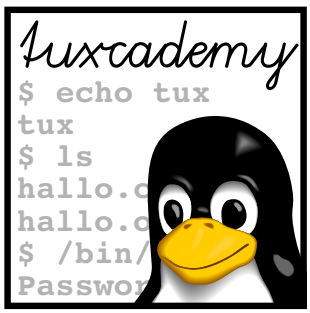
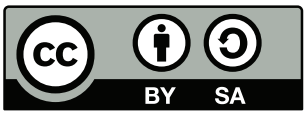
Linux kompakt

Ein Einstieg in Linux-Anwendung und -Administration





tuxcademy – Linux- und Open-Source-Lernunterlagen für alle
www.tuxcademy.org · info@tuxcademy.org

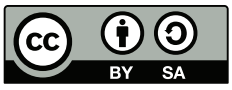
Diese Schulungsunterlage wurde vom Linux Professional Institute *(LPI) im LPI-ATM-Programm zertifiziert. Sie ist inhaltlich und didaktisch zur Vorbereitung auf die LPIC-1-Zertifizierung geeignet.*

Das Linux Professional Institute *empfiehlt keine speziellen Prüfungsvorbereitungsmaterialien oder -techniken – wenden Sie sich für Details an* `info@lpi.org`.

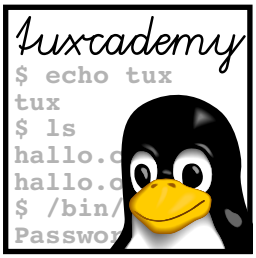
Das tuxcademy-Projekt bietet hochwertige frei verfügbare Schulungsunterlagen zu Linux- und Open-Source-Themen – zum Selbststudium, für Schule, Hochschule, Weiterbildung und Beruf.
Besuchen Sie `https://www.tuxcademy.org/`! Für Fragen und Anregungen stehen wir Ihnen gerne zur Verfügung.

Linux kompakt **Ein Einstieg in Linux-Anwendung und -Administration**

Revision: `lxk1:155fdd73d8ffc3ef:2015-08-20`
`adm1:9f8c99fa2fddd494:2015-08-08` 10–18, 26–27
`adm2:7ff0ace1377051dd:2015-08-20` 20–25
`grd1:a13e1ba7ab759bab:2015-08-04` 1–9, B
`grd2:fbedd62c7f8693fb:2015-08-05` 19
`lxk1:EhLhE5y2iHeik2BEFB7p5A`



Autoren: Tobias Elsner, Thomas Erker, Anselm Lingnau
Technische Redaktion: Anselm Lingnau ⟨*anselm.lingnau@linupfront.de*⟩
Gesetzt in Palatino, Optima und DejaVu Sans Mono



Inhalt

1 Einführung **15**
1.1 Was ist Linux? . . . 16
1.2 Die Geschichte von Linux . . . 16
1.3 Freie Software, »Open Source« und die GPL . . . 18
1.4 Linux – Der Kernel . . . 22
1.5 Die Eigenschaften von Linux . . . 24
1.6 Linux-Distributionen . . . 27

2 Die Bedienung des Linux-Systems **33**
2.1 Anmelden und Abmelden . . . 34
2.2 An- und Ausschalten . . . 36
2.3 Der Systemadministrator . . . 36

3 Keine Angst vor der Shell **41**
3.1 Warum? . . . 42
3.2 Was ist die Shell? . . . 42
3.3 Kommandos . . . 44
3.3.1 Wozu Kommandos? . . . 44
3.3.2 Wie sind Kommandos aufgebaut? . . . 44
3.3.3 Arten von Kommandos . . . 45
3.3.4 Noch mehr Spielregeln . . . 46

4 Hilfe **49**
4.1 Hilfe zur Selbsthilfe . . . 50
4.2 Der `help`-Befehl und die `--help`-Option . . . 50
4.3 Die Handbuchseiten . . . 51
4.3.1 Überblick . . . 51
4.3.2 Struktur . . . 51
4.3.3 Kapitel . . . 52
4.3.4 Handbuchseiten anzeigen . . . 52
4.4 Die Info-Seiten . . . 53
4.5 Die HOWTOs . . . 54
4.6 Weitere Informationsquellen . . . 54

5 Der Editor `vi` **57**
5.1 Editoren . . . 58
5.2 Der Standard – `vi` . . . 58
5.2.1 Überblick . . . 58
5.2.2 Grundlegende Funktionen . . . 59
5.2.3 Erweiterte Funktionen . . . 63
5.3 Andere Editoren . . . 65

6 Dateien: Aufzucht und Pflege 67
6.1 Datei- und Pfadnamen . . . 68
6.1.1 Dateinamen . . . 68
6.1.2 Verzeichnisse . . . 70
6.1.3 Absolute und relative Pfadnamen . . . 70
6.2 Kommandos für Verzeichnisse . . . 71
6.2.1 Das aktuelle Verzeichnis: `cd` & Co. . . . 71
6.2.2 Dateien und Verzeichnisse auflisten – `ls` . . . 72
6.2.3 Verzeichnisse anlegen und löschen: `mkdir` und `rmdir` . . . 74
6.3 Suchmuster für Dateien . . . 75
6.3.1 Einfache Suchmuster . . . 75
6.3.2 Zeichenklassen . . . 77
6.3.3 Geschweifte Klammern . . . 78
6.4 Umgang mit Dateien . . . 79
6.4.1 Kopieren, Verschieben und Löschen – `cp` und Verwandte . . . 79
6.4.2 Dateien verknüpfen – `ln` und `ln -s` . . . 81
6.4.3 Dateiinhalte anzeigen – `more` und `less` . . . 86
6.4.4 Dateien suchen – `find` . . . 86
6.4.5 Dateien schnell finden – `locate` und `slocate` . . . 90

7 Standardkanäle und Filterkommandos 95
7.1 Ein-/Ausgabeumlenkung und Kommandopipelines . . . 96
7.1.1 Die Standardkanäle . . . 96
7.1.2 Standardkanäle umleiten . . . 97
7.1.3 Kommando-Pipelines . . . 101
7.2 Filterkommandos . . . 102
7.3 Dateien lesen und ausgeben . . . 103
7.3.1 Textdateien ausgeben und aneinanderhängen – `cat` und `tac` . . . 103
7.3.2 Anfang und Ende von Dateien – `head` und `tail` . . . 105
7.3.3 Mit der Lupe – `od` und `hexdump` . . . 106
7.4 Textbearbeitung . . . 109
7.4.1 Zeichen für Zeichen – `tr`, `expand` und `unexpand` . . . 109
7.4.2 Zeile für Zeile – `fmt`, `pr` und so weiter . . . 112
7.5 Datenverwaltung . . . 117
7.5.1 Sortierte Dateien – `sort` und `uniq` . . . 117
7.5.2 Spalten und Felder – `cut`, `paste` & Co. . . . 122

8 Mehr über die Shell 129
8.1 `sleep`, `echo` und `date` . . . 130
8.2 Shell-Variable und die Umgebung . . . 131
8.3 Arten von Kommandos – die zweite . . . 133
8.4 Die Shell als komfortables Werkzeug . . . 135
8.5 Kommandos aus einer Datei . . . 138
8.6 Die Shell als Programmiersprache . . . 139
8.7 Vorder- und Hintergrundprozesse . . . 143

9 Das Dateisystem 147
9.1 Begriffe . . . 148
9.2 Dateitypen . . . 148
9.3 Der Linux-Verzeichnisbaum . . . 150
9.4 Verzeichnisbaum und Dateisysteme . . . 158
9.5 Wechselmedien . . . 159

10 Systemadministration 163
10.1 Administration allgemein . . . 164
10.2 Das privilegierte `root`-Konto . . . 164
10.3 Administratorprivilegien erlangen . . . 166
10.4 Distributionsabhängige Administrationswerkzeuge . . . 169

11 Benutzerverwaltung **173**
11.1 Grundlagen . . . 174
11.1.1 Wozu Benutzer? . . . 174
11.1.2 Benutzer und Gruppen . . . 175
11.1.3 »Natürliche Personen« und Pseudobenutzer . . . 177
11.2 Benutzer- und Gruppendaten . . . 178
11.2.1 Die Datei `/etc/passwd` . . . 178
11.2.2 Die Datei `/etc/shadow` . . . 181
11.2.3 Die Datei `/etc/group` . . . 184
11.2.4 Die Datei `/etc/gshadow` . . . 185
11.2.5 Das Kommando `getent` . . . 185
11.3 Benutzerkonten und Gruppeninformationen verwalten . . . 186
11.3.1 Benutzerkonten einrichten . . . 186
11.3.2 Das Kommando `passwd` . . . 188
11.3.3 Benutzerkonten löschen . . . 190
11.3.4 Benutzerkonten und Gruppenzuordnung ändern . . . 190
11.3.5 Die Benutzerdatenbank direkt ändern — `vipw` . . . 191
11.3.6 Anlegen, Ändern und Löschen von Gruppen . . . 191

