Arnold Willemer

Wie werde ich UNIX-Guru?

Galileo Computing

Auf einen Blick

Inhalt

Vorwort

Titel

Der Titel des Buches ist ungewöhnlich. Und da die Entstehungsgeschichte einiges über die Zielsetzung verrät, sei sie hier erzählt. Galileo-Press trat an mich heran und fragte, ob ich Lust habe, ein Buch über UNIX zu schreiben. Das UNIX-Buch sollte an UNIX-Anfänger gerichtet sein. Es sollten aber auch Administration und Programmierung beschrieben werden. Ich war und bin der Meinung, dass ein Buch über UNIX ohne das Netzwerk TCP/IP unvollständig ist. Auch die grafische Oberfläche X11 darf in einem UNIX-Buch nicht unterschlagen werden. Da es ein Buch und kein Prospekt werden sollte, musste jedes dieser Themen mit einer angemessenen Tiefe behandelt werden. Da ich es selbst hasse, in einem Buch auf die Phrase zu stoßen, dass das Thema den Rahmen dieses Buches sprenge, wurde mir klar, dass das Projekt wohl sehr anspruchsvoll werden würde. Zu diesem Zeitpunkt sandte mir der Verlag auch einen Fragebogen zum Buchprojekt und wollte unter anderem einen Vorschlag für den Titel haben. Und da ich in diesem Augenblick nicht sicher war, ob ich mich bei diesem Projekt geschmeichelt fühlen oder lieber verzweifelt aufgeben sollte, schrieb ich etwas albern: »Wie werde ich UNIX-Guru?« Ich war sicher, dass der Lektor darüber einmal herzlich lachen und einen besseren Titel vorschlagen würde. Aber anstatt mich vor meinen eigenen dummen Witzen zu retten, blieb der Verlag bei dem vorgeschlagenen Titel, und so ich stehe seither unter dem Druck, dem Titel gerecht zu werden. Das Leben steckt voller Gemeinheiten.

Zweite Auflage

Seit dem Erscheinen der ersten Auflage hat sich Linux zu einer oft eingesetzten Alternative für Arbeitsplatzrechner entwickelt. Die grafischen Oberflächen KDE und GNOME lassen sich auch nicht schwerer erlernen als diejenigen aus dem Hause Microsoft. Wer auf Sicherheit und Stabilität setzt, verwendet Linux mit einer grafischen Oberfläche. So wird der Anfänger also zunächst eine Einführung in die grafischen Oberflächen brauchen. Daraus ergibt sich beinahe zwangsläufig eine neue Kapitelordnung gegenüber der ersten Auflage:

- In Kapitel 1 werden die unter UNIX geläufigen grafischen Oberflächen erläutert. Die Zielgruppe ist der Anfänger. Es geht also ausschließlich um die Bedienung und nicht so sehr um die technischen Hintergründe.
- Im Folgekapitel werden das Konzept von UNIX, einige technische Hintergründe und die Geschichte von UNIX vorgestellt.
- In Kapitel 3 findet sich nun die Bedienung von der Konsole aus. Dieses Kapitel ist nicht nur der praktische Einstieg für Leser, die mit einer

Servermaschine arbeiten sollen, sondern gibt auch Benutzern von grafischen Oberflächen einen Einblick in Möglichkeiten, die auch mit der besten grafischen Oberfläche nicht zu erreichen sind. Die Kommandozeile ist unter UNIX so mächtig, dass man sie auch bei einer gelungenen grafischen Oberfläche immer wieder verwenden wird. Die Konsole ist also nicht obsolet. »Die Shell ist schnell«, schrieb mir ein Leser[1] und er hat recht. Die besondere Leistungsfähigkeit der UNIX-Shell gibt Möglichkeiten an die Hand, die grafische Oberflächen nicht bieten.

- Kein normaler Mensch kauft sich einen Computer, um ein schönes Betriebssystem zu besitzen.[2] In diesem Kapitel werden die wichtigsten Anwendungen genannt und kurz beschrieben. Damit soll auch aufgezeigt werden, dass man mit UNIX den Aufgabenbereich abdecken kann, der von einem heutigen Computersystem erwartet wird.
- Dieses Kapitel befasst sich mit der Administration. Die klassischen Administrationstätigkeiten wie Benutzerverwaltung und Datensicherung werden hier genauso erläutert wie grundlegende Ansätze zur Analyse von Systemproblemen. Die verschiedenen Drucksysteme unter UNIX kommen zur Sprache, und der Abschnitt zum Thema CUPS wurde erweitert. Aber auch Informationen zur Behandlung von Hardwaregeräten wie CD-Brennern oder USB-Sticks finden sich hier.
- Das Kapitel über Netzwerke behandelt TCP/IP und alle wichtigen Dienste. UNIX und TCP/IP gehören zusammen. Sowohl der Bereich der lokalen Netzwerke als auch die Dienste im Internet werden hier beschrieben.
- Das Kapitel über das X Window System unterscheidet sich vom ersten Kapitel dadurch, dass hier in erster Linie Hintergründe und Konfigurationsfragen bis hin zum grafischen Einloggen über das Netz beschrieben werden.
- Das Erstellen kleiner Skripten für Alltagsaufgaben ist unter UNIX sehr einfach. Die Skriptsprache ist aber so mächtig, dass auch komplexere Abläufe damit programmiert werden können. Wer einen Einblick in die Grundlagen der Programmierung sucht, hat unter UNIX durch die Shellskripte alles Notwendige zur Hand.
- Da die Skriptsprache Perl sowohl für die Administration als auch im Internet-Bereich häufig verwendet wird, lohnt sich die Einarbeitung.

1 Mario Schröder von der Webseite http://www.linux4us.de
2 Wir nehmen Informatiker an dieser Stelle aus. Schließlich würden die meisten Informatiker zugeben, dass sie keine normalen Menschen sind.

- Unter UNIX gibt es eine große Zahl hilfreicher Werkzeuge. Im Kapitel über Programmierwerkzeuge werden Compiler, make, einige Versionsverwaltungen und andere hilfreiche Werkzeuge beschrieben.
- Wer Software für UNIX erstellen will oder auch nur einen tieferen Einblick in den Umgang mit Prozessen und Dateien bekommen möchte, sollte sich die Systemaufrufe anschauen.
- Im Glossar findet vor allem der Anfänger Begriffe erläutert, die im Text vielleicht etwas zu kurz kommen, um den Lesefluss nicht zu stören.

Natürlich wird man nicht durch die Lektüre dieses Buchs allein zum UNIX-Guru. Das wird auch sicher nicht wirklich jemand vermuten. Guru wird man, wenn man sich gern und intensiv immer wieder mit UNIX befasst. Aber ich bin doch sicher, dass dieses Buch auf dem Weg dorthin immer wieder eine Hilfe darstellt und die Richtung weist. Und so hoffe ich, dass dieses Buch zu einem praktischen Nachschlagewerk am Computer wird und Sie nie im Regen stehen lässt.

Was ist UNIX?

UNIX ist ein feines Betriebssystem. Vor allem ist es ein offenes System. Das bedeutet, dass man alles über UNIX erfahren kann, wenn man es will. Aber man muss sich etwas damit befassen. Denn Dilettantismus fliegt einem zu, Wissen nicht. Aber entgegen allen Klischees über UNIX ist es eigentlich kein wirklich kompliziertes System. Ich halte MS Windows für wesentlich komplizierter. Aber das wird Bill Gates wahrscheinlich anders sehen. UNIX ist sicher, stabil und leistungsfähig, und darum ist es sinnvoll, sich damit zu befassen.

Linux und UNIX

Inzwischen besteht kein ernsthafter Zweifel mehr daran, dass Linux ein »echtes« UNIX ist. UNIX verdankt seine derzeitige Popularität vor allem Linux, und selbst denjenigen, die sich noch mit Wehmut an die Tage der Exklusivität ihres Expertentums zurückerinnern, ist inzwischen klar, dass UNIX ohne Linux heute vielleicht nur noch eine Fußnote der EDV-Geschichte wäre. Da Linux in einem wesentlich schnelleren Tempo entwickelt wird, kann man unter Linux bereits vorab sehen, was in den anderen UNIX-Varianten später einmal Standard sein wird.

Mac OS X und UNIX

Auch Apple ist mit Mac OS X zu einem wichtigen Mitspieler in der UNIX-Arena geworden. Das bislang proprietäre Betriebssystem basiert nun auf FreeBSD. Zu dem Darwin genannten Kernel, den es als Open Source gibt, kommen eine eigene grafische Benutzeroberfläche namens Aqua sowie entsprechende Programmierschnittstellen (Carbon und Cocoa) hinzu, auf die dieses Buch allerdings nicht näher eingehen wird. Wenn Sie also etwas über den neuen Kernel von Mac OS X erfahren wollen, sind Sie

hier richtig. Norbert M. Doerner, der als Autor des Programms CDFinder auf dem Macintosh bekannt ist, hat beim Korrekturlesen immer wieder darauf hingewiesen, welche Eigenheiten das Mac OS X hat, und natürlich sind seine Anmerkungen in dieses Buch eingeflossen.

Schreibweisen

Zur besseren Lesbarkeit sind folgende Konventionen eingeführt worden: `Aufrufe` und `Kommandos` werden in nichtproportionaler Schrift gesetzt. **Dateien** und **Verzeichnisse** erscheinen in fetter Schrift. Funktionsaufrufe wie `open()` sind grundsätzlich mit Klammern dargestellt. KONSTANTEN und UMGEBUNGSVARIABLEN sind unter UNIX meist in Großbuchstaben gesetzt. Da dies bereits leicht aus dem Schriftbild heraussticht, habe ich es dabei belassen.

Dieses Symbol am Rand soll darauf hinweisen, dass hier ein Beispielszenario aufgezeigt wird, das im Text weiterverfolgt wird. Da viele Beispiele im Buch verwendet werden, sind nicht alle so auffällig gekennzeichnet, sondern nur solche, die etwas ausführlicher behandelt werden.

Mit diesem Symbol werden Hinweise gekennzeichnet, wie Sie sich das Leben etwas erleichtern.

Dieses Symbol soll auf Stolperfallen hinweisen. Hier schleichen sich entweder leicht Fehler ein, oder es geht um Dinge, die zu einem Datenverlust oder zum Verlust der Systemsicherheit führen können.

Konsolenprompt

Bei diversen Beispielen werden Bildschirmausschnitte dargestellt. Dabei zeigt der Prompt immer den Rechnernamen und ein Größerzeichen für einen normalen Anwender bzw. ein Hashzeichen (#) für den Administrator root.

Hier stehen sie untereinander:

```
gaston >
silver #
```

Dabei sind die Rechnernamen bei mir etwas bunt. Mein Linux-Arbeitsplatz heißt gaston, dann gibt es noch simba und silver (Linux), powermac (Mac OS), hpsrv (HP-UX), note (FreeBSD), sol (Solaris), sparc (SunOS), aix (AIX) und andere.

Herzlichen Dank

Wenn man ein Buch schreibt, arbeitet man nicht im luftleeren Raum. Da gibt es ein paar Leute, die mich unterstützt haben, und das war sehr nett. Meine Frau Andrea und meine Söhne haben mir einen Freiraum gewährt, in dem ich arbeiten konnte. Dankenswerterweise wurde ich mit Nahrungsmitteln versorgt, sodass ich bei Abschluss der Arbeiten sogar

noch mehr wiege als vorher. Da ich eher ein Nachtarbeiter bin, hat meine Frau unter den Büchern besonders zu leiden. Ich hänge den ganzen Abend bis in die Nacht am Computer, und sie verteidigt meinen Schlaf in den Morgenstunden, damit ich nicht den ganzen Tag aussehe wie ein ausgewrungener Waschlappen. Stephan Matteschеck hat mich als Lektor betreut, mir ständig Bücher zugeschickt, Fragen gestellt und auch manche beantwortet. Er hat mich weitgehend walten lassen und sich als Partner angeboten, wenn ich mir unsicher wurde. Daniel Lauer hat mich beim Layout und bei meinen Problemen mit den Untiefen von LaTeX unterstützt, und Iris Warkus hat mir einige Geheimnisse aus dem Bereich der Grafik verraten und ist für das gute Aussehen des Buches verantwortlich. Frau Friederike Daenecke formte aus meinen Verbrechen gegen die deutsche Sprache druck- und lesbare Sätze. Die Firma Tacoss hat mich mit Unterlagen unterstützt, mir freien Zugang zu ihrem Maschinenpark gewährt und mir eine HP-UX-Maschine zur Verfügung gestellt. Insbesondere Claus Erichsen und Leif Hansen seien hier mit Namen genannt. Sehr viel Arbeit haben sich diejenigen gemacht, die dieses Buch zur Korrektur gelesen und viele Anregungen eingebracht haben. Zu ihnen zählt Ralf Lenz mit seinen Erfahrungen, die er bei diversen Internet-Providern als Programmierer, Projektleiter und Netzwerkexperte gesammelt hat. Norbert M. Doerner hat seine Administrationserfahrungen aus dem Umfeld von Solaris und seine Kenntnisse als Entwickler auf dem Macintosh eingebracht. Er hat auch die Informationen dieses Buches darauf geprüft, ob sie für Mac OS X gelten, und Hinweise auf Unterschiede gegeben. Jörg Osarek von der Firma Oracle hat mit seinen UNIX- und Linux-Kenntnissen aus der Perspektive des professionellen EDV-Einsatzes wichtige Aspekte einbringen können und trotz engem Terminkalender noch Zeit für dieses Buch gefunden. Stephan Engelmann hat mit seiner Erfahrung als Webadministrator und Netzwerkpraktiker wichtige Fragen gestellt und gute Hinweise gegeben. Bei der zweiten Auflage hat mir Stephan Engelmann Material zur Verfügung gestellt. Mark Mitzkus hat insbesondere das Kapitel zur Administration auf Verständlichkeit abgeklopft und einige hilfreiche Hinweise gegeben.

Norgaardholz, den 24.7.2004
Arnold Willemer

1 Das Thema UNIX

UNIX ist ein benutzerfreundliches System. Es ist nur manchmal etwas eigen in der Auswahl seiner Freunde.

Dieser Satz charakterisiert UNIX vielleicht besonders. Es ist tatsächlich benutzerfreundlich und nicht so schwer zu lernen, wie von mancher Seite behauptet wird. Aber man muss schon etwas mehr tun, als mit der Maus wahllos auf Bildchen zu klicken. Man lernt UNIX nicht durch Trial and Error (also Versuch und Irrtum), sondern dadurch, dass man die angebotenen Hilfen nutzt, um zu verstehen, was passiert.

1.1 Besonderheiten von UNIX

UNIX ist immer noch nicht unmodern. Und das will bei einem Alter von über 30 Jahren etwas heißen. In den 70er Jahren war es revolutionär. Selbst in den 80er Jahren war es noch so vorbildlich, dass viele seiner Ideen in MS-DOS einflossen.[1] Auch heute noch gilt UNIX als Vorbild an Effizienz und Zuverlässigkeit. Immer noch gibt es viele UNIX-Konzepte, die erst nach und nach Eingang in andere Systeme finden.

1.1.1 Dateien und Verzeichnisse

Computer legen Ihre Daten in Paketen ab, die einen Namen bekommen und über diesen ansprechbar sind. Ein solches Paket nennt man Datei. Um die Flut der Dateien übersichtlich zu gestalten, wurden Verzeichnisse eingeführt. Auch diese besitzen einen Namen. Sie enthalten aber selbst keine Daten, sondern sind nur ein Gefäß für Dateien und weitere Verzeichnisse. Das Modell des Datenstroms, der beispielsweise aus einer Datei kommt, durch verschiedene Programme hindurchläuft und dann hinterher wieder in eine Datei umgeleitet wird, ist eines der Kennzeichen von UNIX.

1.1.2 Das offene System

Eine wichtige Eigenschaft von UNIX ist seine Offenheit. UNIX ist nach seiner Entstehung lange Zeit an der Berkeley-Universität weiterentwickelt worden. Der Sourcecode stand Studenten zur Verfügung, damit sie lernen, wie ein Betriebssystem funktioniert. Vieles wurde an UNIX demonstriert, und was UNIX nicht konnte, das brachte man ihm bei. Vor allem

1 Man kann eher bedauern, dass nicht mehr Ideen von UNIX in die Entwicklung eingeflossen sind.

aber sorgte man dafür, dass es keine Geheimnisse gab. Jeder Prozess ist sichtbar, und man kann ihm auf die Finger klopfen. Alle Konfigurationsdateien sind reine Textdateien oder leicht in solche zu verwandeln. Solche Konfigurationen kann man leicht sichern, ausdrucken oder durchsuchen. Eine solche Umgebung ist aber auch sicher, weil einfach nichts versteckt werden kann.

Komplex, aber nicht kompliziert

UNIX gilt als kompliziert. Dieses Klischee wird von Leuten verbreitet, die glauben, MS Windows sei simpel. Ein modernes Betriebssystem mit Multitasking, Netzwerkanschluss und grafischer Oberfläche ist komplex. Wer den Zugriff auf alle Details erhält, wie das bei UNIX der Fall ist, der mag zu Anfang über die vielen Informationen erschrecken. Aber man muss nicht alles wissen, um mit UNIX produktiv umgehen zu können. Dennoch ist es gut zu wissen, dass man alles wissen könnte.

1.1.3 Mehrbenutzersystem

UNIX ist immer ein Mehrbenutzersystem gewesen. Daraus erwachsen zwei Ziele: Fairness und Sicherheit.

