/ttyS1 38400 crtscts defaultroute
```

Listing 26.1 Starten des PPP-Dämons `pppd`

Die serielle Schnittstelle `/dev/ttyS1` wird in den PPP-Modus versetzt. Die Geschwindigkeit wird auf 38400 bps eingestellt. Dabei wird ein Hardware-Handshake verwendet. Dieser Link wird als Default-Route installiert. Das ist einer der Gründe dafür, dass `pppd` als `root` gestartet werden muss. Sollen auch nicht pri-

vilegierte Benutzer einen PPP-Link öffnen, so müssen Sie das Programm `setuid` starten (siehe `chmod(1)`). Sie können dabei die Ausführbarkeit auf eine Gruppe von Benutzern beschränken, siehe auch Abschnitt 5.6, »Berechtigungen für Dateien«.

Bevor auf dieser Verbindung Nutzdaten übertragen werden können, müssen die Verbindungsparameter, wie IP-Adresse und MTU (Maximum Transfer Unit), festgelegt werden. Wenn Sie keine speziellen Eintragungen vorgenommen haben, so wird dies mit dem Server mit Hilfe des IP Control Protocol (IPCP) ausgehandelt. Das ist ein Dialog zwischen den Rechnern, in dem die Verwendung einzelner Einstellungen vorgeschlagen und angenommen bzw. abgelehnt wird. Die Log-Datei zeigt, insbesondere wenn `pppd` mit der Option `debug` gestartet wird, den Austausch der entsprechenden Parameter. Die Log-Datei wird vom `syslogd` verwaltet. Dazu werden die Facility `daemon` und die Priorität `debug` verwendet. Danach ist die PPP-Schnittstelle (`ppp0` für die erste, `ppp1` für die zweite usw.) initialisiert und konfiguriert.

Die Parameter der Verbindung können aber durch die Verwendung einer Konfigurationsdatei weiter eingeschränkt werden. In der Datei `/etc/ppp/options` lassen sich globale Optionen für das System festlegen. Listing 26.2 zeigt ein einfaches Beispiel.

```
# Globale PPP-Optionen
auth       Verlange Authentifizierung bei jeder Verbindung
lock       UUCP-artige Lock-Dateien anlegen
```

Listing 26.2 Die Datei `/etc/ppp/options`

Mit dem Schlüsselwort `auth` legen Sie fest, dass bei jeder PPP-Verbindung eine Authentifizierung notwendig ist. Damit kann nur zu den Rechnern eine Verbindung aufgebaut werden, die lokal bekannt sind. Es existieren zwei Protokolle, das Password Authentification Protocol (PAP) und da Cryptographic Handshake Authentification Protocol (CHAP), die von PPP verwendet werden können. Falls möglich, verwenden Sie CHAP, da es sicherer als PAP ist, aber auf eine Authentifizierung sollten Sie auf keinen Fall verzichten.

Diese Art der Authentifizierung kann vom Server und vom Client verlangt werden. Damit können sich beide Seiten relativ sicher sein, dass der Partner auch derjenige ist, als der er sich ausgibt. Dieser Problembereich wird von SLIP nicht gelöst.

Verschiedene Optionen dieser Datei können mit anderen Einstellungen in der Datei `~/.ppprc`, `/etc/ppp/options.`*`ttyname`* oder bei ausgehenden Verbindungen in der Datei `/etc/ppp/peers/`*`peername`* verändert werden. Besonders sicherheitsrelevante Optionen können jedoch nicht mehr verändert werden. Sie sollten daher unbedingt eine globale Optionendatei erstellen.

Verbindungsaufbau mit chat

Die Modemverbindung kann mit Hilfe eines Chat-Skripts aufgebaut werden. In den `pppd`-Dämon ist eine entsprechende Skript-Sprache bereits integriert. Damit können Sie den Verbindungsaufbau weitgehend automatisieren.

In vielen Beispielen (z. B. im Skript `ppp-on`) wird das Chat-Skript als Kommandozeilenoption übergeben. In diesem Skript sind der Benutzername und das Passwort enthalten. Mit dem Befehl `ps -auxwww` kann jeder Benutzer eine ausführliche Prozessliste erstellen und das Passwort lesen, solange der `pppd`-Prozess nicht aus dem Speicher herausgeswappt wurde. Besser ist es, das Chat-Skript in einer Datei zu speichern und an den `chat`-Interpreter die Option `-f` *`Chat-Datei`* zu übergeben.

Dieses Chat-Skript ist ähnlich aufgebaut wie UUCP-Chats. Abwechselnd folgen ein String, der vom Modem kommt, und eine an das Modem zu sendende Zeichenkette. Der erste zu empfangende String muss leer sein, später können weitere Leerstrings folgen. Nach der Testphase sollten Sie, anders als in Listing 26.3, das Chat-Skript nicht in der Kommandozeile des `pppd` angeben, sondern in eine eigene Datei auslagern. Dies ist hier aus Platzgründen sowie aus Gründen der möglichst einfachen Darstellung nicht gezeigt.

```
pppd connect 'chat -v "" ATDTRufnummer \
                      CONNECT "" \
                      ogin: ppp \
                      word: Passwort' \
     /dev/ttyS1 38400 debug crtscts modem \
     defaultroute
```

Listing 26.3 Aufbau einer PPP-Verbindung

Heute wird man sich bei den meisten Providern nicht mehr mit Namen und Passwort anmelden und dann erst den `pppd` starten. Man wartet nur noch auf das »Connect« und überlässt die Authentifizierung den PPP-Dämonen. Diese werden dann aushandeln, ob CHAP oder PAP verwendet wird, und die Authentifizierung durchführen.

Auf meinem System wird als `connect`-Skript nicht direkt ein `chat`-Skript gestartet, sondern ein normales Shell-Skript. Dieses versucht für die verschiedenen Rufnummern meines Providers einen `CONNECT`. Gelingt dieser, so wird das Skript mit `exit 0` beendet, andernfalls wird ein Fehler an den `pppd` zurückgegeben. Das Skript finden Sie in Listing 26.4, ein Beispiel für ein aufgerufenes Chat-Skript sehen Sie in Listing 26.5.

```
#!/bin/sh
MAXTRIES=3                      # Anzahl der Versuche
host=$(basename $0)             # Wie wurden wir aufgerufen?
```

```
# Wiederhole, bis Anzahl der Versuche erreicht
for i in $( seq 1 $MAXTRIES ); do
    # Für jede Rufnummer ein Chat-Skript
    for j in /etc/ppp/chat/$host-*; do
        chat $CHATFLAGS -f $j   # Und einwählen
        rc=$?
        if [ $rc = 0 ]; then    # erfolgreich?
            exit 0              # ja, dann raus hier
        fi
    done
done
exit 1    # Alle Nummern ohne Erfolg versucht
```

Listing 26.4 Chat-Skript mit Wiederholung und mehreren Nummern

```
TIMEOUT 5
ABORT "NO CARRIER"
ABORT BUSY
ABORT "NO DIALTONE"
ABORT ERROR
"" ATZ
OK ATDTRufnummer
CONNECT ""
sername: Username
word: Passwort
"Your IP" "\d\d\d"
```

Listing 26.5 Chat-Skript als Eingabedatei für `chat`

Die Meldungen von `pppd` werden via `syslog` verwaltet. Dazu wird die Facility `daemon` verwendet. In welcher Datei Sie diese Meldungen wiederfinden, ist in der Datei `/etc/syslog.conf` festgelegt. Hier finden Sie das Debug-Protokoll, das mit dem Parameter `debug` angefordert wird. Dieses Protokoll hilft Ihnen, Konfigurationsproblemen auf die Spur zu kommen. Außerdem werden hier Beginn und Ende der Verbindungen festgehalten.

Das Programm `pppd` startet nach dem erfolgreichen Aufbau einer Verbindung das Skript `/etc/ppp/ip-up`. Als Parameter bekommt dieses Skript den Namen der Netzwerkschnittstelle, das Modem-Device, die Geschwindigkeit der Verbindung sowie die eigene und die fremde IP-Adresse. Hier kann z. B. automatisch die Mail mittels `fetchmail` abgeholt werden. Auf meinem System werden Mail und News mittels UUCP ausgetauscht, so dass hier `uucp` über eine TCP-Verbindung gestartet wird. Analoge Skripte gibt es für das Ende der Verbindung (`/etc/ppp/ip-down`) und für andere Protokolle (`/etc/ppp/ipx-up` und `/etc/ppp/ipx-down`). Bei der Debian-Distribution können Sie eigene Erweiterungen der Skripte im Verzeichnis `/etc/ppp/ip-up.d` speichern, diese bleiben dann bei Upgrades erhalten.

Ausführliche Dokumentationen zu PPP finden Sie in der Manpage zu `pppd(8)`, `chat(8)`, in der PPP-HowTo des Linux Documentation Project und in den im `pppd`-Paket enthaltenen README-Dateien.

Beenden einer PPP-Verbindung

Um die PPP-Verbindung zu beenden, senden Sie dem `pppd` das Signal `SIGINT` mit dem Befehl `kill`. Die PPP-Treiber werden dann die Verbindung geordnet abbauen und das Modem legt auf. Die Prozess-ID des PPP-Dämons finden Sie in der Datei `/var/run/pppN.pid`. Dabei steht *N* für die Nummer des PPP-Device. Die Debian-Distribution stellt hierfür das Skript `poff` zur Verfügung.

Sie können mit dem Eintrag `idle` in der PPP-Konfiguration eine Anzahl von Sekunden festlegen, nach denen der `pppd` eine unbenutzte Verbindung automatisch beendet. Mit dem Schlüsselwort `active-filter` können Sie einen Filter festlegen, die Syntax entspricht den Regeln von `tcpdump`.

26.2.2 Authentifizierung

In der Regel müssen Sie sich bei Ihrem Provider mit Benutzerkennung und Passwort identifizieren, bevor dort der PPP-Treiber gestartet wird. Der Anrufer kann sich nicht sicher sein, dass auf der Gegenseite auch tatsächlich sein Provider ist. Die Authentifizierung in der Gegenrichtung wird bisher (auch im täglichen Leben) selten verwendet. So nimmt zwar jeder an, dass ein Geldautomat sich korrekt verhält, aber eine Garantie dafür existiert nicht. Hat jemand eine Attrappe aufgestellt, so merkt man erst nach der Eingabe der PIN, dass der Automat kein Geld herausgibt, sondern nur zur Erlangung der PIN diente.

Wenn in der Optionsdatei `auth` eingetragen ist, kann ein PPP-Link nur nach der Authentifizierung des Partners aufgebaut werden. Es sind zwei Verfahren implementiert, das *Cryptographic Handshake Authentification Protocol* (CHAP, [rfc1994]) und das *Password Authentification Protocol* (PAP). Nach Möglichkeit sollten Sie CHAP verwenden, aber PAP ist besser als nichts.

Cryptographic Handshake Authentification Protocol

Das Cryptographic Handshake Authentification Protocol basiert auf dem Austausch von geheimen Phrasen. Jeder Partner, egal ob Anrufer oder Angerufener, kann die Authentifizierung verlangen. Dabei kann man durch Optionen darauf bestehen, dass CHAP verwendet wird, oder auch PAP zulassen.

Wenn ein Partner sich authentifizieren muss, dann sucht er die entsprechende Phrase aus der Datei `/etc/ppp/chap-secrets`. Dabei benötigen beide Partner beide Phrasen, um sich gegenseitig identifizieren zu können. Manche ISDN-Router unterstützen hier nur die Konfiguration einer Phrase. Genaueres kann Ihnen Ihr Provider bzw. Netzwerkverwalter sagen.

Wenn der Rechner `jupiter` eine PPP-Verbindung zu `typhon` aufbaut und dabei CHAP verwendet wird, muss die Secrets-Datei wie in Listing 26.6 aufgebaut sein.

```
#remote  local  secret                        IP address list
jupiter    typhon   "Hey, what's the password?" 192.168.30.202
typhon     jupiter  "Ken sent me"               192.168.31.151
```

Listing 26.6 Die Datei `/etc/ppp/chap-secrets`

Bei CHAP wird eine Challenge gesendet, die mit dem Secret verschlüsselt ist. Wenn die Antwort des Partners mit dem lokal berechneten Ergebnis identisch ist, dann war die Authentifizierung erfolgreich. Das Feld `IP-Adresse` kann dazu verwendet werden, die Verbindung nur bei bestimmten IP-Adressen endgültig zu aktivieren.

Password Authentification Protocol

Ist die Authentifizierung mit CHAP nicht möglich, so sollten Sie zumindest das Password Authentification Protocol (PAP) verwenden. Auch in der Datei `/etc/ppp/pap-secrets` sind das eigene und das fremde Passwort gespeichert.

```
#account remote     password          IP address list
jupiter    typhon       "kw9n3k"          10.10.10.2
typhon     jupiter      "auemf6"          10.10.10.1
```

Listing 26.7 Die Datei `/etc/ppp/pap-secrets`

Der erste Eintrag in Listing 26.7 wird bei der Anmeldung an den Rechner `typhon` verwendet. Wenn der Rechner `typhon` sich mittels PAP authentifiziert, wird die zweite Zeile zur Prüfung herangezogen. Beachten Sie, dass manche Systeme nur ein Passwort bzw. ein Secret für jeden Kommunikationspartner kennen. In diesem Fall ist das Passwort für beide Richtungen gleich.

Bei PAP geht das Passwort im Gegensatz zu CHAP im Klartext über die Leitung. In manchen Fällen ist das nicht erwünscht, so dass man lieber CHAP verwendet. Als weiteres Protokoll existiert noch MS-CHAP, hier wird das Passwort zwar verschlüsselt übertragen, aber nicht mit einer Challenge gesichert. Damit kann ein Angreifer ein abgehörtes Passwort bei einer Replay-Attacke verwenden.

26.2.3 Automatischer Verbindungsaufbau

Bei den bisher vorgestellten Verfahren muss die Verbindung vom Benutzer manuell aufgebaut und beendet werden. Dies ist unkomfortabel und kann dazu führen, dass eine bestehende Verbindung »vergessen« wird und dauerhaft Telefongebühren kostet. Wesentlich komfortabler ist es, wenn die Verbindung dynamisch bei Bedarf aufgebaut wird, ohne dass der Benutzer hierfür etwas unternehmen muss. Wird die Verbindung eine Zeit lang nicht mehr benutzt, kann sie ebenfalls automatisch beendet werden. Früher konnte man hierfür `diald` verwenden, seit einiger Zeit ist diese Funktion in den PPP-Dämonen integriert. Die notwendigen Konfigurationsoptionen finden Sie in Listing 26.8.

```
# Dial-On-Demand
demand
# Nach 900 Sekunden Leerlauf auflegen
idle 900
# Nach Fehlern/Auflegen erstmal 900 Sekunden warten
# (gilt nicht für Ende durch "idle")
holdoff 900
# Filter: Welche Pakete gelten als "aktiv" für "idle"
# active-filter <tcpdump-syntax>
```

Listing 26.8 PPP-Konfiguration für Dial-On-Demand

Das Schlüsselwort `demand` versetzt `pppd` in den Dial-On-Demand-Modus. Wenn die Leitung für die nach `idle` genannte Anzahl Sekunden nicht mehr benutzt wurde, so wird die Verbindung beendet. Sollte die Verbindung abbrechen, dann wird der nächste Anwahlversuch erst dann gestartet, wenn die Anzahl Sekunden nach `holdoff` verstrichen ist.

In der Praxis werden Sie verschiedene Pakete nicht zählen wollen, so dass die Verbindung auch dann abgebaut wird, wenn noch diese Pakete übertragen wurden. Mit dem Schlüsselwort `active-filter` können Sie einen Ausdruck angeben, welche Pakete ignoriert werden sollen. Der Ausdruck entspricht denjenigen von `tcpdump` (siehe Abschnitt 28.3.1, »IP-Pakete untersuchen«).

26.2.4 Linux als Einwahlserver

Für die Konfiguration eines Einwahlservers müssen Sie zunächst einen Anruf entgegennehmen – das geht am besten mit dem Programm `mgetty`. Starten Sie das Programm in der Datei `/etc/inittab` (Listing 26.9). Weitere Einstellungen können Sie in der Datei `/etc/mgetty/mgetty.conf` vornehmen, in den meisten Fällen wird das nicht nötig sein. Wenn Sie einen Fehler suchen müssen, dann tragen Sie in diese Datei einen höheren Wert für `debug` ein und probieren Sie es erneut. Ein ausführliches Log finden Sie dann in der Datei `/var/log/mgetty/mg_`*`tty`*`.log`.