12 Zugriffsrechte **195**
12.1 Das Linux-Rechtekonzept . . . 196
12.2 Zugriffsrechte auf Dateien und Verzeichnisse . . . 196
12.2.1 Grundlagen . . . 196
12.2.2 Zugriffsrechte anschauen und ändern . . . 197
12.2.3 Dateieigentümer und Gruppe setzen – `chown` und `chgrp` . . . 198
12.2.4 Die *umask* . . . 199
12.3 Zugriffskontrolllisten (ACLs) . . . 201
12.4 Eigentum an Prozessen . . . 202
12.5 Besondere Zugriffsrechte für ausführbare Dateien . . . 202
12.6 Besondere Zugriffsrechte für Verzeichnisse . . . 203
12.7 Dateiattribute . . . 205

13 Prozessverwaltung **209**
13.1 Was ist ein Prozess? . . . 210
13.2 Prozesszustände . . . 211
13.3 Prozessinformationen – `ps` . . . 212
13.4 Prozesse im Baum – `pstree` . . . 213
13.5 Prozesse beeinflussen – `kill` und `killall` . . . 214
13.6 `pgrep` und `pkill` . . . 216
13.7 Prozessprioritäten – `nice` und `renice` . . . 217
13.8 Weitere Befehle zur Prozessverwaltung – `nohup`, `top` . . . 219

14 Platten (und andere Massenspeicher) **221**
14.1 Grundlagen . . . 222
14.2 Bussysteme für Massenspeicher . . . 222
14.3 Partitionierung . . . 225
14.3.1 Grundlagen . . . 225
14.3.2 Die traditionelle Methode (MBR) . . . 226
14.3.3 Die moderne Methode (GPT) . . . 227
14.4 Linux und Massenspeicher . . . 229
14.5 Platten partitionieren . . . 231
14.5.1 Prinzipielles . . . 231
14.5.2 Platten partitionieren mit `fdisk` . . . 233
14.5.3 Platten formatieren mit GNU `parted` . . . 236
14.5.4 `gdisk` . . . 238
14.5.5 Andere Partitionierungsprogramme . . . 238
14.6 Loop-Devices und `kpartx` . . . 239
14.7 Der Logical Volume Manager (LVM) . . . 241

15 Dateisysteme: Aufzucht und Pflege 245
15.1 Linux-Dateisysteme . . . 246
15.1.1 Überblick . . . 246
15.1.2 Die ext-Dateisysteme . . . 249
15.1.3 ReiserFS . . . 257
15.1.4 XFS . . . 259
15.1.5 Btrfs . . . 260
15.1.6 Noch mehr Dateisysteme . . . 262
15.1.7 Auslagerungsspeicher (*swap space*) . . . 263
15.2 Einbinden von Dateisystemen . . . 264
15.2.1 Grundlagen . . . 264
15.2.2 Der mount-Befehl . . . 264
15.2.3 Labels und UUIDs . . . 266
15.3 Das Programm dd . . . 268

16 Linux booten 271
16.1 Grundlagen . . . 272
16.2 GRUB Legacy . . . 275
16.2.1 Grundlagen von GRUB . . . 275
16.2.2 Die Konfiguration von GRUB Legacy . . . 276
16.2.3 Installation von GRUB Legacy . . . 277
16.3 GRUB 2 . . . 278
16.3.1 Sicherheitsaspekte . . . 279
16.4 Kernelparameter . . . 280
16.5 Probleme beim Systemstart . . . 282
16.5.1 Fehlersuche . . . 282
16.5.2 Typische Probleme . . . 282
16.5.3 Rettungssysteme und Live-Distributionen . . . 284

17 System-V-Init und der Init-Prozess 287
17.1 Der Init-Prozess . . . 288
17.2 System-V-Init . . . 288
17.3 Upstart . . . 295
17.4 Herunterfahren des Systems . . . 297

18 Systemd 303
18.1 Überblick . . . 304
18.2 Unit-Dateien . . . 306
18.3 Typen von Units . . . 310
18.4 Abhängigkeiten . . . 311
18.5 Ziele . . . 312
18.6 Das Kommando systemctl . . . 315
18.7 Installation von Units . . . 318

19 Zeitgesteuerte Vorgänge – at und cron 321
19.1 Allgemeines . . . 322
19.2 Einmalige Ausführung von Kommandos . . . 322
19.2.1 at und batch . . . 322
19.2.2 at-Hilfsprogramme . . . 324
19.2.3 Zugangskontrolle . . . 325
19.3 Wiederholte Ausführung von Kommandos . . . 325
19.3.1 Aufgabenlisten für Benutzer . . . 325
19.3.2 Systemweite Aufgabenlisten . . . 327
19.3.3 Zugangskontrolle . . . 328
19.3.4 Das Kommando crontab . . . 328
19.3.5 Anacron . . . 328

20 Systemprotokollierung 333

20.1 Das Problem . . . 334
20.2 Der Syslog-Daemon . . . 334
20.3 Die Protokolldateien . . . 337
20.4 Protokoll des Systemkerns . . . 338
20.5 Erweiterte Möglichkeiten: Rsyslog . . . 339
20.6 Die »nächste Generation«: Syslog-NG . . . 343
20.7 Das Programm `logrotate` . . . 348

21 Systemprotokollierung mit systemd und »dem Journal« 353
21.1 Grundlagen . . . 354
21.2 Systemd und journald . . . 355
21.3 Protokollauswertung . . . 358

22 Grundlagen von TCP/IP 363
22.1 Geschichte und Grundlagen . . . 364
22.1.1 Die Geschichte des Internet . . . 364
22.1.2 Verwaltung des Internet . . . 365
22.2 Technik . . . 366
22.2.1 Überblick . . . 366
22.2.2 Protokolle . . . 367
22.3 TCP/IP . . . 370
22.3.1 Überblick . . . 370
22.3.2 Kommunikation von Ende zu Ende: IP und ICMP . . . 371
22.3.3 Die Basis für Dienste: TCP und UDP . . . 374
22.3.4 Die wichtigsten Anwendungsprotokolle . . . 378
22.4 Adressen, Wegleitung und Subnetting . . . 379
22.4.1 Grundlagen . . . 379
22.4.2 Wegleitung . . . 380
22.4.3 IP-Netzklassen . . . 381
22.4.4 Subnetting . . . 382
22.4.5 Private IP-Adressen . . . 382
22.4.6 Masquerading und Portweiterleitung . . . 383
22.5 IPv6 . . . 384
22.5.1 Überblick . . . 384
22.5.2 IPv6-Adressierung . . . 385

23 Linux-Netzkonfiguration 389
23.1 Netzschnittstellen . . . 390
23.1.1 Hardware und Treiber . . . 390
23.1.2 Netzwerkkarten konfigurieren mit `ifconfig` . . . 391
23.1.3 Wegleitung konfigurieren mit `route` . . . 392
23.1.4 Netzkonfiguration mit `ip` . . . 395
23.2 Dauerhafte Netzkonfiguration . . . 396
23.3 DHCP . . . 398
23.4 IPv6-Konfiguration . . . 400
23.5 Namensauflösung und DNS . . . 401

24 Fehlersuche und Fehlerbehebung im Netz 405
24.1 Einführung . . . 406
24.2 Lokale Probleme . . . 406
24.3 Erreichbarkeit von Stationen prüfen mit `ping` . . . 407
24.4 Wegleitung testen mit `traceroute` und `tracepath` . . . 409
24.5 Dienste überprüfen mit `netstat` und `nmap` . . . 412
24.6 DNS testen mit `host` und `dig` . . . 415
24.7 Andere nützliche Diagnosewerkzeuge . . . 418
24.7.1 `telnet` und `netcat` . . . 418
24.7.2 `tcpdump` . . . 420
24.7.3 `wireshark` . . . 420