Fairness

Fairness bedeutet, dass jeder Anwender gleich behandelt wird. Dazu muss die Rechenzeit gerecht verteilt werden, und der Umschaltprozess muss effizient arbeiten.

Benutzerverwaltung

Naheliegenderweise muss ein Mehrbenutzersystem mehrere Benutzer verwalten können. Das heißt, jeder Benutzer kann sich anmelden, hat sein Passwort, seine Umgebung und seine Ressourcen. Unter UNIX heißt das: Jede Datei hat einen Besitzer, und jeder Prozess[2] wird einem Eigentümer zugeordnet.

Datenschutz

Eine der wichtigsten Aufgaben eines Mehrbenutzersystems ist der Schutz der Daten eines Benutzers vor jedem anderen. Jeder Benutzer muss den Eindruck gewinnen, dass er der einzige Benutzer der Maschine ist. Andererseits muss die Zusammenarbeit mehrerer Benutzer problemlos möglich sein. Diese Eigenheit wurde nicht nachträglich aufgeflanscht, sondern war und ist bei jeder Weiterentwicklung des Systems bereits im Hinterkopf der Programmierer vorhanden.

Systemsicherheit

Ein Mehrbenutzersystem muss auch Angriffen boshafter Anwender standhalten. Ein übel gesinnter Benutzer darf die Stabilität der Maschine nicht verletzen können. Das bedeutet in der Konsequenz, dass die Programme, die von mehreren Anwendern eingesetzt werden, nur durch den

2 Ein Prozess ist ein Programm, das gerade abläuft.

Administrator verändert werden dürfen. Dieser Schutz hilft heute bei der Absicherung des Systems gegen schädliche Software wie Viren, Würmer oder Trojaner.

Abgrenzung

Ein Mehrbenutzersystem besitzt Mechanismen, damit der Systembenutzer nicht unbedingt auch Administratorrechte haben muss. Die Aufgaben werden klar abgegrenzt, und die Rechte können so vergeben werden, dass nur bestimmte Aufgaben abgegeben werden können.

Verschlüsselung

UNIX bietet in seiner Standardbibliothek die Funktion `crypt()` an, mit der jeder Programmierer Passwörter oder andere schützenswerte Dinge leicht verschlüsseln kann, ohne dafür extra einen eigenen Algorithmus zu erfinden, der in der Praxis dann vermutlich auch eher unzuverlässig ist.[3]

1.1.4 Dateirechte

In einer Multiuser-Umgebung ist es wichtig, dass die Daten des einzelnen Anwenders vor der Neugier der übrigen Anwender geschützt werden können. Dazu ist es erforderlich, dass Dateien einen Besitzer haben. Ferner gibt es Zugriffsrechte auf Dateien. UNIX unterscheidet zwischen Schreib-, Lese- und Ausführungsrechten. Ein Programmierer kann also ein Programm erstellen, das von aller Welt ausgeführt wird, aber nicht veränderbar und vielleicht nicht einmal lesbar ist.

Zielgruppen

Um einem Team von Programmierern die Arbeit zu erleichtern, kennt UNIX das Konzept einer Gruppe. Damit gibt es drei Kategorien von Benutzern: den Eigentümer, die Gruppe und die Welt. Für jede dieser drei Gruppen kann jedes der drei Rechte separat gesetzt werden. Ein Programmierer kann ein Programm entwickeln, das er selbst lesen, schreiben und ausführen kann. Die Gruppe aller Programmierer soll das Programm vielleicht lesen und ausführen können. Aber die Kollegen sollen bitte nicht an seinem Code herumbasteln. Der Rest der Anwender, unter UNIX gern »die Welt« genannt, soll das Programm ausführen und staunen, aber nicht hinter die Kulissen schauen dürfen.

Verzeichnisse

Die gleichen Rechte und Gruppen können nicht nur Dateien, sondern auch Verzeichnissen zugeordnet werden. Dabei bedeutet das Leserecht, dass Sie den Inhalt des Verzeichnisses auslesen dürfen. Haben Sie kein Schreibrecht auf das Verzeichnis, so dürfen Sie eine Datei weder löschen noch anlegen oder umbenennen. Als Eselsbrücke hilft es, sich ein Verzeichnis als besondere Datei vorzustellen, in der eine Liste der Dateina-

3 vgl. dazu ab Seite 804.

men steht.[4] Wenn die Liste der Dateien schreibgeschützt ist, können Sie keine Dateien hinzufügen oder entfernen. Und auch das Verändern der Dateinamen ist dann logischerweise verboten. Das Ausführungsrecht für ein Verzeichnis erlaubt es Ihnen, in dieses Verzeichnis zu wechseln. Hier greift die Eselsbrücke leider nicht mehr so gut.

1.1.5 Der Administrator

Auf jeder UNIX-Maschine gibt es einen absoluten Herrscher. Das ist der Superuser root. Er darf sich über alle Dateirechte hinwegsetzen, legt Benutzer an oder sperrt sie aus.

Der User root wird niemals zur normalen Arbeit eingesetzt. Auch der Administrator hat neben dem root-Zugang normalerweise ein gewöhnliches Benutzerkennwort, das er für seine tägliche Arbeit benutzt. Er wird sich nur dann als root anmelden, wenn er eine Aufgabe zu erledigen hat, die ohne diese Rechte nicht zu lösen ist. Wer also als root Texte erstellt, Programme schreibt oder sonstige Anwendungstätigkeit erledigt, gibt sich sofort als UNIX-Anfänger zu erkennen.

Sicherheit vor Viren

Durch diese scharfe Trennung wird erreicht, dass UNIX-Systeme schwerer angreifbar sind. Jeglicher Schreibzugriff auf Betriebssystembestandteile oder Anwendungsprogramme bleibt dem Administrator als root vorbehalten. Alle normalen Nutzer haben nur das Lese- oder Ausführungsrecht. Hierin besteht auch der Grund, warum Linux als echtes UNIX wesentlich besser gegen Viren geschützt ist als beispielsweise Windows. Dabei werden Viren – außer durch die derzeit geringere Verbreitung von UNIX-Arbeitsplätzen – durch folgende Mechanismen behindert:

- Die klassischen Viren sorgen dafür, dass sie beim Systemstart mitgestartet werden. Ansonsten würde das Virus beim nächsten Reboot der Maschine nicht mehr aktiv sein. Auf alle Dateien, die den Boot-Mechanismus betreffen, kann nur root zugreifen. Wurde das Virenprogramm durch einen Anwender eingeschleppt, so kann der Start nur duch die Start-Skripten der Login-Shell erreicht werden. Hier ist ein Virus allerdings sehr schnell entdeckt.
- Viren wollen sich vervielfältigen. Dazu brauchen sie neue Opfer. Einige Viren lesen die Adressbücher der E-Mail-Programme aus und können sich so im Freundeskreis des Opfers verbreiten. Dies ist theoretisch auch im UNIX-Umfeld denkbar.

4 In den ersten UNIX-Varianten wurde ein Verzeichnis tatsächlich etwa so realisiert.

- Viel effizienter als das Auslesen des Adressbuches ist es für das Virus, den Netzverkehr abzuhorchen und darauf zu warten, dass eine E-Mail-Adresse über die Leitung geht. An diese Komponenten kommt ein normales Anwenderprogramm allerdings unter UNIX nicht heran.
- Die klassischen Viren wurden durch infizierte Anwendungsprogramme übertragen. Auch dieser Infektionsweg ist unter UNIX nicht praktikabel, da Anwenderprogramme einmal vom Administrator installiert werden und danach von keinem Anwender mehr verändert werden können.
- Die Makroviren der Textverarbeitung MS-Word könnten in vergleichbarer Form natürlich auch für andere programmierbare Anwendersoftware entwickelt werden. Inzwischen sind Makroviren allerdings etwas aus der Mode gekommen.
- Die meisten als Viren bezeichneten Schädlinge sind eigentlich Würmer. Damit bezeichnet man Software, die nur den Fortpflanzungscharakter hat und keine Schadensroutine besitzt. Ein klassischer Virus mit Schadensroutine wird zu einem bestimmten Zeitpunkt wie eine Zeitbombe Schaden anrichten. Auch hier gilt unter UNIX, dass der Schädling nur so viel Schaden anrichten kann, wie der einschleppende Anwender Rechte hat. Damit könnte er keinesfalls eine Platte formatieren, sondern schlimmstenfalls die persönlichen Daten des infizierten Anwenders zerstören. Das System selbst ist keinesfalls gefährdet.

1.1.6 Verzeichnisbaum und Speichermedien

In einem Computersystem besitzen Sie immer eine oder mehrere Festplatten und typischerweise mehrere Wechselmedien wie Disketten, CDs und die USB-Memory-Sticks. Einige Betriebssysteme bezeichnen die Laufwerke mit Buchstaben oder Namen, und die Daten werden durch Angabe ihrer physischen Position gefunden. Diese Methode wird schnell zum Fluch, wenn eine neue Platte gekauft wird und die Laufwerksbuchstaben dadurch gehörig durcheinander kommen.

UNIX ordnet dagegen alle Dateien in einem einzigen Verzeichnisbaum an. Dabei haben bestimmte Verzeichnisse bestimmte Aufgaben. Der Baum aus Abbildung 1.1 zeigt eine für UNIX typische Verzeichnisstruktur.

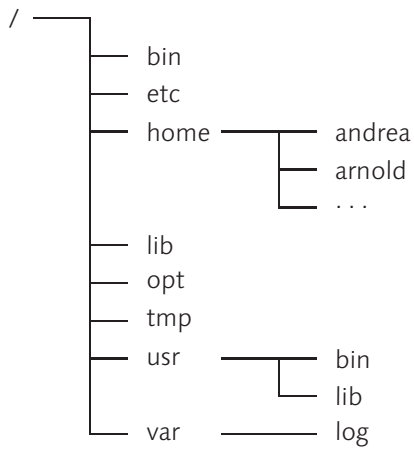

Abbildung 1.1 Verzeichnisbaum

Auf welchen Medien sich die Verzeichnisse befinden, ist frei konfigurierbar. So könnten beispielsweise die Arbeitsverzeichnisse der Benutzer auf Wechselplatten oder gar über das Netzwerk auf einen anderen Computer gelegt werden. Andererseits kann ein Medium nur benutzt werden, wenn es in den Verzeichnisbaum eingebunden wird. Dazu stehen die Befehle `mount` (siehe Seite 296) und die Konfigurationstabelle **fstab** (siehe Seite 297) zur Verfügung.

mount Mit dem Befehl `mount` wird ein Medium in den Verzeichnisbaum eingebunden. Das war ursprünglich dem Administrator vorbehalten. Als Parameter wird angegeben, welches Medium an welcher Stelle im Verzeichnisbaum eingebunden wird. Klassischerweise wurde zum kurzfristigen Einbinden von Medien das Verzeichnis **/mnt** verwendet. Mit dem Befehl `umount` wird eine solche Einbindung wieder gelöst. Das Aushängen eines Mediums ist aber nur dann möglich, wenn kein Prozess mehr damit arbeitet. Unter den grafischen Oberflächen wird dies durch Menübefehle an die Mediensymbole erreicht. Beim Mac OS X und bei Solaris werden eingelegte CDs automatisch eingebunden.

/etc/fstab Wie der Verzeichnisbaum auf die Platten verteilt wird, muss nicht beim Booten jedes Mal neu angegeben werden. Stattdessen kann der Administrator in einer Textdatei (in den meisten Fällen heißt diese **/etc/fstab**) festlegen, welche Medien an welcher Stelle im Verzeichnisbaum eingebunden werden sollen. Sie können also jede Zeile dieser Datei als Parameterliste des `mount`-Befehls verstehen, der beim Booten ausgeführt wird. Diese einfache Tabelle macht es aber leicht möglich, eine neue Platte einzubinden, wenn der Platz an einer bestimmten Stelle im Verzeichnisbaum

knapp wird. Die Anwender und die Programme merken davon nichts, außer natürlich, dass nun genügend Platz ist.

1.2 Die UNIX-Varianten

Es gibt diverse UNIX-Varianten und -Lizenzen. Um zu verstehen, wie es zu dieser Vielfalt kam und was die Besonderheiten der einzelnen Systeme sind, ist es hilfreich, die Geschichte von UNIX zu betrachten. Tatsächlich können Sie im Internet eine Vielzahl von Seiten finden, in denen die Geschichte von UNIX erzählt wird. Hier eine Auswahl von Seiten:

- **Twenty Years of Berkeley Unix**
  http://www.oreilly.com/catalog/opensources/book/kirkmck.html
- **The Creation of the UNIX* Operating System**
  http://www.bell-labs.com/history/unix/
- **Unix at 25**
  http://www.byte.com/art/9410/sec8/art3.htm
- **Dennis Ritchie: The Evolution of the Unix Timesharing System**
  http://cm.bell-labs.com/cm/cs/who/dmr/hist.html
- **Dennis Ritchie's homepage**
  http://www.cs.bell-labs.com/who/dmr/
- **A short history of UNIX**
  http://www-cs.canisius.edu/UNIX/history.html

1.2.1 AT&T

MULTICS

Die Bell Laboratories hatten eine Zusammenarbeit mit MIT und General Electrics an einem Betriebssystem namens MULTICS (Multiplexed Information and Computing System) erfolglos beendet. MULTICS sollte ein Betriebssystem der Zukunft werden. Viele Ideen wurden eingebracht, und bald erwies sich die Aufgabe als zu komplex. Als Plattform war das IBM 7094 Data Processing System vorgesehen. MULTICS enthielt aber viele gute Ideen:[5]

- ein hierarchisches Dateisystem
- virtuellen Speicher
- die Pipe
- die Shell

5 http://www-cs.canisius.edu/UNIX/history.html,
Derek Morr: A History of Unix.

- Hintergrundprozesse
- Verwendung einer Hochsprache (PL/I)[6]

Schlankheitskur

Zwei der Entwickler, die von dem Projekt MULTICS abgezogen wurden, Dennis Ritchie und Ken Thompson, entwickelten in den Bell Laboratories bei AT&T auf einer PDP-7 um 1969 eine abgespeckte Version, die sie zunächst UNICS nannten. Später verwendeten sie eine PDP-11. Im Jahre 1973 wurde der Code in der Sprache C vollkommen neu geschrieben. Dadurch war UNIX portierbar geworden.

1.2.2 UNIX an der Uni

Die kommerziellen Möglichkeiten wurden zunächst völlig unterschätzt, sodass UNIX und sein Quellcode recht freizügig verteilt wurden. So kam UNIX etwa 1974 an die Universitäten Berkeley und Columbia. Generationen von Studenten lernten anhand von UNIX, wie ein Betriebssystem funktioniert. Hier wurden Erweiterungen bzw. Korrekturen vorgenommen. Berkeley gab mit BSD eigene Versionen von UNIX heraus.

In Berkeley wurde der Editor vi geschrieben. Die Terminalansteuerung mit termcap und der virtuelle Speicher wurde von der Universität Berkeley entwickelt. 1979 entstand die legendäre Version 7, die unter anderem die Utilities awk, cpio, expr, find, lint, make, sh, sed, tail, tar und touch enthielt. Die Programmierschnittstellen ioctl, malloc, stdio und string waren ebenfalls implementiert.

Netzwerk und TCP/IP

Das amerikanische Verteidigungsministerium gab 1980 im Zuge der Entwicklung des Internets der Universität Berkeley den Auftrag, Netzwerkschichten in UNIX zu implementieren. 1983 gehörte TCP/IP zum Umfang der Version 4.2 von BSD.

1.2.3 UNIX wird kommerziell

Seit 1979 vertrieb AT&T UNIX kommerziell und sicherte sich 1986 den Markennamen UNIX. Gleichzeitig wurden die Lizenzgebühren für UNIX immer mehr erhöht. Dadurch wurden auch die Kosten für das BSD-UNIX immens, da man zur Verwendung des BSD-UNIX auch eine Lizenz von AT&T brauchte.

Sun

1982 verließ Bill Joy Berkeley und gründete die Firma Sun, die Hardware zum Betrieb von UNIX herstellte. Sun gehörte aber auch zu den Firmen, die am intensivsten die Weiterentwicklung von UNIX betrieben. 1983

6 http://www-cs.canisius.edu/UNIX/history.html

wurde die eigene UNIX-Variante SunOS 1.0 veröffentlicht. 1984 erschien NFS (Network Filesystem).

SGI

Im Jahre 1982 entwickelte die Firma SGI (Silicon Graphics Inc.), die vor allem im Multimedia-Bereich eine führende Rolle übernahm, seine UNIX-Variante IRIX.

System V

Etwa 1984 entwickelte AT&T System V.[7] AT&T und Sun entwickelten 1988 System V Rel 4. Diese Version sollte vor allem die vielen separaten Entwicklungen zusammenfassen und einen neuen Standard bieten. Wegen der Notwendigkeit, die Portabilität zu wahren, die zu den ursprünglichen Stärken von UNIX gehörte, wurden auch Besonderheiten von BSD in das System V aufgenommen.

HP-UX

1986 brachte die Firma Hewlett Packard ihre UNIX-Version HP-UX auf den Markt. HP-UX basiert auf System V.