```
# Starten des mgetty auf /dev/ttyS1 (COM2:)
S1:23:respawn:/sbin/mgetty ttyS1
```

Listing 26.9 Aufruf vom `mgetty` *in der* `/etc/inittab`

Bei einem Anruf nimmt `mgetty` diesen an und sendet einen `login`-Prompt. Nun gibt es mehrere Möglichkeiten:

- Der Benutzer gibt Namen und Passwort ein und `mgetty` startet die Login-Shell des Anwenders. Wenn der Benutzer über geeignete Rechte verfügt, dann kann er auch später noch eine PPP-Sitzung starten.
- Der Anruf kommt von einem Faxgerät, dann wird `mgetty` das Fax annehmen (sofern das Modem faxfähig ist) und per Mail an den Faxadministrator (meist der Benutzer `root`) senden.
- Mit der Voice-Erweiterung kann `mgetty` als Anrufbeantworter fungieren – und wenn das Modem es unterstützt, auch automatisch zwischen Daten, Fax und Voice unterscheiden.
- Nun der Fall, der uns eigentlich interessiert: Wenn in der Datei `/etc/mgetty/login.conf` der Eintrag `AutoPPP` aktiviert ist, dann kann der Anrufer direkt mit PPP starten. Das ist das, was die meisten Provider und Clients tun. Die Authentifizierung erfolgt in diesem Fall mittels PAP oder CHAP.

Wenn Sie die Benutzer bereits auf dem Linux-System angelegt haben, dann können Sie PAP verwenden und in der PPP-Konfiguration `login` eintragen. In diesem Fall wird die Authentifizierung die Linux-Benutzerdatenbank verwenden. Provider und Firmen verwenden in der PAM-Konfiguration (`/etc/pam.d/ppp`) möglicherweise RADIUS oder LDAP zur Authentifizierung. Mit diesen Protokollen können Sie die Benutzer zentral verwalten und den Zugriff auch über mehrere verschiedene Router oder Einwahl-Server gestatten.

Im letzten Schritt können Sie dem Anrufer mit der Option `ms-dns` die Nameserver dynamisch mitteilen. Damit muss man bei einem Windows-System nicht mehr als die Rufnummer, Namen und Passwort eintragen, um PPP zu verwenden.

Auf dem Server muss IP-Routing aktiviert sein, dazu schreiben Sie beim Systemstart den Wert `1` in die Datei `/proc/sys/net/ipv4/ip_forward`. Der Kernel wird die per Proxy-ARP »eingefangenen« Pakete an die richtigen PPP-Clients weiterreichen.

26.2.5 Weitere Möglichkeiten mit pppd

In der Regel werden Sie mit PPP nur einen einzelnen Rechner anbinden, dabei wird der Einwahl-Server Proxy-ARP verwenden, damit kein eigenes Transfernetz notwendig ist. Sollten Sie in die Verlegenheit kommen, Netze mit PPP verbinden zu müssen, so geht das natürlich auch.

Wenn Sie Probleme mit der PPP-Verbindung haben, ergänzen Sie die Option `debug` in der PPP-Konfiguration und sehen Sie in das Systemlog. Zusammen mit der Manpage `pppd(8)` sollten Sie eine Option für Ihren Fall finden.

Viele Unternehmen erhöhen die Sicherheit der Einwahl durch eine Callback-Konfiguration. Dabei wird nach der Authentifizierung entweder eine vorbestimmte oder vom Client angegebene Rufnummer zurückgerufen. Damit ist in den Log-Dateien der Standort des Anrufers (hoffentlich) nachvollziehbar und die Kosten werden von der Firma getragen. In der PPP-Konfiguration richten Sie Callback mit dem Schlüsselwort `callback` ein. Sie können auf den Rückruf entweder mit Hilfe eines Chat-Skripts warten (die Debian-Distribution enthält Beispiele dazu) oder einen `mgetty` laufen lassen.

In der Manpage finden Sie außerdem weitere Hinweise zur Systemsicherheit und zu den verschiedenen Authentifizierungsmöglichkeiten. Wenn Sie einen Einwahl-Server betreiben wollen, dann schauen Sie unbedingt in die Manpage.

26.2.6 Konfiguration mit wvdial

Wenn Ihnen das Ändern von Konfigurationsdateien zu lästig ist, dann ist vielleicht `wvdial` etwas für Sie. Eine erste Konfiguration erstellen Sie mit dem Befehl `wvdialconf /etc/wvdial.conf`. Ihr System wird nach einem Modem durchsucht und eine passende Konfiguration wird in der Datei `/etc/wvdial.conf` abgelegt. Nun müssen Sie nur noch Rufnummer, Namen und Passwort in dieser Datei ergänzen und können mit `wvdial` die Verbindung zu Ihrem Provider herstellen – es sollte einfach funktionieren.

Wenn Sie das Programm mit Strg+C beenden, wird auch die Verbindung zum Provider abgebaut. Als grafische Oberfläche stehen die Programme `kwvdial` und `kppp` zur Verfügung.

26.3 ISDN anstelle eines Modems

Neben den bisherigen analogen Telefonanschlüssen sind digitale ISDN-Anschlüsse inzwischen weit verbreitet. Diese haben für Computeranwender den Vorteil, dass 64 Kbit je Sekunde in jeder Richtung übertragen werden können. Erforderlich ist in diesem Fall eine ISDN-Karte in Ihrem PC.

Im Gegensatz zur Verwendung eines Modems werden die Daten direkt digital übertragen und müssen daher nicht auf ein Trägersignal aufmoduliert werden. Daher sind ISDN-Verbindungen in der Regel sehr stabil und gleichbleibend schnell. Reicht diese Übertragungsrate des einen ISDN-Kanals nicht mehr aus, kann eine Kanalbündelung vorgenommen werden (wenn der Provider dies un-

terstützt). Außerdem ist der Verbindungsaufbau sehr schnell, so dass ein »Dial-on-Demand« einfach zu realisieren ist und auch kaum Verzögerungen für die Benutzer verursacht.

In Deutschland gibt es zwei Arten eines ISDN-Anschlusses: nationales und Euro-ISDN. Neue Anschlüsse werden normalerweise als Euro-ISDN geschaltet. Bei der Übersetzung des Kernels werden Sie gefragt, für welche ISDN-Art Sie den Kernel kompilieren wollen. Außerdem müssen Sie den Treiber für Ihre spezielle ISDN-Karte anwählen. Beachten Sie, dass derzeit nur aktive Karten eine Zulassung haben, da bei diesen die Prüfung unabhängig von der verwendeten Software erfolgt. Bei passiven Karten wird die Kombination von Karte und Software geprüft, aber für die Linux-Treiber ist bisher keine Prüfung erfolgt.

Diese Treiber ermöglichen verschiedene Verbindungstypen. Zunächst gibt es eine »Modememulation« (X.75), die z. B. mit `minicom`, `kermit` oder `uucp` benutzt werden kann. Über diese Verbindung kann auch SLIP oder (asynchrones) PPP übertragen werden. Der einzige Unterschied zur Modemkonfiguration ist, dass ein `ttyI*`-Gerät anstelle von `ttyS*` verwendet wird. Der Begriff »Modememulation« bezieht sich darauf, dass der lokale Computer die ISDN-Karte mit `AT`-Befehlen steuern kann, wie man das von einem Modem gewöhnt ist. Leider ist eine Verbindung zu Systemen, die nur über ein Modem verfügen, nicht möglich.

Wenn Ihr Provider X.75 anbietet, dann unterscheidet sich die PPP-Konfiguration praktisch nicht von derjenigen bei der Verwendung eines Modems. Allerdings bieten die meisten Provider diese Art der Verbindung nicht an. Daher wird im Folgenden nicht genauer auf diesen Verbindungstyp eingegangen.

26.3.1 Synchrones PPP

Als zweiten Verbindungstyp gibt es synchrones PPP (syncPPP). Hier erfolgt die Verbindung nicht über ein simuliertes Modem. Damit ist auch eine Authentifizierung mittels Benutzernamen und Passwort in einem Chat-Skript nicht möglich. Stattdessen wird CHAP oder PAP verwendet, außerdem kann die Verbindung so konfiguriert werden, dass nur Anrufe von bekannten Rufnummern entgegengenommen werden.

Es ist durchaus möglich, dass Ihr Provider Ihnen nicht mitteilen kann, welche Art von PPP er unterstützt. In diesen Fällen bleibt Ihnen nur gezieltes Raten (mit ziemlicher Sicherheit ist es syncPPP) und Ausprobieren. (Ist ein Chat-Skript erforderlich? Dann ist's wohl asynchrones PPP.)

Für die synchrone PPP-Verbindung benötigen Sie einen speziellen PPP-Dämon, den `ipppd`. Dessen Konfiguration ist praktisch identisch mit der des normalen `pppd`. Alle wesentlichen Konfigurationen werden mit dem Programm `isdnctrl` durchgeführt. Das Listing 26.10 zeigt einige wichtige Parameter, mehr finden Sie in der entsprechenden Manpage.

```
# Erzeugen des ISDN-Interfaces ippp0
/sbin/isdnctrl addif      ippp0
# Welche Rufnummer soll gewählt werden?
/sbin/isdnctrl addphone   ippp0 out 4711
# Setzen der eigenen »Endgeräte-Auswahl-Ziffer« (MSN)
/sbin/isdnctrl eaz        ippp0 4712
# Auflegen nach 60 Sekunden ohne Datenverkehr
/sbin/isdnctrl huptimeout ippp0 60
# Synchrones PPP verwenden
/sbin/isdnctrl encap      ippp0 syncppp
# Keine Anrufe entgegennehmen
/sbin/isdnctrl secure     ippp0 on
```

Listing 26.10 Einsatz des Programms `isdnctrl`

Wichtig ist der Parameter `huptimeout`, der standardmäßig auf zehn Sekunden eingestellt ist. Wenn Sie ihn nicht ändern, dann wird die Verbindung nach sehr kurzer Untätigkeit bereits abgebaut. Mit `isdnctrl` können Sie außerdem die weiteren Verbindungsparameter wie Encapsulation und ISDN-Protokoll wählen.

Anschließend wird das ISDN-Subsystem automatisch Verbindung mit dem Provider aufnehmen, wenn ein entsprechendes IP-Paket gesendet werden soll. Hier stehen leider nicht dieselben ausgefeilten Auswahlmöglichkeiten wie bei `pppd` zur Verfügung, dafür kann der Verbindungsaufbau hier sehr viel schneller stattfinden.

Die Verwaltung der Netzwerkschnittstelle `/dev/ippp`*N* übernimmt ein speziell angepasster PPP-Dämon, das Programm `ipppd`. Listing 26.11 zeigt ein Beispiel für den entsprechenden Aufruf.

```
# Interface konfigurieren
/sbin/ifconfig ippp0 192.168.99.5 \
        pointopoint -arp metric 3
# Route setzen
/sbin/route add default dev ippp0
# ipppd starten, ein Prozess kann viele Interfaces betreuen
/sbin/ipppd name zeus remotename ppp.example.org -bsdcomp \
        debug -pred1comp \
        -vj 192.168.99.5:/dev/ippp0
```

Listing 26.11 Start des `ipppd`*-Dämons*

Weitere Verbindungsmöglichkeiten sind u. a. Raw-IP, Ethernet über ISDN und Cisco-HDLC. Damit ist Linux für fast alle Anforderungen gerüstet.

Die Authentifizierung erfolgt bei syncPPP stets mit PAP oder CHAP, dabei gilt das bei Modemverbindungen Gesagte auch hier. Bei Raw-IP erfolgt die Authentifizierung nur durch die übertragene Rufnummer.

26.3.2 Linux als ISDN-Einwahlserver

Um eine ISDN-Verbindung anzunehmen, müssen Sie nur ein Interface mit `isdnctrl` konfigurieren (Listing 26.10), aber Anrufe zulassen (Listing 26.12). Die Authentifizierung erfolgt wie bei PPP beschrieben, so dass Sie wiederum entweder eigene CHAP- oder PAP-Passwörter vergeben oder die Unix-Passwörter verwenden können (und mittels PAM wieder RADIUS oder LDAP). Wenn Sie nur Anrufe von bestimmten Partnern annehmen wollen, geben Sie die gewünschten Rufnummern mit `isdnctrl addphone` an.

```
# Alle Anrufer annehmen:
# isdnctrl secure interface off
# Nur bestimmte Rufnummern zulassen:
isdnctrl secure interface on
# Anrufe folgender Rufnummer(n) annehmen
isdnctrl addphone interface in Rufnummer
```

Listing 26.12 Vorbereiten eines Interface für ISDN-Einwahl

Auch hier kann wieder mittels Callback (`isdnctrl callback`) gearbeitet werden (Listing 26.13). Dabei ist es aufgrund der Rufnummernübertragung möglich, es nur »einmal klingeln« zu lassen und dann diese Nummer zurückzurufen. Damit hat der Anrufer keinerlei Kosten.

```
# Callback für einen "Client" (der Angerufene ruft zurück)
isdnctrl callback interface out
# Callback für einen Einwahl-Server (Anrufer werden
zurückgerufen)
isdnctrl callback interface out
# Anrufer ablehnen vor dem Callback?
isdnctrl cbhup interface on
```

Listing 26.13 ISDN Callback

26.3.3 Weitere ISDN-Tools

Zu ISDN-4-Linux gehören außerdem eine Reihe von zusätzlichen Programmen, die hier nur sehr kurz vorgestellt werden sollen. Das Programm `iprofd` dient zur Speicherung der Einstellungen, die bei der Modememulation verwendet werden. Nützlich ist der `imon`, der den Status der ISDN-Leitungen darstellt.

Mit `isdnlog` können Sie viele Informationen zum Verbindungsauf- und -abbau anzeigen und abspeichern. Damit ist auch eine Gebührenabrechnung möglich. Mit `isdnlog` ist auch eine einfache Callback-Funktion möglich: Sie können für Anrufe von bestimmten Nummern auf einer MSN beliebige Befehle starten, also auch den Aufbau einer Internetverbindung. Die meist dynamisch zugewiesene

IP-Adresse können Sie zum Beispiel per Mail versenden oder mittels DynDNS oder ähnlichen Diensten zugreifbar machen. Anschließend können Sie von überall auf der Welt auf Ihren Rechner zugreifen.

26.4 Internetzugang mittels DSL

Für den Internetzugang via DSL (Digital Subscriber Line) wird PPP-over-Ethernet (PPPoE, [rfc2516]) verwendet. Hierfür wird ein spezieller Dämon, der `pppoe`, benötigt.

Sie brauchen eine von Linux unterstützte Netzwerkkarte, dieser muss allerdings keine IP-Adresse zugewiesen werden. Wenn dies die einzige Netzwerkkarte in Ihrem Rechner ist, dann wird das Interface mit dem Namen `eth0` angesprochen.

Bei der Debian-Distribution tragen Sie Ihren Benutzernamen in der Datei `/etc/ppp/peers/dsl-provider` ein und den Namen und das Passwort in der Datei `/etc/ppp/pap-secrets`. Anschließend können Sie mit dem Befehl `pon dsl-provider` die Verbindung herstellen und mit `poff dsl-provider` wieder beenden.

Wenn Sie eine Flatrate haben, dann können Sie die Verbindung bereits beim Systemstart herstellen. Dazu führen Sie die in Listing 26.14 gezeigten Befehle aus. Diese Datei wird von den Startskripten der Debian-Distribution ausgewertet.

```
(linux):~# cd /etc/ppp
(linux):~# ln -s ppp_on_boot.dsl ppp_on_boot
```

Listing 26.14 Starten der DSL-Verbindung beim Systemstart

Mit dem Eintrag `persist` in der PPP-Konfiguration versucht der `pppd` die Verbindung wieder aufzubauen, wenn die Gegenstelle aufgelegt hat. Auf diese Art und Weise ist Ihr Rechner praktisch immer »online«. Möchten Sie dies nicht, so können Sie mit `demand` auf Dial-On-Demand umstellen, so dass die Verbindung immer dann aufgebaut wird, wenn ein Paket zu übertragen ist.

Bei vielen DSL-Zugängen wird als Protokoll PPP-over-Ethernet (PPPoE) verwendet. Dabei wird zu einem PPP-Paket noch der Ethernet-Header ergänzt, so dass innerhalb eines Ethernet-Frame von 1500 Byte nur PPP-Pakete mit 1492 Byte transportiert werden können. Ist die MTU zu groß, so müssen Sie mit verschiedensten Problemen rechnen. Einige Webseiten lassen sich aufrufen, andere nicht. `ssh`-Verbindungen funktionieren, Dateien lassen sich aber mit `scp` nicht kopieren. Dieser Fehler ist als »Path MTU Discovery blackhole« bekannt.

Bei Verbindungen, bei denen NAT im Spiel ist, sollten Sie die MTU sogar noch kleiner, auf den Wert 1452, setzen. Die technischen Hintergründe sind in [rfc2923] erklärt. Sie können die MTU für eine bestehende Verbindung dyna-

misch ändern (zum Beispiel mit `ifconfig ppp0 mtu 1452`). Auf die Dauer werden Sie die Einstellung aber in der PPP-Konfiguration speichern.

Weitere Informationen zu DSL und PPPoE finden Sie in der DSL-HowTo und [rfc2516].

26.5 Linux als Router verwenden

Vielfach wird Linux auch als Router für ein ganzes Netz eingesetzt. Dabei kann intern Ethernet oder Token Ring zum Einsatz kommen, nach außen findet man häufig die hier besprochenen Verbindungsarten. Dabei kann Dial-On-Demand eingesetzt werden, manche Anwender verwenden Skripte auf dem Router, um die Verbindung manuell aufzubauen.

Für eine ISDN- oder DSL-Verbindung kann ein älterer Rechner verwendet werden. Und mit einem Floppy-Linux kann bei genügend großem Hauptspeicher auf eine Festplatte verzichtet werden. Damit lässt sich mit einfachen und preisgünstigen Mitteln ein Router bauen, der auch besondere Anforderungen erfüllen kann.

Auf dem Router muss IP-Forwarding eingeschaltet werden. Da die meisten Provider heute bei Wählverbindungen nur noch eine IP-Adresse (und diese zumeist dynamisch) zuteilen, muss der Router für das Netzwerk die Adressen umsetzen. Diese Umsetzung kennt man unter dem Namen Masquerading oder Network Address Translation (NAT).

In der IP-Masquerading-HowTo und der Masquerading-Simple-HowTo finden Sie einfache Beispiele für Skripte. In der einfachsten Version sind keine Firewall-Regeln zum Schutz der Rechner oder andere Policies implementiert. In vielen Fällen werden Sie mit den in den HowTos erläuterten Beispielen oder Variationen davon auskommen.

Die Konzeption einer Firewall und die Implementation einer Policy mit Hilfe von `iptables`-Regeln ist ein größeres Thema, das Kenntnisse von TCP/IP und ein Gespür für Sicherheitsfragen erfordert. Einen Einstieg in das Thema finden Sie in der Firewall-FAQ unter *http://www.iks-jena.de/mitarb/lutz/usenet/Firewall.html*. Ganz am Ende werden Sie ohne die Manpages und weiterführende Literatur wie [Zwicky2000] nicht auskommen.

26.6 Zusammenfassung

Linux unterstützt praktisch alle heute geläufigen Netzwerkanbindungen, entsprechend natürlich auch die Netzwerkanbindung über Wählverbindungen. Wenn der Rechner als Router konfiguriert ist, dann können Sie auch komplette

Netzwerke miteinander verbinden. Haben die Netzwerke keine offiziell reservierten IP-Adressen, so können (und müssen) Sie Masquerading einsetzen.

Über eine IP-Anbindung mittels Wählverbindung können Sie praktisch jedes Anwendungsprotokoll benutzen, nur der Durchsatz und die Round-Trip-Zeit werden die Geschwindigkeit begrenzen. Für einen Arbeitsplatz zu Hause oder die Anbindung weniger Arbeitsplätze in einer Außenstelle ist diese Art der Verbindung gut geeignet.

Über eine IP-Verbindung können Sie zum Beispiel mittels `ssh` einzelne Ports tunneln und damit den Datentransfer verschlüsseln. Alternativ können Sie auch ein »Virtual Private Network« (VPN) zum Beispiel mit Hilfe von IPsec aufbauen.

27 Virtuelle Private Netze (VPN)

27.1 Sparen mit Virtuellen Privaten Netzen

Für die Vernetzung von Filialen, Außenstellen, Außendienstmitarbeitern (»Road Warriors«) oder auch Heimarbeitsplätzen hat man früher je nach Datenvolumen und Entfernung eine Stand- oder Wählleitung verwendet. Heute nutzt man für diese Zwecke gerne das Internet, auch zum Transport von internen Daten. In diesem Fall sind nur die normalen Internetgebühren zu bezahlen, oder beispielsweise eine Flatrate.

Verwendet man hier jedoch nur »normale« Anwendungsprotokolle, so könnte ein Angreifer die meisten Daten auf einem Router im Internet abfangen. Um dies zu verhindern, ist also die Verbindung zu verschlüsseln. Die meisten Anwendungen könnten zwar über einen `ssh`-Tunnel verschlüsselt werden, das erfordert jedoch eine Menge Konfigurationsaufwand.

Die Verschlüsselung der Anwendungsdaten kann auf verschiedenen Ebenen erfolgen. Heute verschlüsseln verschiedene Anwendungen ihre Daten, zum Beispiel S/MIME oder GnuPG für Mail. Das hat den Vorteil, dass für den Zweck geeignete Verfahren verwendet werden können, aber den Nachteil, dass es für Anwender und Anwendungen nicht transparent ist. Dazu kommt, dass die Schlüsselverwaltung für viele Kommunikationspartner und Anwendungen ein Alptraum werden kann.

Ein weiterer Ansatzpunkt ist die Verschlüsselung auf Socket-Ebene. Dabei wird eine Zwischenschicht verwendet, die die Anwendung von der Arbeit der Verschlüsselung entlastet. Ein Beispiel dafür ist die OpenSSL-Bibliothek, die zum Beispiel für https-Verbindungen verwendet wird. Die Programmierung ist relativ einfach, die Verfahren zur Schlüsselverwaltung sind bekannt. In letzter Zeit wird diese Bibliothek immer häufiger verwendet – allerdings werden nur Daten verschlüsselt, wenn die Anwendung entsprechend programmiert ist.

In diesem Kapitel beschäftigen wir uns mit der Verschlüsselung auf dem Transport – die Daten aller Anwendungen werden verschlüsselt. Dabei ist Arbeit bei der Systemkonfiguration zu leisten, dafür entfällt die Verschlüsselung auf Applikationsebene. Gerade für die Verbindung von Mitarbeitern oder Filialen über unsichere Netze wie das Internet verwendet man gerne diese Art der Verschlüsselung.

Bei einem *Virtual Private Network* (VPN) wird der gesamte Datenverkehr zwischen zwei Rechnern oder Routern verschlüsselt und die Daten werden über ein öffentliches Netz wie das Internet übertragen. Für Anwendungen und Anwen-

der ist das Verfahren transparent und Angreifer können die Daten beim Transport nicht mehr lesen.

Es gibt eine Reihe von Möglichkeiten, VPNs aufzubauen. Ein sehr einfacher Fall ist das Tunneln von PPP über `ssh`. Dabei werden alle PPP-Pakete mit Hilfe von `ssh` verschlüsselt. Problematisch ist dabei, dass hier TCP über TCP getunnelt wird. Bei einem Retransmit eines Pakets durch PPP, zum Beispiel weil das TCP-Paket der `ssh` im Internet verloren gegangen ist, wird auch schon das reguläre TCP einen Retransmit durchführen. Es kann eine Weile dauern, bis sich die Konfusion wieder gelegt hat. Derartige Tunnel sind sehr einfach einzurichten, für Dauerbetrieb mit größeren Datenmengen aber eher nicht geeignet.

Ein weiteres Paket ist CIPE (Crypto IP Encapsulation). Diese Paket ist einfach zu konfigurieren und für Linux-Verbindungen gut zu gebrauchen.

Das wohl am häufigsten verwendete Programm ist FreeS/WAN, das den IPsec-Standard ([rfc2401]) implementiert. Dieses erfordert einen Kernel-Patch und ein Kernel-Modul, kann allerdings auch mit einigen kommerziellen Produkten zusammenarbeiten. In den folgenden Abschnitten werden wir uns FreeS/WAN genauer ansehen.

27.2 Das Design eines Virtuellen Privaten Netzes

Bei dem Design eines VPN müssen Sie zwei Netzwerke einrichten, die miteinander verwoben sind. Auf der unteren Ebene benötigen Sie ein IP-Netzwerk, das die betreffenden Geräte miteinander verbindet. Hier haben Sie es mit den üblichen Geräten (Routern) und Verbindungen (Wähl- oder Standleitungen) zu tun. Darauf setzen Sie ein logisches Netzwerk, das die VPN-Endgeräte oder Router direkt miteinander verbindet. Jedes VPN-Gerät kennt seine Gegenstelle, aber nicht die Router dazwischen.

Wenn Sie das Netzwerk aufmalen, dann verwenden Sie für das VPN eine andere Farbe oder gar eine transparente Folie, das macht später die Konfiguration anschaulicher. Die Tunnel sind Punkt-Zu-Punkt-Verbindungen, also direkte Verbindungen zwischen jeweils zwei VPN-Teilnehmern.

Neben dem komplexeren Routing muss das Problem der Schlüsselverwaltung gelöst werden. Auch hierfür gibt es verschiedene Ansätze: Die Schlüssel können als Shared Secret auf beiden Rechnern gespeichert werden, es kann ein Public-Key-Verfahren verwendet werden oder es wird eine (bestehende) Public-Key-Infrastruktur verwendet. Für IPsec kann ein »opportunistisches« Verfahren zum Einsatz kommen, hier werden die Public Keys mit Hilfe des DNS verwaltet.

Viele Provider vergeben an Dialup-Rechner nur noch dynamische IP-Adressen, diese Rechner können in der Regel einfach als Clients (»Road Warrior«) verwendet werden. Sollen diese Systeme auch als Ziel einer VPN-Verbindung verwendet werden, so muss die IP-Adresse auf anderen Wegen ermittelt werden und kann nicht zur Authentifizierung eingesetzt werden.

Eine Warnung noch: VPNs sind nicht einfach zu verwalten. Außerdem verwendet man sie gerne, um Zugriffe auf interne Netze zu kontrollieren. Beides zusammen verlangt besondere Vorsicht. Lesen Sie die verfügbare Dokumentation und nehmen Sie nur geplante Änderungen vor. Einfach »ausprobieren« könnte am Ende den erhofften Schutz zerstören.

27.3 Crypto IP Encapsulation (CIPE)

CIPE (Crypto IP Encapsulation, *http://www.inka.de/~bigred/devel/cipe.html*) überträgt alle IP-Pakete zwischen den beteiligten Rechnern verschlüsselt mit Hilfe von UDP-Paketen. CIPE erfordert ein Kernel-Modul, aber keinen Kernel-Patch, so dass es einfacher als FreeS/WAN zu installieren ist.

Auch die Konfiguration ist einfacher, da CIPE nur zwischen zwei Rechnern tunnelt. Für eventuell hinter dem anderen Rechner liegende Systeme ist der Partner verantwortlich. Auch das Key-Management ist einfacher, Sie können aber auch eine eigene PKI verwenden (mit dem Programm `pkcipe`).

Für jeden Peer (Tunnel) legen Sie eine Konfigurationsdatei unter `/etc/cipe/peers` an, die den Tunnel beschreibt. In Listing 27.1 finden Sie ein Beispiel für die Konfiguration eines Clients mit dynamischer IP-Adresse.