25 Die Secure Shell **423**
25.1 Einführung . . . 424
25.2 Anmelden auf entfernten Rechnern mit ssh . . . 424
25.3 Andere nützliche Anwendungen: scp und sftp . . . 428
25.4 Client-Authentisierung über Schlüsselpaare . . . 428
25.5 Portweiterleitung über SSH . . . 431
25.5.1 X11-Weiterleitung . . . 431
25.5.2 Beliebige TCP-Ports weiterleiten . . . 432

26 Paketverwaltung mit Debian-Werkzeugen **437**
26.1 Überblick . . . 438
26.2 Das Fundament: dpkg . . . 438
26.2.1 Debian-Pakete . . . 438
26.2.2 Paketinstallation . . . 439
26.2.3 Pakete löschen . . . 440
26.2.4 Debian-Pakete und ihr Quellcode . . . 441
26.2.5 Informationen über Pakete . . . 442
26.2.6 Verifikation von Paketen . . . 444
26.3 Debian-Paketverwaltung der nächsten Generation . . . 445
26.3.1 APT . . . 445
26.3.2 Paketinstallation mit apt-get . . . 446
26.3.3 Informationen über Pakete . . . 447
26.3.4 aptitude . . . 449
26.4 Integrität von Debian-Paketen . . . 451
26.5 Die debconf-Infrastruktur . . . 452
26.6 alien: Pakete aus fremden Welten . . . 453

27 Paketverwaltung mit RPM & Co. **455**
27.1 Einleitung . . . 456
27.2 Paketverwaltung mit rpm . . . 457
27.2.1 Installation und Update . . . 457
27.2.2 Deinstallation von Paketen . . . 457
27.2.3 Datenbank- und Paketanfragen . . . 458
27.2.4 Verifikation von Paketen . . . 460
27.2.5 Das Programm rpm2cpio . . . 461
27.3 YUM . . . 461
27.3.1 Überblick . . . 461
27.3.2 Paketquellen . . . 461
27.3.3 Pakete installieren und entfernen mit YUM . . . 462
27.3.4 Informationen über Pakete . . . 464
27.3.5 Pakete nur herunterladen . . . 466

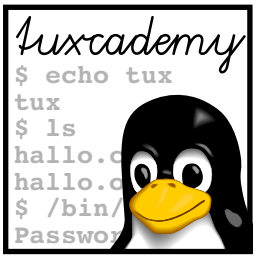
A Musterlösungen **469**

B Beispieldateien **491**

C LPIC-1-Zertifizierung **495**
C.1 Überblick . . . 495
C.2 Prüfung LPI-101 . . . 496
C.3 Prüfung LPI-102 . . . 496
C.4 LPI-Prüfungsziele in dieser Schulungsunterlage . . . 497

D Kommando-Index **509**

Index **515**



Tabellenverzeichnis

4.1 Gliederung der Handbuchseiten 51
4.2 Themenbereiche der Handbuchseiten 52

5.1 Tastaturbefehle für den Eingabemodus von `vi` 61
5.2 Tastaturbefehle zur Cursorpositionierung in `vi` 62
5.3 Tastaturbefehle zur Textkorrektur in `vi` 63
5.4 Tastaturbefehle zur Textersetzung in `vi` 63
5.5 `ex`-Kommandos von `vi` 65

6.1 Einige Dateitypenkennzeichnungen in `ls` 72
6.2 Einige Optionen für `ls` 73
6.3 Optionen für `cp` 79
6.4 Tastaturbefehle für `more` 86
6.5 Tastaturbefehle für `less` 87
6.6 Testkriterien von `find` 88
6.7 Logische Operatoren für `find` 89

7.1 Standardkanäle unter Linux 96
7.2 Optionen für `cat` (Auswahl) 103
7.3 Optionen für `tac` (Auswahl) 104
7.4 Optionen für `od` (Auszug) 106
7.5 Optionen für `tr` 109
7.6 Zeichen und Zeichenklassen für `tr` 110
7.7 Optionen von `pr` 114
7.8 Optionen für `nl` (Auswahl) 115
7.9 Optionen für `wc` (Auswahl) 116
7.10 Optionen für `sort` (Auswahl) 120
7.11 Optionen für `join` (Auswahl) 125

8.1 Wichtige Variable der Shell 132
8.2 Tastaturkürzel innerhalb der Bash 137
8.3 Optionen für `jobs` 145

9.1 Linux-Dateitypen 149
9.2 Zuordnung einiger Verzeichnisse zum FHS-Schema 157

12.1 Die wichtigsten Dateiattribute 206

14.1 Verschiedene SCSI-Varianten 224
14.2 Partitionstypen (hexadezimal) für Linux 227
14.3 Partitionstyp-GUIDs für GPT (Auswahl) 228

18.1 Gängige Ziele für systemd (Auswahl) 313
18.2 Kompatibilitäts-Ziele für System-V-Init 314

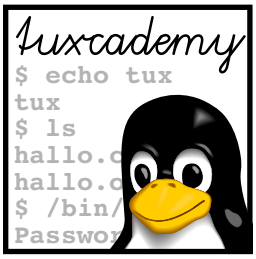
20.1 Kategorien für den `syslogd` 335
20.2 Prioritäten für den `syslogd` (nach aufsteigender Dringlichkeit) 336

20.3 Filterfunktionen für Syslog-NG 345

22.1 Gängige Anwendungsprotokolle auf TCP/IP-Basis 378
22.2 Beispiel für Adressenvergabe 380
22.3 Traditionelle IP-Netzklassen 381
22.4 Beispiel für Subnetting 382
22.5 Private IP-Adressbereiche nach RFC 1918 383

23.1 Optionen innerhalb `/etc/resolv.conf` 402

24.1 Wichtige Optionen von `ping` 408



Abbildungsverzeichnis

1.1 Ken Thompson und Dennis Ritchie an einer PDP-11 17
1.2 Die Weiterentwicklung von Linux 18
1.3 Organisationsstruktur des Debian-Projekts 29

2.1 Die Anmeldebildschirme einiger gängiger Linux-Distributionen . . 34
2.2 Programme als anderer Benutzer ausführen in KDE 38

4.1 Eine Handbuchseite 53

5.1 Arbeitsmodi von `vi` 60
5.2 Der Editor `vi` (direkt nach dem Start) 61

7.1 Standardkanäle unter Linux 97
7.2 Das Kommando `tee` 102

8.1 Die synchrone Arbeitsweise der Shell 143
8.2 Die asynchrone Arbeitsweise der Shell 144

9.1 Inhalt des Wurzelverzeichnisses (SUSE) 150

13.1 Verdeutlichung der Zusammenhänge zwischen den verschiedenen Prozesszuständen 211

15.1 Die Datei `/etc/fstab` (Beispiel) 265

17.1 Eine typische `/etc/inittab`-Datei (Auszug) 289
17.2 Upstart-Konfigurationsdatei für Job `rsyslog` 296

18.1 Eine systemd-Unit-Datei: `console-getty.service` 307

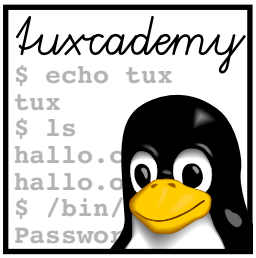
20.1 Beispiel-Konfiguration für `logrotate` (Debian GNU/Linux 8.0) . . . 348

21.1 Vollständige Protokollausgabe mit `journalctl` 360

22.1 Protokolle und Dienstschnittstellen 368
22.2 ISO/OSI-Referenzmodell 369
22.3 Aufbau eines IP-Datagramms 372
22.4 Aufbau eines ICMP-Pakets 373
22.5 Aufbau eines TCP-Segments 374
22.6 Aufbau einer TCP-Verbindung: Der Drei-Wege-Handshake 375
22.7 Aufbau eines UDP-Datagramms 376
22.8 Die Datei `/etc/services` (Auszug) 377

23.1 Beispiel für `/etc/resolv.conf` 402
23.2 Die Datei `/etc/hosts` (SUSE) 403

26.1 Das Programm `aptitude` 450



Vorwort

Diese Unterlage bietet eine kompakte Einführung in die Anwendung und Administration von Linux. Sie richtet sich vor allem an Teilnehmer, die schon mit anderen Betriebssystemen Erfahrung gesammelt haben und auf Linux „umsteigen“ wollen, aber ist auch als Grundlage für den Unterricht an Schulen und Hochschulen geeignet.