AIX

Auch auf der Basis von System V brachte etwa 1990 auch IBM seine UNIX-Version AIX auf den Markt. Für AIX stellt IBM die RISC-Maschinen RS/6000 her. Die bereits 1988 gegründete Open Software Foundation OSF[8] hatte sich zur Aufgabe gestellt, gemeinsam die Weiterentwicklung von UNIX zu betreiben, und wählte AIX als Ausgangssystem aus.

POSIX von IEEE

Während seiner Entwicklungsgeschichte drohte immer wieder die Zersplitterung von UNIX in diverse Dialekte. Es gab mehrere Bestrebungen, verbindliche Normen zu schaffen. Das IEEE (Institute for Electrical und Electronic Engineers) definierte mit POSIX (Portable Operating System Interface) eine Familie von Standards für die UNIX-Schnittstellen.

X/Open

Von Seiten der großen Computerhersteller wurde die X/Open gegründet, die einen Industriestandard für offene Systeme schaffen sollte. Das wichtigste Ergebnis war der XPG (X/Open Portability Guide).

1.2.4 Die Rache der Enterbten

AT&T hatte natürlich das Recht, für eigene Entwicklungen Lizenzen zu fordern. Allerdings war UNIX gerade durch die Entwicklungen an der Universität Berkeley ein Erfolg geworden. Durch die immer restriktiveren Lizenzbedingungen wurde es immer schwieriger, UNIX im Lehrbetrieb einzusetzen. Und so mussten die Universitäten unter dem Erfolg von UNIX leiden, den sie selbst herbeigeführt hatten.

7 Das V steht für eine römische Fünf und wird üblicherweise englisch five ausgesprochen.

8 Nicht zu verwechseln mit der Free Software Foundation FSF

GNU Als Reaktion auf diesen Zustand gründete 1984 Richard M. Stallmann die Free Software Foundation. Software, die von ihren Autoren ohne Bezahlung erstellt wurde, sollte frei bleiben. Das Projekt GNU wurde geboren. GNU war die Abkürzung für »GNU is Not Unix«. Damit war nicht gemeint, dass man sich von UNIX als Betriebssystem distanzierte. Vielmehr sollte ausgedrückt werden, dass es dieser Software nicht so ergehen sollte wie UNIX und dass es auch nichts mit dem UNIX zu tun hatte, an dem AT&T irgendwelche Rechte geltend machen konnte. Eine Kommerzialisierung auf Kosten der Autoren sollte nicht möglich sein. Das Fernziel war ein eigenes Betriebssystem. Zunächst wurden aber die Utilities entwickelt. Das wichtigste Produkt des GNU-Projekts war sicherlich der Compiler, der kostenlos auf beinahe allen Plattformen zur Verfügung steht. Nun hatte man einen eigenen Compiler, mit dem man plattformübergreifend entwickeln konnte. Dazu kam, dass immer mehr Hersteller ihre Compiler als »Entwicklungspaket« bezeichneten und für teures Geld separat verkauften.

XINU Immer noch fehlte den Lehrstühlen für Betriebssysteme an den Hochschulen die Möglichkeit, Studenten »am lebenden Objekt« auszubilden. Der UNIX-Quellcode durfte nicht mehr frei veröffentlicht werden. Lizenzen für interessierte Studenten waren unerschwinglich. Es gab verschiedene Ansätze, um in dieser Situation Abhilfe zu schaffen. An den Universitäten entstand zunächst XINU[9] von Douglas Comer als ein Systemkern ohne Anbindung an irgendwelche Peripherie. Erst spät entstand eine Portierung auf einen Personal-Computer.

MINIX Zu dieser Zeit hatte Andrew Tanenbaum bereits MINIX entwickelt. Wie schon bei XINU entstand das Betriebssystem im Zusammenhang mit einem Buch über Betriebssysteme.[10] Tanenbaums System wurde vollständig für den IBM-PC entwickelt und enthielt alle wichtigen Komponenten eines Betriebssystems und auch die Treiber. So wurde sein Buch zu einem der Standardwerke zum Thema Betriebssysteme. Neu war auch das Konzept: MINIX wurde aus mehreren Modulen zusammengesetzt, die mittels Message Passing miteinander kommunizierten. Dieser Ansatz war elegant und ermöglichte den späteren Umstieg auf ein verteiltes Betriebssystem, war aber nicht auf Performance ausgelegt. Immerhin konnten Studenten dieses System für wenig Geld erwerben und hatten nicht nur ein vollständiges UNIX, sondern auch die Sourcen zur Verfügung, um dem System

9 Comer, Douglas: Operating System Design – The XINU Approach. Prentice Hall International Editions, 1984.

10 Tanenbaum, Andrew S.: Operating Systems – Design and Implementation. Prentice Hall, 1987.

auf die Finger zu schauen. Das System war unter Studenten recht populär, und es existierten sogar Portierungen auf Apple Macintosh, Atari ST und Commodore Amiga.

Freie BSD

Die Universität Berkeley blieb nicht untätig. Sie entfernte allen Code, der in ihrer UNIX-Version noch Rechte von AT&T enthielt, und entwickelte diese Funktionalitäten völlig neu. Da nun alle Teile dieses System selbst geschrieben waren, wurde es als freie Software vertrieben. 1991 folgte ein längerer Rechtsstreit zwischen der Universität und AT&T und schließlich war FreeBSD geboren. FreeBSD enthält den Original-Quellcode des TCP/IP, wie er auch in den anderen UNIX-Ablegern läuft.

Linux

In dieser Situation erstellte Linus Torvalds einen Kernel in UNIX-Machart. Er nannte ihn Linux und stellte ihn in der MINIX-Newsgroup vor. Andrew S. Tanenbaum reagierte mit einer Mail, deren Betreff »Linux is obsolete« heute oft zitiert wird. Sie wird gern dahingehend missverstanden, dass Tanenbaum das Zukunftspotenzial von Linux verkannt hätte. Das Wort »obsolete« bedeutet aber »althergebracht« und bezieht sich auf die Struktur des Systemkerns. Im Gegensatz zu MINIX, das ein modulares Betriebssystem darstellt und dadurch die Ansätze für ein verteiltes System aufzeigt, erreicht Linux seine wesentlich bessere Performance durch seine Rückkehr zu einem monolithischen Betriebssystemkern. Der schnelle Kern von Linux mag zwar nicht elegant gewesen sein, aber der Gedanke, ein UNIX-ähnliches System zu besitzen, das die Geschwindigkeit des damals neuen 80386-Prozessors voll ausnutzte, hatte seinen Charme.

Linux, GNU und BSD

Durch das Hinzufügen der GNU-Tools und des MINIX-Dateisystems konnte man aus Linux schnell ein lauffähiges Grundsystem bauen. Von FreeBSD wurden die Netzwerkbibliotheken verwendet. Das grafische X Window System wurde, da es frei verfügbar war, auch bald eingebunden. Sun stellte seine grafische Oberfläche SunView unter dem Namen OpenView frei, und so hatte Linux bereits früh eine durchaus professionelle grafische Oberfläche zur Verfügung. Anfangs wurde Linux noch als Betriebssystemspielzeug belächelt. Dann entwickelte sich durch die weltweite Beteiligung in unglaublicher Geschwindigkeit ein System, das seine Spuren auch in anderen UNIX-Systemen hinterließ. Wollte früher Linux wie UNIX sein, versuchen heute die UNIX-Systeme, die neuen Möglichkeiten von Linux zu integrieren. Heute gibt es wohl niemanden mehr, der daran zweifelt, dass Linux ein »richtiges« UNIX ist.

UNIX auf dem Desktop

Nachdem Linux im Bereich der Server bewiesen hatte, dass es ein sicheres und stabiles Betriebssystem ist, wollten immer mehr Anwender auch ein stabiles und sicheres Betriebssystem als Arbeitsplatz. Es wurden die bei-

den Desktops KDE und GNOME entwickelt, die in ihrer Benutzerführung dem Marktführer MS-Windows in nichts nachstanden.

KDE Die erste Version von KDE wurde in einer unglaublichen Geschwindigkeit auf die Beine gestellt. KDE wurde auf Basis der Qt-Bibliothek der Firma Troll Tech entwickelt. Der entstandene Desktop erfüllte die gestellten Aufgaben. Nichts war mehr von der leichten Muffigkeit des alten X Window Systems zu spüren. Schnell wuchs die Zahl der Dienstprogramme und der Anwender. Die Firma SUSE, die in Deutschland Marktführer unter den Linux-Distributoren ist, setzt KDE als Standardoberfläche ein.

GNOME Die Verwendung einer kommerziellen Bibliothek führte zu Aufruhr in der Linux-Gemeinde. Während bei allen anderen Betriebssystemen die Verwendung der Bibliotheken kostenlos war, sollte dies ausgerechnet beim freien Linux anders sein? Man entschied sich, auf der Basis der Bibliotheken des Grafikprogramms GIMP, das vollständig der GNU-Lizenz unterliegt, einen eigenen Desktop zu schaffen. GNOME orientierte sich auch weniger an MS Windows, sondern mehr am Macintosh. Darüber hinaus basiert GNOME auf CORBA, einer objektorientierten Komponentenarchitektur. GNOME wird von Red Hat, dem wichtigsten amerikanischen Linux-Distributor gefördert. Außerdem hat sich Sun dafür entschieden, GNOME als Standard-Desktop für Solaris einzusetzen.

KDE oder GNOME? Die Firma Troll Tech hat der KDE-Gemeinde einige Zugeständnisse gemacht. So ist die Bibliothek unter ähnlichen Rechten zu verwenden wie die GNU, solange die entstehende Software unter GNU-Lizenz weitergegeben wird. Zahlen müssen nun nur Unternehmen, die ihre Software verkaufen wollen. Da die Entwicklung für Linux immer noch ein Risiko ist, werden diese Unternehmen aber vermutlich eher für GNOME entwickeln als für KDE. Für Shareware-Autoren wird die Entscheidung ebenfalls eindeutig sein. Die wichtigste Errungenschaft ist, dass die Anwendungen für den einen Desktop auch auf dem anderen laufen.

1.2.5 Mac OS X

Single-Tasking Aus einer ganz anderen Richtung kommt die Entwicklung bei Apple. Mitte der 80er Jahre stellte Apple den Macintosh vor. Im Gegensatz zum IBM-PC besaß der Mac keine Textkonsole mehr, sondern arbeitete auf einer grafischen Oberfläche mit einer Maus. Der neue Computer sollte auch für Anfänger leicht zu benutzen sein. Alles war leicht zu verstehen, und so passte auch das Single-Tasking dazu. Der Anwender konnte nur ein Programm gleichzeitig starten. Wollte er ein anderes Programm benutzen, musste er das bisherige zunächst beenden.

NextStep

Wenige Jahre später trennte sich Apple von dem Firmenmitbegründer Steve Jobs, der das Macintosh-Projekt veranlasst hatte. Jobs gründete eine neue Computerfirma namens NeXT und entwickelte dort einen Computer der Superlative, den er 1988 vorstellte. Es wurde leistungsfähigste Hardware verwendet. Aber auch die Software war vom Feinsten. So bildete Mach den Kern. Mach war ein Mikrokernel, der durch Module erweitert wurde, die durch Versenden von Nachrichten miteinander kommunizierten. Erweitert wurde der Kern um das BSD-UNIX der Version 4.3. Darauf lief eine objektorientierte Systembibliothek zum Erstellen von Anwenderprogrammen. Die Grafik wurde sowohl am Bildschirm als auch auf dem Drucker komplett in PostScript realisiert. Dieser Computer war sehr beeindruckend, aber leider auch sehr teuer, und der kommerzielle Erfolg blieb aus.[11]

Comeback

Inzwischen erwartete der Markt, dass ein Computer multitasking-fähig war. So wurde auch beim Apple Macintosh das zugrunde liegende Betriebssystem immer mehr erweitert. Obwohl die Benutzeroberfläche noch hoch gelobt wurde, wurden die Mängel des zugrunde liegenden Systems immer deutlicher. Aber auch die Oberfläche wirkte inzwischen ein wenig altbacken. Schließlich wurde 1996 die Firma NeXT inklusive Steve Jobs von Apple übernommen, und die UNIX-Basis von NextStep wurde zum neuen Basissystem des Mac OS. Apple veröffentlichte die Sourcen zu diesem Unterbau unter der Apple Public Licence und nannte ihn Darwin. Auch die grafische Oberfläche wurde um die Ideen von NextStep bereichert und grafisch aktualisiert. Da die nächste Mac OS-Version die Nummer 10 sein würde und um auf den UNIX-Kern anzuspielen, nannte man diese Version Mac OS X.

Mac und UNIX

Damit ist der Macintosh eine UNIX-Maschine. Allerdings hat der Macintosh ein paar Besonderheiten. Statt des in UNIX-Kreisen üblichen X Window System verwendet Apple eine eigene grafische Oberfläche namens Aqua, die nicht auf X11 beruht. Der Macintosh ist von seinem Ursprung her kein Server, sondern ein Arbeitsplatzrechner. Weil die anerkannt einfache Bedienung der Wettbewerbsvorteil ist, der den Macintosh aus der Masse der MS-Windows-PCs heraushebt, legte Apple einen größeren Wert darauf, die grafische Oberfläche geschmeidig zu halten, als darauf, alle UNIX-Dogmen und -Standards einzuhalten.

11 vgl. Bresink, Marcel: Mac OS X – Das Jaguarbuch. 2003, mitp-Verlag, Bonn. S. 157ff.

1.2.6 UNIX wird verkauft

Novell kaufte UNIX

Im Jahre 1993 verkaufte AT&T seine Unix Systems Laboratories an Novell. Dadurch kamen die Rechte an UNIX in die Hand von Novell. Novell vertrieb zu diesem Zeitpunkt ein eigenes PC-UNIX namens UnixWare. Die grundlegende Idee war, Netware, das Hauptprodukt von Novell, auf die Basis von UNIX und TCP/IP zu stellen, weil der bisherige Kern nicht mehr zukunftsfähig erschien. Aber es kam anders. Microsoft setzte Novell mit seinem Marketing für die NT-Server so unter Druck, dass Novell die UNIX-Pläne aufgab.[12]

Novell und SCO

1995 wurden die Eigentums- und Nutzungsrechte an UNIX zusammen mit UnixWare an SCO (Santa Cruz Operation Inc.) verkauft. Interessanterweise sind in dem Vertrag wohl die Urheberrechte und Patente explizit ausgenommen.[13] Die Firma SCO war zu diesem Zeitpunkt der Hersteller des verbreitetsten UNIX für PCs. Der Markenname UNIX wurde dagegen an die Open Group weitergegeben. 2001 wurde SCO von Caldera aufgekauft. Caldera war zu diesem Zeitpunkt ein Linux-Distributor. Die Distribution von Caldera zeichnete sich dadurch aus, dass sie einige kommerzielle Tools enthielt, und peilte den professionellen Markt an. Zunächst ging SCO in Caldera auf. Später benannte Caldera sich in SCO um und beteiligte sich an einem Standardisierungprojekt für Linux. Seit 2003 versuchte SCO seine UNIX-Rechte zu Geld zu machen, indem die Firma behauptete, Linux enthalte Code-Passagen, die aus dem Original-UNIX stammen. Die Firma forderte Linux-Anwender auf, Lizenzrechte zu kaufen, und verklagte parallel IBM und andere Firmen wegen Urheberrechtsverletzungen. Dennoch entstand der Eindruck, dass SCO es nicht eilig hatte, die Vorwürfe der Code-Übernahme zu belegen.

12 vgl. Joe Firmage: Insider-Story: Warum Novell die Unix-Rechte an SCO verkauft hat. ZD-Net. 17. November 2003.

13 vgl. Vertrag sichert SCO die Unix-Rechte – Novell aber auch. Computerwoche, 05.06.2003

Teil 1
Anwendung

haben sie meistens zwei Bindestriche[4]. Schließlich haben die meisten Befehle Argumente. Dies sind die Objekte, auf denen die Befehle ausgeführt werden sollen, meist Dateien oder Verzeichnisse. Je nach Art des Befehls kann es gar keine oder beliebig viele Argumente geben.

Fehlermeldungen

UNIX-Programme sind normalerweise schweigsam. Wenn das Programm keine besonderen Ausgaben hat und kein Fehler aufgetreten ist, erscheint oft keine Meldung, sondern nur der Prompt.[5] Bei einem Fehler gibt es allerdings eine Fehlermeldung. Diese kommt vom aufgerufenen Programm, oder die Shell meldet sich, wenn sie den Befehl bzw. das Programm nicht kennt oder die Struktur des Befehls ihr nicht behagt.

So könnte nach Eingabe von `abcdefg` als Kommando eine Meldung wie die folgende erscheinen:

```
sh: abcdefg: not found.
```

Da UNIX kein Programm namens `abcdefg` finden konnte, meldet der Kommandointerpreter `sh` den Fehler. `sh` ist die Standard-Shell. Sie kann bei Ihnen vielleicht anders heißen. Kommt ein korrekt aufgerufenes Programm mit der Eingabe nicht zurecht, meldet es sich. Zum Beispiel erscheint nach der Eingabe `grep o p` folgende Meldung:

```
> grep o p
sh: grep: 0652-033 cannot open p.
>
```

Vor der Fehlermeldung sieht man sehr schön die Aufrufhierarchie. `sh` rief `grep` auf, und `grep` meldet den Fehler. Das Programm `grep` gibt es also, und es wurde auch gestartet. `grep` hatte mit `o` keine Probleme, aber `p` wollte es wohl öffnen, konnte es aber nicht finden oder hatte nicht das Recht, die Datei zu öffnen.