```
# Die IP-Adresse des Peers (UDP-Port)
peer            jupiter.jochen.org:6543
# die eigene IP-Adresse
me              0.0.0.0:6789
# die IP-Adresse ist dynamisch
dynip

# Die VPN-Adresse des Peers
ptpaddr         192.168.222.202
# Die eigene VPN-Adresse
ipaddr          192.168.222.118

# Der statische Key, der muss geheim bleiben!
# Der Key ist 128 Bit lang und hexadezimal notiert.
key             3248fd20adf9c00dcf9ecc4393fbb3e4
```

Listing 27.1 Konfiguration eine CIPE-Clients

Die Konfiguration des Peers ist zu der in Listing 27.1 gezeigten symmetrisch, d. h., was auf der einen Seite `peer` ist, ist auf der anderen Seite `me` und umgekehrt. Genauso verhält es sich mit `ipaddr` und `ptpaddr`. Wenn beide Kommunikationspartner dynamische IP-Adressen verwenden, so muss zumindest eine der beteiligten IP-Adressen publiziert werden, zum Beispiel mit Hilfe eines dynamischen DNS-Eintrags.

Bei der Verwendung von statischen IP-Adressen kann der Link einfach aufgebaut werden, bei dynamischen IP-Adressen haben Sie mit einigen Einschränkungen zu leben. Im Wesentlichen müssen Sie auf der einen Seite des Links im `ip-up`-Skript einen `ping` auf den Peer einbauen, so dass die notwendigen Daten ausgetauscht werden können. In der Info-Dokumentation von CIPE finden Sie eine Reihe von Beispielen und ausführliche Erläuterungen zum Design.

Die Netzwerkgeräte heißen `cipcb0` und sind als Punkt-zu-Punkt eingetragen. Wenn das Gerät aktiviert wird, dann startet das Skript `/etc/cipe/ip-up`. Hier können Sie Routen setzen oder andere Einstellungen vornehmen. Das Skript `/etc/cipe/ip-down` wird aufgerufen, wenn die Verbindung beendet bzw. das Gerät gelöscht wird.

27.4 Konfiguration von FreeS/WAN

Das Paket FreeS/WAN (*http://www.freeswan.org/*) implementiert einen Großteil von IPsec ([rfc2401] und [rfc2411]). Insbesondere bei der Verbindung von Endgeräten verschiedener Hersteller ist FreeS/WAN die erste Wahl. Mit IPsec bzw. FreeS/WAN können Sie einzelne Rechner miteinander verbinden, einen Rechner mit einem Netzwerk oder zwei Netzwerke aneinander koppeln. Je nach Umfeld wird sich die Konfiguration unterscheiden.

Die Installation von FreeS/WAN erfordert einen Kernel-Patch und damit eine Neu-Übersetzung des Kernels. Nach dem Einfügen des Patchs finden Sie in der Kernel-Konfiguration unter »Network-Options« die Einstellungen für IPsec. Sie können verschiedene Authentifizierungsalgorithmen verwenden. Welche Verfahren Sie einsetzen können oder wollen, sollten Sie mit Ihrer Gegenstelle absprechen.

Die Konfiguration der Tunnel erfolgt in der Datei `/etc/ipsec.conf`. In Listing 27.2 finden Sie ein Beispiel für einen einfachen Tunnel zwischen zwei Hosts, also kein VPN bzw. Zugang für Road-Warrior. Ausführliche Informationen erhalten Sie in den mitgelieferten Manpages und Dokumenten. In der FAQ sind fast alle Probleme erläutert, außerdem finden Sie eine Reihe von Konfigurationsbeispielen.

```
# basic configuration
config setup
# THIS SETTING MUST BE CORRECT or almost nothing will work;
# %defaultroute is okay for most simple cases.
interfaces=%defaultroute
# Debug-logging controls:  "none" for (almost) none, "all" for
lots.
klipsdebug=none
plutodebug=none
# Use auto= parameters in conn descriptions to control startup
actions.
plutoload=%search
plutostart=%search
# Close down old connection when new one using same ID shows up.
uniqueids=yes

# defaults for subsequent connection descriptions
# (mostly to fix internal defaults which, in retrospect, were
badly chosen)
conn %default
keyingtries=0
disablearrivalcheck=no
authby=rsasig
leftrsasigkey=%dns
rightrsasigkey=%dns

conn jupiter-hermes
type=transport
left=192.168.30.118
        leftrsasigkey=0sAQNY9u...
right=192.168.30.202
rightrsasigkey=0sAQNui...
auto=start
```

Listing 27.2 Eine einfache FreeS/WAN-Konfiguration

Vor dem Start müssen auf beiden Rechnern die Hostkeys generiert werden. Dies erfolgt, sofern die Distribution das nicht bei der Installation schon erledigt hat, mit dem Befehl `ipsec newhostkey --output /etc/ipsec.secrets` auf allen beteiligten Rechnern. Anschließend können Sie sich die Public Keys mit Hilfe von `ipsec showhostkey` anzeigen lassen oder aus der Datei `/etc/ipsec.secrets` in die Datei `/etc/ipsec.conf` in die betreffende `conn`-Konfiguration kopieren.

Die einzige Abweichung zur bereits installierten Standardkonfiguration in Listing 27.2 ist die Verbindung `conn jupiter-hermes`. Der Typ `type=transport` beschreibt einen Tunnel zwischen zwei Rechnern. Für einen Tunnel zwischen Netzwerken oder zu einem Netzwerk müssen Sie `type=tunnel` verwenden. Die Einstellungen für `left` und `right` sind austauschbar, im aktuellen Beispiel ist die Konfigurationsdatei auf beiden Rechnern identisch.

Die Schlüsselwörter `leftrsasigkey` bzw. `rightrsasigkey` enthalten die Host-Keys, wie man sie sich mit `ipsec showhostkey` anzeigen lassen kann (am Anfang muss ein 0s ergänzt werden, um die Kodierung anzugeben) oder sie in der Datei `/etc/ipsec.secrets` findet. Anschließend kann die Verbindung aufgebaut werden. Bei der Verwendung von `auto=start` erfolgt das automatisch beim Booten, andernfalls kann `ipsec auto --up jupiter-hermes` verwendet werden.

Wenn Sie viele Tunnel konfigurieren müssen, dann wird die Datei `/etc/ipsec.conf` recht schnell unübersichtlich. Sie können mit dem Schlüsselwort `include ipsec.d/*.conf` weitere Dateien einlesen. Jede dieser Dateien könnte beispielsweise einen Tunnel beschreiben.

Die VPN-Tunnel werden mit dem Befehl `ipsec` verwaltet, alle Optionen sind in der Manpage erläutert. In der Regel hat Ihre Distribution die passenden Start-Skripts installiert, die Dokumentation finden Sie in der Manpage `ipsec(8)` und `ipsec_Befehl`.

Für die Anbindung von Road-Warriors und Netzwerken finden Sie ausführliche Beispiele in der FreeS/WAN-Dokumentation. Wenn Sie Probleme mit der Verbindung haben, so können Sie Debugging einschalten (`plutodebug=all` und/oder `klipsdebug=all`) und die Logs analysieren – die Meldungen sind recht aussagekräftig.

27.5 Key-Management

Wieder ist das Key-Management auch für IPsec das zentrale Problem. Wie bekommt man den Key sicher zum Kommunikationspartner? Wenn Sie die Keys in die Konfigurationsdateien eintragen, dann sind Sie selbst für den Transport zuständig. Alternativ können Sie die Keys im DNS speichern und von dort aus verwenden. Statt des Key tragen Sie in der Konfiguration dann `%dns` ein.

Dieses Verfahren kann auch dazu eingesetzt werden, ein »opportunistisches« IPsec zu verwenden. Dabei werden Sessions nicht im Vorfeld vereinbart, sondern dynamisch anhand der im DNS enthaltenen Daten aufgebaut. Damit ist es möglich, auch zwischen sich sonst unbekannten Systemen eine verschlüsselte Verbindung aufzubauen. Um die Verbindung abzuhören, müsste der Key kompromittiert oder ausgetauscht werden. Das kann durch einen kompromittierten DNS-Server oder DNS-Spoofing der Fall sein. Solange kein Secure DNS verwendet wird, ist opportunistisches IPsec nicht wirklich sicher. Das FreeS/WAN-Paket enthält eine gute HowTo zu diesem Thema.

27.6 Fazit

Neben den hier vorgestellten VPN-Lösungen für Linux gibt es eine ganze Reihe von weiteren Programmen. Je nach Bedarf könnten auch diese interessant für Sie sein. Wenn Sie jedoch an der Interoperabilität mit kommerziellen Produkten interessiert sind, dann führt kaum ein Weg an FreeS/WAN vorbei.

Wenn das VPN erst einmal entworfen und konfiguriert ist, dann besteht eine einfach zu handhabende, sichere und in der Regel genügend schnelle Verbindung zwischen den Systemen. Die größten Probleme werden in der Praxis das Routing und die Schlüsselverwaltung aufwerfen. Hier sollten Sie einen sauberen Entwurf machen und diesen und die Implementation gut dokumentieren.

28 Netzwerkadministration

SNAFU – Situation Normal, All Fucked Up
Eine Situationsbeschreibung

28.1 Aufgaben eines Netzwerkadministrators

Ein lokales Netz entwickelt sich ständig weiter. Neue Rechner werden angeschlossen, alte Rechner entfernt oder mit einem neuen Betriebssystem versehen. Diese Veränderungen und viele andere mehr haben auch Auswirkungen auf das Netz. Für eine geordnete und zielgerichtete Entwicklung ist in der Regel ein Netzwerkadministrator zuständig. Neben diesen »planbaren« Tätigkeiten nimmt auch die Fehlerbehebung (Troubleshooting) großen Raum ein. Für beide Anforderungen existieren eine Reihe von Programmen, die den Administrator unterstützen können. Einige Programme werden hier vorgestellt, viele weitere Programme werden in [rfc1470] erwähnt.

28.2 Troubleshooting im Netz

The only problem with troubleshooting is that the trouble shoots back.
Motto für die Fehlersuche

Ein lokales Netz ist ein komplexes Gebilde, das für Fehler relativ anfällig ist. Es besteht aus unterschiedlichen Hardware-Komponenten verschiedener Hersteller, die dennoch miteinander umgehen können müssen. Oft ist bereits diese Struktur kaum zu überblicken. Hinzu kommen verschiedene Netzprotokolle unter mehreren Betriebssystemen.

Obwohl diese Protokolle standardisiert sind, gibt es dennoch Implementationen, die nicht mit allen anderen Versionen zusammenarbeiten können. Im Internet werden regelmäßig Wettkämpfe veranstaltet, um die Interoperabilität und Stabilität der verschiedenen Programme zu testen. Dabei wird versucht, gültige und ungültige Pakete zu versenden und die Reaktion der Systeme zu prüfen.

Daneben gibt es eine Reihe von Fehlersituationen, in denen die Fehlerursache nicht offensichtlich ist. In solchen Fällen hilft nur eine zielorientierte Suche nach der Ursache, da in vielen Firmen dem lokalen Netz eine strategische Bedeutung zukommt. Viele Unternehmen sind von ihrem Netz so abhängig, wie früher von einem Großrechner, aber die entsprechenden Vorsichtsmaßnahmen werden oft nicht getroffen.

Die Stabilität und Ausfallsicherheit eines Netzes lässt sich erhöhen, indem bei Ethernet Twisted-Pair- statt Coax-Kabel eingesetzt werden. Als Konzentrator sollte ein Switch anstelle eines einfachen Hubs eingesetzt werden, da dieser eine Lasttrennung zwischen den verschiedenen Segmenten durchführen kann. Die einzelnen Netzsegmente können durch Bridges oder Router voneinander getrennt werden. Außerdem sollte für jede zentrale Komponente mindestens ein Backup vorhanden sein. Der Netzwerkadministrator sollte sich mit Tools wie einem Analyzer vertraut machen und das Netz regelmäßig untersuchen, damit ihm Abweichungen vom normalen Verhalten auffallen.

Tipp

Nehmen Sie sich die Zeit dafür, einfach mal zu schauen, was im Netz passiert. Wenn Sie nicht wissen, was normal ist, dann können Sie nicht erkennen, was im Fehlerfall an der aktuellen Situation besonders ist. Die Zeit, die Sie im Vorfeld investieren, zahlt sich bei der Fehlersuche häufig aus.

Wenn Sie Probleme haben, grenzen Sie diese ein. Isolieren Sie einzelne Rechner oder Netzsegmente und testen Sie zunächst relativ lokal. Gerade Probleme wie vergessene oder abgefallene Terminatoren und defekte Kabel sind ohne Analyzer nicht einfach zu finden.

28.2.1 Konfigurationsfehler des lokalen Systems

Bereits bei der Konfiguration eines einzelnen Rechners kann man eine Reihe von Fehlern machen, die die Einbindung in das Netz verhindern. Bei Problemen sollte man also zunächst die Konfiguration der betroffenen Rechner überprüfen. Dabei sind folgende Fehler besonders einfach festzustellen:

- Wurde die Netzwerkkarte vom Kernel erkannt? Sehen Sie sich mit dem Programm `dmesg` die Kernel-Meldungen an. Möglicherweise sind die Meldungen beim Systemstart in eine Datei (oft `/var/log/dmesg`, `kernel` oder `messages`) gespeichert worden.
- Ist dem Netzwerk-Interface die richtige IP-Adresse zugewiesen? Hier hilft Ihnen die Ausgabe des Befehls `ifconfig` weiter. Dieses Programm zeigt auch die Netmask und Broadcast-Adresse an, die Sie ebenfalls überprüfen sollten.
- Sind die entsprechenden Routen gesetzt? Verwenden Sie den Befehl `route` bzw. `netstat -r`. Wenn Sie Probleme mit der Namensauflösung haben, verwenden Sie zusätzlich die Option `-n`.
- Ist ein `ping localhost` möglich? Funktioniert ein `ping` *`eigene IP-Adresse`* bzw. `ping` *`eigener Host-Name`*? Geht ein `ping` von einem anderen Rechner aus?

- Transferiert die Netzwerkkarte Daten? Viele Netzwerkkarten sind mit LEDs ausgestattet, die gesendete und empfangene Pakete anzeigen. Manchmal kann ein `tcpdump` auf einem anderen Rechner hierüber Aufschluss geben.
- Ist der Port am Repeater oder Hub auf »Rot«? Eventuell ist dann das Kabel beschädigt.
- Zeigt die Ausgabe von `ifconfig` viele Kollisionen an? Dann könnte es zum Beispiel sein, dass der Switch und der Rechner verschiedene Auffassungen von der Geschwindigkeit oder Duplex-Einstellung haben.
- Ist ein eventueller Nameserver ansprechbar? Ist ein `ping` zu dessen IP-Adresse möglich? Können Sie Anfragen mittels `host` oder `nslookup` abgeben und erhalten Sie Antwort?
- Läuft der `inetd` und sind in der Datei `/etc/inetd.conf` die richtigen Dämonen eingetragen? Sind auffällige Einträge im Log-File des `inetd` zu finden? Liefert `netstat -a` das erwartete Ergebnis?
- Sind die RPC-Dämonen, soweit notwendig, gestartet und funktionsfähig? Das können Sie mit `rpcinfo -p` bzw. `rpcinfo -u` *`Host Dienst`* überprüfen.
- Gibt der Kernel ungewöhnliche Meldungen aus? Zurückliegende Meldungen können Sie sich mit `dmesg` erneut anzeigen lassen, allerdings ohne Zeitstempel. In diesem Fall sollten Sie in das entsprechende System-Log sehen.

Je nach Installation und Verwendung des Rechners können weitere, einfach zu überprüfende Funktionen hinzukommen oder einzelne Funktionen entfallen. Insbesondere bei Problemen mit der Verkabelung kommt man nur mit Software bei der Fehlersuche nicht weiter. Für größere Netze lohnt sich daher die Anschaffung eines Netzwerkanalysators. Damit lassen sich z. B. Kollisionen, Kabelunterbrechungen oder fehlende Terminierungen feststellen.

28.2.2 Staus im Internet

Oft sind einzelne Rechner im Internet nicht oder nur schwer zu erreichen. Mit dem Befehl `traceroute` können Sie sich eine Übersicht über alle Router bzw. Gateways verschaffen, über die der Datentransfer vermutlich abgewickelt wird. Eine wirkliche Sicherheit kann die Ausgabe nicht gewährleisten, da für jedes einzelne Paket dynamisch eine andere Route gewählt werden kann. `traceroute` verwendet dabei Funktionen des IP und ICMP, um diese Informationen zu erhalten. Das Programm existiert auch für Windows-Systeme, dort heißt es `tracert`.

Das Programm sendet ein IP-Paket, das nur eine begrenzte Lebensdauer hat. Diese Lebensdauer ist im Feld `ttl` (Time To Live, Max Hops in IPv6) im IP-Header abgelegt. Zunächst wird mit einer `ttl` von 1 gestartet. Das Paket enthält eine vermutlich nicht benutzte Port-Nummer als Ziel. Das Gateway, das das Paket wegen überschrittener Lebensdauer vom Netz entfernt, sendet eine `ICMP`-Nach-

richt vom Typ »time exceed«. Aufgrund dieser Nachricht kann dieses Gateway identifiziert werden. Danach wird die `ttl` um eins erhöht und das Paket erneut gesendet. Erhält das Programm die ICMP-Nachricht »Port unreachable«, dann ist der gesuchte Rechner erreicht.

Mit der Option `-n` verhindern Sie eine Reverse-DNS-Abfrage für jedes Gateway, so dass nur die IP-Adressen angezeigt werden. Wenn Sie ohnehin Probleme mit Ihrem Netz oder Name-Service haben, sollten Sie diese Option in jedem Fall aktivieren. Weitere Einstellungen können Sie mit verschiedenen Kommandozeilenparametern vornehmen, die in der Manpage erläutert werden. Ein Beispiel für die Ausgabe von `traceroute` finden Sie in Listing 28.1.