Zu den Themen gehören eine umfassende Einführung in den Gebrauch der Linux-Shell, des Editors `vi` und der wichtigsten Werkzeuge zum Umgang mit Dateien sowie ein Einstieg in grundlegende Administrationsaufgaben wie die Benutzer-, Zugriffsrechte- und Prozessverwaltung. Wir geben einen Einblick in die Organisation des Dateisystems und die Administration von Plattenspeicher, beschreiben die Vorgänge beim Systemstart, die Konfiguration von Diensten, die zeitgesteuerte Automation von Aufgaben und die Funktion des Systemprotokolldiensts. Abgerundet wird die Darstellung durch eine Einführung in TCP/IP und die Konfiguration und den Betrieb von Linux-Rechnern als Netzwerk-Clients mit besonderem Augenmerk auf die Fehlersuche und -behebung sowie Kapiteln über die „Secure Shell“ und das Drucken mit Linux auf lokalen und Netzwerkdruckern.

Gemeinsam mit dem Folgeband *Linux kompakt – Aufbaukurs* deckt diese Schulungsunterlage den kompletten Stoffumfang der Prüfungen zum LPIC-1-Zertifikat des *Linux Professional Institute* ab und eignet sich darum auch zur Prüfungsvorbereitung.

Diese Schulungsunterlage soll den Kurs möglichst effektiv unterstützen, indem das Kursmaterial in geschlossener, ausführlicher Form zum Mitlesen, Nach- oder Vorarbeiten präsentiert wird. Das Material ist in Kapitel eingeteilt, die jeweils für sich genommen einen Teilaspekt umfassend beschreiben; am Anfang jedes Kapitels sind dessen Lernziele und Voraussetzungen kurz zusammengefasst, am Ende finden sich eine Zusammenfassung und (wo sinnvoll) Angaben zu weiterführender Literatur oder WWW-Seiten mit mehr Informationen.

Kapitel

Lernziele
Voraussetzungen

Zusätzliches Material oder weitere Hintergrundinformationen sind durch das »Glühbirnen«-Sinnbild am Absatzanfang gekennzeichnet. Zuweilen benutzen diese Absätze Aspekte, die eigentlich erst später in der Schulungsunterlage erklärt werden, und bringen das eigentlich gerade Vorgestellte so in einen breiteren Kontext; solche »Glühbirnen«-Absätze sind möglicherweise erst beim zweiten Durcharbeiten der Schulungsunterlage auf dem Wege der Kursnachbereitung voll verständlich.

Absätze mit dem »Warnschild« weisen auf mögliche Probleme oder »gefährliche Stellen« hin, bei denen besondere Vorsicht angebracht ist. Achten Sie auf die scharfen Kurven!

Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind numeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad angegeben. Aufgaben, die mit einem Ausrufungszeichen (»!«) gekennzeichnet sind, sind besonders empfehlenswert.

Übungsaufgaben

Auszüge aus Konfigurationsdateien, Kommandobeispiele und Beispiele für die Ausgabe des Rechners erscheinen in `Schreibmaschinenschrift`. Bei mehrzeiligen Dialogen zwischen Benutzer und Rechner werden die Benutzereingaben in **`fetter Schreibmaschinenschrift`** angegeben, um Missverständnisse zu vermeiden. Wenn Teile einer Kommandoausgabe ausgelassen wurden, wird das durch »◁◁◁◁◁« kenntlich gemacht. Manchmal sind aus typografischen Gründen Zeilenumbrüche erforderlich, die in der Vorlage auf dem Rechner nicht stehen; diese werden als »▷ ◁« dargestellt. Bei Syntaxdarstellungen stehen Wörter in spitzen Klammern (»⟨*Wort*⟩«) für »Variable«, die von Fall zu Fall anders eingesetzt werden können; Material in eckigen Klammern (»[`-f` ⟨*Datei*⟩]«) kann entfallen und ein vertikaler Balken trennt Alternativen (»`-a`|`-b`«).

Wichtige Konzepte
Definitionen

Wichtige Konzepte werden durch »Randnotizen« hervorgehoben; die **Definitionen** wesentlicher Begriffe sind im Text fett gedruckt und erscheinen ebenfalls am Rand.

Verweise auf Literatur und interessante Web-Seiten erscheinen im Text in der Form »[GPL91]« und werden am Ende jedes Kapitels ausführlich angegeben.

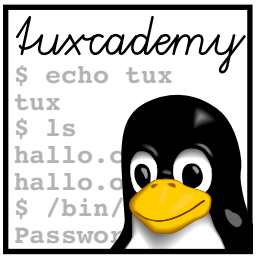
Wir sind bemüht, diese Schulungsunterlage möglichst aktuell, vollständig und fehlerfrei zu gestalten. Trotzdem kann es passieren, dass sich Probleme oder Ungenauigkeiten einschleichen. Wenn Sie etwas bemerken, was Sie für verbesserungsfähig halten, dann lassen Sie es uns wissen, etwa indem Sie eine elektronische Nachricht an

```
info@tuxcademy.org
```

schicken. (Zur Vereinfachung geben Sie am besten den Titel der Schulungsunterlage, die auf der Rückseite des Titelblatts enthaltene Revisionsnummer sowie die betreffende(n) Seitenzahl(en) an.) Vielen Dank!

LPIC-1-Zertifizierung

Diese Unterlage ist Teil eines Kurskonzepts zur Vorbereitung auf die LPIC-1-Zertifizierung. Anhang C enthält hierzu mehr Informationen.



1

Einführung

Inhalt

1.1 Was ist Linux? 16
1.2 Die Geschichte von Linux 16
1.3 Freie Software, »Open Source« und die GPL 18
1.4 Linux – Der Kernel 22
1.5 Die Eigenschaften von Linux 24
1.6 Linux-Distributionen 27

Lernziele

- Linux mit seinen Eigenschaften und seiner Entstehungsgeschichte kennen
- Verstehen, was Kernel und Distributionen sind
- Die Begriffe »GPL«, »freie Software« und »Open Source« einordnen können

Vorkenntnisse

- Kenntnisse eines anderen Betriebssystems sind hilfreich, um Parallelen und Unterschiede erkennen zu können

1.1 Was ist Linux?

Linux ist ein Betriebssystem. Als Betriebssystem stellt es die elementaren Funktionen zum Betrieb eines Rechners zur Verfügung. Anwendungsprogramme bauen auf dem Betriebssystem auf. Es bildet die Schnittstelle zwischen der Hardware und den Anwendungsprogrammen, aber auch die Schnittstelle zwischen Hardware und Mensch (Benutzer). Ohne ein Betriebssystem ist der Computer nicht in der Lage, unsere Eingaben zu »verstehen« bzw. zu verarbeiten.

Die verschiedenen Betriebssysteme unterscheiden sich in der Weise, wie sie die oben genannten Aufgaben ausführen. Linux ist in seiner Funktionalität und seiner Bedienung dem Betriebssystem Unix nachempfunden.

1.2 Die Geschichte von Linux

Die Entstehungsgeschichte von Linux ist etwas Besonderes in der Computer-Welt. Während die meisten anderen Betriebssysteme kommerzielle Produkte von Firmen sind, wurde Linux von einem Studenten als Hobbyprojekt ins Leben gerufen. Inzwischen arbeiten weltweit Hunderte von Profis und Enthusiasten daran mit – von Hobbyprogrammierern und Informatikstudenten bis zu Betriebssystemexperten, die von renommierten Computerfirmen dafür bezahlt werden, Linux weiterzuentwicken. Grundlage für die Existenz eines solchen Projekts ist das Internet: Die Linux-Entwickler nutzen intensiv Dienste wie E-Mail, verteilte Revisionskontrolle und das World Wide Web und haben so das Betriebssystem Linux zu dem gemacht, was es heute ist. Linux ist also das Ergebnis einer internationalen Zusammenarbeit über Länder- und Firmengrenzen hinweg, nach wie vor geleitet von Linus Torvalds, dem ursprünglichen Autor.