Übrigens brauchen Sie nicht zu erschrecken. Die meisten UNIX-Umgebungen haben inzwischen auch deutsche Meldungen.

3.4 Operationen mit Dateien

Groß- und Kleinschreibung von Dateien

Eine Datei ist eine DATenEInheit, also ein Rudel Daten in einem Sack. Das können Texte, Programme oder Datenbanken sein. Eine Datei hat immer einen Namen. UNIX unterscheidet sehr genau zwischen Groß- und

4 Eine Ausnahme bildet beispielsweise find, das Optionswörter, aber nur einen Bindestrich verwendet.

5 Als Prompt bezeichnet man das, was in Ruhestellung links neben Ihrer Eingabemarke steht.

Kleinschreibung, auch bei Dateinamen. Die Dateien **montag** und **Montag** können nebeneinander im gleichen Verzeichnis stehen und sind zwei voneinander völlig unabhängige Dateien. Mac OS X behandelt aus Kompatibilitätsgründen zum alten System Klein- und Großbuchstaben gleich. Dieser Effekt tritt ebenfalls auf, wenn man es mit an sich UNIX-fremden Systemen zu tun hat, die sich so verhalten. Dies betrifft beispielsweise SAMBA (siehe S. 505) oder Medien, die unter MS Windows erzeugt worden sind.

Der Name einer UNIX-Datei hat im Gegensatz zu MS-DOS oder VMS oft keine durch einen Punkt abgetrennte Endung, die den Inhalt einer Datei beschreibt. Insbesondere haben ausführbare Dateien spezielle Endungen wie .EXE, .COM oder .BAT. Das kennt UNIX nicht. Ausführbare Dateien haben unter UNIX normalerweise keine Endung. Dagegen gibt es sie durchaus bei C-Programmquelltexten mit .c, Objekten mit .o oder HTML-Dokumenten mit .html.

Wer mit UNIX arbeitet, wird feststellen, dass dort fast alles kleingeschrieben wird. In Kombination mit der Kürze der Kommandos kann man schnell zu dem Schluss kommen, dass UNIX-Benutzer faul sind. Auf jeden Fall ist es der UNIX-Anwender gewohnt, mit wenigen Tastenanschlägen maximale Leistung zu erzielen.

3.4.1 Eine kleine Beispielsitzung

Anhand einer kleinen Beispielsitzung sollen die ersten Befehle von UNIX erklärt werden. Am deutlichsten wird es, wenn Sie alles direkt ausprobieren. Dazu melden Sie sich zunächst einmal an. Als Erstes soll ein Verzeichnis angelegt werden, in dem man nach Herzenslust experimentieren kann.

```
mkdir spielwiese
cd spielwiese
```

Verzeichnis anlegen

Der Befehl `mkdir` (mkdir steht für make directory) erzeugt das Verzeichnis **spielwiese**. Mit `cd` (cd steht für change directory) wechselt man hinein. Ein Verzeichnis ist wie ein Ordner, in dem man mehrere Dateien aufbewahrt. Bei manchen Systemen sehen Sie sogar den Namen **spielwiese** links neben Ihrem Cursor stehen. Als Erstes brauchen Sie eine Datei zum Spielen. Dazu kopieren Sie mit dem ersten Befehl eine Datei namens **passwd** aus dem Systemverzeichnis **/etc** hierher:

```
cp /etc/passwd .
cp passwd anton
cp anton paula
```

Der / ist der Verzeichnistrenner

Verzeichnisnamen werden durch einen normalen Schrägstrich (/, der über der 7, auch *Slash* genannt) und nicht durch den umgekehrten Schrägstrich (\, genannt *Backslash*) getrennt, wie das unter MS Windows oder MS-DOS üblich ist. Im ersten Befehl kopieren Sie aus dem Verzeichnis **/etc** die Datei **passwd** in das aktuelle Verzeichnis. Das aktuelle Verzeichnis wird mit einem Punkt bezeichnet. Jetzt steht im aktuellen Verzeichnis eine Datei **passwd**. Der zweite cp-Befehl kopiert die Datei **passwd** und gibt ihr den Namen **anton**. Da kein Schrägstrich im Argument vorkommt, bewegt sich alles im aktuellen Verzeichnis **spielwiese**. Zuletzt wird **anton** nach **paula** kopiert. Jetzt liegen drei Dateien im Verzeichnis. Das kann man sich anschauen, indem man den Befehl ls (ls steht für list) verwendet:

```
ls
ls -l
ls -l paula
```

Anzeigen der Dateien

Nach dem ersten Befehl haben Sie eine Liste von drei Namen nebeneinander (passwd anton paula). Das sind die Dateien, die Sie oben kopiert hatten. ls zeigt Dateinamen an. Kombiniert mit der Option -l zeigt ls wieder Dateinamen an, aber diesmal in der Langform. Unter UNIX ist es (wie bereits erwähnt) üblich, Befehle mit Optionen zu steuern. Diese Optionen bestehen aus einem Bindestrich und einem Buchstaben oder aus zwei Bindestrichen und einem Wort. Die Langform zeigt unter anderem das Datum der Entstehung und die Größe der Datei in Bytes. Im letzten Befehl wird nach der Option ein Dateiname angegeben. Der Befehl heißt: Zeige die Langform des Dateinamens der Datei **paula**.

Im nächsten Befehl spielen wir ein wenig mit den Argumenten. Im ersten Befehl wird aufgelistet, welche beiden Dateien in Langform angezeigt werden sollen. Im zweiten Befehl wird ein Stern eingesetzt, der als Platzhalter für beliebig viele Zeichen verwendet wird:

```
ls -l passwd paula
ls -l p*
```

*** als Platzhalter**

p* bedeutet: Zeige alle Dateien, die mit p anfangen. Das sind in diesem Fall **passwd** und **paula**. Der Stern bedeutet also: Setze an dieser Stelle beliebige Zeichen ein. Mit p*d sieht man die Datei **passwd**, da **passwd** mit p anfängt und mit d aufhört. p*d* findet ebenfalls die Datei **passwd**,

da der Name mit p anfängt und ein d enthält, wenn auch an der letzten Stelle. Danach folgen beliebig viele Zeichen, in diesem Fall keine.

Die Shell verteilt die Sterne

Wichtig zu erwähnen ist, dass nicht der Befehl `ls` den Stern interpretiert, sondern dass dies von der Kommandozeile, der Shell, getan wird. Dies ist deswegen wichtig, weil so garantiert ist, dass alle Befehle den Stern gleich interpretieren, weil sie ihn eben nicht selbst interpretieren, sondern die Shell dies tut. Sie liefert an das Programm eine Liste aller Dateien, die auf die Maske passen. Die Shell sucht beispielsweise bei `p*` alle Dateien, die mit p beginnen, und listet sie auf. Das Programm `ls` bekommt als Argument von der Shell **passwd paula** geliefert. Noch ein paar Befehle:

```
mv paula erna
rm anton
ls
```

Umbenennen und Löschen

paula wurde nun in **erna** umbenannt. Der Befehl lautet `mv` für move (engl. bewegen). Man kann mit diesem Befehl also Dateien auch in andere Verzeichnisse schieben. **anton** ist nun verschwunden. `rm`, kurz für remove (engl. entfernen), hat diese Datei entsorgt. Zuletzt soll wieder aufgeräumt werden. Die folgenden Befehle löschen alle Dateien, wechseln wieder in das Verzeichnis zurück, aus dem Sie kamen, und entfernen das Übungsverzeichnis:

```
rm *
cd ..
rmdir spielwiese
```

Im ersten Befehl wird `rm *` von der Shell zu `rm erna passwd` expandiert. Der »waagerechte« Doppelpunkt (..) hinter dem `cd` bezeichnet das übergeordnete Verzeichnis. Man spricht auch vom Elternverzeichnis. Sie verlassen also Ihre **spielwiese** und stehen nun dort, wo Sie angefangen hatten. `rmdir` löscht leere Verzeichnisse. Da Sie mit dem `rm` vorher aufgeräumt haben, ist **spielwiese** leer und wird entsprechend auch gelöscht. Ein Verzeichnis kann nicht mit `rm` gelöscht werden,[6] sondern nur mit dem Befehl `rmdir`.

3.4.2 Dateien anzeigen: ls

Die Aufgabe von `ls` ist die Anzeige von Dateinamen. Es folgen Optionen, die mit einem Bindestrich beginnen, und dann die anzuzeigenden Datei-

6 Abgesehen von dem Spezialfall eines rekursiven Löschens mit der Option -r, die später behandelt wird.

en. Werden keine Angaben zu den Dateien gemacht, werden alle Dateien des aktuellen Verzeichnisses angezeigt.

Verzeichnisse anzeigen: ls -d

Wird als Parameter der Name eines Verzeichnisses eingegeben, zeigt `ls` den kompletten Inhalt des angegebenen Verzeichnisses. Dies kann leicht irritierend sein, wenn man nur nähere Informationen zu einem Verzeichnis haben wollte. Mit der Option `-d` (d wie directory) kann man diesen Effekt unterbinden. Dann wird nur das Verzeichnis angezeigt und nicht hineingeschaut.

Beispielsweise legen Sie ein Verzeichnis **spielwiese** an und lassen es sich hinterher mit `ls spielwiese` anzeigen, um zu sehen, ob es noch da ist. Sie sehen – nichts. Der Grund liegt darin, dass `ls` das Verzeichnis **spielwiese** nicht selbst anzeigt, sondern seinen Inhalt. Da dieser bisher leer ist, wird die leere Liste angezeigt. Es sieht also so aus, als wäre **spielwiese** verschwunden. Mit `ls -ld spielwiese` kann man sie wiederum sichtbar machen.

ls -l zeigt die Langform

Einige Optionen werden häufiger gebraucht. Die Langform mit `-l` zeigt alle Informationen zu den Dateien, die man sich wünschen kann. Von rechts nach links erkennt man den Dateinamen, es folgen der Zeitpunkt der Erstellung und die Größe der Datei. Das erste Zeichen der Zeile zeigt ein `d` für Verzeichnisse und ein Minuszeichen für eine normale Datei. Die anderen Informationen werden in den späteren Kapiteln noch ausführlich erläutert. Hier sehen Sie ein Beispiel für die Ausgabe von `ls -l`:

```
-rwx------    1 arnold   users       13654 Jan 17 04:41 a.out
drwxr-x---    2 arnold   users        4096 Mär  6 23:35 awk
-rwxr-xr-x    1 arnold   users       13856 Feb  8 10:48 copy
-rw-r--r--    1 arnold   users         175 Feb  8 10:48 copy.c
drwxr-xr-x    2 arnold   users        4096 Jan 13 12:37 dir
-rw-rw-r--    1 arnold   users          98 Apr 13 13:55 doppel.c
-rwxr-xr-x    1 arnold   users       13516 Feb  8 21:25 env
-rw-r--r--    1 arnold   users         165 Feb  8 21:25 env.c
drwx------    2 arnold   users        4096 Feb 21 14:13 ipc
drwxr-xr-x    2 arnold   users        4096 Feb 13 13:51 make
-rwxr-xr-x    1 arnold   users       13489 Apr 11 21:34 moin
-rw-r--r--    1 arnold   users          60 Apr 13 13:55 moin.c
```

Der Dateiname kann unter UNIX relativ frei gestaltet werden. Er darf keinen Schrägstrich enthalten, da dieser als Verzeichnistrenner interpretiert wird. Enthält der Dateiname Sonderzeichen, die die Shell interpretiert, muss man den Namen in Anführungszeichen setzen, wenn man ihn von

der Kommandozeile aus benutzt. Alternativ kann man auch einen Backslash vor das Sonderzeichen stellen. Für die Shell bedeutet der Backslash, dass sie das folgende Zeichen nicht interpretieren, sondern einfach durchreichen soll.

Option	Wirkung
-l	Zeigt alle Informationen über die Datei an.
-a	Zeigt auch die Dateien, die mit einem Punkt beginnen.
-d	Zeigt das Verzeichnis und nicht dessen Inhalt.
-t	Sortiert nach letzter Änderung. Neueste Dateien kommen zuerst.
-r	Dreht die Sortierreihenfolge um.
-R	Zeigt alle Unterverzeichnisse.

Tabelle 3.3 Einige Optionen zu ls

ls -a zeigt alle Dateien

Die Option `-a` zeigt auch Dateien an, die mit einem Punkt beginnen. Unter UNIX werden Dateien, die zur Konfiguration verwendet werden, gern so benannt, dass das erste Zeichen ein Punkt ist. So stören sie nicht, wenn man eigentlich nur die Arbeitsdateien betrachten will. Auch ein versehentliches Löschen wird unwahrscheinlicher, weil der Befehl `rm *` diese Dateien nicht erfasst. Will man `-l` und `-a` kombinieren, kann man `ls -a -l` oder einfacher `ls -al` angeben.

ls -t sortiert nach der Zeit

Mit der Option `-t` wird statt nach dem Alphabet nach der Zeit sortiert. Dabei erscheint zuerst die neueste Datei. Mit der Option `-r` wird die Reihenfolge umgekehrt.

Die Option `-R` zeigt eine Liste aller Unterverzeichnisse mit ihren Unterverzeichnissen.

ls hat erheblich mehr Optionen. Wer sie gern alle kennen lernen möchte, muss nur `man ls` eingeben.

3.4.3 Dateien kopieren: cp

cp kopiert Dateien

Der Befehl cp kopiert Dateien an genau ein Ziel. Wird nur eine Datei kopiert, kann das Ziel ein Dateiname sein. Dann wird von der Quelldatei eine Kopie angefertigt und diese unter dem Zielnamen abgespeichert. Werden dagegen mehrere Dateien kopiert, muss das Ziel ein Verzeichnis sein, da ja nicht alle Quelldateien den gleichen Namen bekommen können. Die Kopien finden sich nach der Ausführung unter ihrem bishe-

rigen Namen im angegebenen Zielverzeichnis. Soll das Ziel das aktuelle Verzeichnis sein, muss der Punkt dafür angegeben werden.

Um mehrere Dateien zu erfassen, kann man sie aufzählen oder durch einen Stern auswählen. Wichtig ist, dass der zuletzt angegebene Name von cp immer als Ziel interpretiert wird. Ein Verzeichnis als Quelle wird von cp einfach übergangen.

cp erstellt neue Dateien

cp erzeugt beim Kopieren immer eine neue Datei. Darum hat eine Dateikopie auch immer den Zeitpunkt des Kopierens als Änderungsdatum, und nicht das Datum der Datei, von der sie kopiert wird. Auch der Eigentümer der neu entstandenen Datei ist immer der Anwender, der den Befehl aufgerufen hat. Dieses Verhalten kann mit der Option -p unterbunden werden, die aber nicht in allen UNIX-Versionen vorhanden ist.

Auch die Möglichkeit, mit der Option -r komplette Verzeichnisbäume zu kopieren, ist nicht auf allen Systemen verfügbar. Darum wird das Kopieren kompletter Verzeichnisbäume unter Beibehaltung aller Eigenschaften auf älteren Systemen normalerweise mit dem Kommando tar realisiert, das an anderer Stelle betrachtet wird (siehe S. 153).

Welche Möglichkeiten cp auf Ihrem System sonst noch bietet, erfahren Sie wieder mit man cp.

3.4.4 Dateien verschieben oder umbenennen: mv

mv verschiebt Dateien

Der Befehl mv hat eigentlich zwei Funktionen. Wie die Abkürzung mv für move impliziert, können Sie eine Datei verschieben, also in ein anderes Verzeichnis bringen. Die Datei ist dann an der Ausgangsposition nicht mehr vorhanden. Das können Sie auch mit mehreren Dateien gleichzeitig machen. Das letzte Argument ist dann immer das Zielverzeichnis.

Umtaufen

Sie können mit dem Befehl mv aber auch einer Datei einen neuen Namen geben. Sobald das Ziel nicht ein existierendes Verzeichnis ist, wird mv davon ausgehen, dass Sie die Datei oder das Verzeichnis umbenennen wollen. Da eine Namensgebung sehr individuell ist, können Sie nur eine Datei gleichzeitig umbenennen.

Schneller Schieber

Oft werden Sie feststellen, dass das Verschieben selbst großer Dateien erstaunlich schnell erledigt ist. Das liegt daran, dass der Befehl mv die Dateien selbst gar nicht anfasst, sondern nur die Datei-Einträge in andere Verzeichnisse verschiebt. Eine Ausnahme gibt es nur, wenn der Ursprung und das Ziel auf verschiedenen Dateisystemen liegen, beispielsweise auf verschiedenen Festplatten. Dann kann mv nicht einfach Einträge

verschieben, sondern kopiert zunächst die Dateien. Dann wird das Original gelöscht, und zu guter Letzt werden die Eigenschaften der Originale übernommen.

Eigenschaften bleiben

Da die Datei selbst beim Befehl `mv` nicht verändert wird, bleiben auch das Datum, der Eigentümer und die Rechte der Datei unverändert.

3.4.5 Dateien löschen: rm

rm -i löscht nur nach Rückfrage

Der Befehl `rm` löscht Dateien. Dies geschieht unwiderruflich, und darum ist es ganz gut, dass es die Option `-i` gibt. Dann fragt `rm` bei jeder einzelnen Datei nach, ob sie wirklich entfernt werden soll.

Will man einen kompletten Verzeichnisbaum löschen, verwendet man die Option `-r`. Damit geht `rm` rekursiv den gesamten Baum durch und entfernt alle enthaltenen Dateien und Verzeichnisse.