```
(linux):~$ traceroute -n ftp.gwdg.de
traceroute to gwdu32.gwdg.de (134.76.12.1) 30 hops max, 40 byte
packets
 1  139.174.2.254  4 ms  2 ms  3 ms
 2  139.174.254.2  3 ms  3 ms  3 ms
 3  * 188.1.132.13  3156 ms  1310 ms
 4  134.76.12.1  519 ms  1066 ms  1188 ms
```

Listing 28.1 Die Verwendung von `traceroute`

Kann ein Rechner nicht innerhalb der Wartezeit erreicht werden, so wird ein Stern (`*`) anstelle der Antwortzeit ausgegeben. Die Wartezeit kann mit der Kommandozeilenoption `-w` angegeben werden. Die vollständige Dokumentation aller Optionen finden Sie in der Manpage.

`traceroute` kann eine recht hohe Netzlast erzeugen, insbesondere, wenn das Netz ohnehin schon am Rande seiner Kapazität ist. In der Regel sollte man `traceroute` nur zur Fehlersuche einsetzen und nicht in (automatisch gestarteten) Skripten. Außerdem leiten einige Firewall-Systeme nur wenige oder keine ICMP-Pakete weiter, so dass der Einsatz von `traceroute` nicht möglich ist.

`traceroute` gibt nur Aufschluss über die mittels UDP und ICMP erreichbaren Rechner. Oft kann man Systeme, die hinter einer Firewall liegen, mit diesem Protokoll erreichen, aber andere (Anwendungsprotokolle) werden von der Firewall gefiltert (oder umgekehrt). Eventuell können Sie die Option `-I` verwenden, die anstelle der UDP-Pakete nur ICMP versendet, bei manchen Firewall-Konfigurationen hilft das. Auch aus diesem Grunde sind die Ausgaben des Programms mit Vorsicht zu genießen. Andere unsinnig erscheinende Ausgaben können möglicherweise durch eine fehlerhafte Implementation des TCP/IP-Stack erklärt werden.

Alternativ zu `traceroute` können Sie z. B. das Programm `mtr` (*http://www.bitwizard.nl/mtr/*) verwenden. Es erzeugt eine ähnliche Ausgabe, hat aber auch eine GTK- und eine Curses-Oberfläche. Beim Aufruf ohne Host-Name als Parameter wird der `localhost` verwendet, unter X können Sie in das passende Feld

einen anderen Namen eingeben. Andere nützliche Optionen sind `-i` (`--interval`) für die Pause zwischen ICMP-Paketen, `-n` (`--no-dns`) für die Ausgabe von IP-Adressen statt Host-Namen, `-c` (`--repeat-cycles`) für die Anzahl der Wiederholungen und `-r` (`--report`) erstellt zusammen mit `-c` einen Report und verwendet keine Benutzeroberfläche.

Ein Beispiel für die Anzeige von `mtr` finden Sie in Listing 28.2. Beachten Sie auch hier, dass Sie möglicherweise eine hohe Netzlast erzeugen und andere Anwender behindern können. Dennoch sind `traceroute` und `mtr` nützliche Programme.

```
                          Matt's traceroute  [v0.45]
jupiter                                                Fri Feb  8
09:28:08 2002
Keys:  D - Display mode    R - Restart statistics    Q - Quit
                                      Packets
Pings
Hostname                           %Loss  Rcv  Snt  Last Best
Avg  Worst
 1. janus.jochen.org                  0%    8    8     1    1
1      1
 2. 217.5.98.15                       0%    7    7    33   31
35     58
 3. 217.237.152.114                   0%    7    7    31   30
31     32
 4. F-gw13.F.NET.DTAG.DE              0%    7    7    32   31
32     33
 5. 213.83.45.1                       0%    7    7    31   31
31     32
 6. 213.83.57.20                      0%    7    7    47   32
37     48
 7. www.heise.de                      0%    7    7    35   33
34     36
```

Listing 28.2 Das Programm `mtr` im Einsatz

Viele Firewalls filtern ICMP- und UDP-Pakete, so dass vielfach `traceroute` nicht wie gewünscht funktioniert. Abhilfe schafft `tcptraceroute` (*http://michael.toren.net/code/tcptraceroute/*). Das Programm versucht, eine TCP-Verbindung auf dem Port 80 aufzubauen, was vielfach gelingt. Sollte der Zielrechner diesen Port nicht verwenden, so kann nach dem Host-Namen zusätzlich ein Zielport angegeben werden. Das Verfahren funktioniert häufiger als ein Standard-`traceroute`, es ist allerdings notwendig, dass die betreffenden Firewalls die ICMP-Nachricht »Time exceeded« nicht filtern.

28.2.3 Protokollieren von ICMP-Paketen

Neben der Verwendung als Diagnose-Tool (z. B. in `ping` und `traceroute`) erfüllt ICMP einige wichtige Aufgaben für die Verlässlichkeit der Protokolle. Die meisten ICMP-Nachrichten werden vom Kernel in aller Stille ausgewertet und bearbeitet. Manchmal kann es sinnvoll sein, diese Pakete zu analysieren, da sie wichtige Informationen zum Routing, zur Stabilität der Verbindungen oder zur Sicherheit des Netzes enthalten können.

Die Analyse kann z. B. mit `tcpdump` erfolgen, das im nächsten Abschnitt genauer vorgestellt wird. Wenn Sie jedoch nur an ICMP-Paketen interessiert sind, ist `icmpinfo` (*ftp://hplyot.obspm.fr/net*) einfacher zu handhaben.

Sie müssen `icmpinfo` als `root` starten (oder `setuid-root` installieren). Die Ausgabe kann entweder auf die Standardausgabe erfolgen oder mittels `syslog` (Option `-l`, Facility `daemon`, Priorität `notice` oder `warning`) weiterverarbeitet werden. Standardmäßig werden nur ungewöhnliche ICMP-Pakete protokolliert. Mit der Option `-v` werden alle Pakete außer `ICMP_ECHO_REPLY` analysiert, die Option `-vv` zeigt wirklich alle Pakete an. Vermutlich werden Sie erstaunt sein, wie viele ICMP-Pakete Ihr Rechner zu bearbeiten hat.

In Einzelfällen interessant ist die Option `-vvv`, die die Pakete komplett anzeigt. Wenn Sie Nameserver-Anfragen vermeiden wollen, dann sollten Sie die Option `-n` angeben. Da das Programm nur sehr wenig Speicher benötigt und praktisch keine CPU-Leistung verbraucht, können Sie es auch für längere Zeit im Hintergrund laufen lassen.

28.3 Programme zur Netzverwaltung

Viele der oben oder in den anderen Kapiteln angesprochenen Probleme lassen sich mit den entsprechenden Tools vorausahnen oder relativ leicht erkennen. Im Folgenden werden einige Programme vorgestellt, mit denen Sie das Netz auf verschiedenen Ebenen untersuchen können. Oft wird dazu die Netzwerkkarte in den Promiscous Mode versetzt, in dem alle Ethernet-Pakete von der Karte angenommen werden und nicht nur die für diesen Rechner bestimmten Pakete. Damit kann der gesamte Verkehr auf dem Netz beobachtet und analysiert werden.

Für Einbrecher in das lokale Netz ist diese Funktion auch von Interesse, da z. B. bei `telnet` das Passwort unverschlüsselt übertragen wird und gelesen werden kann. Abhilfe können Sie mit Programmen und Protokollen wie `ssh` schaffen, die keine Passwörter im Klartext übertragen. Daher sollten Sie regelmäßig Ihre Systeme auf Netzwerkkarten überprüfen, die im Promiscous Mode betrieben werden. Das Programm `ifconfig` zeigt diese Informationen an.

Es gibt allerdings so genannte Root-Kits, die eine Reihe von Programmen enthalten, die einen erfolgreichen Angriff verschleiern sollen, dazu gehören u. a. Programme wie `ifconfig`, `netstat` oder `ps`. Mit `chkrootkit` (*http://www.chkrootkit.org/*) können Sie Ihr System überprüfen – wenn es allerdings kompromittiert ist, dann können Sie sich auf *kein* installiertes Programm mehr verlassen. Eine Neuinstallation ist dann unvermeidbar.

28.3.1 IP-Pakete untersuchen

Viele Probleme entstehen dadurch, dass fehlerhafte IP-Pakete im Netz unterwegs sind. Derartige Pakete können mit dem Programm `tcpdump` analysiert werden. Einzelne Protokolle oder Pakete von oder für bestimmte Rechner können angezeigt werden. Die genauer zu untersuchenden Pakete lassen sich flexibel eingrenzen, so dass auch bei einem stark belasteten Netz die Ausgabe des Programms nicht zu groß wird.

Das Programm `tcpdump` versetzt die Ethernet-Karte in den Promiscous Mode, in dem alle Pakete vom Netz gelesen werden. Damit können alle Verbindungen[1], die von Rechnern im lokalen Netz aufgebaut oder angenommen werden, durch den Systemadministrator mitverfolgt werden. In der Standardversion zeigt `tcpdump` nur den Header der Pakete an. Sie können auch alle Daten mitschneiden und so z. B. den Verbindungsaufbau einer `ftp`- oder `telnet`-Sitzung mitverfolgen. In diesem Fall können Sie dann auch das Passwort des entsprechenden Benutzers aus den Daten ermitteln.

Das Programm `tcpdump` kann nur von `root` verwendet werden, da es die Netzwerkkarte in den Promiscous Mode versetzen muss. Ähnliche Pakete gibt es auch für DOS-Systeme, wo dies keine Einschränkung darstellt. Nach dieser Einführung folgen einige Beispiele für die Verwendung von `tcpdump`. Die hier abgedruckten Daten wurden zunächst vollständig in einer Datei abgespeichert. Diese Datei wurde dann als Eingabe für die `tcpdump`-Befehle verwendet, auch wenn die dafür notwendigen Optionen aus Platzgründen nicht angegeben sind. Außerdem wurden die IP-Adressen durch für den internen Gebrauch reservierte ersetzt.

Der sinnvolle Einsatz von `tcpdump` setzt ein intensives Verständnis der zu beobachtenden Protokolle voraus, besonders wenn Sie anhand der in den Paketen transportierten Daten selektieren wollen. Wenn Sie intensiver mit diesem Programm arbeiten wollen, sollten Sie die entsprechenden RFCs greifbar haben. Für alle, die sich wirklich für die Interna von TCP/IP interessieren, ist die Serie »TCP/IP illustrated« von Richard W. Stevens [Stevens1994] eine wahre Fund-

1. Auch ein Switch hilft nicht weiter, die meisten kann man so überlasten, dass sie wie ein Hub arbeiten. Im Normalfall sieht man allerdings nur die Verbindungen aus derselben »Collision-Domain«, also demselben Switch-Port.

grube. Dort werden, insbesondere im ersten Teil, viele Protokolle untersucht und mittels `tcpdump` beobachtet.

Später werden Sie sicher für die verschiedenen Anwendungsfälle ein entsprechendes Skript erstellen. In Listing 28.3 werden alle `tcp`-Verbindungen angezeigt.

```
(linux):~# tcpdump -n -t -q
192.168.31.150.23 > 192.168.31.155.1808: tcp 1
192.168.31.155.1808 > 192.168.31.150.23: tcp 0
192.168.31.150.23 > 192.168.31.155.1808: tcp 47
192.168.31.155.1808 > 192.168.31.150.23: tcp 0
192.168.31.155.513 > 192.168.31.151.1023: tcp 47 (DF) [tos 0x10]
192.168.31.151.1023 > 192.168.31.155.513: tcp 0 [tos 0x10]
192.168.31.155.1808 > 192.168.31.150.23: tcp 0
192.168.31.151.1023 > 192.168.31.155.513: tcp 0 [tos 0x10]
```

Listing 28.3 Anzeige aller `tcp`-Verbindungen im Netz

Die Option `-n` verhindert den Reverse-Lookup beim Nameserver und die Anzeige der symbolischen Port-Namen aus der Datei `/etc/services`. Die Option `-t` unterdrückt die Ausgabe der Uhrzeit und `-q` sorgt für eine kürzere Ausgabe. In der ersten Spalte sind der Source-Rechner und der dort verwendete Port angegeben. In der nächsten Spalte (nach dem Größer-Zeichen) steht der Zielrechner mit dem Ziel-Port. Nach dem Doppelpunkt werden weitere Informationen aus dem Header des Pakets angezeigt.

Im nächsten Beispiel werden nur die Verbindungen vom bzw. zum Port `login` angezeigt. Wenn Sie die Ausgabe in Listing 28.4 mit der in Listing 28.3 vergleichen, finden Sie hier weniger Pakete. Neben der Einschränkung auf Ports können Sie auch nach Rechnern (`host`), Netzen (`net`) oder Gateways (`gateway`) selektieren. Darüber hinaus lässt sich die Anzeige auf ein- oder ausgehende Pakete (`dst` oder `src`) und einzelne Protokolle (`udp`, `tcp` oder `icmp`) beschränken.

```
(linux):~# tcpdump -n -q -t port login
192.168.31.155.login > 192.168.31.151.1023: tcp 47 (DF) [tos
0x10]
192.168.31.151.1023 > 192.168.31.155.23: tcp 0 [tos 0x10]
192.168.31.151.1023 > 192.168.31.155.23: tcp 0 [tos 0x10]
192.168.31.155.23 > 192.168.31.151.1023: tcp 40 (DF) [tos 0x10]
192.168.31.151.1023 > 192.168.31.155.23: tcp 0 [tos 0x10]
```

Listing 28.4 Nur `rlogin`-Verbindungen

Im Listing 28.5 finden Sie ein Beispiel für die logische Verknüpfung von mehreren Merkmalen. Hier werden nur die `rlogin`-Verbindungen zu einem bestimmten Rechner angezeigt. Bedingungen können anstelle von `and` auch mit `or` verknüpft oder mit einem Ausrufezeichen (`!`) negiert werden. Wenn das Ausrufezeichen ein

Shell-Sonderzeichen ist, dann müssen Sie dieses, z. B. mit einem Backslash, entwerten. Diese Ausdrücke können Sie auch in runde Klammern einschließen, um eine bestimmte Reihenfolge der Auswertung zu erzwingen.

```
(linux):~# tcpdump -n -q -t port login and host 192.168.31.152
192.168.31.153.1023 > 192.168.31.152.login: tcp 0
192.168.31.152.23 > 192.168.31.153.1023: tcp 0 (DF)
192.168.31.153.1023 > 192.168.31.152.23: tcp 0
192.168.31.153.1023 > 192.168.31.152.23: tcp 1 (DF)
192.168.31.152.23 > 192.168.31.153.1023: tcp 0 (DF)
192.168.31.153.1023 > 192.168.31.152.23: tcp 25 (DF)
```

Listing 28.5 Nur `rlogin`-Verbindungen zu einem Host

Das letzte Beispiel (siehe Listing 28.6) zeigt den Zugriff auf in dem Paket gespeicherte Daten. Hier werden die Pakete angezeigt, die beim Aufbau oder Ende einer Verbindung gesendet werden. Wenn Sie genügend große Pakete (Option `-s`) mitschneiden, dann können Sie auf beliebige Datenfelder im Paket zugreifen. Diese Funktionen verlangen sehr gute Kenntnisse der Protokolldefinition, wie sie in den entsprechenden RFCs zu finden ist.