Um den Hintergrund von Linux erklären zu können, müssen wir etwas weiter ausholen: Unix, das Vorbild von Linux, entstand ab 1969. Es wurde von Ken
Bell Laboratories Thompson und seinen Kollegen bei den Bell Laboratories (dem Forschungsinstitut des US-amerikanischen Telefonmagnaten AT&T) entwickelt[1]. Es verbreitete sich rasch vor allem in Universitäten, da die Bell Labs die Quellen und Dokumentation zum Selbstkostenpreis abgaben (AT&T durfte aufgrund kartellrechtlicher Einschränkungen keine Software verkaufen). Unix war zunächst ein Betriebssystem für die PDP-11-Rechner von Digital Equipment, wurde aber im Laufe der 1970er Jahre auf andere Plattformen portiert – ein vergleichsweise einfaches Unterfangen, da die Unix-Software, einschließlich des Betriebssystemkerns, zum allergrößten Teil in der von Dennis Ritchie speziell für diesen Zweck erfundenen Program-
C miersprache **C** geschrieben war. Die vielleicht wichtigste Portierung war die auf
VAX die PDP-11-Nachfolgeplattform VAX an der University of California in Berkeley, die als »BSD« (kurz für *Berkeley Software Distribution*) in Umlauf kam. Im Laufe der Zeit entwickelten verschiedene Computerhersteller unterschiedliche Unix-Varianten teils auf der Basis des AT&T-Codes, teils auf der Basis von BSD (z. B. Sinix von Siemens, Xenix von Microsoft (!), SunOS von Sun Microsystems, HP/UX von Hewlett-Packard oder AIX von IBM). Auch AT&T selbst durfte schließlich Unix
System V verkaufen – die kommerziellen Versionen System III und (später) System V. Das führte zu einer ziemlich unübersichtlichen Fülle verschiedenener Unix-Produkte. Es kam nie wirklich zu einer Standardisierung, aber man kann in etwa zwischen BSD- und System-V-nahen Unix-Varianten unterscheiden. Zum größten Teil wur-
SVR4 den die BSD- und die System-V-Entwicklungslinien in »System V Release 4« zusammengeführt, das die wesentlichen Eigenschaften beider Strömungen aufweist.

Die allerersten Teile von Linux wurden 1991 von Linus Torvalds, einem damals 21-jährigen Studenten aus Helsinki, entwickelt, als dieser die Möglichkeiten des Intel-386-Prozessors in seinem neuen PC genauer untersuchte. Nach einigen Mo-

[1] Der Name »Unix« ist ein Wortspiel mit »Multics«, dem Betriebssystem, an dem Ken Thompson und seine Kollegen vorher mitgearbeitet hatten. Das frühe Unix war viel simpler als Multics. Wie es dazu kam, den Namen mit »x« zu schreiben, ist nicht mehr bekannt.

Bild 1.1: Ken Thompson (sitzend) und Dennis Ritchie (stehend) an einer PDP-11, ca. 1972. (Abdruck mit freundlicher Genehmigung von Lucent Technologies.)

naten war aus den Assemblerstudien ein kleiner, lauffähiger Betriebssystemkern entstanden, der in einem Minix-System eingesetzt werden konnte – Minix war ein kleines Unix-artiges Betriebssystem, das der Informatikprofessor Andrew S. Tanenbaum an der Freien Universität Amsterdam für seine Studenten geschrieben hatte. Das frühe Linux hatte ähnliche Eigenschaften wie ein Unix-System, enthielt aber keinen Unix-Quellcode. Linus Torvalds gab den Programmcode über das Internet frei, und die Idee wurde mit Begeisterung von vielen Programmierern aufgegriffen und weiter entwickelt. Die im Januar 1992 herausgegebene Version 0.12 war bereits ein stabil laufender Betriebssystemkern. Es gab – dank Minix – den `gcc` (GNU C-Compiler), die `bash`, `emacs` und viele der anderen GNU-Hilfsprogramme. Dieses Betriebssystem wurde über anonymes FTP weltweit verteilt. Die Zahl der Programmierer, Tester und Unterstützer wuchs rasend schnell. Das ermöglichte in kurzer Zeit Fortschritte, von denen mächtige Softwareunternehmen nur träumen können. Innerhalb weniger Monate wurde aus dem Minikernel ein ausgewachsenes Betriebssystem mit ziemlich vollständiger (wenn auch simpler) Unix-Funktionalität.

Minix

Das Projekt »Linux« ist auch heute nicht abgeschlossen. Linux wird ständig aktualisiert und erweitert, und zwar von Hunderten von Programmierern auf der ganzen Welt, denen inzwischen mehrere Millionen zufriedene private und kommerzielle Anwender gegenüberstehen. Man kann auch nicht sagen, dass das System »nur« von Studenten und anderen Amateuren entwickelt wird – viele Leute, die am Linux-Kern mitarbeiten, haben wichtige Posten in der Computerindustrie und gehören zu den fachlich angesehensten Systementwicklern überhaupt. Inzwischen läßt sich mit Berechtigung behaupten, dass Linux das Betriebssystem mit der breitesten Hardwareunterstützung überhaupt ist, nicht nur bezogen auf die Plattformen, auf denen es läuft (vom PDA bis zum Großrechner), sondern auch auf die Treiberunterstützung zum Beispiel auf der Intel-PC-Plattform. Linux dient auch als Test- und Forschungsplattform für neue Betriebssystem-Ideen in Industrie und Hochschule; es ist zweifellos eines der innovativsten derzeit verfügbaren Betriebssysteme.

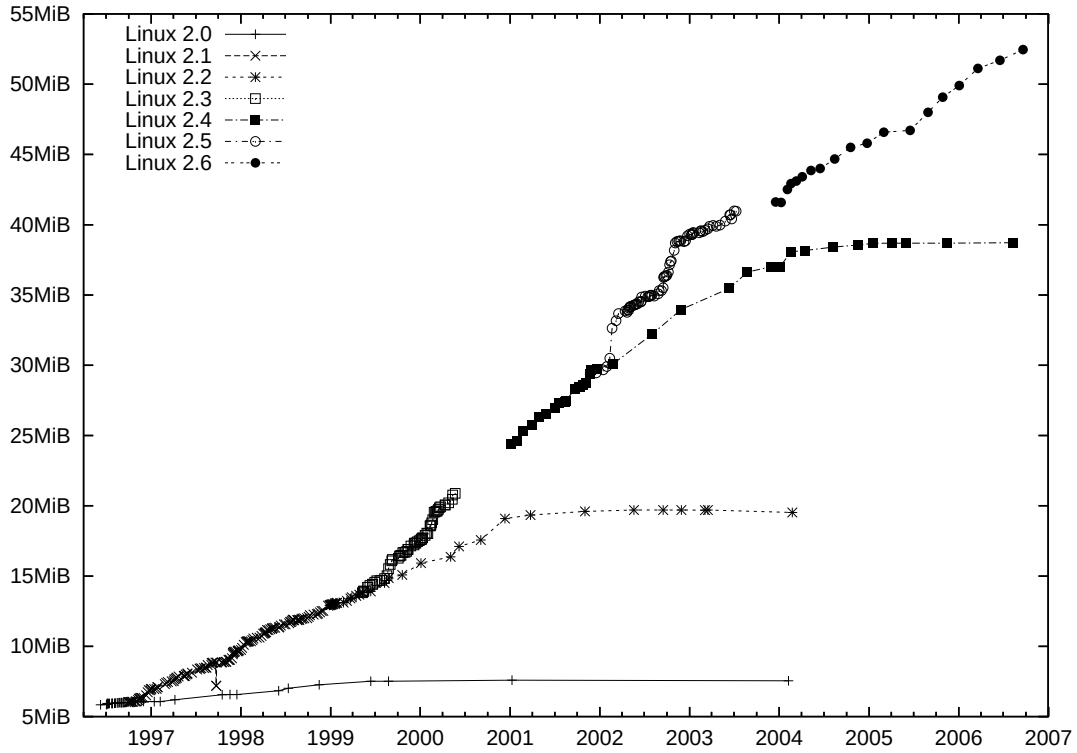


Bild 1.2: Die Weiterentwicklung von Linux, gemessen an der Größe von `linux-*.tar.gz`. Jede Marke entspricht einer Linux-Version. In den gut 10 Jahren von Linux 2.0 bis Linux 2.6.18 hat der Umfang des komprimierten Linux-Quellcodes sich nahezu verzehnfacht.

Übungen

1.1 [4] Suchen Sie im Internet nach der berühmt-berüchtigten Diskussion zwischen Andrew S. Tanenbaum und Linus Torvalds, in der Tanenbaum sagt, Linus Torvalds wäre mit etwas wie Linux bei ihm im Praktikum durchgefallen. Was halten Sie davon?

1.2 [2] Welche Versionsnummer hat der älteste Linux-Kern-Quellcode, den Sie noch finden können?