3.4.6 Verzeichnisbefehle: mkdir, rmdir, cd und pwd

Um Dateien zu ordnen und zu gruppieren, legt man Verzeichnisse an. Der Befehl zum Erzeugen von Verzeichnissen lautet `mkdir`.[7]

cd wechselt das Verzeichnis

Der Anwender befindet sich immer in einem Verzeichnis. Mit `cd` wechselt er das Verzeichnis. In diesem arbeitet er ab sofort, darum spricht man auch von seinem Arbeitsverzeichnis. `cd` ohne Parameter wechselt in das Heimatverzeichnis. Das Heimatverzeichnis ist das Verzeichnis, in dem sich der Anwender nach dem Anmelden zunächst befindet.

pwd zeigt den aktuellen Pfad an

Um zu ermitteln, in welchem Verzeichnis man sich aktuell befindet, verwendet man den Befehl `pwd` (print work directory).

rmdir löscht leere Verzeichnisse

Um ein Verzeichnis wieder verschwinden zu lassen, verwendet man `rmdir` (remove directory). Allerdings kann man mit `rmdir` nur leere Verzeichnisse löschen und auch nur solche, die derzeit nicht verwendet werden. In Verwendung sind auch Verzeichnisse, in denen noch eine Sitzung von einem anderen Terminal stattfindet. Man darf nämlich nicht einfach jemand anderem das Verzeichnis unter den Füßen wegziehen. Leider ist es nicht immer ganz leicht herauszufinden, wer mit welchem Prozess dort gerade arbeitet. Hier helfen die Befehle `fuser` und `ps` (siehe S. 414) und auch `lsof` (siehe S. 415).

7 Die Abkürzung md für mkdir gibt es unter UNIX nicht. Wer sie vermisst, kann sie sich aber als alias (siehe S. 165) definieren.

Funktion	Wirkung
print	Anzeigen
printf	Entspricht der C-Funktion printf.
sprintf	Für die Speicherung von Ausgaben in einer Variablen
length(s)	Ermittelt die Länge einer Zeichenkette.
substr(s, anf, anz)	Teilstringbildung
index(s, t)	Gibt die Stelle an, an der in s die Zeichenkette t beginnt.
sqrt(n)	Wurzel
log(n)	Natürlicher Logarithmus
exp(n)	e^n, e=2,71828
int(n)	Ganzzahliger Anteil

Tabelle 3.26 awk-Funktionen

3.10.9 Weitere Werkzeuge im Überblick

Es gibt noch eine Reihe sehr praktischer kleiner Werkzeuge. Häufig sind es kleine Programme, die sehr spezielle Aufgaben lösen. Sie sind manchmal beinahe unbekannt, aber vielleicht in der einen oder anderen Situation hilfreich. Da sie zugegebenermaßen ein wenig exotisch sind, sollen sie hier nur tabellarisch aufgezählt werden.

Programm	Funktion
split	Teilt eine große Textdatei in mehrere kleine auf.
cut	Schneidet einen angegebenen Teil aus einer Datei heraus.
fold	Umbricht Zeilen ab einer bestimmten Zeilenlänge.
tr	Wandelt Zeichen um.
diff	Zeigt die Unterschiede zwischen zwei Textdateien (siehe S. 727).

Tabelle 3.27 Diverse UNIX-Werkzeuge

Genauere Informationen finden Sie in den passenden Manpages.

3.11 Reguläre Ausdrücke

Für den Suchbegriff kann in vielen UNIX-Programmen wie `grep`, `sed`, `awk` oder `vi` ein regulärer Ausdruck verwendet werden. Zunächst einmal ist ein regulärer Ausdruck nichts anderes als ein Suchbegriff, und man kann

ganz naiv den Begriff verwenden, den man sucht. Wenn Sie also das Wort »Maus« suchen, können Sie auch »Maus« als regulären Ausdruck angeben. Reguläre Ausdrücke können komplexeste Suchmuster beschreiben. Dann sehen diese Ausdrücke auf den ersten Blick allerdings auch ziemlich erschreckend aus. Es lohnt sich aber, diese Beschreibungssprache zu lernen, da UNIX den Programmierern eine Bibliothek anbietet, in der die Suche nach regulären Ausdrücken unterstützt wird. Es ist so für den Programmierer leichter, die regulären Ausdrücke zu verwenden, als selbst eine Suche mit Platzhaltern zu basteln.

Anders als Wildcards

Zunächst werden einfache Platzhalter verwendet. Bei den Dateimasken der Shell, den so genannten Wildcards, gibt es solche Platzhalter auch. Die regulären Ausdrücke haben allerdings nichts mit den Wildcards zu tun, die die Shell verwendet. Dort hat der Stern beispielsweise eine andere Bedeutung als hier. Das einfachste Sonderzeichen ist der Punkt. Er steht stellvertretend für genau ein beliebiges Zeichen. Die Suche nach `M..s` findet die Wörter Maus, Moos und Muks, aber nicht Murks, da hier zwischen M und s drei Zeichen stehen. Der Punkt ist also in der Wirkung mit dem Fragezeichen bei den Wildcards vergleichbar.

Multiplikatoren

Der Stern und das Pluszeichen sind Multiplikatoren und beziehen sich immer auf das Zeichen links neben sich. Das Pluszeichen besagt, dass das Zeichen einmal oder mehrfach auftreten kann. Beim Stern ist es auch denkbar, dass das Zeichen gar nicht erscheint. Die Suche nach `abc*` findet also abc, abcc, abccccc, aber auch ab. Wirklich interessant werden die Multiplikatoren in Verbindung mit dem Punkt. So findet `M.*s` Maus und Moos, aber eben auch Murks und Meeresfrüchte.

Anfang und Ende

Hier werden Sie vielleicht stutzen, denn Meeresfrüchte enden doch gar nicht auf s. Das ist richtig, aber im regulären Ausdruck wurde ja auch gar nicht erwähnt, dass das Wort hinter s enden soll. Das müsste man explizit mit einem \> angeben. Das Gegenstück lautet \< und bedeutet Wortanfang. So wie nach dem Wortanfang und -ende gesucht werden kann, so gibt es auch das ^ für den Zeilenanfang und das $ für das Zeilenende. Eine wichtige Anwendung ist, die Verzeichnisse anzeigen zu lassen. Unter UNIX unterscheidet man Dateien von Verzeichnissen an dem kleinen d am Zeilenanfang, wenn man `ls -l` aufruft. Dementsprechend würde folgende Befehlskombination nur die Verzeichnisse anzeigen:

```
gaston> ls -l | grep ^d
drwxr-xr-x    3 arnold   users        4096 Jun 25 20:57 pic
drwxr-xr-x    2 arnold   users        4096 Jun 28 20:55 unprog
gaston>
```

grep sucht in der Standardeingabe ein d, das direkt dem Zeilenanfang folgt, oder anders ausgedrückt, das am Anfang der Zeile steht. Ohne das Dach hätte man alle Zeilen erhalten, in denen ein d steht. Da der Benutzer arnold heißt, wären das wohl alle Dateien des Verzeichnisses.

Rechteckige Klammern

Die eckigen Klammern haben bei den regulären Ausdrücken die gleiche Bedeutung wie bei den Wildcards. Sie stehen für ein Zeichen, das durch den Inhalt der Klammern beschrieben wird. Hier ist es möglich, die Zeichen einfach aufzuzählen oder aber den Bindestrich zu nutzen, um Bereiche anzugeben. Typisch sind hier die Zahlen, geschrieben als 0-9, oder die Kleinbuchstaben als a-z. Das folgende Beispiel beschreibt ein Wort, das mit einem Großbuchstaben beginnt, dem beliebig viele Großbuchstaben oder Zahlen folgen.

```
\<[A-Z][A-Z0-9]*\>
```

Alle Zeichen, außer . . .

Bisher wurde nach Mustern gesucht, die existieren. Es gibt Situationen, da werden alle Zeichenfolgen gesucht, in denen ein bestimmtes Zeichen nicht vorkommt. Wenn Sie beispielsweise in einem TEX-Dokument über die Programmiersprache PASCAL nach der Zeichenkette »begin« suchen, werden Sie in erster Linie die Folge \begin finden, die TEX sehr intensiv verwendet. Sie würden also einen regulären Ausdruck verwenden wollen, der besagt, dass Sie alle »begin« suchen, die nicht durch einen Backslash eingeleitet werden. Dazu wird zunächst die eckige Klammer verwendet, die auch benutzt wird, um eine Menge von Zeichen zu beschreiben, die an einer Position auftreten kann. Steht als erstes Zeichen ein ^, so bedeutet das, dass die angeführten Zeichen nicht vorkommen sollen. Danach führt man den Backslash an, den man wiederum verdoppeln muss, damit er nicht als Kommando missinterpretiert wird:

```
[^\\]begin
```

Ersetzen im vi

Vielfältige Möglichkeiten gewinnt man im vi dadurch, dass man als Suchwort einen regulären Ausdruck verwenden kann. Ganz besondere Möglichkeiten tun sich dadurch auf, dass man Markierungen innerhalb eines Ausdrucks setzen und diese beim Ersetzen verwenden kann. Ein praktisches Beispiel findet sich beim Umsetzen von TEX-Dokumenten nach HTML. In der ersten Zeile sehen Sie eine Überschrift in TEX und darunter eine in HTML:

```
\section{Dies ist ein spannendes Kapitel}
<H1>Dies ist ein spannendes Kapitel</H1>
```

Um alle Vorkommen von section in die entsprechenden <H1> umzuwandeln, wird ein regulärer Ausdruck verwendet. Zunächst wird das Muster beschrieben, das eine section erkennt.

```
\\section{.*}
```

Der doppelte Backslash muss sein, damit er nicht fälschlich als Kommando interpretiert wird. In den geschweiften Klammern steht schlicht Punkt Stern, also der Ausdruck für eine beliebige Zeichenfolge. Das ist unsere Überschrift, die wir gern übernehmen wollen. Also wird davor und dahinter eine Markierung gesetzt.

```
\\section{\(.*\)}
```

Nun wird das Ganze in den Ersetzungsbefehl von `vi` eingesetzt. Der komplette Aufruf lautet also:

```
:1,$ s/\\section{\(.*\)}/<H1>\1<\/H1>/g
```

Der letzte Backslash der Zeile muss sein, sonst glaubt `vi`, dass der Schrägstrich des </H1> der Befehl dafür wäre, dass der Ersetzungsbereich hier endet. Die Zeichenfolge \1 in der Ersetzung liefert den in der Markierung gemerkten Wert und befördert die Überschrift in die gewünschte, neue Umklammerung.

Machen Sie sich klar, dass Sie sich mit diesem zugegeben etwas kryptischen Befehl vielleicht stundenlange Arbeit ersparen, wenn Sie in einem langen Dokument die Überschriften austauschen müssen. Und überlegen Sie sich auch, ob Sie so etwas mit einem normalen Editor ohne reguläre Ausdrücke auch könnten.

Dass so viele Programme mit regulären Ausdrücken umgehen können, liegt daran, dass UNIX dem Programmierer die Suche nach regulären Ausdrücken aus einer Bibliothek anbietet (siehe S. 806).

In der folgenden Tabelle sehen Sie eine Übersicht über die regulären Ausdrücke. Sie können diese überall da einsetzen, wo die Dokumentation eine »regular expression« nennt.

Ausdruck	Bedeutung
. (Punkt)	Steht für ein einzelnes beliebiges Zeichen.
[afg]	Das Zeichen a, f oder g muss an dieser Stelle erscheinen.
[0-9]	Eine Ziffer muss an dieser Stelle stehen.
*	Das vorangehende Zeichen kommt beliebig oft vor.
+	Das vorangehende Zeichen kommt mindestens einmal vor.
^	Zeilenanfang
$	Zeilenende
\<	Wortanfang
\>	Wortende
\	Das folgende Zeichen wird nicht als Metazeichen interpretiert.
\(\)	Markierung eines Bereichs
\1 \2 . . .	Referenz auf die erste und zweite Markierung

Tabelle 3.28 Reguläre Ausdrücke (regular expressions)

3.12 Pack deine Sachen und geh ...

Wenn man ein Rudel Dateien, das vielleicht auch noch in einem größeren Verzeichnis untergebracht ist, transportieren möchte, dann hat man zwei Probleme. Erstens sollte das Paket handlich sein und zweitens möglichst klein. Dafür gibt es unter UNIX je eine Lösung: `tar` und `compress`. Und natürlich kann man auch beide kombinieren.

3.12.1 Verschnüren: tar

Pflichtoption c, x oder t

Das Programm `tar` (tape archiver) kommt aus dem Bereich der Datensicherung. Aber es ist ungeheuer praktisch im Umgang mit Dateien. Mit `tar` kann man packen, auspacken und Pakete anschauen. Gesteuert wird die Funktion durch die erste Option: c (create) zum Erzeugen, x (extract) zum Auspacken und t zum Anschauen. Genau eine von diesen Optionen braucht `tar`, damit klar ist, was zu tun ist.

Dateien in ein Archiv packen

Da die Daten in eine Datei gepackt werden sollen, braucht man die Option f (file) für Datei. Denn dadurch arbeitet dann tar nicht auf dem Standardbandlaufwerk, sondern auf der angegebenen Datei. An dieser Stelle wird deutlich, dass `tar` ursprünglich für Magnetbänder entwickelt wurde. Und zu guter Letzt ist die Option v (verbose) hilfreich. Dann erzählt `tar`, welche Dateien bearbeitet werden.

Prozesse von jemandem unterbricht. Der Wechsel selbst erfolgt einfach durch Aufruf von:

```
init 1
```

Man kehrt in den bisherigen Modus zurück, indem man die Shell mit `ctrl-D` schließt oder durch den Befehl `init 2` bzw. `init 3`.

5.5 Benutzerverwaltung

Die Benutzerverwaltung gehört zu den Routinetätigkeiten des Administrators. Für jeden Benutzer wird eine Kennung, ein Passwort und ein Bereich auf der Platte angelegt, in dem er arbeiten kann. Es sind Aspekte der Einbruchsicherheit zu berücksichtigen, und hin und wieder müssen den Benutzern auch Grenzen gesetzt werden.

5.5.1 Benutzerverwaltung unter UNIX

Informationen über die Anwender werden in der Datei **/etc/passwd** gehalten. Hier steht für jeden Benutzer, wo das Heimatverzeichnis liegt, welche Shell gestartet wird, und hier werden die User-ID und die Group-ID gespeichert. Traditionsgemäß enthält die Datei **/etc/passwd** auch das verschlüsselte Passwort. Daher hat sie ihren Namen.

Die Datei **/etc/passwd** ist für jeden Benutzer des Systems lesbar. Dies ist auch dann nicht unbedingt leichtsinnig, wenn man weiß, dass der Algorithmus zur Verschlüsselung des Passworts öffentlich zugänglich ist. Sie können sich die Verschlüsselung eines Passworts ansehen, indem Sie einen einfachen Perl-Aufruf ausführen:

```
perl -e "print crypt('Mein2Passwd','aw');"
```

crypt

Das erzeugte Passwort lautet: aw7JrjDPWvzqE. Sie erkennen die beiden Zeichen »aw« am Anfang des Passwortes wieder, die beim Aufruf der Funktion `crypt()` an zweiter Stelle stehen. Diese beiden Zeichen können beliebige Buchstaben und Ziffern sein. Dadurch ist es möglich, das gleiche Passwort auf verschiedene Weisen zu verschlüsseln. Die Funktion `crypt()` gehört zum POSIX-Standard und kann damit auf jedem UNIX-System von fast jeder Programmiersprache aus aufgerufen werden. Durch den freien Zugriff auf die Funktion `crypt()` ist es jedem UNIX-Programm leicht möglich, Passwörter zu verschlüsseln und es damit zu vermeiden, dass unverschlüsselte Passwörter in Dateien stehen oder über Netzwerkleitungen übertragen werden.

Nun könnte man vermuten, dass ein Verfahren zur Verschlüsselung von Passwörtern kaum sicher sein kann, wenn dessen Quellcode jedermann zur Verfügung steht. Tatsächlich ist es trotz der Öffentlichkeit des Quellcodes von `crypt()` nicht möglich, aus dem verschlüsselten Passwort das Passwort im Klartext zu generieren. Der Grund dafür ist ein so genannter Falltür-Algorithmus. Das bedeutet, dass man aus dem Passwort im Klartext zwar leicht die Verschlüsselung ermitteln kann, dass der umgekehrte Weg aber nicht möglich ist. Es ist also wie bei einer Falltür: Es ist leicht, durch die Falltür nach unten zu kommen. Von unten kann man aber nicht mehr zur Ausgangsposition zurück. Ein anschauliches Beispiel für einen Falltür-Algorithmus ist die Modulo-Rechnung. Sie ermittelt den Rest einer ganzzahligen Division. So ergibt 25 geteilt durch 7 immer den Rest 4. Aus der Kenntnis der 4 und des Algorithmus können Sie aber immer noch nicht schließen, ob eine 25 oder eine 18 als Ausgangswert verwendet wurde.

Angriff per Lexikon

Wie so oft sitzt aber das Sicherheitsproblem vor dem Computer, und Sie können mit den Passwörtern Auto, Bier und Sonne in erstaunlich viele Systeme einbrechen. Diese Sorte Passwörter können leicht geknackt werden, indem der Angreifer ein Programm schreibt, das eine Lexikondatei und ein Namensregister durchgeht und von jedem Eintrag ein Passwort generiert. Ist das verschlüsselte Wort in der Passwortdatei vorhanden, kennt der Angreifer das Passwort. Aus diesem Grund soll ein Passwort weder ein normaler Begriff aus dem Lexikon noch ein Name sein.