```
(linux)~# tcpdump 'tcp[13] & 3 != 0 and not src and dst net
192.168.31'
192.168.31.151.1094 > 192.168.3.151.23: tcp 0
192.168.3.151.23 > 192.168.31.151.1094: tcp 0 (DF)
192.168.31.151.1022 > 192.168.3.150.514: tcp 0
192.168.3.150.514 > 192.168.31.151.1022: tcp 0 (DF)
192.168.31.151.1022 > 192.168.3.150.514: tcp 0
192.168.3.150.514 > 192.168.31.151.1022: tcp 0 (DF)
```

Listing 28.6 Nur Beginn und Ende von Verbindungen

Das Programm `tcpdump` wird zusammen mit einigen einfachen `awk`-Skripten geliefert, die zur Auswertung der gesammelten Daten dienen können. In der Praxis werden Sie sich für häufig auftretende Aufgaben entsprechende Skripten erstellen, um z. B. bei Problemen schnell reagieren zu können und nicht erst in den Manpages nachschlagen zu müssen.

28.3.2 Weitere Tools

Neben `tcpdump` existieren noch eine ganze Reihe anderer Netzwerk-Sniffer. Für den Textmode gibt es `iptraf`, das Programm zeigt eine Übersicht der aktuellen Verbindungen an sowie welche Art von Paketen gerade im Netz unterwegs ist (Abbildung 28.1).

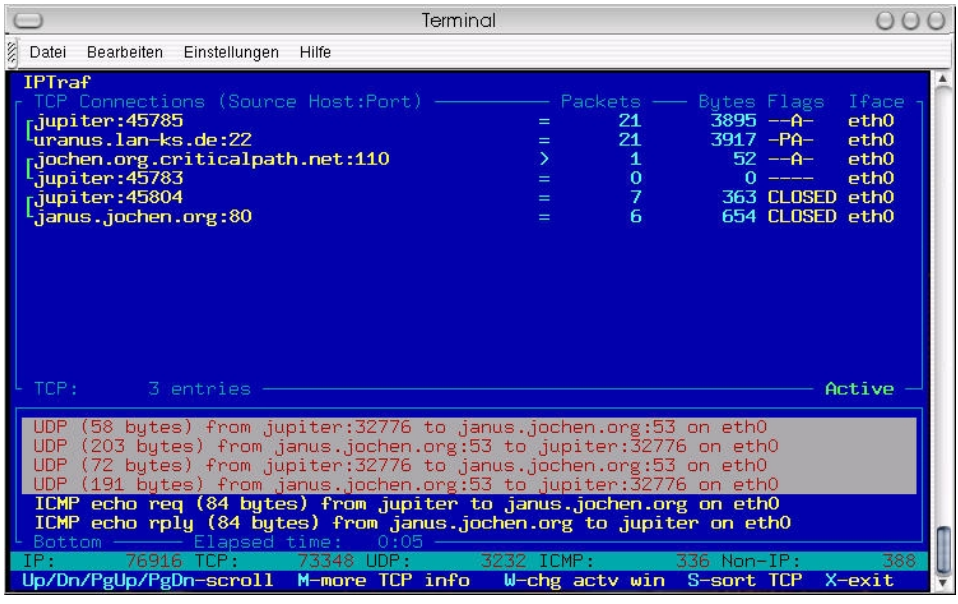


Abbildung 28.1 Das Programm `iptraf`

`iptraf` ist gut dazu geeignet, einen Überblick über den aktuellen Verkehr auf dem Netzwerk zu erhalten. Sie können via Menü verschiedene Ansichten und Anzeigen auswählen und diese beobachten. Neben dem interaktiven Aufruf kann das Programm auch im Hintergrund laufen und die Daten protokollieren. Dazu müssen Sie die Option `-B` angeben, die `iptraf` veranlasst, in den Hintergrund zu gehen. Zusätzlich müssen Sie noch auswählen, welche Daten protokolliert werden sollen, zum Beispiel `-i all` für die in der Abbildung 28.1 gezeigten Daten.

Für einen Überblick ist `iptraf` sehr hilfreich, wenn Sie bestimmte Pakete oder Verbindungen im Detail analysieren möchten, dann kann Ihnen aber nur ein Programm wie `tcpdump` weiterhelfen. `ethereal` (Abbildung 28.2) stellt die gesammelten Pakete grafisch dar, Sie können diese per Mausklick genauer untersuchen. Eine sehr nützliche Funktion finden Sie unter TOOLS->FOLLOW TCP STREAM – damit können Sie auch in einem belasteten Netz einzelne Sitzungen analysieren. Bei `telnet`-, `pop3`- oder `imap`-Sitzungen werden Sie hier das Passwort im Klartext finden.

Damit ist das Programm nicht nur zur Analyse von Netzwerkverkehr einsetzbar, sondern macht, insbesondere weil es so einfach zu bedienen ist, jedem unmissverständlich klar, warum Protokolle wie `telnet`, `ftp` oder unschlüsseltes POP3 heute einfach nicht mehr zeitgemäß sind.

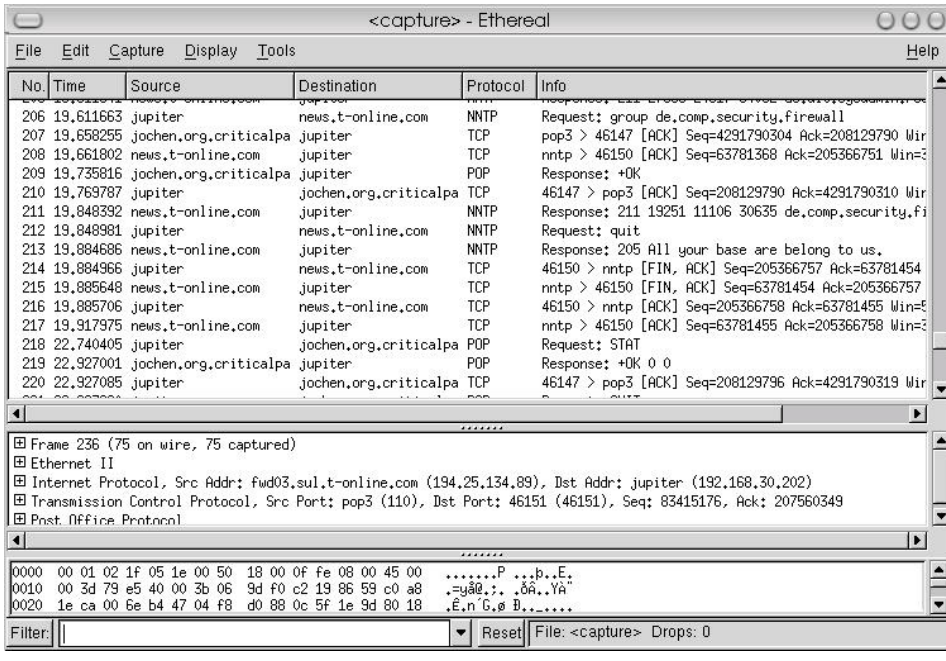


Abbildung 28.2 Das Programm `ethereal`

28.4 Performance im Netz

Eine häufige Klage von Netzbenutzern ist, dass der Datendurchsatz über das Netz viel zu langsam ist. In der Regeln speichern die Benutzer alle ihre Daten zentral auf einem (oder mehreren) Server(n). Das hat den Vorteil, dass die Daten einfach zu sichern sind und zwischen den verschiedenen Mitarbeitern ausgetauscht werden können. Die Übertragung über das Netz ist langsamer als die Speicherung auf einer lokalen Festplatte, aber die Vorteile wiegen diesen Nachteil in der Regel auf.

Noch wichtiger wird der Netzwerkdurchsatz, wenn zusätzlich das Betriebssystem und die Programme aus dem Netz geladen werden. Der Vorteil ist wieder eine einfache und zentrale Verwaltung, der Nachteil die hohe Netzlast. Weitere Netzwerkkiller sind X-Programme und das Swappen via NFS (bei plattenlosen Geräten). Für den Einbruch der Datenübertragungsrate können mehrere Ursachen verantwortlich sein.

28.4.1 Kollisionen im Netz

Zunächst kann die Netzauslastung einfach nur zu hoch sein. Das Ethernet-Protokoll (CSMA/CD, Carrier Sense Multiple Access/Collision Detect) ist so definiert, dass jeder Rechner zu jedem Zeitpunkt auf das Netz zugreifen kann. Bei einer Kollision (mehrere Rechner haben gleichzeitig Daten über das Netz geschickt) wird die Datenübertragung wiederholt.

Dabei wird nach jeder Kollision eine kurze Zeit gewartet und dann die Übertragung erneut versucht. Erkennt der Rechner erneut eine Kollision, so wird die Wartezeit verlängert. Diese Zeiten sind in gewisser Weise auch zufällig, damit nicht die beiden kollidierenden Rechner beim nächsten Versuch erneut eine Kollision verursachen.

Ein Nachteil des Ethernet-Protokolls ist, dass ab einer gewissen Netzlast die Netto-Übertragungsrate aufgrund von Kollisionen einbricht. Dieses Problem hat das Token-Ring-Protokoll nicht. Hier kreist ein spezielles Paket (Token) auf dem Ring. Eine Station, die dieses Token hat, kann dann mit dem Netz arbeiten. Nach der Übertragung wird das Token an die nächste Station weitergereicht. Sind sehr viele Stationen im Ring, so kann es (allerdings erst wesentlich später als bei Ethernet) auch zum Einbruch des Durchsatzes kommen. Wenn das Token verschwindet, weil z. B. der Rechner, der das Token gerade hat, abstürzt, dann wird von einem Rechner im Ring nach einer gewissen Wartezeit ein neues Token generiert.

Abhilfe kann prinzipiell mit zwei Aktionen geschaffen werden: Die Netzlast kann verringert werden, indem z. B. Daten lokal kopiert und dann bearbeitet werden. Das widerspricht aber der Grundidee eines Netzes, dass die Daten zentral und nur einmal gespeichert werden sollen, so dass diese Option häufig nicht praktikabel ist. Die zweite Möglichkeit ist die Teilung des Netzes (z. B. mit einer »intelligenten« Bridge, einem Switch oder gar einem Router) und die Lasttrennung zwischen den beiden Netzen. Wenn Daten zwischen zwei Rechnern in einem Netzsegment übertragen werden, ist das andere Segment frei.

Dieses Verfahren funktioniert nur dann gut, wenn die Netzwerklast relativ lokal ist. Müssen fast alle Pakete über die Bridge oder den Router, so wird dieses Gerät schnell zum Engpass. Ein weiterer Nachteil ist, dass die Latenzzeit (oder Round-Trip-Time beim `ping`) durch jedes Gerät etwas länger wird.

Unter Linux steht leider kein Programm zur Verfügung, das die Anzahl der aufgetretenen Kollisionen anzeigen kann. Man kann also nur aufgrund der Auslastung des Netzes vermuten, dass viele Kollisionen entstehen. Für größere kommerziell genutzte Netze empfiehlt sich die Anschaffung eines Analyzers, der diese Fälle untersuchen kann.

28.4.2 Überlastung des Servers

Eine weitere Ursache für den Einbruch im Netzdurchsatz kann die Überlastung des Servers sein. In vielen Netzen wird ein einzelner Rechner zunächst zum Server für viele Dienste. Im Laufe der Zeit werden die Services in der Regel immer häufiger und intensiver genutzt, so dass bald die Leistung des Servers (Prozessor- oder Plattenleistung) nicht mehr ausreicht. Hinweis auf eine Überlastung kann eine hohe Load (wird z. B. mit `uptime` angezeigt), Swapping (`vmstat`) oder wenig freier Speicher (`free`) sein.

In diesem Fall hilft es nur noch, einzelne Dienste auf einen anderen, weniger genutzten Rechner zu übertragen. Hier erweist es sich dann als sinnvoll, wenn man für jeden Service einen Aliasnamen für diesen Rechner eingetragen hat. Nach der Veränderung des Aliasnamens im Nameserver wird dieser Service von einem anderen Rechner übernommen. Es ist also nicht notwendig, auf allen Rechnern im Netz Änderungen an der Systemkonfiguration oder gar in der Benutzerkonfiguration durchzuführen. Jede Änderung an Rechnern oder am Netz sollte für Benutzer so unsichtbar wie möglich sein.

Auch die Netzwerkkarte kann ein Nadelöhr darstellen. In einem Server sollte eine gute und schnelle Karte eingebaut werden, die die CPU entlastet. Ist der Datendurchsatz durch die Karte selbst der Engpass, dann kann möglicherweise eine zweite Netzwerkkarte Abhilfe schaffen.

28.4.3 Performancemessungen mit spray

Mit dem Programm `spray` kann mit relativ einfachen Mitteln der mögliche Durchsatz über das Netz festgestellt werden. Auf einem Rechner muss der `sprayd`-Dämon gestartet werden. Anschließend können Sie mit `spray` die Verbindung zu diesem Rechner testen. Die Pakete werden mit Hilfe des UDP-Protokolls übertragen.

Wenn Sie die Werte für ein leeres und ein normal belastetes Netz kennen, können Sie möglicherweise Aussagen über den Zustand des Netzes machen. Denken Sie daran, dass Sie mit diesem Programm andere Benutzer in ihrer Tätigkeit behindern können, da das Netz restlos ausgelastet wird. Das Programm `spray` gibt am Ende eine Statistik über die gesendeten und empfangenen Pakete aus. Mittels der Optionen `-c` und `-l` können die Anzahl und die Größe der Pakete verändert werden. Die weiteren Optionen entnehmen Sie bitte der Manpage.

Listing 28.7 zeigt ein Beispiel für die Ausgabe von `spray`. Das Programm wurde nur für eine kurze Zeit gestartet, damit die anderen Benutzer im Netz nicht behindert werden. Beachten Sie, dass Sie mit vielen hier vorgestellten Programmen die Netzlast deutlich vergrößern können und damit die Lage möglicherweise verschlimmern.

```
(linux):$spray jupiter
sending 1162 packets of length 86 to jupiter...
        in 2.36 seconds elapsed time,
        0 packets (0.00Sent:   491 packets/sec, 42.3 KBytes/sec
Recvd:  523 packets/sec, 45.0 KBytes/sec
```

Listing 28.7 Das Programm `spray`

`spray` verwendet UDP als Protokoll. Damit ist eine zuverlässige Übertragung der Pakete nicht gewährleistet und die Messung ist stark von der Geschwindigkeitsdifferenz der beiden Rechner abhängig. Daher ist in der Regel das TCP-basierte `tcpspray` besser geeignet, wenn Sie Aussagen über die Netzwerk-Performance treffen wollen.

28.4.4 Performance-Messungen mit tcpspray

Alternativ zu `spray` können Sie auch `tcpspray` verwenden. Anstelle von UDP wird hier TCP zur Datenübertragung verwendet. Auch hier gilt wieder das oben Gesagte. `tcpspray` verwendet als Gegenstelle den internen `discard`-Service des `inetd` auf dem als Parameter angegebenen Rechner. Alternativ kann auch der `echo`-Port verwendet werden, wenn die Option `-e` angegeben wird.

28.4.5 Welche Systeme und Dienste belasten das Netz?

Eine etwas an `top` erinnernde Anzeige der Netzbelastung erhalten Sie mit dem Programm `ntop` (*http://www.ntop.org/*). `ntop` wird einmal gestartet und stellt seine Ergebnisse mit Hilfe eines eingebauten Web-Servers dar. In der Standardeinstellung wird der Port 3000 verwendet, so dass Sie zum Zugriff die URL *http://localhost:3000/* verwenden müssen (Abbildung 28.3). Im Web-Browser können Sie dann verschiedene Übersichten, zum Beispiel über gesendete und empfangene Pakete oder verschiedene Protokolle, ansehen. Ein Vorteil von `ntop` ist, dass das Programm nur einmal gestartet werden muss und dann mehrere Benutzer gleichzeitig mittels Web-Browser auf die gesammelten Daten zugreifen können.

ACHTUNG

Beachten Sie, dass `ntop` eine eigene Benutzerverwaltung mitbringt. Das Standard-Passwort für den Administrationsbenutzer `admin` ist »admin«. Das sollten Sie unmittelbar nach der Installation ändern.

Neben der Anzeige können Sie die von `ntop` gesammelten Daten auch in einer Datenbank exportieren (mit der Option `-b`), so dass man auch später noch Analysen durchführen kann. Eine weitere Option ist `-R`, mit der Sie verschiedene Filterregeln für Warnmeldungen hinterlegen können. Wenn Sie keinen Web-

Browser haben, dann können Sie für einige Funktionen das Kommandozeilen-Tool `intop` verwenden, allerdings sind deutlich weniger Funktionen als in `ntop` implementiert.

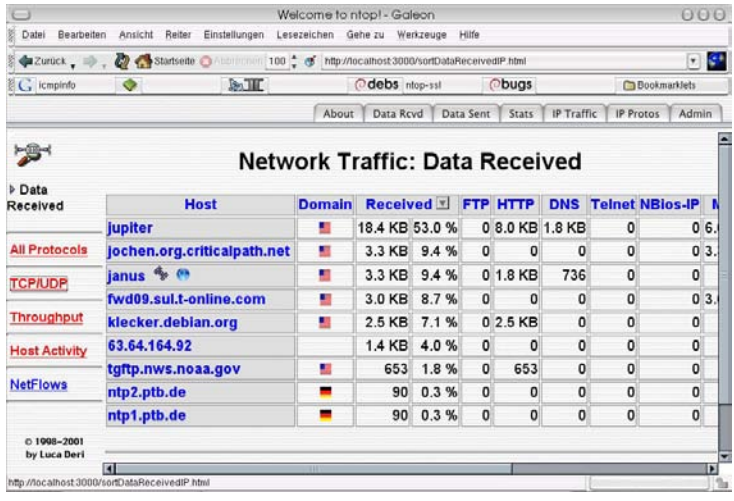


Abbildung 28.3 Die Ausgabe von `ntop` im Browser

28.5 TCP-Dienste prüfen

Zum Testen eines TCP-Dienstes kann das Programm `telnet` verwendet werden. Dazu baut man eine Verbindung direkt zum gewünschten Port auf. Anschließend können Sie den Dienst direkt testen. Versuchen Sie einfach ein `telnet localhost smtp`. Wenn Sie wissen, wie das Protokoll abläuft (das ist in [rfc2821] festgelegt), dann können Sie diesen Service manuell testen.

Oft können Sie eine neue Version des Dämons unter einem anderen Port starten und dort in Ruhe testen, bevor Sie die alte Version ersetzen. Wenn ein System bereits von Benutzern verwendet wird, sollte man viel Wert darauf legen, dass Änderungen für diese nach Möglichkeit nicht sichtbar sind.

Listing 28.8 zeigt ein Beispiel für das Testen des POP3-Servers, mit dem viele Internet-Provider ihren Kunden Zugriff auf das Postfach auf dem Server geben. Beachten Sie, dass Sie das Passwort hier im Klartext sehen, genauso wie es zwischen den Rechnern übertragen wird. Oft kann man hier Meldungen lesen, die vom Client-Programm falsch interpretiert werden.

```
telnet localhost pop
Trying 127.0.0.1...
Connected to localhost.
```

```
Escape character is ']'.
+OK POP3 localhost v6.50 server ready
user jochen
+OK User name accepted, password please
pass Passwort
+OK Mailbox open, 0 messages
quit
+OK Sayonara
Connection closed by foreign host.
```

Listing 28.8 Testen eines POP3-Servers

Ein weiteres Programm, das zur Fehlersuche und in Skripten nützlich sein kann, ist `netcat` (*ftp://avian.org/src/hacks/*). Es gilt als das »Schweizer Taschenmesser« für TCP/IP. Das Programm ermöglicht es Shell-Skripten, auf TCP/IP-Ports zuzugreifen. Neuere Versionen der `bash` und zum Beispiel `awk` unterstützen dies heute auch. `netcat` (auf den Systemen meist als `nc` installiert) kann allerdings einige Dinge mehr (wie `telnet`-Negotiation oder Source-Routing). Zum Vergleich finden Sie eine ähnliche Sitzung wie im `telnet`-Beispiel in Listing 28.9.