1.3 Freie Software, »Open Source« und die GPL

GPL
Free Software Foundation
Freie Software
Open Source

Linux steht seit Beginn der Entwicklung unter der *GNU General Public License* (**GPL**) der *Free Software Foundation* (FSF). Die FSF wurde von Richard M. Stallman, dem Autor des Editors Emacs und anderer wichtiger Programme, mit dem Ziel gegründet, qualitativ hochwertige Software »frei« verfügbar zu machen – in dem Sinne, dass Benutzer »frei« sind, sie anzuschauen, zu ändern und im Original oder geändert weiterzugeben, nicht notwendigerweise in dem Sinne, dass sie nichts kostet[2]. Insbesondere ging es ihm um ein frei verfügbares Unix-artiges Betriebssystem, daher »GNU« als (rekursive) Abkürzung für *"GNU's Not Unix"*. Die wesentliche Aussage der GPL liegt darin, dass in ihrem Sinne geschützte Software zwar jederzeit verändert und auch verkauft werden darf, aber immer zusammen mit dem (gegebenenfalls veränderten) Quellcode weitergegeben werden muss – deshalb auch *Open Source* – und der Empfänger dieselben Rechte der Veränderung und Weitergabe erhalten muss. Es hat darum keinen großen Sinn, GPL-Software

[2] Die FSF sagt "free as in speech, not as in beer"

»pro Rechner« zu verkaufen, da der Empfänger das Recht bekommt, die Software so oft zu kopieren und zu installieren, wie er mag. (Was man allerdings durchaus »pro Rechner« verkaufen darf, ist Unterstützung und Service für die GPL-Software.) Neue Software, die durch Erweiterung oder Veränderung von GPL-Software entsteht, muss als »abgeleitetes Werk« ebenfalls unter die GPL gestellt werden.

Die GPL bezieht sich also auf den Vertrieb der Software, nicht den Gebrauch, und gibt dem Empfänger der Software Rechte, die er sonst überhaupt nicht hätte – zum Beispiel das Recht zur Vervielfältigung und Weitergabe der Software, das nach dem Gesetz dem »Urheber« vorbehalten bleibt (darum »Urheberrecht«). Sie unterscheidet sich deshalb deutlich von den *End User License Agreements* »proprietärer« Software, die dem Benutzer normalerweise Rechte zu *nehmen* versuchen. (Manche EULAs wollen dem Empfänger der Software zum Beispiel verbieten, in der Öffentlichkeit schlecht – oder überhaupt – über das Produkt zu reden.)

EULAs

Die GPL ist eine *Lizenz*, kein Vertrag, da sie dem Empfänger der Software einseitig etwas erlaubt (wenngleich unter Bedingungen). Der Empfänger der Software muss die GPL nicht ausdrücklich »akzeptieren«. Bei den gängigen EULAs handelt es sich dagegen um *Verträge*, da der Empfänger der Software im Austausch für das Recht, die Software benutzen zu dürfen, auf gewisse Rechte verzichten soll. Aus diesem Grund müssen EULAs auch ausdrücklich akzeptiert werden. Die rechtlichen Anforderungen hierfür liegen recht hoch – etwa müssen in Deutschland einem Software-Käufer die EULA-Bedingungen vor dem Kauf bekannt sein, damit sie wirksamer Bestandteil des Kaufvertrags werden können. Da die GPL die Rechte des Käufers (insbesondere was den Gebrauch der Software angeht) in keiner Weise dem gegenüber einschränkt, was er bei irgendeinem Sachkauf zu erwarten hätte, gelten diese Anforderungen nicht für die GPL; die zusätzlichen Rechte, die der Käufer unter der GPL erhält, sind eine Art Sonderbonus.

Im Moment sind zwei Versionen der GPL in Gebrauch. Die neuere Version 3 (auch als »GPLv3« bezeichnet) wurde Ende Juni 2007 veröffentlicht und unterscheidet sich von der älteren Version 2 (auch »GPLv2«) durch Präzisierungen in Bereichen wie Softwarepatenten, der Kompatibilität mit anderen freien Lizenzen und der Einführung von Restriktionen dafür, Änderungen an der theoretisch »freien« Software von Geräten unmöglich zu machen, indem man sie durch spezielle Hardware ausschließt (die »Tivoisierung«, nach einem digitalen Videorekorder auf Linux-Basis, dessen Linux-Kern man nicht verändern oder austauschen kann). Die GPLv3 erlaubt ihren Benutzern auch das Hinzufügen weiterer Klauseln. – Die GPLv3 stieß nicht auf die uneingeschränkte Zustimmung der Szene, so dass viele Projekte, allen voran der Linux-Kernel, mit Absicht bei der einfacheren GPLv2 geblieben sind. Viele Projekte werden auch »unter der GPLv2 oder einer neueren Version der GPL« vertrieben, so dass Sie sich entscheiden können, welcher Version der GPL Sie bei der Weitergabe oder Änderung solcher Software folgen wollen.

GPLv3

Nicht verwechseln sollten Sie GPL-Software auch mit »Public-Domain-Software«. Letztere gehört niemandem, jeder darf damit machen, was er möchte (dieses Konzept existiert im deutschen Recht übrigens nur als »Gemeinfreiheit«, die eintritt, wenn der Urheber länger als 70 Jahre tot ist – im Computerbereich ist das noch nicht relevant). Die Urheberrechte an GPL-Software liegen aber in der Regel auch weiterhin beim Entwickler oder den Entwicklern, und die GPL sagt sehr deutlich, was mit der Software gemacht werden darf und was nicht.

Public Domain

Es gilt unter Entwicklern freier Software als guter Stil, Beiträge zu einem Projekt unter dieselbe Lizenz zu stellen, die das Projekt schon benutzt, und die meisten Projekte bestehen tatsächlich darauf, dass das zumindest für solchen Code gilt, der in die »offizielle« Version einfließen soll. Manche

Projekte bestehen sogar auf *copyright assignments*, bei denen der Urheber des Codes seine Rechte an das Projekt (oder eine geeignete Organisation) abtritt. Dieser Schritt hat den Vorteil, dass urheberrechtlich nur das Projekt für den Code verantwortlich ist und Lizenzverstöße – die nur der Rechte-Inhaber verfolgen kann – leichter geahndet werden können. Ein entweder gewollter oder ausdrücklich unerwünschter Nebeneffekt ist auch, dass es einfacher möglich ist, die Lizenz für das ganze Projekt zu ändern, denn auch das darf nur der Rechte-Inhaber.

Im Falle des Linux-Betriebssystemkerns, wo ausdrücklich keine *copyright assignments* verlangt werden, ist eine Lizenzänderung praktisch sehr schwierig bis ausgeschlossen, da der Code ein Flickenteppich von Beiträgen von über tausend Autoren ist. Die Angelegenheit wurde im Zuge der Veröffentlichung der GPLv3 erörtert, und man war sich einig, dass es ein riesengroßes Projekt wäre, die urheberrechtliche Provenienz jeder einzelnen Zeile des Linux-Quellcodes zu klären und die Zustimmung der Autoren zu einer Lizenzänderung einzuholen. Manche Linux-Entwickler wären auch vehement dagegen, andere sind nicht mehr zu finden oder sogar verstorben, und der betreffende Code müsste durch etwas Ähnliches mit klarem Copyright ersetzt werden. Zumindest Linus Torvalds ist aber nach wie vor Anhänger der GPLv2, so dass das Problem sich in der Praxis (noch) nicht wirklich stellt.

GPL und Geld

Die GPL sagt nichts über den möglichen Preis des Produkts aus. Es ist absolut legal, dass Sie Kopien von GPL-Programmen verschenken oder auch Geld dafür verlangen, solange Sie auch den Quellcode mitliefern oder auf Anfrage verfügbar machen und der Empfänger der Software auch die GPL-Rechte bekommt. GPL-Software ist damit also nicht unbedingt »Freeware«.

Mehr Informationen erhalten Sie durch Lesen der GPL [GPL91], die übrigens jedem entsprechenden Produkt (auch Linux) beiliegen muss.