Um solche Attacken zu erschweren, stehen in den heutigen UNIX-Systemen die Passwörter nicht mehr in der Datei **/etc/passwd**, die nach wie vor öffentlich lesbar ist, sondern werden in der Datei **/etc/shadow** oder an anderen Orten abgelegt, die für den normalen Anwender nicht lesbar sind. Sollte Ihr System nicht /etc/shadow verwenden, finden Sie mit dem Befehl »man passwd« Informationen darüber, in welcher Datei die Passwörter stehen.

Pseudo-Benutzer

In der Benutzerdatei **/etc/passwd** finden Sie nicht nur Anwender aus Fleisch und Blut, sondern auch Benutzer, die für interne Zwecke verwendet werden. So ist auf vielen Systemen ein Benutzer mail oder lp eingetragen. Dabei steht mail für die Postverteilung und lp für die Druckerverwaltung. Diese Dienste müssen nicht unbedingt von root durchgeführt werden. Der normale Schutz vor Angreifern, den jeder Benutzer genießt, reicht für diese Zwecke völlig aus. Da diese Benutzer aber nur für Hintergrundprozesse verwendet werden, ist es nicht wünschenswert, dass sich jemand unter diesen Namen anmelden kann.

5.5.2 Die Benutzerdatei /etc/passwd

In den älteren UNIX-Systemen manipulierte man für einen Benutzereintrag direkt die Datei **/etc/passwd**. Dieser Weg steht prinzipiell auch heute noch offen, sofern die Passwörter nicht zentral im Netz unter NIS (siehe S. 467) verwaltet werden.

Die Datei **/etc/passwd** zeigt, wie der Benutzer eines UNIX-Systems definiert ist. Ein Eintrag in der **/etc/passwd** hat folgenden Aufbau:

```
Name:Passwort:User-ID:Group-ID:Kommentar:Verzeichnis:Shell
```

Hier folgt ein Beispiel für eine **/etc/passwd** Datei:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:daemon:/sbin:/bin/bash
lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash
ftp:x:40:2:ftp account:/usr/local/ftp:/bin/bash
named:x:44:44:Nameserver Daemon:/var/named:/bin/bash
nobody:x:65534:65534:nobody:/var/lib/nobody:/bin/bash
arnold:x:501:100:Arnold Willemer:/home/arnold:/bin/bash
andrea:x:502:100::/home/andrea:/bin/bash
```

Die Einträge bedeuten im Einzelnen:

- **Name**
  Der Benutzername wird beispielsweise zur Anmeldung am System verwendet. Unter diesem Namen wird der Benutzer bei Rechtezuordnungen angesprochen. Häufig wird dazu der Nachname, der Vorname oder eine Mischung aus beidem verwandt. Der Benutzername ist alles andere als geheim und sollte leicht der wirklichen Person zuzuordnen sein. Bei einigen Systemen ist er auf acht Buchstaben begrenzt.
- **Passwort**
  In Systemen ohne Shadow-Datei steht hier das verschlüsselte Passwort.

  Beim Anlegen eines neuen Benutzers lässt man diesen Eintrag frei. Ein leerer Eintrag erlaubt den Zugang ohne Passwort. Gleich anschließend sollte der neue Benutzer den Eintrag durch den Aufruf des Befehls `passwd` füllen.

  Soll der Benutzer für interne Zwecke angelegt werden, dann ist es durchaus möglich, dass man vermeiden will, dass sich jemand unter dieser Kennung anmeldet. Das erreicht man, indem man einen Stern

oder ein X als Passwort einsetzt. Der Verschlüsselungsalgorithmus kann nämlich niemals einen einstelligen Eintrag an dieser Stelle erzeugen. Damit gibt es kein Passwort, das hierzu passt.

Ein kleines x an dieser Stelle in allen Einträgen deutet darauf hin, dass das System eine Shadow-Datei für die Passwörter verwendet.

- **User-ID**
  Jeder Benutzer hat seine eigene Nummer. Normale Benutzer werden auf einigen Systemen ab 50, ab 100 oder neuerdings ab 500 angelegt. Die kleineren Nummern sind teilweise für Systemdienste festgelegt, so ist das Administrationskennwort root mit der User-ID 0 verbunden. Der Eigentümer von Dateien wird in den i-nodes (siehe S. 101) mit der User-ID gekennzeichnet.

- **Group-ID**
  Jeder Benutzer gehört zu mindestens einer Gruppe. Die Hauptgruppe, zu der der Benutzer gehört, wird hier eingetragen. Der Benutzer kann in der Datei **/etc/group** auch in weiteren Gruppen angemeldet werden. Die Gruppen werden ab Seite 271 ausführlich behandelt.

- **Kommentar**
  Hier wird im Klartext eingetragen, wer der Benutzer ist. Der Eintrag hat informativen Charakter.

- **Verzeichnis**
  Das Heimatverzeichnis des Benutzers. Hier hat der Anwender seinen Arbeitsbereich, und hier wird er auch landen, sobald er sich eingeloggt hat. Auch Einstellungen wie die .profile-Datei stehen hier. Der neue Benutzer muss in das Verzeichnis wechseln und es lesen können. Das Verzeichnis muss also mindestens die Rechte x und r haben und dem Benutzer gehören. In den meisten Fällen werden die Benutzer es schätzen, wenn sie in ihrem Bereich auch schreiben können.

  Unter den meisten Systemen wird als Pfad **/home** verwendet. Darin wird der Benutzername als Verzeichnisname benutzt. Für den Anwender arnold wird also das Verzeichnis **/home/arnold** erzeugt. Bei Solaris funktioniert dies nicht. Hier wird das Verzeichnis **/export/home/arnold** erzeugt. Der Automounter (siehe S. 502) wird dieses Verzeichnis automatisch in **/home/arnold** übersetzen.

- **Shell**
  Hier wird normalerweise die Shell eingetragen, die für den Benutzer beim Einloggen gestartet wird. Je nach Geschmack kann neben der klassischen Bourne-Shell (sh) auch die Korn-Shell (ksh), die C-Shell (csh) oder die Bourne-Again-Shell (bash) eingesetzt werden. Lediglich

für den root sollte man eine Shell wählen, die auch dann zur Verfügung steht, wenn nur die Bootpartition ansprechbar ist. Die Shell ist mit vollem Pfadnamen einzutragen.

Um neue Benutzer direkt in die **/etc/passwd**-Datei einzutragen, kopiert man am einfachsten einen bisherigen Eintrag, korrigiert die benutzerbezogenen Daten, insbesondere den Benutzernamen und erstellt ein neues Heimatverzeichnis. Zum Beispiel:

```
meier::237:106::/home/meier:/bin/sh
```

Für den Benutzer wird ein Verzeichnis eingerichtet. In unserem Beispiel:

```
mkdir /home/meier
```

Das Heimatverzeichnis muss nicht zwingend unterhalb des Verzeichnisses **/home** angelegt werden. Prinzipiell können Sie jedes Heimatverzeichnis an jeder beliebigen Stelle des Verzeichnisbaums anlegen. Zum Heimatverzeichnis wird es ausschließlich durch den Eintrag in die Datei **etc/passwd**. Allerdings hat sich das Anlegen des Benutzerverzeichnisses **/home** als sinnvoll erwiesen. Gegebenenfalls wird der Benutzer in die Gruppendatei **/etc/group** aufgenommen (siehe S. 271). Zum Beispiel:

```
projekt_a::106:petersen,meier
post::107:schulz,mueller,meier
```

Das Heimatverzeichnis des neuen Benutzers wird mit dessen Zugriffsrechten versehen. Zum Beispiel:

```
gaston# chown meier /home/meier
gaston# chgrp projekt_a  /home/meier
```

Das Heimatverzeichnis **/home/meier** gehört nun auch dem Benutzer meier. Der Gruppeneigentümer ist projekt_a. Üblicherweise verwendet man als Gruppe diejenige, die in der Datei **/etc/passwd** als Standardgruppe des Benutzers eingetragen ist. Mit dem Befehl `passwd` wird schließlich das Passwort gesetzt:

```
passwd meier
```

Wenn allerdings Schattenpasswörter in der Datei **/etc/shadow** verwendet werden, muss vor dem ersten Aufruf von `passwd` noch ein Eintrag für meier in dieser Datei angelegt werden. Ansonsten setzt der Befehl `passwd` das verschlüsselte Passwort doch wieder in die Datei **/etc/passwd**.

5.5.3 Verborgene Passwörter: shadow

Die Datei **/etc/passwd** enthält heute normalerweise keine verschlüsselten Passwörter mehr. Diese werden im System in einer zweiten Datei namens **/etc/shadow** abgelegt, die nur noch von root lesbar ist. Der Grund für diese Maßnahme ist, dass viele Anwender Passwörter verwenden, die leicht zu knacken sind. Das ist vor allem der Fall, wenn das Passwort Wörtern entspricht, die man in einem Lexikon findet oder die übliche Vornamen sind.

Angriff auf /etc/passwd

Ein gängiger Angriff auf eine Maschine erfolgte dadurch, dass man die für jeden lesbare Datei **/etc/passwd** kopierte. Anschließend ließ man ein kleines Programm über ein Wörterbuch laufen, das jedes Wort verschlüsselt und die Verschlüsselung mit der Zeichenfolge in der **/etc/passwd** vergleicht. Ein solches Programm zu schreiben ist nicht weiter schwierig, da das Verfahren, mit dem UNIX seine Passwörter verschlüsselt, öffentlich bekannt ist.

So wie man einen Benutzer in der **/etc/passwd** von Hand eintragen kann, ist das auch in der **/etc/shadow** möglich. Am einfachsten ist es auch hier, einen existierenden Eintrag zu kopieren und an den neuen Benutzer anzupassen. Wie in der **passwd**-Datei werden auch hier die Einträge durch Doppelpunkte voneinander getrennt. Dabei stehen folgende Einträge hintereinander:

- Die Benutzerkennung, die auch in der **/etc/passwd**-Datei in der ersten Spalte steht
- Das Passwort
- Der Tag der letzten Änderung des Passworts. Das Datum wird als Zahl der seit dem 1.1.1970 vergangenen Tage[12] codiert.
- Die Anzahl der Tage, nach denen das Passwort erstmals geändert werden darf
- Die Anzahl der Tage, nach denen das Passwort geändert werden muss
- Die Anzahl der Tage, die der Benutzer vor dem Ablauf des Passworts gewarnt wird
- Eine Frist in Tagen, die angibt, wann das Passwort endgültig abgelaufen ist.

12 Im Gegensatz zur sonst bei UNIX üblichen Codierung eines Zeitpunkts als die Anzahl der Sekunden seit dem 1.1.1970 wird hier tatsächlich die Anzahl der Tage verwendet.

- Das Datum, an dem der Zugang geschlossen wurde, und zwar in Tagen seit dem 1.1.1970
- Reserviertes Feld

Beispiel eines Eintrags:

```
arnold::11545:0:99999:7:0::
```

Passwörter zeitlich begrenzen

Man sieht, dass man durch die **/etc/shadow**-Datei neben der Sicherung der verschlüsselten Passwörter auch die Möglichkeit gewinnt, Passwörter ablaufen zu lassen. Wenn Sie diese Option einsetzen möchten, denken Sie daran, dass ein häufiger Wechsel kontraproduktiv sein kann. Sobald die Anwender den Überblick verlieren, werden sie die Passwörter aufschreiben, anstatt sie sich zu merken, oder sie lassen sich Kreationen wie januar1, februar2 und so fort einfallen.

5.5.4 Benutzerpflege automatisieren

useradd

Bei den meisten Systemen existiert ein Dienstprogramm namens useradd. Manchmal heißt es auch adduser. Damit kann man einen Benutzer anlegen. Das Programm prüft die Verträglichkeit des Eintrags und setzt sinnvolle Vorgaben, damit man nicht so viel tippen muss. Im folgenden Beispiel wird ein Benutzer mit der Kennung testperson angelegt:

```
gaston# useradd testperson
gaston# cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
  ...
nobody:x:65534:65534:nobody:/var/lib/nobody:/bin/bash
willemer:x:500:100:Arnold Willemer:/home/willemer:/bin/bash
arnold:x:501:100:Arnold Willemer:/home/arnold:/bin/bash
andrea:x:502:100::/home/andrea:/bin/bash
testperson:x:503:100::/home/testperson:/bin/bash
gaston# ls /home
.  ..  andrea  arnold  cvsroot  t345-1.33  willemer
gaston# mkdir /home/testperson
gaston# chown testperson /home/testperson
```

Durch useradd wird der Eintrag in der **/etc/passwd** generiert. Das zeigt die Ausgabe der Datei. Das Kommentarfeld und das Passwort lässt das Programm frei. Das Heimatverzeichnis des Benutzers wird zwar in der Datei **/etc/passwd** benannt, muss aber erst noch angelegt werden. Solche Tools sind durchaus praktisch, man sollte aber genau wissen, wo sie ihre

Grenzen haben und wann man von Hand nacharbeiten muss. Analog gibt es auch kleine Programme, die eine Änderung oder das Austragen des Benutzers durchführen.

Mit allen Administrationstools kann man auch Benutzer verwalten. Im Allgemeinen leisten sie mehr als die oben genannten Skripten.

Probleme beim Anlegen des Verzeichnisses

Wenn Sie plötzlich Probleme haben, im Verzeichnis **/home** einen Benutzer anzulegen, und dies auch nicht durch das Administrationstool möglich ist, kann es sein, dass ein Automount-Dämon (siehe S. 502) das Verzeichnis **/home** verwaltet. In diesem Fall ist zu prüfen, ob die Heimatverzeichnisse im Netz per Automount verwaltet werden. Falls das nicht beabsichtigt ist, muss man diesen Dämon deinstallieren, bevor man im Verzeichnis **/home** Benutzerverzeichnisse anlegt.

5.5.5 Benutzer-Konfigurationsdateien

Die Datei **/etc/profile** wird beim Einloggen jedes Benutzers gestartet, der nicht die C-Shell verwendet. Hier können Umgebungsvariablen gesetzt und Programme eingetragen werden, die von jedem Anwender nach dem Einloggen ausgeführt werden sollen. Die Datei **/etc/profile** wird vom Administrator gepflegt. Normale Benutzer haben darauf keinen Zugriff. Bei Benutzern, die die C-Shell verwenden, wird die Datei **/etc/profile** nicht ausgeführt. Stattdessen heißt die entsprechende Datei **/etc/csh.cshrc**. Die Unterscheidung ist insofern sinnvoll, als die C-Shell eine andere Syntax für das Setzen von Variablen hat. In diesen beiden Dateien können Einstellungen vorgenommen werden, die für alle Benutzer gelten sollen.

Lokale Dateien

Nachdem die Datei **/etc/profile** ausgeführt worden ist, wird die lokale Datei **.profile** ausgeführt. Diese liegt im Heimatverzeichnis des Benutzers und kann von ihm frei verändert werden. Hier kann jeder Anwender seine eigenen Umgebungen definieren oder Programme starten lassen. Arbeitet der Anwender mit der C-Shell, wird die Datei **.cshrc** abgearbeitet.

Shell-Startdateien

Während die Datei **.profile** beim Einloggen gestartet wird, werden die so genannten rc-Dateien der Shells immer dann ausgeführt, wenn eine neue Shell gestartet wird. Bei der bash (Bourne Again Shell) heißt die zentrale Startdatei **/etc/bash.bashrc**, während die lokale Datei im Heimatverzeichnis des Benutzers unter dem Namen **.bashrc** zu finden ist. Beide Dateien werden nacheinander ausgeführt, sobald eine neue Shell gestartet wird. Es reicht also aus, das Kommando bash einzugeben. Dagegen wird die Datei **.profile** nur ausgeführt, wenn sich der Anwender einloggt. Um das auszu-

probieren, können Sie ja eine Zeile mit einem echo-Befehl in die beiden Dateien schreiben und beobachten, wann die Meldungen erscheinen.

Die folgenden Einstellungen und Befehle werden fast immer im Zusammenhang mit Startup-Dateien eingesetzt.

ulimit

Mit dem Befehl `ulimit` können den Anwendern Beschränkungen auferlegt werden, die die Größen von einigen Ressourcen betreffen. Der Befehl würde sich typischerweise in der Datei **/etc/profile** finden. Die Optionen von `ulimit` sind in Tabelle 5.3 aufgeführt.

Option	Wirkung
-a	Anzeige aller aktuellen Limits
-c	Maximale Größe eines Coredumps
-d	Maximale Größe des Datensegments eines Prozesses
-f	Maximale Größe einer Datei
-n	Maximale Anzahl offener Dateien (nicht bei allen Systemen)
-s	Maximale Größe des Stacks eines Prozesses
-t	Maximale CPU-Zeit in Sekunden
-v	Maximale Größe des virtuellen Speichers

Tabelle 5.3 Optionen von ulimit

Die durch `ulimit` vorgegebenen Grenzen können natürlich vom Anwender wieder zurückgesetzt werden. Durch diesen Befehl gibt der Administrator aber schon einmal eine Grundpolitik vor.

ulimit vs. quota

Die Größe einer Datei zu beschränken, um den Plattenplatz zu limitieren, dürfte nicht der richtige Weg sein, da viele kleine Dateien genauso viel Platz einnehmen können wie eine große. Ein solches Problem löst man besser mit Quotas (siehe S. 305). Einen Sammler von Musik- oder Filmdateien kann aber die Beschränkung der Größe daran erinnern, dass eine solche Sammeltätigkeit nicht im Interesse der Firma ist.

Informieren!

Die hier festgelegten Einschränkungen sollten allgemein bekannt gemacht werden. Im Allgemeinen wird das Überschreiten der Limits zu einem Programmabbruch führen, und die Ursache kann gegebenfalls schwer zu ermitteln sein.

umask

Der Befehl umask findet sich fast immer in mindestens einer der Startdateien. Damit wird voreingestellt, welche Rechte neu angelegte Dateien haben sollen. Nicht jeder Anwender denkt daran, dass eine neu angelegte Datei vor Veränderungen durch boshafte Mitbenutzer geschützt sein sollte.

umask verhindert zu großzügige Rechte

Der Befehl umask setzt eine Maske auf diejenigen Rechte, die bei der Erzeugung einer Datei nicht gesetzt werden. Ein typischer Wert ist 022. Das bedeutet, dass die Gruppe und die Welt keine Schreibrechte auf die Dateien und Verzeichnisse haben. Jemand, der Sorge hat, ausgespäht zu werden, könnte 026 als Parameter für umask verwenden. Dann sind seine Dateien für die Gruppe noch lesbar, für die Welt aber weder les- noch schreibbar. Letztlich kann der Anwender natürlich die Rechte nach ihrer Erzeugung mit dem Befehl chmod beliebig setzen. Allerdings weiß er dann auch, was er tut. Der Befehl umask dient also in erster Linie dazu, den Anfänger davor zu schützen, dass er versehentlich Dateien anlegt, die dann von aller Welt verändert werden können.

Umgebungsvariablen

Die Vorbelegung von Umgebungsvariablen, die für alle Benutzer der Maschine gelten sollen, kann ebenfalls in **/etc/profile** erfolgen.

PATH

Eine der wichtigsten ist die Variable PATH, die beschreibt, wo die startbaren Programme zu finden sind. Einige Anwender legen sich gern eigene Skripten oder selbst geschriebene Programme in ein eigenes Verzeichnis. Um diese Befehle ohne explizite Pfadangaben verwenden zu können, würde man die lokale Datei **.profile** durch den folgenden Befehl ergänzen:

```
export PATH=$PATH:~/bin
```

Andere Einstellungen, wie der Standardprinter oder Einstellungen zur grafischen Umgebung können hier systemweit vorgegeben werden. Aber auch Variablen, die als Schalter für Anwendungssoftware dienen, können hier definiert werden.

5.5.6 Verzeichnisprototyp: /etc/skel

Das Heimatverzeichnis eines neuen Benutzers ist typischerweise nicht völlig leer, sondern enthält diverse rc-Dateien und Konfigurationsdateien. Viele dieser Dateien beginnen mit einem Punkt, damit der Benutzer sie nicht versehentlich durch rm * löschen kann. Sie sind auch für ls unsichtbar, sofern nicht die Option -a verwendet wird.

Damit sie nicht für jeden Benutzer mühselig neu erstellt werden müssen, gibt es unter **/etc** oder **/usr/share** ein Verzeichnis namens **skel**, dessen Inhalt man einfach in das neu angelegte Verzeichnis kopieren kann. So würde das Anlegen eines neuen Benutzerverzeichnisses durch folgende Befehle erfolgen:

Verzeichnis skel

```
gaston# mkdir /home/meier
gaston# chown meier /home/meier
gaston# chgrp projekt_a  /home/meier
gaston# cp /etc/skel/.??* /home/meier
gaston# cd /home/meier
gaston# chmod 644 .??*
gaston# chown meier .??*
gaston# chgrp projekt_a .??*
```

Der Grund für die Maske .??* ist, dass verhindert werden muss, dass das Verzeichnis .. ebenfalls dem Benutzer meier zugeordnet wird. Die beiden Fragezeichen gewährleisten, dass nach dem Punkt noch mindestens zwei Zeichen folgen. Das trifft für die Dateien in **/etc/skel** normalerweise zu. Alternativ könnte auch .[a-zA-Z]* als Maske verwendet werden. Diese Maske gewährleistet, dass nach dem Punkt ein Buchstabe folgt.

Maske gegen ..

5.5.7 Gruppenverwaltung

Eine Gruppe ist eine Menge von Benutzern. Durch die Gruppe ergibt sich die Möglichkeit, die Berechtigungen von Dateien so zu wählen, dass ein Team gemeinsam auf Ressourcen zugreifen kann, ohne dass die Daten gleich allen Benutzern öffentlich angeboten werden.

Die Verfahrensweise ist simpel. Die Gruppen werden in der Datei **group** im Verzeichnis **/etc** verwaltet. Hinter dem Gruppennamen stehen die Benutzerkennungen der Mitglieder der Gruppe. Eine Datei, die nur für die Gruppe gedacht ist, wird durch den Befehl `chgrp` zum Eigentum der Gruppe. Mit dem Befehl `chmod` werden die Zugriffsrechte der Gruppe gesetzt.

/etc/group definiert Gruppen

Jeder Benutzer gehört zu einer Standardgruppe, die in der **passwd**-Datei festgelegt wird. Das bedeutet, dass Dateien, die er erzeugt, automatisch dieser Gruppe zugeordnet werden. Der Benutzer kann aber in beliebig vielen anderen Gruppen eingetragen sein. Dazu wird der Administrator root in der Datei **/etc/group** seinen Namen an die Liste der Mitglieder der jeweiligen Gruppe hängen:

```
dialout:x:16:root,arnold,andrea
prog:x:101:arnold,ralf
```

Zur Gruppe dialout gehören root, arnold und andrea, zur Gruppe prog arnold und ralf. An der dritten Stelle steht die eindeutige Nummer der Gruppe. Die Definition eigener Gruppen sollte man bei 101 beginnen. Bis hierher reichen die Gemeinsamkeiten der Systeme, die immerhin durchaus eine klar geregelte Gruppenbildung ermöglichen.

Gruppenpasswort

Wie in der Datei **/etc/passwd** ist in der zweiten Spalte der **group**-Datei Platz für ein Passwort. Einige Systeme verfügen dennoch über keinen Befehl, um dieses Passwort zu setzen. Dieses Passwort ist normalerweise auch gar nicht erforderlich, solange der Administrator root den Zugang zu den Gruppen regelt. Einige Systeme erlauben aber ein Passwort. Zur Verwaltung wird unter Linux beispielsweise der Befehl `gpasswd` verwendet. Der Befehl `gpasswd` ermöglicht es root mit der Option `-A`, einen Gruppenadministrator für jede Gruppe festzulegen, der dann ebenfalls mit dem Befehl `gpasswd` Mitglieder hinzufügen oder austragen kann. Dadurch kann der Gruppenchef die Gruppe verwalten, ohne dass er Administrationsrechte bekommt. Alternativ dazu wird für die Verwaltung der Gruppenpasswörter der normale Befehl `passwd` mit der Option `-g` verwendet.

Gruppe wechseln

Einige Systeme verfügen über den Befehl `newgrp`. Damit kann ein Anwender für den Rest der Sitzung die Gruppe wechseln. Die Syntax des Befehls ähnelt dem Kommando su (siehe S. 274). Sofern der Benutzer in der Zielgruppe eingetragen ist, wird die reale Gruppe gewechselt. Danach werden also alle von ihm angelegten Dateien automatisch der neuen Gruppe gehören. Fall er nicht zur Gruppe gehört, diese aber ein Passwort besitzt, wird er danach gefragt. Ansonsten ist ein Wechsel nicht möglich. Leider reagieren nicht alle Systeme gleich. Bei einigen Systemen wird trotz des Eintrags in einer anderen Gruppe nach dem Passwort gefragt, bei anderen ist ein Passwort gar nicht vorgesehen.

5.5.8 Benutzerüberwachung

Das Überwachen von Benutzern hat einen schlechten Beigeschmack. Normalerweise befasst sich ein Administrator nicht sehr gern damit. Allerdings ist es wichtig zu wissen, welche Möglichkeiten existieren, wenn ein Benutzer das in ihn gesetzte Vertrauen missbraucht.

Accounting

Aufzeichnungen

Das Accounting ist das Berechnen der Kosten, die ein Benutzer auf dem System verursacht. In Zeiten, da eine UNIX-Maschine ein Vermögen kos-

tete, lohnte es sich durchaus, den hoch bezahlten Administrator berechnen zu lassen, welchen Kostenanteil welcher Benutzer verursachte. Heute ist der Kostenaspekt so weit zurückgegangen, dass ein Accounting fast nicht mehr durchgeführt wird. In einer Hinsicht ist es aber immer noch interessant: Da dabei alle Aktivitäten der Benutzer protokolliert werden, hat man Informationen zur Hand, wenn auf der Maschine Unregelmäßigkeiten vorkommen.

accton startet und stoppt Accounting

Das Programm accton kontrolliert das Accounting. Wird ihm eine Datei als Argument mitgegeben, startet das Accounting und protokolliert in diese Datei. Als Datei wird üblicherweise **acct** oder **pacct** im Verzeichnis **/var/adm/** verwendet. Die Datei muss beim Aufruf existieren. In der Datei werden die Anzahl der Programmaufrufe, die verbrauchte CPU-Zeit, die I/O-Operationen und die Speicherbenutzung für jeden Anwender protokolliert. Das Accounting wird durch den Aufruf von accton ohne Parameter abgeschaltet.

/var/adm/wtmp

Um die Verbindungszeiten zu protokollieren, gibt es die Datei **wtmp** im Verzeichnis **/var/adm**. Sie wird von init und login gefüllt und kann recht schnell recht groß werden. Darum ist es sinnvoll, von Zeit zu Zeit zu kontrollieren, ob die Datei nicht gestutzt werden sollte (siehe S. 403). Braucht man die Verbindungsdaten nicht, kann man sie einfach löschen. Existiert die Datei nicht, wird sie auch nicht von init oder login gefüllt. Zum Auswerten der Datei zum Zwecke des Accounting dient das Programm ac. Mit der Option -u wertet das Programm nach dem Benutzer, mit der Option -d nach dem Datum aus.

who und finger

Übersicht über aktuelle Aktivitäten

who zeigt an, welche Benutzer an welchen Terminals angemeldet sind. finger zeigt alle angemeldeten Benutzer. Für jeden wird angezeigt, seit wann er angemeldet ist und seit wann keine Aktivität mehr feststellbar ist. Auf manchen Systemen gibt es den Befehl whodo. Damit kann man sehen, welches Programm welcher Benutzer aktuell gestartet hat.

```
gaston > who
arnold   pts/0        May 13 08:21
arnold   pts/3        May 13 09:39
arnold   pts/4        May 13 09:53
gaston > finger
Login      Name               Tty      Idle  Login Time   Where
arnold     Arnold Willemer    pts/0    1:52     Thu 08:21
arnold     Arnold Willemer   *pts/3       7     Thu 09:39
```

```
arnold     Arnold Willemer   *pts/4        -     Thu 09:53
gaston >
```

/var/run/utmp

`finger` und `who` werten die Datei **utmp** aus, die sich normalerweise in **/var/run** oder in **/var/log** befindet. Hier sollen sich alle für das Einloggen zuständigen Programme eintragen. Allerdings gibt es da Unterschiede in der Interpretation. So nimmt `xterm`[13] einen Eintrag vor, `konsole`, die xterm-Variante von KDE, nimmt dagegen keinen Eintrag vor. Die KDE-Entwickler begründet dies damit, dass es sich bei einer `xterm`-Sitzung nicht um einen weiteren Benutzer des Systems handelte.

5.5.9 Kurzfristiger Benutzerwechsel: su

su wechselt die Benutzeridentität

Obwohl der Befehl su (set user) oft als Abkürzung für »superuser« bezeichnet wird, dient er dazu, während der aktuellen Terminalsitzung die eigene Identität gegen eine beliebige andere zu tauschen. Wenn Sie su ohne Parameter aufrufen, werden Sie nach dem Administrationspasswort gefragt und mutieren zum Benutzer root. Der Wechsel zu root wird normalerweise protokolliert.

Administratoren melden sich übrigens selten als root am Terminal an. Typischerweise verwenden sie so weit möglich einen gewöhnlichen Account. Erst wenn sie die Macht des Superusers brauchen, loggen sie sich per su ein, erledigen die anstehende Aufgabe und loggen sich sofort wieder aus.

su - bewirkt ein komplettes Login

Beim einfachen Aufruf von su werden die **profile**-Skripten nicht durchlaufen und das Arbeitsverzeichnis wird nicht gewechselt. Als Ergebnis hat man zwar root-Rechte, aber die Umgebungsvariablen für die Werkzeuge des root sind nicht gesetzt. Insbesondere stimmt die Variable PATH nicht, sodass die Programme im Verzeichnis **sbin** beispielsweise nicht gefunden werden. Will man sich vollständig als root anmelden, verwendet man su -. Dann wird ein regulärer Login vollzogen.

Benutzer ohne Anmeldemöglichkeit

Man kann bei su auch einen anderen Benutzernamen als Parameter angeben und wechselt dann seine Identität. Normale Benutzer müssen dann natürlich das Passwort des Zielbenutzers angeben. Nur root muss das nicht. Auf diese Weise ist es möglich, Benutzer anzulegen, die kein Passwort besitzen. Diese können sich zwar nicht am Terminal anmelden, aber root kann mit su deren Identität annehmen, da er kein Passwort braucht. Ein Beispiel ist der Benutzer news, der für die Administration des News-

13 Das Programm xterm erzeugt unter der grafischen Oberfläche X11 ein virtuelles Terminal. Im Gegensatz zu realen Terminals ist es nicht ungewöhnlich, dass ein Benutzer mehrere xterm-Fenster gestartet hat.

groupservers (siehe S. 551) gebraucht wird. Dieser Benutzer kann einen Stern als Passwort bekommen. Auf diese Weise kann sich niemand über diesen Account von außen anmelden. Die Newsgroup wird normalerweise sowieso vom Systemadministrator gepflegt. Er kann sich leicht per `su news` anmelden, ohne dass ein Passwort existiert.

Befehl	Wirkung
su	root-Login ohne Umgebungswechsel
su -	root-Login inklusive Umgebungswechsel
su - *username*	Login als anderer Benutzer

Tabelle 5.4 Varianten des Aufrufs von su

5.5.10 Administrationsaufgaben starten: sudo

Es gibt diverse Aufgaben, für die man die Berechtigung von root braucht, die aber durchaus auch in die Hände einzelner Benutzer gelegt werden können. Dennoch möchte der Administrator diesen Benutzern deswegen nicht gleich das Passwort von root verraten. Diese Lücke schließt das Programm sudo. Damit ist es möglich, Programmaufrufe festzulegen, die bestimmte Benutzer unter root-Rechten ausführen dürfen.

Damit ein solcher Programmaufruf privilegiert abläuft, stellt der Aufrufer der Zeile einfach das Kommando sudo voran. Das System fragt mit einem Passwort nach, ob der berechtigte Benutzer wirklich am Terminal sitzt. Das nun geforderte Passwort ist also nicht das des Superusers root, sondern das des Anwenders, der gerade eingeloggt ist. Stimmt die Berechtigung, wird die Zeile anschließend so ausgeführt, als hätte sie der Administrator ausgeführt. Jede Benutzung von sudo wird exakt protokolliert.

/etc/sudoers wird mit visudo editiert

Vor dem Aufruf von sudo muss festgelegt werden, wer welche Befehle mit root-Rechten ausführen darf. Die Konfigurationsdatei **/etc/sudoers** enthält die Information, welche Benutzer den Befehl sudo verwenden dürfen. Sie wird nicht direkt, sondern von root mit dem Programm `visudo` editiert. Die Datei **sudoers** wird in mehrere Gruppen eingeteilt. Zu Anfang können Alias-Spezifikationen vorgenommen werden. Diese haben den Zweck, kompliziertere Rechte einfacher zu beschreiben. Als Benutzerprivilegien ist von vornherein eingetragen, dass root auf allen Rechnern alles darf. Als Beispiel ist hier hinzugekommen, dass der Benutzer arnold seine Datensicherung auf gaston mit dem Skript `cddasi` ausführen darf. Dies könnte beispielsweise erforderlich sein, weil arnold ansonsten keinen Zugriff auf den CD-Brenner bekommt.