```
(linux):~> nc localhost pop3 >>EOF
user jochen
pass Passwort
list
quit
EOF
+OK Qpopper (version 4.0.3) at jupiter starting.
>867.1013595502@jupiter>
+OK Password required for jochen.
+OK jochen has 2 visible messages (0 hidden) in 5745 octets.
+OK 2 visible messages (5745 octets)
1 2270
2 3475
.
+OK Pop server at jupiter signing off.
```

Listing 28.9 TCP am Shellprompt mit `netcat`

`netcat` wurde von mir beispielsweise als TCP-Relay eingesetzt, ich wollte eine NNTP-Sitzung stets über einen speziellen Rechner aufbauen. Hier wäre prinzipiell auch der Einsatz von `xinetd` möglich gewesen, auf dem betreffenden Rechner sollte allerdings der `inetd` nicht ersetzt werden. `netcat` wird ohne Manpage ausgeliefert, die Dokumentation finden Sie in der zugehörigen `README`-Datei.

Wenn ich Ihnen nun schon den Mund wässrig gemacht habe, die `bash` interpretiert die Dateinamen `/dev/tcp/`*`Host`*`/`*`Port`* bzw. `/dev/udp/`*`Host`*`/`*`Port`* und öffnet eine TCP- bzw. UDP-Verbindung zu dem angegebenen Rechner und Port.

28.6 Fehlersuche bei RPC-Services

Bei der Verwendung von RPC-Services können neben den oben bereits besprochenen Problemen weitere Fehler auftreten. Zunächst sollten Sie überprüfen, ob der Portmapper läuft. Anschließend können Sie mit `rcpinfo -p` *`Host-Name`* feststellen, welche Services beim Portmapper registriert sind. Im letzten Schritt können Sie mit `rpcinfo -u` *`Host-NameService`* jeden Dämon testen.

28.7 Network File System

Das Network File System basiert auf RPCs, so dass Sie zunächst die oben bei RPC-Services angegebenen Dinge überprüfen sollten. Mit dem Programm `showmount` können Sie feststellen, welche Volumes der entsprechende Rechner exportiert. Sie können auch feststellen, auf welchem Client welche NFS-Volumes verwendet werden.

Wenn Sie Probleme mit dem Datendurchsatz durch NFS haben, so können Sie möglicherweise die Puffergrößen für Schreib- und Lesezugriffe auf 4 bzw. 8 Kbyte erweitern. Dies sollte in der Regel eine deutlich größere Schreib- bzw. Lesegeschwindigkeit ermöglichen.

28.8 Hacker-Tools

Im Internet stehen viele Tools zur Verfügung, die einerseits zum Angriff auf fremde Rechner dienen können, andererseits für Administratoren nützlich sind, um die Verletzbarkeit der eigenen Systeme zu prüfen. Hier möchte ich nur zwei Programme kurz vorstellen, `nmap` und `nessus`.

28.8.1 Der Portscanner nmap

Das Programm `nmap` ist ein einfach zu bedienender, aber trotzdem sehr leistungsfähiger Portscanner. Wenn man als einzigen Parameter einen Host-Namen angibt, so erfolgt ein Portscan auf diesen Host – dabei werden allerdings nicht alle Ports untersucht, sondern nur eine Auswahl (Listing 28.10). Es ist immer wieder überraschend, wie viele Dienste auf einem Rechner aktiviert sind, von denen der Administrator nicht immer weiß, wozu sie gut sind.

```
(linux):~$ nmap jupiter
Starting nmap V. 2.54BETA30 ( www.insecure.org/nmap/ )
Interesting ports on jupiter.jochen.org (192.168.30.202):
(The 1539 ports scanned but not shown below are in state: closed)
Port       State       Service
```

```
22/tcp      open        ssh
80/tcp      open        http
111/tcp     open        sunrpc

Nmap run completed -- 1 IP address (1 host up) scanned in 14
seconds
```

Listing 28.10 Ein Portscan mit `nmap`

Beachten Sie, dass manche der in diesem Kapitel vorgestellten Programme entweder selbst Portscans durchführen (können) oder sogar `nmap` verwenden. Eine Verbindung zu einem Port aufzubauen, ist im Internet die einzige Möglichkeit, festzustellen, ob ein Rechner einen bestimmten Dienst anbietet. Aus diesem Grund ist es sicher legitim, wenn man ein Tool wie `nmap` benutzt, um herauszufinden, ob ein Rechner einen Dienst anbietet. Auf der anderen Seite kann ein Portscan über mehrere Ports oder mehrere Rechner Vorbote für einen gezielten Angriff sein – daher fassen viele Administratoren einen Portscan bereits als Angriff auf. Sagen Sie nicht, ich hätte Sie nicht gewarnt.

Mit der Option `-O` versucht `nmap` zu raten, welches Betriebssystem auf dem Rechner aktiv ist. Das ist für einen gezielten Angriff nicht unwichtig ... Etwas ausgefeilter finden Sie diese Funktion übrigens in dem Programm `queso`.

28.8.2 Der Security-Scanner nessus

In der Praxis erlebt man es häufig, dass ein Angreifer einfach (per Skript gesteuert) diverse Angriffe ausführt, egal, ob diese auf das System zutreffen könnten oder nicht. Bei einem Angriff auf viele Rechner macht es nichts, wenn der Angriff bei den meisten Rechnern ins Leere läuft. Irgendeiner der Rechner wird schon verwundbar sein...

Gegen solche Angriffe kann man sich nur durch eine restriktive Konfiguration und aktuelle Sicherheitsupdates schützen. Das Programm `nessus` (*http://www.nessus.org/*) enthält Informationen über eine Reihe von bekannten Sicherheits- und Konfigurationsproblemen. Ein Rechner wird intensiv mittels verschiedener Plugins gescannt und das Ergebnis wird übersichtlich aufbereitet (Abbildung 28.4). Als Admin muss man dann »nur noch« die richtigen Konsequenzen ziehen.

Versetzen Sie sich einmal in die Lage eines Angreifers und versuchen Sie von außen Informationen über Ihre Systeme zu sammeln. Auch hier wieder werden Sie erstaunt sein, wie viele Informationen Sie finden werden. Die hier vorgestellten Programme unterstützen Sie nicht unbedingt gegen jeden Angriff, können Ihnen aber helfen, Ihre Systeme besser zu verstehen. Nutzen Sie diese Gelegenheit.

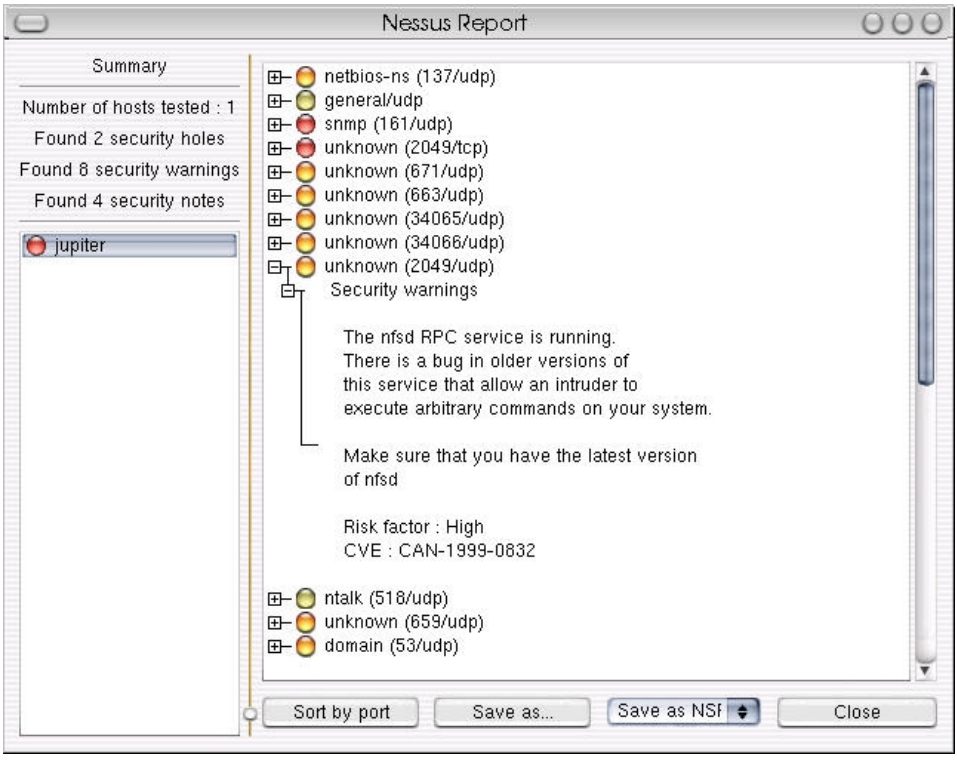


Abbildung 28.4 Das Scan-Ergebnis von `nessus`

28.9 IP-Netzverwaltung

In vielen Firmen wird das IP-Netz mit Hilfe von SNMP (Simple Network Management Protocol, [rfc1157]) verwaltet. Jedes mit SNMP verwaltete Gerät – das können Workstations, Router, Switches, Drucker und vieles andere mehr sein – strukturiert die Informationen in der Management Information Base (MIB). Die MIB ist in den Internetstandards [rfc1441] und [rfc2578] ff. definiert. Dort werden neben den (statischen) Informationen über das Gerät (z. B. Standort, verantwortlicher Mitarbeiter) auch dynamische Daten verwaltet. Dazu zählen Routing-Tabellen, Interface-Statistiken oder allgemeine Informationen über das Gerät (z. B. Load bei Unix-Workstations oder der verfügbare Plattenplatz).

Der Zugriff auf die in der MIB gespeicherten Informationen wird über die Community geregelt. Die Standard-Community ist `public` und erlaubt den lesenden Zugriff auf die Daten. Die Veränderung der Daten sollte nur für eine andere Community erlaubt werden. Es kann außerdem sinnvoll sein, den Zugriff auf die MIB auf die IP-Adresse der Netzverwaltungsstation zu beschränken.

Einzelne Geräte können sich bei der Netzwerkkonsole mit einem SNMP-Trap melden und dadurch verschiedene Aktionen auslösen. Ein großer Nachteil in heterogenen Netzen ist jedoch, dass hier nur IP-Geräte erfasst werden können. Andere Protokolle werden nicht erfasst, so dass Sie zu diesem Zweck möglicherweise ein weiteres Tool benötigen.

28.9.1 Linux als SNMP-Client

Für Linux ist z. B. das `smnp`-Paket der Carnegie-Mellon University (CMU) verfügbar. Damit können viele Informationen über einen Linux-Rechner von einer Netzwerkverwaltungsstation aus abgerufen werden. Diese Informationen reichen von Daten über den Netzanschluss des Rechners bis hin zur CPU-Auslastung oder Belegung der Festplatten.

In diesem Paket enthalten sind einfache Programme, mit denen Sie Einträge aus der MIB (oder die gesamte MIB) anzeigen lassen können. Diese Programme sind aber für den täglichen Einsatz viel zu unkomfortabel.

28.9.2 Linux zur Netzwerkverwaltung

An der TU Braunschweig wurde zur Netzwerkverwaltung mit SNMP das Paket `tkined` (*ftp://ftp.tu-bs.de//pub/local/tkined*) entwickelt (das Source-Archiv heißt scotty, möglicherweise heißt das Paket Ihrer Distribution genauso). Dieses Programmpaket wurde in Tcl/Tk (mit Erweiterungen zum Netzzugriff) implementiert. Den Quellcode dazu finden Sie auf dem `ftp`-Server der TUÂ Braunschweig (*ftp://ftp.tu-bs.de/local/tkined/*).

Mit diesem Programm erstellen Sie eine Übersichtskarte (Map) Ihres Netzes. Basierend auf dieser Map können Sie verschiedene Verwaltungsaufgaben erledigen und Statistiken (auch grafisch) erstellen. Dieses Tool ist eine kostengünstige Alternative zu kommerziellen Paketen, die Sie z. B. von IBM, Sun oder HP bekommen können.

Abbildung 28.5 zeigt die Darstellung einer einfachen Map. In der Mitte ist ein Switch dargestellt, der die verschiedenen Netzsegmente miteinander verbindet. Diese Segmente sind hier als Icon dargestellt, können aber bei Bedarf aufgeklappt werden. Es ist dann der Zugriff auf alle Geräte in diesem Segment möglich. Jedem Segment ist hier eine Statistik der Netzauslastung zugeordnet.

Eine neuere Entwicklung war das Programm `cheops`, dessen Nachfolger `cheops-ng` unter *http://cheops-ng.sourceforge.net/* zu finden ist. Abbildung 28.6 zeigt mein lokales Netz.

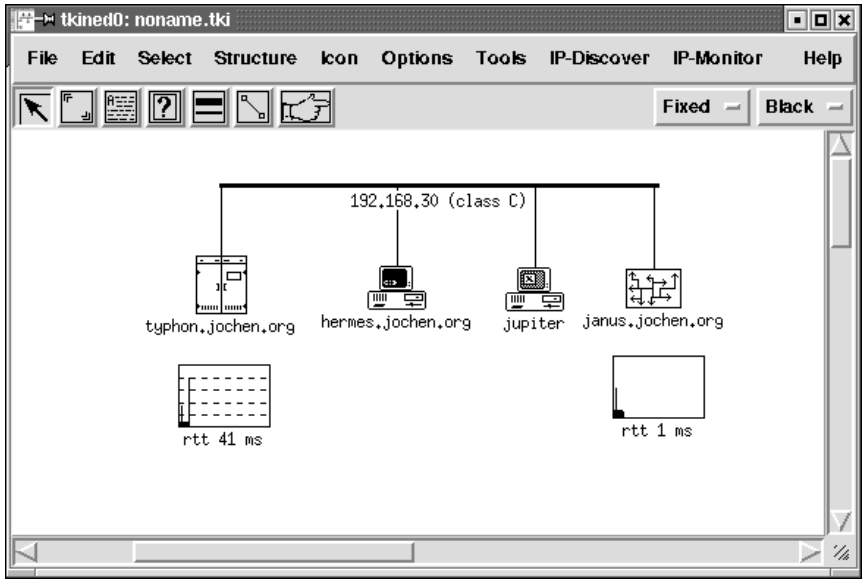


Abbildung 28.5 Netzwerkverwaltung mit `tkined`

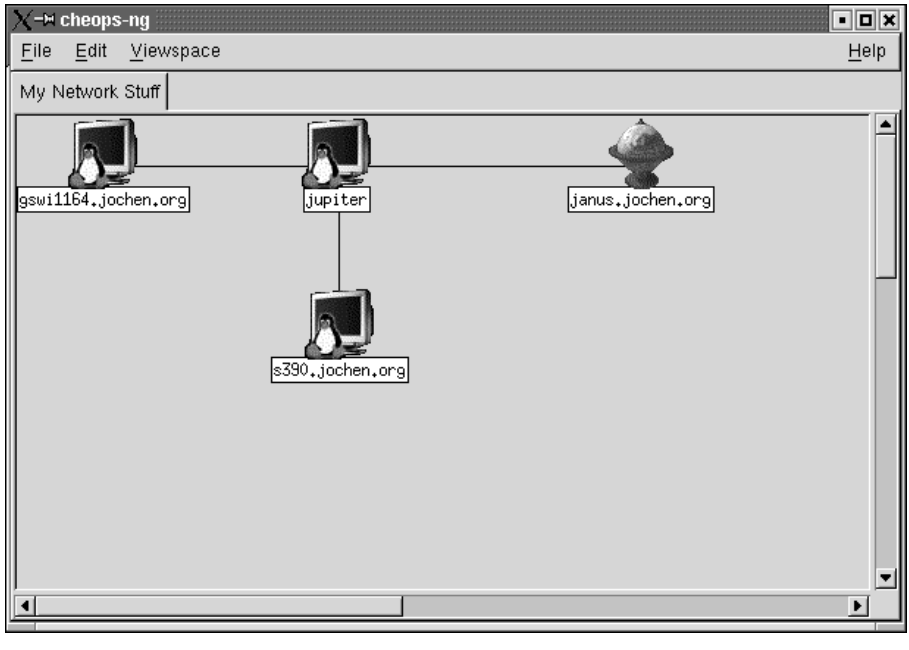


Abbildung 28.6 Mein lokales Netzwerk in `cheops-ng`

Ein weiteres Project ist OpenNMS (*http://www.opennms.org/*, ein System, das nicht nur Netzwerke und deren Komponenten (wie Router und Switches) überwacht, sondern auch Rechner und darauf laufende Applikationen überwachen kann. Die Scan-Ergebnisse werden mit Hilfe eines Web-Browsers dargestellt, dabei gibt es eine übersichtliche Zusammenfassung, Grafiken und Statistiken zu Performance und Verfügbarkeit.

Wenn Sie nur ein Tool benötigen, das z. B. die aktuelle Auslastung des Netzes grafisch darstellt, dann können Sie das Programm `mrtg` (Multi Router Traffic Grapher, *http://www.mrtg.org/*) verwenden. In der Konfigurationsdatei legen Sie fest, welche Werte überwacht werden sollen. In der Regel werden Sie hierfür die entsprechenden Geräte mittels SNMP abfragen. Das Programm wird alle fünf Minuten mittels `cron` gestartet, fragt die Daten ab und generiert Grafiken der täglichen, wöchentlichen, monatlichen und jährlichen Auslastung.

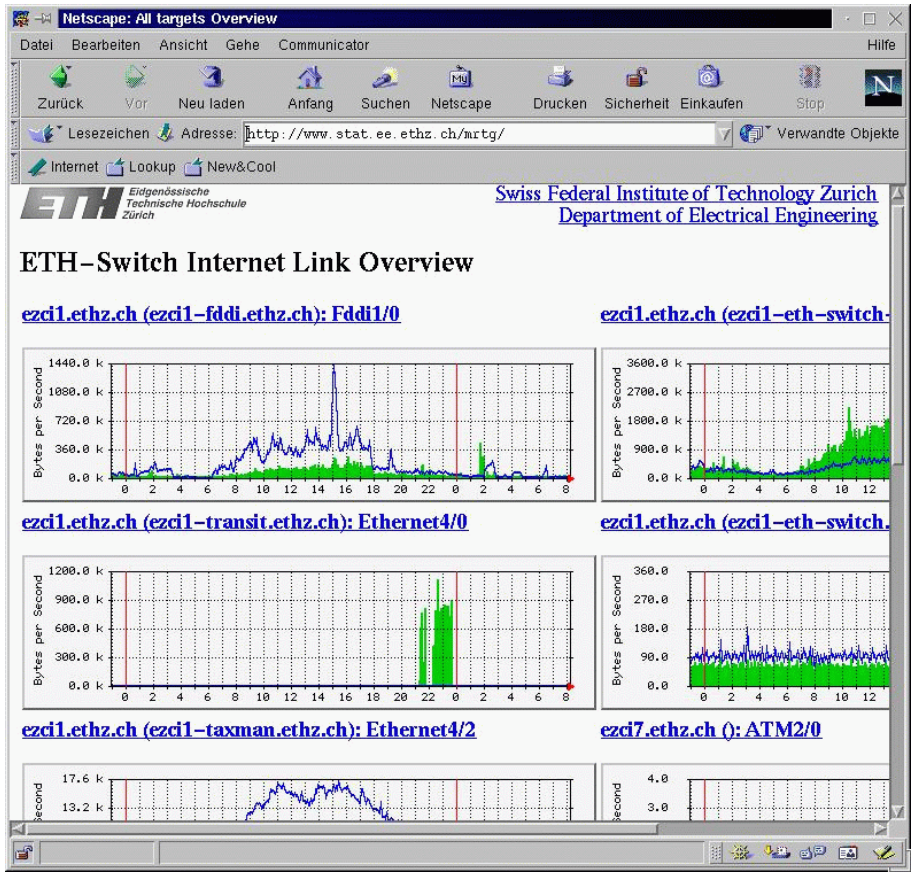


Abbildung 28.7 Grafik mit `mrtg`

Abbildung 28.7 zeigt eine Webseite (ein Beispiel, auf das auf der Webseite *http://www.mrtg.org* verwiesen wird), die mit Hilfe von `mrtg` erstellt wurde. Sie können aber nicht nur Daten mittels SNMP abfragen. Es ist auch möglich, ein externes Programm aufzurufen, das beliebige Daten beschaffen und an `mrtg` übergeben kann. `mrtg` verdichtet die gelesenen Daten sofort, so dass auch für die Messwerte der Jahresgrafik nur sehr wenig Plattenplatz benötigt wird. Für einige wenige Werte ist die Belastung des Rechners vernachlässigbar – wenn Sie viele Systeme und Parameter darstellen wollen, so benötigen Sie möglicherweise einen eigenen Rechner dafür.

A Der Standardeditor vi

The BEST EDITOR EVER
Stuart Woolford über VIM

A.1 Der Editor vi

Von vielen gehasst, von manchen geliebt, ist `vi` der Standardeditor unter den Unix-Systemen geworden. Vor `vi` gab es nur Zeileneditoren (wie `ex` oder `ed`), seitdem (das war etwa 1978) kann man »visuell« editieren. Man sieht einen großen Ausschnitt aus dem Text und alle Änderungen, die man dort durchführt. Aus dieser Zeit stammt auch die Umgebungsvariable `VISUAL`, die wie `EDITOR` zur Auswahl des Standardeditors dient.

Die meisten Linux-Distributionen enthalten einen der zwei verbreiteten `vi`-Clones `elvis` oder `vim`. Beide Editoren sind zum Vorbild weitgehend kompatibel, besitzen aber erweiterte Funktionen. `vim` kann z. B. spaltenweise Text markieren und besitzt einen Makrorekorder. Innerhalb der frei verfügbaren Systeme (*BSD und GNU) wird als `vi`-Clone der `nvi` eingesetzt. Hier ist das Ziel, einen wirklich kompatiblen `vi`-Clone zu entwickeln, der sogar die Fehler des Originals hat.

Es ist sehr zu empfehlen, sich zumindest mit den Grundfunktionen von `vi` vertraut zu machen. `vi` befindet sich meist als einziger Editor auf Boot- oder Notfalldisketten, da er sehr klein ist (ca. 100 Kbyte). Außerdem steht dieser Editor, im Gegensatz zu Emacs und anderen freien Editoren, auf jedem kommerziellen Unix-System zur Verfügung. `vi` ist sehr schnell. Er wird daher oft von Systemadministratoren verwendet, um »mal eben« eine Datei zu editieren, und kann sehr einfach auf anderen Rechnern via `ssh` genutzt werden.

`vi` steht für »visual editor«, da Änderungen am Text sofort am Bildschirm angezeigt werden. Wird das Programm mit dem Namen `ex` aufgerufen, so wird nur ein Zeilenmodus verwendet. Wenn Sie nur eine Datei ansehen wollen, so können Sie mit `view` diese Datei nur-lesbar in den Editor laden.

Der einfachste Aufruf ist `vi` *`Dateiname`*. Der Editor `vi` arbeitet in verschiedenen Modi (Kommandomodus, Einfügemodus und Kommandozeilenmodus), in denen die Tastatur unterschiedlich belegt ist. Diese Unterscheidung entstand aus dem Wunsch, dass `vi` auch bei langsamen Terminals und Netzverbindungen vernünftig zu bedienen sein sollte. Mit dem Befehl `set showmode` in der Datei `~/.exrc` können Sie sich anzeigen lassen, in welchem Modus `vi` sich befindet. Heute wirkt diese Art der Bedienung antiquiert, wenn jedoch die Tastaturbefehle nicht mehr aus dem Kopf, sondern aus den Fingern kommen, geht die Arbeit mit `vi` leicht von der Hand.

Nach dem Laden des Textes befindet man sich normalerweise im Kommandomodus. In diesem Modus werden Befehle durch einfache Tastendrücke ausgelöst. So bewirkt das Drücken der Taste [x] nicht, dass »x« auf dem Bildschirm erscheint, sondern dass das Zeichen unter dem Cursor gelöscht wird. Beinahe jede Taste ruft im Kommandomodus ein bestimmtes Kommando auf. Dabei wird zwischen Groß- und Kleinschreibung unterschieden. Es existieren auch Kommandos, die zusammen mit der Steuerungstaste [Strg] eingegeben werden müssen.

Andere, komplexere Kommandos lassen sich im Kommandozeilenmodus eingeben, in den man im Kommandomodus mit dem Doppelpunkt [:] kommt. Daher heißt der Modus auch Colon- oder Doppelpunktmodus. Der Cursor befindet sich dann in der untersten Bildschirmzeile hinter einem Doppelpunkt. Eine Übersicht über die wichtigsten Kommandos finden Sie am Ende dieses Abschnitts (siehe Tabelle A.1).

Ein anderer wichtiger Modus ist der Einfügemodus, der mit der Taste [i] eingeschaltet wird. Um nach der Eingabe von Text aus dem Einfügemodus wieder in den Kommandomodus zurückzugelangen, muss man die Taste [Esc] drücken. Wenn Sie nicht wissen, in welchem Modus Sie bzw. vi sich befinden, dann betätigen Sie einfach die [Esc]-Taste. Soll bestehender Text überschrieben werden, so kann man das im Überschreibmodus erledigen. In den Überschreibmodus gelangt man durch Drücken der Taste [R] im Kommandomodus. Wenn Sie nur ein einzelnes Zeichen ersetzen wollen, können Sie das im Kommandomodus mit der Taste [r], gefolgt von dem neuen Zeichen, tun. Text kann durch Drücken der Taste [x] zeichenweise gelöscht werden. Soll eine ganze Zeile gelöscht werden, so kann dies durch zweimaliges Drücken der Taste [d] geschehen.

Will man sich in der Textdatei bewegen, so funktioniert dies meist mit den Cursor-Tasten, aber nur im Kommandomodus (das ist allerdings je nach vi-Version und Terminal unterschiedlich). Sollte einmal die Terminal-Anpassung nicht stimmen, so kann man sich trotzdem (im Kommandomodus) durch Drücken der Tasten [h] nach links, [l] nach rechts, [j] nach unten und [k] nach oben im Dokument bewegen. Seitenweises Blättern ist durch Drücken von [Strg]-[f] (vorwärts) und [Strg]-[b] (rückwärts) möglich. Ein Nachteil der Cursor-Tasten ist (abhängig von der Kompatibilität des vi-Clone), dass sie ein Kommando auslösen. Danach ist weder ein Undo mit [u] noch die Wiederholung des letzten Befehls mit [.] möglich.

Um die Textdatei zu speichern, muss man im Kommandomodus [:] [w] eingeben. Dadurch wird die Datei unter dem aktuellen Namen gespeichert. Soll die Datei unter einem anderen Namen abgespeichert werden, kann der Dateiname mit angegeben werden, beispielsweise [:] [w] **Dateiname**. Wenn die Datei schreibgeschützt ist, kann man diese mit [:] [w] [!] trotzdem überschreiben. Moderne vi-Versionen haben hier mit der [Tab]-Taste eine Dateinamensvervollständigung eingebaut.

vi kann durch Eingeben des Kommandos [:][q] verlassen werden. Wurde die Datei verändert, so wird dies durch die Meldung »Use q! to abort changes, or wq to save changes« mitgeteilt. Es besteht die Möglichkeit, das Dokument zu speichern und vi zu beenden ([:][w][q]) oder die Änderungen im Dokument zu verwerfen und vi zu beenden ([:][q][!]).

Viele Kommandos können durch ein numerisches Präfix mehrfach ausgeführt werden. Die Tastenkombination [5][x] löscht im Kommandomodus beispielsweise die nächsten fünf Zeichen. Für viele Funktionen stehen auch einfache Kürzel bereit, wie z. B. [c][w] bzw. [d][w] für »change word« bzw. »delete word«. Auch hier können Sie wieder ein Präfix angeben, damit der Befehl mehrfach ausgeführt wird.

Im vi können Sie im Kommandomodus mit [/] suchen. vi verfügt natürlich auch über eine Suchen-und-Ersetzen-Funktion. Die hierbei verwendete Schreibweise erinnert an die bei sed verwendete (siehe Abschnitt 9.7, »Textdateien bearbeiten mit sed«). Zunächst ist das Suchen und Ersetzen ein Doppelpunkt-Befehl, muss also im Kommandozeilenmodus eingegeben werden. Dann ist der Bereich anzugeben, in dem ersetzt werden soll. Das kann durch Zeilennummern (hier ist der Befehl :set nu hilfreich) oder Suchbegriffe geschehen.

Der Befehl zum Suchen und Ersetzen lautet s, gefolgt von dem Suchbegriff und der Ersetzung. Anschließend muss ein g angehängt werden, wenn diese Ersetzung in einer Zeile auch mehrfach durchgeführt werden soll. Das klingt kompliziert, deshalb folgt in Listing A.1 ein kurzes Beispiel. Dabei kann vi nach regulären Ausdrücken suchen. Moderne vi-Versionen verfügen im Doppelpunktmodus über eine History, durch die Sie mit den Cursor-Tasten blättern können.