Andere »freie« Lizenzen

Es gibt andere »freie« Software-Lizenzen, die dem Empfänger der Software ähnliche Rechte einräumen, etwa die »BSD-Lizenz«, die die Integration von entsprechend lizenzierter Software in eigene, nicht freie Produkte gestattet. Die GPL gilt aber als konsequenteste der freien Lizenzen in dem Sinne, dass sie sicherzustellen versucht, dass einmal unter der GPL veröffentlichter Code auch frei *bleibt*. Es haben schon öfters Firmen versucht, GPL-Code in ihre eigenen Produkte zu integrieren, die dann nicht unter der GPL freigegeben werden sollten. Allerdings haben diese Firmen bisher immer nach einer nachdrücklichen Ermahnung durch (meist) die FSF als Rechteinhaberin eingelenkt und die betreffenden Programme freigegeben. Zumindest in Deutschland ist die GPL auch schon gerichtlich für gültig erklärt worden – einer der Linux-Kernelprogrammierer konnte vor dem Landgericht Frankfurt ein Urteil gegen die Firma D-Link (einen Hersteller von Netzwerkkomponenten, hier einem NAS-Gerät auf Linux-Basis) erwirken, in dem letztere zu Schadensersatz verklagt wurde, weil sie bei der Verbreitung ihres Geräts den GPL-Auflagen nicht genügte [GPL-Urteil06].

Warum funktioniert die GPL? Manche Firma, der die Anforderungen der GPL lästig waren, hat versucht, sie für ungültig zu erklären oder erklären zu lassen. So wurde sie in den USA schon als »unamerikanisch« oder »verfassungswidrig« apostrophiert; in Deutschland wurde versucht, das Kartellrecht heranzuziehen, da die GPL angeblich illegale Preisvorschriften macht. Die Idee scheint zu sein, dass Software, die unter der GPL steht, für jeden beliebig benutzbar ist, wenn mit der GPL irgendetwas nachweislich nicht stimmt. Alle diese Angriffe verkennen aber eine Tatsache: Ohne die GPL hätte niemand außer dem ursprünglichen Autor überhaupt das Recht, mit dem Code irgendetwas zu tun, da Aktionen wie das Weitergeben oder gar Verkaufen nach dem Urheberrecht nur ihm vorbehalten sind. Fällt die GPL weg, dann stehen alle anderen Interessenten an dem Code also viel schlechter da als vorher.

Ein Prozess, bei dem ein Softwareautor eine Firma verklagt, die seine GPL-Software vertreibt, ohne sich an die GPL zu halten, würde aller Wahrscheinlichkeit nach so aussehen:

Richter Was ist jetzt das Problem?

Softwareautor Herr Vorsitzender, die Beklagte hat meine Software vertrieben, ohne die Lizenz dafür einzuhalten.

Richter (zum Rechtsanwalt der Beklagten) Stimmt das?

An dieser Stelle kann die beklagte Firma »Ja« sagen und der Prozess ist im Wesentlichen vorbei (bis auf das Urteil). Sie kann auch »Nein« sagen, aber sie muss dann begründen, warum das Urheberrecht für sie nicht gilt. Ein unangenehmes Dilemma und der Grund, warum wenige Firmen sich diesen Stress machen und die meisten GPL-Streitigkeiten außergerichtlich geklärt werden.

Verstößt ein Hersteller proprietärer Software gegen die GPL (etwa indem er ein paar hundert Zeilen Quellcode aus einem GPL-Projekt in seine Software integriert), bedeutet das übrigens in keinem Fall, dass seine Software dadurch automatisch komplett unter die GPL fällt und offengelegt werden muss. Es bedeutet zunächst nur, dass er GPL-Code gegen dessen Lizenz vertrieben hat. Lösen kann der Hersteller das Problem auf verschiedene Arten:

- Er kann den GPL-Code entfernen und durch eigenen Code ersetzen. Die GPL wird dann irrelevant für sein Programm.
- Er kann mit dem Urheberrechtsinhaber des GPL-Codes verhandeln und zum Beispiel eine Zahlung von Lizenzgebühren vereinbaren (wenn dieser aufzutreiben ist und mitmacht). Siehe auch den Abschnitt über Mehrfachlizenzierung weiter unten.
- Er kann sein komplettes Programm *freiwillig* unter die GPL stellen und so den Anforderungen der GPL entsprechen (die unwahrscheinlichste Methode).

Unabhängig davon könnte für den vorher stattgefundenen Lizenzverstoß Schadensersatz verhängt werden. Der urheberrechtliche Status der proprietären Software bleibt davon allerdings unberührt.

Freiheits-Kriterien
Debian Free Software Guidelines

Wann gilt eine Software als »frei« oder »Open Source«? Dafür gibt es keine verbindlichen Regeln, aber weithin anerkannt sind die *Debian Free Software Guidelines* [DFSG]. Die FSF fasst ihre Kriterien in den *Four Freedoms* zusammen, die für eine freie Software gegeben sein müssen:

- Man muss das Programm für jeden beliebigen Zweck benutzen dürfen (Freiheit 0)
- Man muss studieren können, wie das Programm funktioniert, und es an seine Bedürfnisse anpassen können (Freiheit 1)
- Man muss das Programm weitergeben dürfen, um seinem Nächsten zu helfen (Freiheit 2)
- Man muss das Programm verbessern und die Verbesserungen veröffentlichen dürfen, um der Allgemeinheit zu nutzen (Freiheit 3).

Zugang zum Quellcode ist Vorbedingung für die Freiheiten 1 und 3. Gängige Freie-Software-Lizenzen wie die GPL und die BSD-Lizenz erfüllen diese Freiheiten natürlich.

Mehrfach-Lizenzierung

Im übrigen darf der Urheber einer Software diese durchaus unter verschiedenen Lizenzen verteilen, etwa alternativ unter der GPL und einer »kommerziellen« Lizenz, die den Empfänger von den Anforderungen der GPL, etwa der Freigabe

von Quellcode für abgeleitete Werke, freistellt. Auf diese Weise können zum Beispiel Privatanwender und Autoren freier Software kostenlos in den Genuss einer leistungsfähigen Unterprogrammbibliothek wie zum Beispiel des Grafikpakets »Qt« (veröffentlicht von Qt Software – ehemals Troll Tech –, einem Tochterunternehmen von Nokia) kommen, während Firmen, die ihren eigenen Programmcode nicht frei zur Verfügung stellen wollen, sich von der GPL »freikaufen« müssen.

Übungen

1.3 [!2] Welche der folgenden Aussagen über die GPL stimmen und welche sind falsch?

1. GPL-Programme dürfen nicht verkauft werden.
2. GPL-Programme dürfen von Firmen nicht umgeschrieben und zur Grundlage eigener Produkte gemacht werden.
3. Der Urheber eines GPL-Programms darf das Programm auch unter einer anderen Lizenz vertreiben.
4. Die GPL gilt nicht, weil man die Lizenz erst zu sehen bekommt, nachdem man das Programm schon hat. Damit Lizenzbestimmungen gültig werden, muss man sie vor dem Erwerb der Software sehen und ihnen zustimmen können.

1.4 [4] Manche Softwarelizenzen verlangen, dass bei Änderungen am Inhalt einer Datei aus der Softwaredistribution die Datei einen neuen Namen bekommen muss. Ist so lizenzierte Software »frei« gemäß der DFSG? Was halten Sie davon?

1.4 Linux – Der Kernel

Betriebssystemkern

Genaugenommen bezeichnet der Begriff »Linux« nur den Betriebssystemkern, der die eigentlichen Aufgaben des Betriebssystems übernimmt. Den Betriebssystemkern nennt man (neudeutsch) auch *Kernel*. Er erledigt elementare Aufgaben wie Speicher- und Prozessverwaltung und die Steuerung der Hardware. Anwenderprogramme müssen sich an den Kernel wenden, wenn sie zum Beispiel auf Dateien auf der Platte zugreifen wollen. Der Kernel prüft solche Vorgänge und kann auf diese Weise dafür sorgen, dass niemand unerlaubten Zugriff auf fremde Dateien erhält. Außerdem kümmert sich der Kernel darum, dass alle Prozesse im System (und damit alle Benutzer) den ihnen jeweils zustehenden Anteil an der verfügbaren Rechenzeit erhalten.

Versionen

Natürlich gibt es nicht nur einen Linux-Kernel, sondern es existieren viele verschiedene Versionen. Bis zur Kernel-Version 2.6 wurde zwischen stabilen »Anwender-Versionen« und instabilen »Entwickler-Versionen« wie folgt unterschieden:

stabile Versionen

- In Versionsnummern wie 1.*x*.*y* oder 2.*x*.*y* bezeichnet ein gerades *x* stabile Versionen. In stabilen Versionen sollen keine durchgreifenden Änderungen gemacht werden; es sollen vor allem Fehlerkorrekturen vorgenommen werden, und ab und zu werden Treiber für neu herausgekommene Hardware oder andere sehr wichtige Eigenschaften neu hinzugefügt, die aus den Entwicklungs-Kernels »zurückportiert« wurden.