```
# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL) ALL
arnold  gaston = /home/arnold/bin/cddasi
```

Da das Skript cddasi per cdrecord (siehe S. 313) auf den CD-Brenner zugreift, braucht es die root-Rechte. Sie könnten das Problem auch lösen, indem Sie die Rechte der Gerätedatei des Brenners mit chmod so einstellen, dass der Brenner allen Anwendern oder einer bestimmten Gruppe von Anwendern freigegeben wird. Damit ist das Gerät zu jeder Verwendung freigegeben.

Sie könnten auch den in den Manpages von cdrecord vorgeschlagenen Weg gehen und für den Befehl cdrecord das SUID-Bit (siehe S. 111) setzen:

```
chmod 4755 cdrecord
```

Alternative User-ID-Bit

Damit würde immerhin erreicht, dass der Zugriff auf den Brenner nur mit Hilfe der Anwendung cdrecord möglich ist. Der Zugriff auf das Programm cdrecord ist nun für jeden Anwender frei. Die Freigabe per sudo ermöglicht aber sogar die Einschränkung auf das Skript cddasi. Der Anwender darf mit dem Brenner also ausschließlich seine Datensicherung durchführen. Er kann beispielsweise keine Audio-CDs brennen. Des Weiteren können Sie mit sudo dieses Recht einzelnen aufgeführten Anwendern gewähren. Und zu guter Letzt wird die Verwendung des Befehls auch noch protokolliert.

Will der Benutzer arnold das Skript cddasi ausführen, gibt er das Kommando mit vorangestelltem sudo ein:

```
arnold@gaston> sudo /home/arnold/bin/cddasi
```

sudo fragt nun nach einem Passwort. Es wird aber nicht das Passwort von root verlangt, sondern das des angemeldeten Benutzers, also hier das von arnold. Damit wird vermieden, dass ein Fremder die nicht geschlossene Konsole bei kurzfristiger Abwesenheit für privilegierte Aufgaben nutzen kann. Tatsächlich können Sie auch das Passwort von root eingeben. Das ist vor allem für den Administrator nützlich, der aus seinem normalen

Account heraus eine Administrationstätigkeit ausführen möchte, die nur eine Zeile umfasst.

Protokoll in der syslog-Datei

In der Protokolldatei des syslog-Dämons (meist **/var/log/messages**, siehe S. 400) wird protokolliert, dass der Benutzer arnold das Skript als root ausgeführt hat. Hier sind zwei Protokolleinträge dargestellt. Der erste wurde ohne Berechtigung durchgeführt. Im zweiten wurde der Eintrag in der Datei **sudoers** eingetragen:[14]

```
Mar  9 05:45:30 gaston sudo:   arnold : command not allowed ;
            TTY=pts/5 ; PWD=/home/arnold ; USER=root ;
            COMMAND=/home/arnold/bin/cddasi
Mar  9 05:46:18 gaston sudo:   arnold : TTY=pts/5 ;
            PWD=/home/arnold ; USER=root ; COMMAND=bin/cddasi
```

Makros

Wie oben schon erwähnt wurde, kann man in der Datei **sudoers** auch mit Makros arbeiten. Dabei können mehrere Benutzer, Maschinen und Kommandomakros gebildet und zur Rechtevergabe verwendet werden:

```
User_Alias     DEPUTIES = wim, wum, wendelin
Host_Alias     SERVERS = gaston, asterix, silver
Cmnd_Alias     HALT = /usr/sbin/halt, /usr/sbin/fasthalt
DEPUTIES SERVERS = HALT
```

Hier können alle Benutzer, die unter DEPUTIES aufgeführt sind, alle Server herunterfahren.

sudoers im Netzwerk

Vielleicht ist es irritierend, dass in der Datei auch die root-Rechte fremder Rechner festgelegt werden. Es ist natürlich nicht möglich, auf einem Rechner festzulegen, welche Rechte ein Anwender auf einem andern Rechner bekommt. Jeder Rechner kann seine root-Rechte nur selbst vergeben. Der Grund, dass hier auch Hostnamen genannt werden, besteht darin, dass die Datei **sudoers** per NIS (siehe S. 467) verteilt werden kann und dann jeder Rechner anhand des Rechnernamens nur den Teil übernimmt, der für ihn gedacht ist.

Liste der Kommandos

Durch den Aufruf von `sudo -l` kann ein Benutzer sehen, welche Kommandos er unter `sudo` ausführen kann.

Das Programm `sudo` hat noch vielfältige Optionen. So kann für einzelne Kommandos oder Gruppen die Passworteingabe abgeschaltet werden. Es kann konfiguriert werden, dass das Protokoll per Mail versandt wird. Diese

14 Aus satztechnischen Gründen wurden die Zeilen umbrochen.

Informationen findet man in den umfangreichen Manpages von sudo und **sudoers**.

5.5.11 Pseudobenutzer zum Shutdown

Bei einer Workstation, also einem Einzelplatzrechner, kann es wichtig sein, dass auch nicht privilegierte Anwender die Maschine herunterfahren können. Der Vorgang sollte dann so einfach wie möglich sein, denn je umständlicher das Herunterfahren ist, desto eher besteht die Neigung, die Maschine einfach abzuschalten.

Der Benutzer shutdown

Eine elegante Lösung ist es, einen Benutzer anzulegen, dessen einziger Zweck das Herunterfahren ist. Dieser Pseudobenutzer kann beispielsweise shutdown heißen, und damit es sich jeder merken kann, bekommt er auch das Passwort shutdown. Will nun ein Workstation-Benutzer seine Maschine herunterfahren, meldet er sich einfach ab und als Benutzer shutdown neu an, und die Maschine fährt herunter.

Nachdem der Benutzer shutdown angelegt worden ist, wird in die Datei **.profile** im Heimatverzeichnis ganz zu Anfang eine Zeile geschrieben:

```
sudo shutdown -h 0
```

sudo shutdown

Damit wird direkt nach dem Anmelden der Befehl shutdown aufgerufen. Da dies nur dem Administrator root erlaubt ist, wird sudo verwendet. Nun muss der Benutzer shutdown noch einen Eintrag in die Datei **/etc/sudoers** bekommen, der es ihm erlaubt, den besagten shutdown auszuführen:

```
shutdown    localhost=NOPASSWD:/sbin/shutdown -h 0
```

Das Attribut NOPASSWD ist wichtig, da die Ausführung von **.profile** nicht interaktiv ist. Damit wird für das folgende Kommando kein Passwort abgefragt.

Alternative

Alternativ kann man für den Benutzer shutdown in der Datei **/etc/passwd** als Shell das Programm shutdown eintragen. Dazu muss der komplette Pfad von shutdown angegeben werden, der durch das Kommando which leicht ermittelt werden kann. Beim Anmelden als Benutzer shutdown wird die Shell gestartet, die in diesem Fall das Kommando zum Herunterfahren des Rechners ist. Einen kleinen Haken gibt es noch: Da shutdown immer noch ein privilegiertes Kommando ist, muss auch der Benutzer shutdown ein privilegierter Benutzer sein. Man erreicht dies am einfachsten, indem man als User-ID wie bei root eine 0 einträgt.

```
shutdown:clShgL2tYojh.:0:0::/tmp:/sbin/shutdown
```

Teil 3
Netzwerk

6 Netzwerk

Die Zeit der zentralen Großrechner mit den passiven Terminals klingt aus. Gerade UNIX findet heute seinen Haupteinsatz als leistungsfähiges und zuverlässiges Serversystem in einer Netzwerkumgebung.

Ein Netzwerk ist zunächst nichts anderes als die Verbindung mehrerer Computer durch ein Kabel, über das sie miteinander kommunizieren können. In Kombination mit geeigneter Software können Sie von Ihrem Arbeitsplatz aus Ressourcen anderer Computer nutzen. Am bekanntesten dürfte die Möglichkeit sein, gemeinsam auf eine Festplatte oder einen Drucker zugreifen zu können. Das Motiv ist nicht allein Sparsamkeit. Netzwerke ermöglichen das Teilen von Daten und gewährleisten so höchste Aktualität.

Server

Auf den ersten Blick scheint es, als würde ein Netzwerk Objekte wie Drucker oder Festplatten zur Verfügung stellen. Bei näherer Betrachtung handelt es sich aber um *Dienste* (engl. *services*). So werden die Druckdaten nicht an den Netzwerkdrucker selbst gesendet, sondern an einen Prozess. Dieser veranlasst dann als Dienstleister (engl. *server*) den Druck. Da die Druckdaten nicht direkt an den Drucker gehen, kann der Prozess vor dem Druck die Berechtigungen prüfen. Der teure Farblaserdrucker der Werbeabteilung soll beispielsweise nicht allen Angestellten zur Verfügung stehen, um ihre Urlaubsfotos auszudrucken.

Internet

Das größte aller Netze ist das Internet. Hier gibt es Abertausende von Dienstanbietern. Neben dem World Wide Web bietet es eine schnelle, kostengünstige Kommunikation per E-Mail. Darum ist eine Firma in der Regel daran interessiert, vielen Arbeitsplätzen einen Zugang zu dieser Informationsquelle zu ermöglichen. Allerdings besteht auch das Risiko, dass das eigene Netz über das Internet ausgespäht oder angegriffen wird.

TCP/IP

Jedes Netzwerk braucht ein Protokoll, in dem festgelegt wird, welcher Teil der Nachricht die Adresse, der Absender, eine Kontrollinformation bzw. Daten ist. Inzwischen ist TCP/IP (Transmission Control Protocol/Internet Protocol) das unangefochten wichtigste Protokoll. TCP/IP ist für UNIX nicht nur das Zugangsprotokoll zum Internet, sondern auch die Basis für lokale Netzwerke. Dabei spielen die vom PC her bekannten Festplatten- und Druckserver keine so große Rolle wie das Verteilen von Anwendungen in Client-Server-Architekturen. In UNIX-Netzen geht es mehr um das

Starten von Prozessen auf entfernten Maschinen oder um das Verteilen von Prozessen auf mehrere Maschinen.

Heterogene Netze

In TCP/IP-Umgebungen finden Sie selten ausschließlich UNIX-Maschinen. So haben Sie immer wieder damit zu tun, auch MS Windows- oder Mac OS-Rechner zur Zusammenarbeit zu bewegen. Sie werden oft als Front-End benutzt, während das Back-End unter UNIX läuft.

6.1 Client-Server-Architekturen

Clients und Server sind Prozesse

Der Zweck eines Netzes ist es, einen Prozess auf einem Rechner mit Informationen zu versorgen, die auf einem anderen Computer vorhanden sind. Der Auslöser ist also immer ein Prozess, der eine Information anfragt und auf die Antwort wartet. Einen »Anfrager« nennt man Client, einen »Antworter« nennt man Server. Ein Server ist also in erster Linie ein Prozess und kein Computer. Derselbe Computer kann durchaus gleichzeitig als Client und als Server auftreten, indem er bestimmte Anfragen beantwortet und auf der anderen Seite Anfragen stellt.

Eine Software, die nach dem Client-Server-Prinzip arbeitet, ist auf zwei Seiten aufgeteilt. Sie ist quasi irgendwo »durchgesägt«. Das Front-End läuft auf dem Arbeitsplatzrechner, und ein anderer Teil arbeitet auf einem anderen, typischerweise zentralen Rechner, der dann auch meist als Server bezeichnet wird, weil auf ihm die Serverprozesse laufen.

6.1.1 Ethernet als Verkabelungsbeispiel

Bei der Verkabelung wird heutzutage meistens Ethernet eingesetzt. Diese Technik ist für lokale Netzwerke inzwischen nahezu konkurrenzlos. In seiner einfachsten Form besteht ein Ethernet aus einem Koaxialkabel[1] und je einem Widerstand an jedem Ende. Ein Computer wird mit dem Kabel durch einen Abgriff verbunden. Bei dem in kleinen Netzwerken teilweise noch verwendeten Koaxialkabel RG58 ist das ein auf dem BNC-Stecker basierendes T-Stück. Dieses T-Stück steckt direkt auf einem Transceiver am Ethernet-Controller.

Twisted Pair

Inzwischen verwendet man auch in kleinen Netzen mehr und mehr eine Twisted-Pair-Verkabelung. Das Kabel besteht, wie der Name schon sagt, aus verdrillten Drähten. Genauer gesagt führen in einem Twisted-Pair-Kabel zwei Drähte zum Computer und zwei zurück zum Hub. Ein Hub ist das Rückgrat des Netzes. Denn obwohl das Ganze so aus-

1 Ein Koaxialkabel besteht aus einem Draht, der von einer Abschirmung umgeben ist. Antennenkabel sind typischerweise auch Koaxialkabel.

sieht, als wäre die Verkabelung sternförmig angeordnet, wird vom Hub jeder Abgang schleifenartig zu einem langen Draht verkoppelt, sobald das Kabel auf der anderen Seite durch den Computer verbunden wird. Der Vorteil von Twisted Pair liegt vor allem in der Robustheit. Das Koaxialkabel RG58 neigt an den Anschlüssen zu Wackelkontakten, insbesondere bei den Abschlusswiderständen an den Kabelenden.

Jeder Ethernet-Controller hat seine eigene, weltweit eindeutige Nummer, die 48 Bit groß ist. Diese Nummer ist meist in das ROM des Controllers eingebrannt.[2] Der Ethernet-Controller lauscht die ganze Zeit am Kabel, und sobald ein Paket kommt, das die Nummer des Controllers als Adresse hat, holt er es in seinen Speicher und gibt es an das Betriebssystem weiter.

Senden

Will der Controller selbst Daten senden, packt er sie in Pakete zu je maximal 1500 Byte und setzt die Ethernet-Adresse des Zielcomputers, gefolgt von der eigenen Adresse als Absender hinein. Zu guter Letzt enthält jedes Paket eine Prüfsumme, die CRC (Cyclic Redundancy Check), und schon ist das Paket sendebereit.

Zum eigentlichen Senden wartet der Controller ab, bis sich auf dem Netzwerk eine Sendepause ergibt. Nun beginnt er damit, ein paar alternierende Folgen von Einsen und Nullen dem Paket vorauszusenden, und lauscht, ob diese von einem anderen Controller gestört werden, der vielleicht im gleichen Moment zu senden versucht. Ist das der Fall, versuchen es beide Controller später noch einmal. Geht alles glatt, setzt der Controller Paket um Paket auf diese Weise hinterher, bis seine Daten übermittelt sind.

CSMA/CD

Dieses Verfahren, mit dem Kabel umzugehen, nennt sich *Carrier Sense Multiple Access Bus with Collision Detect*, abgekürzt CSMA/CD. Es bedeutet so viel wie »Abtasten des Mediums mit mehrfachem Zugriff bei Kollisionserkennung«. Die Kollision von Paketen gehört also zum Protokoll und ist kein Drama. Allerdings häufen sich die Kollisionen bei höherer Netzlast und können dann zu Problemen mit dem Durchsatz führen.

Wenn Sie sich im Detail dafür interessieren, was wirklich auf Controller-Ebene passiert, wie die Pakete genau aussehen und wie die Prüfsummen arbeiten, sollten Sie die Werke von Tanenbaum[3] und Comer[4] lesen.

2 Bei Sun-Maschinen befinden sie sich in einem batteriegepufferten RAM. Darum empfielt es sich, sich diese Nummer zu notieren. Die Batterie könnte ja mal schlappmachen. Die Nummer wird beim Booten angezeigt.

3 Andrew S. Tanenbaum: Computer Networks. Prentice Hall, Englewood Cliffs, 1987.

4 Douglas E. Comer: Internetworking with TCP/IP. Prentice Hall, 2nd ed., 1991

6.1.2 Die Pseudoschnittstelle loopback

TCP/IP wird nicht nur verwendet, um mit anderen Rechnern Kontakt aufzunehmen. Manchmal führt der Rechner auch Selbstgespräche. So arbeitet die grafische Oberfläche von UNIX über TCP/IP. Dabei erfolgt die Kommunikation auf modernen Workstations vom eigenen Rechner zum eigenen Display. Bei Maschinen, die überhaupt keine Netzwerkanschlüsse haben, wird eine Schnittstelle gebraucht, über die der Rechner mit sich selbst Kontakt aufnehmen kann. Diese Schnittstelle nennt sich loopback, weil sie wie eine Schleife auf den Rechner selbst zurückführt.

6.1.3 Pakete in Paketen

ARP

Auf der Basis von Ethernet-Paketen werden TCP/IP-Pakete versandt. Als Adresse werden IP-Nummern und der Port verwendet (siehe unten). Dabei kennt ein Ethernet-Controller nur Ethernet-Nummern. Die IP-Nummern müssen also auf die Ethernet-Nummern abgebildet werden. Das Protokoll, das diese Abbildung steuert, nennt sich ARP (Address Resolution Protocol) und ist eigentlich unterhalb des TCP/IP anzusiedeln.

Bearbeiten der ARP-Tabelle

Es kann ein Problem auftreten, wenn eine Netzwerkkarte zwischen Rechnern getauscht wurde. Da die IP-Nummer nun unter einer anderen Ethernet-Nummer zu finden ist, ist die ARP-Tabelle, die die Zuordnung von Ethernet-Nummern zu IP-Nummern enthält, nicht mehr gültig. Nach dem Tausch läuft diese IP-Nummer bei anderen Rechnern im Netz noch unter einer anderen Ethernet-Nummer oder umgekehrt. Die ARP-Tabelle kann unter UNIX mit dem Befehl `arp` bearbeitet werden. Mit der Option -a wird die Tabelle angezeigt, mit -d werden Einträge gelöscht, und mit -s kann ein Eintrag gesetzt werden. Näheres finden Sie in der Manpage von `arp`.

6.2 TCP/IP, der Standard

TCP/IP ist das Standardprotokoll für den Zugriff auf das Netzwerk. Es wurde von der Berkeley Universität seinerzeit für UNIX entwickelt. Diese enge Verbindung merkt man TCP/IP auch in anderen Umgebungen noch an.

6.2.1 Die TCP/IP-Nummer

Die Netzkennung in der IP-Adresse

Die TCP/IP- oder Internet-Nummer ist eine 32-Bit-Zahl, die die Netzwerkschnittstelle eines Computers im Netz eindeutig bestimmt. Der erste Teil der Nummer bezeichnet das Netz, in dem sich der Computer be-

Index

C

D

E

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z