```
1,\$ s/gesucht/ersetzt/g
```

Listing A.1 Suchen und Ersetzen im vi

Im gesamten Text (von Zeile 1 bis zum Ende, eine Kurzform dafür ist %) wird die Zeichenkette gesucht durch die Zeichenkette ersetzt ausgetauscht. Durch diese zunächst kompliziert wirkende Syntax ist vi sehr flexibel einsetzbar, insbesondere die regulären Ausdrücke können oft sehr sinnvoll eingesetzt werden.

Wenn die im vi eingebauten Funktionen nicht ausreichen, können externe Programme aufgerufen werden. Dabei kann die ganze Datei oder nur der aktuelle Absatz als Eingabe verwendet werden und die Ausgabe des Programms tritt an die Stelle der Eingabe. Außerdem können Sie im Doppelpunktmodus mit [!] jedes beliebige Programm direkt aufrufen. Listing A.2 zeigt, wie die Ausgabe des date-Befehls in die Datei eingefügt und die gesamte Datei durch den Filter fmt verarbeitet wird.

```
:r !date
:% |fmt
```

Listing A.2 Doppelpunktbefehle im `vi`

Das Kommando [:][r] kann auch dazu verwendet werden, eine Datei vollständig einzulesen und an der Cursorposition einzufügen. Das Beispiel `:% | fmt` formatiert den gesamten Text mit Hilfe des Programms `fmt`. Mit `!}fmt` formatieren Sie nur den aktuellen Absatz.

Jeder Benutzer kann sich eigene Makros definieren oder `vi` an die eigenen Bedürfnisse anpassen. Als Konfiguration wird die Datei `~/.exrc` gelesen, ein Beispiel finden Sie in Listing A.3. Hier können Sie Funktionstasten belegen und andere Einstellungen vornehmen. Je nach Einstellung der Option `exrc` kann zusätzlich zur Datei `~/.exrc` aus dem Home-Verzeichnis auch die Datei `.exrc` aus dem aktuellen Verzeichnis eingelesen werden. Als Kommentarzeichen werden die doppelten Anführungszeichen verwendet.

```
" Ein Auszug aus einer .exrc-Datei
" Anzeige des aktuelles Modes
set showmode
" Anzeige von Zeilennummern
set number
" Tastatur-Makros
map <F3> :r !date<CR>
```

Listing A.3 Eine `~/.exrc`

Damit ist die Leistungsfähigkeit von `vi` noch lange nicht erschöpft. `vi` arbeitet z. B. mit Hilfsprogrammen wie `ctags` zusammen und wird daher von vielen Programmierern und Systemadministratoren eingesetzt. Es empfiehlt sich, weitere Literatur zu Rate zu ziehen, z. B. das `vi`-Tutorial, das gemeinsam mit `nvi` vertrieben wird. Auch im Paket des `vim` befindet sich ein gutes, aber in Englisch geschriebenes Tutorial. Daneben ist auch die `vi`-FAQ eine gute Informationsquelle. Tabelle A.1 enthält eine Übersicht über einige `vi`-Befehle im Kommandomodus.

Taste(n)	Funktion
h	Cursor links
l	Cursor rechts
j	Cursor nach unten
k	Cursor nach oben
Strg-f	Seite vorwärts
Strg-b	Seite zurück
i	Einfügemodus aktivieren
r	Ein Zeichen überschreiben
R	Überschreibmodus aktivieren
x	Zeichen löschen
.	Letztes Kommando wiederholen
u	Letztes Kommando zurücknehmen
d d	Zeile löschen
/ *Suchbegriff*	Suche nach dem Suchbegriff
n	Weitersuchen
: *Zeilennummer*	Gehe zur Zeile *Zeilennumer*
: $	Gehe zur letzten Zeile
: q	`vi` beenden
: q!	`vi` beenden, ohne zu speichern
Z Z	`vi` beenden und speichern
: w q	`vi` beenden und speichern
: w *Dateiname*	Datei speichern
: r *Dateiname*	Datei laden

Tabelle A.1 Die wichtigsten `vi`*-Befehle*

B Passwörter generieren

»Hey, what's the password?« - »Ken sent me«
aus Leisure Suite Larry

B.1 Passwörter als Zugriffsschutz

Die Sicherheit der meisten Computersysteme hängt von der Sicherheit der verwendeten Passwörter ab. Für Bereiche mit besonders hohen Anforderungen werden stellenweise Chipkarten und andere Hardware-Lösungen eingesetzt. Für die normalerweise erforderliche Sicherheit sind diese jedoch nicht notwendig.

In der Regel sollten Benutzer ihre Passwörter selbst wählen, damit sie sich diese auch merken können. Als Administrator sollten Sie allerdings darauf achten, dass Ihre Benutzer gute Passwörter wählen. Dabei können Ihnen Programme wie `npasswd`, `passwd+` oder `crack` helfen. Außerdem sollten Sie Ihre Benutzer dazu anhalten, Passwörter regelmäßig zu wechseln. In der Shadow-Password-Suite ist diese Funktion neben vielen anderen bereits enthalten.

Für spezielle Benutzerkennungen ist es auch in normalen Umgebungen sinnvoll, besonders sichere Passwörter zu wählen. Dies ist z. B. bei allen Administratorkennungen nützlich, damit diese nicht mittels einer `crack`-Attacke erraten werden können. Leider kennt Unix keine feinere Aufteilung der Administratorrechte, wie sie beispielsweise unter VMS existiert, so dass ein kompromittierter `root`-Benutzer bedeutet, dass das ganze System nicht mehr vertrauenswürdig ist.

B.2 Passwörter mit Würfeln festlegen

Eine einfache Methode, ein Passwort zu erzeugen, besteht darin, es mit einem Passwortgenerator auszuwürfeln. Damit schließt man Unsicherheiten bei der Verwendung eines maschinellen Zufallszahlengenerators aus und verhindert das einfache Erraten des Passworts. Diese Art von Passwörtern bietet sich besonders für privilegierte Benutzerkennungen wie z. B. `root` an. Die Tabelle auf der nächsten Seite wurde von Jörg Czeranski (*mailto:joerch@joerch.org*) freundlicherweise zur Verfügung gestellt.

Für jedes Zeichen des Passworts würfeln Sie mit drei verschiedenen Würfeln (bzw. dreimal mit einem Würfel). Der erste Würfel bzw. Wurf gibt an, welche Tabelle verwendet wird. Der zweite Würfel (oder Wurf) bestimmt die Zeile, in der das Zeichen steht, und der dritte die Spalte. Damit ist ein Zeichen zufällig festgelegt. Für jedes weitere Zeichen wiederholen Sie dieses Verfahren, bis Sie

sechs bis acht Zeichen haben. Ein Passwort, das aus weniger als sechs Zeichen besteht, ist in der Regel zu kurz, um wirklich sicher zu sein. Passwörter sollten aber auch nicht länger als acht Zeichen sein, damit sie sich sinnvoll merken lassen und weil einige Programme nur die ersten acht Stellen akzeptieren.

Wenn Sie mehrere Rechner mit unterschiedlichen Tastaturlayouts (z. B. mit US-amerikanischer und deutscher Belegung) verwalten, so ist es möglicherweise sinnvoll, nur solche Zeichen zuzulassen, die auf beiden Tastaturlayouts an der gleichen Stelle zu finden sind. Auch die Verwendung eines Leerzeichens ist nicht immer sinnvoll, da diese Taste vom Hinsehen und Hinhören recht deutlich auffällt. Ähnliches mag, je nach Tastatur, auch für die Umschalttasten gelten.

1	**1**	**2**	**3**	**4**	**5**	**6**
1	o	p	3	l	a	%
2	1	k	M	B	7	m
3	D	B	w	m	v	0
4	e	R	8	z	2	u
5	d	x	4	M	C	Z
6	2	$	i	N	O	g
2	**1**	**2**	**3**	**4**	**5**	**6**
1	j	V		A	u	2
2	x	t	9	S	4	r
3		C	P	j	o	J
4	d	-	T	K	n	#
5	8	9	X	E	,	e
6	6	U	3	k	!	b
3	**1**	**2**	**3**	**4**	**5**	**6**
1	8	b	Q	C	W	.
2	f	Z	P	0	g	q
3	s	X	*	7	O	q
4	Z	&	N	y	6	W
5	|	P	n	J	l	7
6	y	F	S	A	;	c
4	**1**	**2**	**3**	**4**	**5**	**6**
1	<	U	s	R	D	K
2	v	c	"	G	=	)
3	g	i	5	;	3	b
4	x	c	=	f	M	a
5	f	(	G	z	I	t
6	/	?	v	5	p	Q

5	**1**	**2**	**3**	**4**	**5**	**6**
1	i	I	d	^	h	/
2	N	y	n	E	T	e
3	H	>	U	t	L	E
4	_	9	r	1	A	h
5	+	j	W	s	H	q
6	o	B	r	K	V	w
6	**1**	**2**	**3**	**4**	**5**	**6**
1	z	J	L	w	5	H
2	O	6	Y	,	F	X
3	Q	S	I	p	Y	V
4	4	G	F	.	l	k
5	m	D	1	T	u	R
6	0	L	Y	-	h	a

Tabelle B.1 Zuordnung von Würfelergebnissen zu Zeichen eines Passworts

B.3 Passwörter maschinell erzeugen

Es gibt natürlich auch einige Programme, mit denen Passwörter maschinell erzeugt werden können. Das größte Problem dabei ist, dass Computer Schwierigkeiten mit der Erzeugung von Zufallszahlen haben – es werden in der Regel so genannte Pseudozufallszahlen generiert. Diese sehen von außen zufällig aus, sind aber möglicherweise für kryptographische Analysen anfällig – für den Fall, dass man nur das erste Passwort eines Benutzers vorgeben möchte, kann man vermutlich damit leben. Ein solches Programm finden Sie unter *http://www.inka.de/~bigred/sw/pwgen.txt*.

C Literaturverzeichnis

C.1 Hinweise zum Literaturverzeichnis

Hier finden Sie Verweise auf Literatur und andere Bücher, die Themen intensiver behandeln. Ein Teil der Bücher gilt als »Standardwerke«, um deren Lektüre man früher oder später nicht mehr herumkommt, wenn man sich mit dem Thema beschäftigt.

C.2 Literaturverzeichnis

[Albitz2001]
DNS and BIND, Paul Albitz, Cricket Liu, O'Reilly and Associates, April 2001, ISBN 0-596-00158-4.

Das Standardwerk zu BIND

[Barret2001]
SSH: Secure Shell – Ein umfassendes Handbuch, David Barret, Richard Silverman, O'Reilly, Köln, 2001, ISBN 3-89721-287-0.

Wenn Ihnen die mitgelieferte Dokumentation nicht ausreicht oder Sie lieber ein Buch in der Hand halten ...

[Beck2001]
Linux-Kernel-Programmierung, Michael Beck, Harald Böhme, Mirko Dziadzka, Ulrich Kunitz, et al., Addison-Wesley, München, 2001, ISBN 3-8273-1659-6.

Dieses Buch beschreibt ausführlich die im Linux-Kernel (Version 2.4) verwendeten Strukturen und Funktionen. Das Buch enthält praktisch keine Hinweise zur Systemadministration oder Anwendung und ist daher fast nur für C- und Kernel-Programmierer interessant. Es ist heute neben [Rubini2002] das Standardwerk für Kernel-Hacker und solche, die es werden wollen.

[Bolinger1995]
Applying RCS and SCCS, From Source Control to Project Control, Don Bolinger und Tan Bronson, O'Reilly & Associates Inc., Sebastopol, CA, 1995, ISBN 1-56592-117-8.

Eine ausführliche Anleitung in die Verwendung von RCS und SCCS.

[Callaghan2000]
NFS Illustrated, Brent Callaghan, Addison-Wesley, Reading, Massachusetts, 2000, ISBN 0-201-32570-5.

[Dougherty1997]
sed & awk, Dale Dougherty, O'Reilly & Associates Inc., Sebastopol, CA, 1997, ISBN 1-56592-225-5.

[DuBois1996]
Software portability with imake, Paul DuBois, O'Reilly & Associates Inc., Sebastopol, CA, 1996, ISBN 1-56592-226-3.

Die Verwendung von `imake` als Werkzeug, um plattformunabhängig Programme zu übersetzen und zu installieren. Es ist das derzeit einzige Buch zu `imake`. Die Installation und Konfiguration von `imake` wird genauso behandelt wie das Schreiben und Testen von `Imakefiles`.

[Friedl1998]
Reguläre Ausdrücke, Jeffrey E.F.Friedl, O'Reilly & Associates Inc., Sebastopol, CA, 1998, ISBN 3-930673-62-2.

Das Standardwerk zu regulären Ausdrücken.

[Garfinkel1996]
Practical UNIX & Internet Security, Simson Garfinkel und Gene Spafford, O'Reilly & Associates Inc., Sebastopol, CA, 1996, ISBN 1-56592-148-8.

[Herold1998]
Linux-Unix-Grundlagen, Helmut Herold, Addison-Wesley, München, 1998, ISBN 3-8273-1435-6.

[Hunt1998]
TCP/IP Netzwerk Administration, Craig Hunt, O'Reilly & Associates Inc., Sebastopol, CA, 1998, ISBN 3-89721-110-6.

[Kirch2001]
Linux – Wegweiser für Netzwerker, Olaf Kirch, Terry Dawson, O'Reilly & Associates Inc., Sebastopol, CA, 2001, ISBN 3-89721-135-1.

[Kofler]
Linux, Michael Kofler, Addison-Wesley, München, 2002, ISBN 3-8273-1854-8.

[Lamb1998]
Learning the vi Editor, Linda Lamb, O'Reilly & Associates Inc., Sebastopol, CA, 1998, ISBN 1-56592-426-6.

[Nemeth2001]
UNIX System Administration Handbook, Evi Nemeth, Garth Snyder, Scott Seebass, Trent M.Hein, Prentice Hall, Upper Saddle River, New Jersey, 2001, ISBN 0-13-020601-6.

Dieses Buch gilt als eines der Standardwerke für die Administration eines Unix-Systems.

[Nemeth2002]
Linux Administration Handbook, Evi Nemeth, Garth Snyder, Trent M. Hein, Prentice Hall, Upper Saddle River, New Jersey, 2002, ISBN 0-13-008466-2.

[Oram1991]
Managing Projects with make, Andrew Oram und Steve Talbott, O'Reilly & Associates Inc., Sebastopol, CA, 1991, ISBN 0-937175-90-0.

[Preston1999]
UNIX Backup & Recovery, W. Curtis Preston, O'Reilly and Associates, Sebastopol, CA1999, ISBN 1-56592-642-0.

[Robinson2001]
Effective awk Programming, Text Processing and Pattern Matching, Arnold Robbins, O'Reilly and Associates, Sebastopol, CA, 2001, ISBN 0-596-00070-7.

[Rubini2002]
Linux-Gerätetreiber, Alessandro Rubini, Jonathan Corbet, O'Reilly and Associates, Sebastopol, CA, 2002, ISBN 3-89721-138-6.

[Stern2001]
Managing NFS and NIS, Hal Stern, Mike Eisler, Ricardo Labiaga, O'Reilly & Associates Inc., Sebastopol, CA, 2001, ISBN 1-56592-510-6.

Das Standardwerk für einen Netzwerkadministrator, der NFS und NIS betreiben will. Die Konzepte, Funktionen und Programme werden ausführlich besprochen. Daneben enthält das Buch auch ausführliche Hilfen zur Fehlersuche.

[Stevens1994]
TCP/IP Illustrated, Richard W.Stevens, Addison-Wesley, Reading, Massachusetts, 1994, ISBN 0-201-63346-9.

Dreibändiges Werk, das sehr gut den Aufbau und die Implementation des TCP/IP-Protokolls erklärt. Für Netzwerkverwalter ist besonders der erste Band interessant.

[Stevens1995]
Programmierung in der UNIX-Umgebung, Richard W. Stevens, Addison-Wesley, München, 1995, ISBN 3-89319-814-8.

Das Buch ist die Standardreferenz (»Bibel«) für die Unix-Programmierung. Es wird sowohl auf BSD- als auch System-V-basierte Systeme eingegangen. Dieses Buch ist auch in englischer Sprache erhältlich.

[Torvalds2001]
Just for Fun, Wie ein Freak die Computerwelt revolutionierte, Linus Torvalds, David Diamond, Hanser Fachbuch, 2001, ISBN 344621684-7.

[Zwicky2000]
Building Internet Firewalls, Elizabeth D. Zwicky, Simon Cooper & D. Brent Chapman, O'Reilly & Associates Inc., Sebastopol, CA, 2000, ISBN 1-56592-871-7.

D Verzeichnis der wichtigsten RFCs

D.1 Überblick

Die Internet-RFCs (Request for Comments) sind die definitiven Beschreibungen der Internetprotokolle. Jeder RFC, egal ob er ein Standard oder eher als Diskussionsgrundlage oder zur Information gedacht ist, erhält eine eindeutige Nummer. Bei Änderungen an dem Dokument wird eine neue Nummer vergeben und die alte als obsolet markiert. Wenn also hier auf einen RFC verwiesen wird, dann halten Sie auch nach einer Aktualisierung davon Ausschau.

Oft wird zu einem RFC ein weiterer mit Änderungen oder zusätzlichen Informationen veröffentlicht. Auch dies ist in der Datei `rfc-index.txt` vermerkt. In der letzten Zeit werden wieder viele RFCs erstellt, so dass Sie für die letzten Neuerungen z. B. unter der URL *ftp://ftp.gwdg.de/pub/rfc* nachsehen sollten.

Die hier enthaltene Aufstellung soll Ihnen einen Überblick über einige der wichtigsten RFCs geben, ist aber bei weitem nicht vollständig. Die englischen Texte sind relativ gut lesbar, obwohl es Standarddefinitionen sind. Insbesondere sind sie eine Fundgrube für den interessierten Netzwerk- oder Systemverwalter, da die verwendeten Protokolle oder Formate im Detail erklärt werden, oft sogar mit Beispielen. Neben der Definition von Standards oder Protokollen werden oft Aufsätze zu neuen Erkenntnissen bei der Programmierung, Entwicklung oder Anwendung von Protokollen veröffentlicht, von denen auch ein Systemverwalter profitieren kann.

D.2 RFC-Verzeichnis

[rfc0768]
User Datagram Protocol, J. Postel, Aug-28-1980.

[rfc0791]
Internet Protocol, J. Postel, Sep-01-1981.

[rfc0792]
Internet Control Message Protocol, J. Postel, Sep-01-1981.

[rfc0793]
Transmission Control Protocol, J. Postel, Sep-01-1981.

[rfc0826]
Ethernet Address Resolution Protocol:Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, D. Plummer, Nov-01-1982.

[rfc0854]
Telnet Protocol specification, J. Postel, J. Reynolds, May-01-1983.

[rfc0855]
Telnet option specifications, J. Postel, J.K. Reynolds, May-01-1983.

[rfc0903]
Reverse Address Resolution Protocol, R. Finlayson, T. Mann, J.C. Mogul, M. Theimer, Jun-01-1984.

[rfc0940]
Toward an Internet standard scheme for subnetting, Gateway Algorithms and Data Structures Task Force, April 1985.

[rfc0950]
Internet standard subnetting procedure, J. Mogul, J. Postel, August 1985.

[rfc0951]
Bootstrap Protocol, W.J. Croft, J. Gilmore, September-1985.

[rfc0952]
DoD Internet host table specification, K. Harrenstien, M. Stahl, E. Feinler, October 1985.

[rfc0959]
File Transfer Protocol, J. Postel, J.K. Reynolds, Oct-01-1985.

[rfc1014]
XDR: External Data Representation standard, Sun Microsystems Inc., Jun-01-1987.

[rfc1032]
Domain administrators guide, M.K. Stahl, Nov-01-1987.

[rfc1033]
Domain administrators operations guide, M. Lottor, Nov-01-1987.

[rfc1034]
Domain names – concepts and facilities, P.V. Mockapetris, Nov-01-1987.

[rfc1035]
Domain names – implementation and specification, P.V. Mockapetris, Nov-01-1987.

[rfc1057]
RPC: Remote Procedure Call Protocol specification: Version 2, Inc. Sun Microsystems, Jun-01-1988.

[rfc1058]
Routing Information Protocol, C. Hedrick, June 1988.

[rfc1084]
BOOTP vendor information extensions, J. Reynolds, 12/01/1988.

[rfc1112]
Host extensions for IP multicasting, S. Deering, August 1989.

[rfc1122]
Requirements for Internet hosts – communication layers, R. Braden, 10/01/1989.

[rfc1123]
Requirements for Internet hosts – application and support, R. Braden, 10/01/1989.

[rfc1144]
Compressing TCP/IP headers for low-speed serial links, V. Jacobson, February 1990.

[rfc1149]
Standard for the transmission of IP datagrams on avian carriers, D. Waitzman, Apr-01-1990.

[rfc1157]
Simple Network Management Protocol (SNMP), J. Case, M. Fedor, M. Schoffstall, C. Davin, May 1990.

[rfc1178]
Choosing a name for your computer, D. Libes, August 1990.

[rfc1179]
Line printer daemon protocol, L. McLaughlin, Aug-01-1990.

[rfc1183]
New DNS RR definitions, C. Everhart, L. Mamakos, R. Ullmann, P. Mockapetris, October 1990.

[rfc1258]
BSD Rlogin, B. Kantor, Sep-01-1991.

[rfc1282]
BSD Rlogin, B. Kantor, December 1991.

[rfc1301]
Multicast Transport Protocol, S. Armstrong, A. Freier, K. Marzullo, February 1992.

[rfc1305]
Network Time Protocol (v3), D. Mills, 04/09/1992.

[rfc1311]
Introduction to the STD Notes, J. Postel, ed., March 1992.

[rfc1413]
Identification Protocol, M. St. Johns, February 1993.

[rfc1441]
Introduction to version 2 of the Internet-standard Network Management Framework, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, April 1993.

[rfc1458]
Requirements for Multicast Protocols, R. Braudes, S. Zabele, May 1993.

[rfc1459]
Internet Relay Chat Protocol, J. Oikarinen, D. Reed, May 1993.

[rfc1470]
FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices, R. Enger, J. Reynolds, 06/25/1993.

[rfc1518]
An Architecture for IP Address Allocation with CIDR, Y. Rekhter, T. Li, 09/24/1993.

[rfc1519]
Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy, V. Fuller, T. Li, J. Yu, K. Varadhan, 09/24/1993.

[rfc1534]
Interoperation Between DHCP and BOOTP, R. Droms, October 1993.

[rfc1536]
Common DNS Implementation Errors and Suggested Fixes, A. Kumar, J. Postel, C. Neuman, P. Danzig & S. Miller, October 1993.

[rfc1542]
Clarifications and Extensions for the Bootstrap Protocol, W. Wimer, October 1993.

[rfc1579]
Firewall-Friendly FTP, S. Bellovin, February 1994.

[rfc1627]
Network 10 Considered Harmful (Some Practices Shouldn't be Codified), E. Lear, E. Fair, D. Crocker, T. Kessler, July 1994.

[rfc1630]
Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web, T. Berners-Lee, 06/09/1994.

[rfc1635]
How to Use Anonymous FTP, P. Deutsch, A. Emtage, A. Marine, 05/25/1994.

[rfc1661]
The Point-to-Point Protocol (PPP), W. Simpson, ed., July 1994.

[rfc1750]
Randomness Recommendations for Security, D. Eastlake, S. Crocker, J. Schiller, December 1994.

[rfc1905]
Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), J. Case, K. McCloghrie, M. Rose, S. Waldbusser, January 1996.

[rfc1907]
Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2), J. Case, K. McCloghrie, M. Rose, S. Waldbusser, January 1996.

[rfc1912]
Common DNS Operational and Configuration Errors, D. Barr, February 1996.

[rfc1918]
Address Allocation for Private Internets, Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot & E. Lear, February 1996.

[rfc1983]
Internet Users' Glossary, G. Malkin, August 1996.

[rfc1994]
PPP Challenge Handshake Authentication Protocol (CHAP), W. Simpson, August 1996.

[rfc2007]
Catalogue of Network Training Materials, J. Foster, M. Isaacs & M. Prior, October 1996.

[rfc2015]
MIME Security with Pretty Good Privacy (PGP), M. Elkins, October 1996.

[rfc2026]
The Internet Standards Process – Revision 3, S. Bradner, October 1996.

[rfc2030]
Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI, D. Mills, October 1996.

[rfc2045]
Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, N. Freed & N. Borenstein, November 1996.

[rfc2046]
Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, N. Freed & N. Borenstein, November 1996.

[rfc2047]
MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text, K. Moore, November 1996.

[rfc2048]
Multipurpose Internet Mail Extension (MIME) Part Four: Registration Procedures, N. Freed, J. Klensin & J. Postel, November 1996.

[rfc2049]
Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples, N. Freed & N. Borenstein, November 1996.

[rfc2060]
INTERNET MESSAGE ACCESS PROTOCOL – VERSION 4rev1, M. Crispin, December 1996.

[rfc2083]
PNG (Portable Network Graphics) Specification, T. Boutell, January 1997.

[rfc2100]
The Naming of Hosts, J. Ashworth, April 1997.

[rfc2131]
Dynamic Host Configuration Protocol, R. Droms, March 1997.

[rfc2132]
DHCP Options and BOOTP Vendor Extensions, S. Alexander, R. Droms, March 1997.

[rfc2151]
A Primer On Internet and TCP/IP Tools and Utilities, G. Kessler, S. Shepard, June 1997.

[rfc2182]
Selection and Operation of Secondary DNS Servers, R. Elz, R. Bush, S. Bradner, M. Patton, July 1997.

[rfc2231]
MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations, N. Freed, K. Moore, November 1997.

[rfc2279]
UTF-8, a transformation format of ISO 10646, F. Yergeau, January 1998.

[rfc2322]
Management of IP numbers by peg-dhcp, K. van den Hout, A. Koopal, R. van Mook, Apr-01-1998.

[rfc2401]
Security Architecture for the Internet Protocol, S. Kent, R. Atkinson, November 1998.

[rfc2402]
IP Authentication Header, S. Kent, R. Atkinson, November 1998.

[rfc2403]
The Use of HMAC-MD5-96 within ESP and AH, C. Madson, R. Glenn, November 1998.

[rfc2404]
The Use of HMAC-SHA-1-96 within ESP and AH, C. Madson, R. Glenn, November 1998.

[rfc2405]
The ESP DES-CBC Cipher Algorithm With Explicit IV, C. Madson, N. Doraswamy, November 1998.

[rfc2406]
IP Encapsulating Security Payload (ESP), S. Kent, R. Atkinson, November 1998.

[rfc2407]
The Internet IP Security Domain of Interpretation for ISAKMP, D. Piper, November 1998.

[rfc2408]
Internet Security Association and Key Management Protocol (ISAKMP), D. Maughan, M. Schertler, M. Schneider, J. Turner, November 1998.

[rfc2409]
The Internet Key Exchange (IKE), D. Harkins, D. Carrel, November 1998.

[rfc2410]
The NULL Encryption Algorithm and Its Use With IPsec, R. Glenn, S. Kent, November 1998.

[rfc2411]
IP Security Document Roadmap, R. Thayer, N. Doraswamy, R. Glenn, November 1998.

[rfc2412]
The OAKLEY Key Determination Protocol, H. Orman, November 1998.

[rfc2516]
A Method for Transmitting PPP Over Ethernet (PPPoE), L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, R. Wheeler, February 1999.

[rfc2518]
HTTP Extensions for Distributed Authoring – WEBDAV, Y. Goland, E. Whitehead, A. Faizi, S. Carter, D. Jensen, February 1999.

[rfc2528]
Internet X.509 Public Key Infrastructure Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates, R. Housley, W. Polk, March 1999.

[rfc2535]
Domain Name System Security Extensions, D. Eastlake, March 1999.

[rfc2549]
IP over Avian Carriers with Quality of Service, D. Waitzman, Apr-01-1999.

[rfc2555]
30 Years of RFCs, RFC Editor, et al., Apr-07-1999.

[rfc2567]
Design Goals for an Internet Printing Protocol, F. Wright, April 1999.

[rfc2568]
Rationale for the Structure of the Model and Protocol for the Internet Printing Protocol, S. Zilles, April 1999.

[rfc2569]
Mapping between LPD and IPP Protocols, R. Herriot, Ed., T. Hastings, N. Jacobs, J. Martin, April 1999.

[rfc2578]
Structure of Management Information Version 2 (SMIv2), K. McCloghrie, D. Perkins, J. Schoenwaelder, April 1999.

[rfc2579]
Textual Conventions for SMIv2, K. McCloghrie, D. Perkins, J. Schoenwaelder, April 1999.

[rfc2580]
Conformance Statements for SMIv2, K. McCloghrie, D. Perkins, J. Schoenwaelder, April 1999.

[rfc2616]
Hypertext Transfer Protocol – HTTP/1.1, R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, June 1999.

[rfc2817]
Upgrading to TLS Within HTTP/1.1, R. Khare, S. Lawrence, May 2000.

[rfc2821]
Simple Mail Transfer Protocol, J. Klensin, Ed., April 2001.

[rfc2822]
Internet Message Format, P. Resnick, Ed., April 2001.

[rfc2828]
Internet Security Glossary, R. Shirey, May 2000.

[rfc2845]
Secret Key Transaction Authentication for DNS (TSIG), P. Vixie, O. Gudmundsson, D. Eastlake, B. Wellington, May 2000.

[rfc2854]
The 'text/html' Media Type, D. Connolly, L. Masinter, June 2000.

[rfc2910]
Internet Printing Protocol/1.1: Encoding and Transport, R. Herriot, Ed., S. Butler, P. Moore, R. Turner, J. Wenn, September 2000.

[rfc2911]
Internet Printing Protocol/1.1: Model and Semantics, T. Hastings, Ed., R. Herriot, R. deBry, S. Isaacson, P. Powell, September 2000.

[rfc2923]
TCP Problems with Path MTU Discovery, K. Lahey, September 2000.

[rfc3007]
Secure Domain Name System (DNS) Dynamic Update, B. Wellington., November 2000.

[rfc3010]
NFS version 4 Protocol, S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, D. Noveck, December 2000.

[rfc3022]
Traditional IP Network Address Translator (Traditional NAT), P. Srisuresh, K. Egevang, January 2001.

[rfc3023]
XML Media Types, M. Murata, S. St.Laurent, D. Kohn, January 2001.

[rfc3130]
Notes from the State-Of-The-Technology: DNSSEC, E. Lewis, June 2001.

[rfc3164]
The BSD Syslog Protocol, C. Lonvick, August 2001.

[rfc3227]
Guidelines for Evidence Collection and Archiving, D. Brezinski, T. Killalea, February 2002.

[rfc3232]
Assigned Numbers: RFC 1700 is Replaced by an On-line Database, J. Reynolds, Ed., January 2002.

[rfc3239]
Internet Printing Protocol (IPP): Requirements for Job, Printer, and Device Administrative Operations, C. Kugler, H. Lewis, T. Hastings, February 2002.

[rfc3253]
Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning), G. Clemm, J. Amsden, T. Ellison, C. Kaler, J. Whitehead, March 2002.

[rfc3275]
(Extensible Markup Language) XML-Signature Syntax and Processing, D. Eastlake 3rd, J. Reagle, D. Solo, March 2002.

Stichwortverzeichnis