Entwicklungs-Versionen

- Die Versionen mit ungeradem *x* sind Entwicklungs-Versionen, die nicht für den Produktiveinsatz geeignet sind. Sie enthalten oft unzureichend getesteten Code und sind eigentlich für die Leute gedacht, die sich aktiv an der Entwicklung von Linux beteiligen wollen. Da Linux ständig weiter entwickelt wird, gibt es auch ständig neue Kernelversionen. Die Änderungen betreffen zumeist Anpassungen an neue Hardware oder die Optimierung verschiedener Subsysteme, manchmal auch komplett neue Erweiterungen.

Kernel 2.6

In Kernel 2.6 hat sich die Vorgehensweise geändert: Statt wie bisher nach einer mehr oder kurzen Stabilisierungsphase die Version 2.7 für Neuentwicklungen anzufangen, haben Linus Torvalds und die anderen Kernel-Entwickler sich entschieden, die Weiterentwicklung von Linux näher an der Endanwender-Version zu halten. Dadurch soll die Divergenz von Entwickler- und stabilen Versionen vermieden werden, wie sie sich zum Beispiel im Vorfeld der Veröffentlichung von Linux 2.6 zu einem großen Problem entwickelt hatte – vor allem weil Firmen wie SUSE oder Red Hat sich viel Mühe gemacht hatten, interessante Eigenschaften der Entwicklerversion 2.5 in ihre Versionen der 2.4-Kernels zurück zu übertragen, bis zum Beispiel ein SUSE-2.4.19-Kernel Aberhunderte von Unterschieden zum »offiziellen« Linux 2.4.19 aufwies.

Die aktuelle Methode besteht darin, vorgeschlagene Änderungen und Erweiterungen in einer neuen Version des Kernels »probezufahren«, die dann in einem kürzeren Zeitraum für »stabil« erklärt wird. Beispielsweise folgt auf die Version 2.6.37 eine Entwicklungsphase, in der Linus Torvalds Erweiterungen und Änderungen für die Version 2.6.38 entgegennimmt. Andere Kernelentwickler (oder wer sonst Lust darauf hat) haben Zugriff auf Linus' interne Entwicklerversion, die, wenn sie vernünftig genug aussieht, als »Release-Kandidat« 2.6.38-rc1 herausgegeben wird. Damit beginnt eine Stabilisierungsphase, in der dieser Release-Kandidat von mehr Leuten getestet wird. Reparaturen führen zu weiteren Release-Kandidaten, bis die neue Version stabil genug aussieht, dass Linus Torvalds sie zur Version 2.6.38 erklären kann. Danach schließt sich eine Entwicklungsphase für 2.6.39 an und so weiter.

Release-Kandidat

Parallel zu Linus Torvalds' »offizieller« Version wartet Andrew Morton eine experimentellere Version, den sogenannten »`-mm`-Baum«. Hierin werden größere und durchgreifendere Änderungen getestet, bis sie reif dafür sind, von Linus in den offiziellen Kernel aufgenommen zu werden.

`-mm`-Baum

Einige andere Entwickler bemühen sich um die Wartung der »stabilen« Kernels. Es könnte etwa Kernels mit den Versionsnummern 2.6.38.1, 2.6.38.2, … geben, in denen jeweils nur kleine und überschaubare Änderungen gemacht werden, etwa um gravierende Fehler oder Sicherheitslücken zu reparieren und Linux-Distributoren die Gelegenheit zu geben, auf längerfristig unterstützte Kernel-Versionen zurückgreifen zu können.

Version 3.0

Am 21. Juli 2011 gab Linus Torvalds offiziell die Version 3.0 des Linux-Kerns frei. Eigentlich hätte das die Version 2.6.40 sein müssen, aber er wollte das Schema für die Versionsnummern vereinfachen. Die »stabilen« Kernels auf der Basis von 3.0 werden entsprechend mit 3.0.1, 3.0.2, … bezeichnet, und in Torvalds' Entwicklungslinie folgt dann 3.1-rc1 usw. bis 3.1 und so fort.

Linus Torvalds legte sehr großen Wert auf die Feststellung, dass es zwischen den Kernels 2.6.39 und 3.0 keine großen Schritte bei der Funktionalität gab – jedenfalls nicht mehr als zwischen irgend zwei anderen aufeinanderfolgenden Kernels der 2.6-Serie –, sondern dass es sich nur um eine Umnummerierung handelte. Die Idee des zwanzigjährigen Linux-Jubiläums wurde ins Spiel gebracht.

»offizielle« Kernels

Quellcode für »offizielle« Kernels können Sie im Internet von `ftp.kernel.org` bekommen. Allerdings setzen die wenigsten Linux-Distributoren Originalkernels ein. Die Kernels der gängigen Distributionen sind zumeist mehr oder weniger umfassend modifiziert, etwa indem weitere Treiber hinzugefügt oder Eigenschaften integriert werden, die von der Distribution erwünscht, aber nicht Teil des Standard-Kernels sind. Der Kernel des *SUSE Linux Enterprise Servers 8* zum Beispiel enthielt angeblich rund 800 Modifikationen gegenüber dem Standardkernel. (Die Änderungen im Linux-Entwicklungsprozess sollten dazu geführt haben, dass die Differenzen heutzutage nicht mehr ganz so krass sind.)

Distributions-Kernels

Kernel-Struktur

Die meisten Kernel sind heute modular aufgebaut. Frühere Kernel waren das

nicht, das heißt, sie bestanden aus einem einzigen Stück Code, das sämtliche Aufgaben erfüllte, etwa die Unterstützung bestimmter Hardware. Wollte man neue Hardware zum System hinzufügen oder eine neue Eigenschaft, etwa ein anderes Dateisystem, verwenden, musste man sich aus den Quellen einen neuen Kernel übersetzen, ein sehr zeitintensiver Vorgang. Um das zu umgehen, wurde der Kernel mit der Fähigkeit ausgestattet, zusätzliche Eigenschaften in Form von Modulen einzubinden.

Module

Module sind Programmteile, die dynamisch (also zur Laufzeit des Kernels) hinzugeladen und auch wieder entfernt werden können. Möchten Sie beispielsweise einen neuen Netzwerk-Adapter verwenden, müssen Sie keinen neuen Kernel kompilieren, sondern lediglich ein neues Kernelmodul hinzuladen. Moderne Linux-Distributionen unterstützen eine automatische Hardwareerkennung, die die Eigenschaften des Systems analysiert und die richtigen Module findet und konfiguriert.

Hardwareerkennung

Übungen

1.5 [1] Welche Versionsnummer hat der aktuelle stabile Linux-Kern? Der aktuelle Entwicklerkern? Welche Linux-Kernversionen werden noch gewartet?

1.5 Die Eigenschaften von Linux

Als moderner Betriebssystemkern hat Linux eine Reihe von Eigenschaften, die zum Teil heute zum »Industriestandard« gehören (also auch bei ähnlichen Systemen in äquivalenter Form anzutreffen sind) und zum Teil Alleinstellungsmerkmale bilden.

Prozessoren

- Linux unterstützt eine große Auswahl von Prozessoren und Rechnerarchitekturen, von Mobiltelefonen (das extrem erfolgreiche Betriebssystem »Android« von Google basiert wie einige andere Systeme in diesem Bereich auf Linux) über PDAs und Tablet-Computer, alle möglichen Sorten von neuen und alten PC-artigen Rechnern, Serversystemen unterschiedlicher Art bis hin zu den größten Großrechnern (in der Rangliste der schnellsten Rechner der Welt läuft die weit überwiegende Mehrheit unter Linux).

  Ein großer Vorteil von Linux im Mobilbereich ist, dass es im Gegensatz zu Microsoft Windows die stromsparenden und leistungsfähigen ARM-Prozessoren unterstützt, auf denen die meisten mobilen Geräte aufbauen. Mit »Windows RT« hat Microsoft 2012 eine auf ARM laufende, teilweise zur Intel-Version kompatible Version von Windows 8 veröffentlicht, die im Markt aber nicht besonders gut ankam.

Hardware

- Von allen aktuell vorhandenen Betriebssystemen unterstützt Linux das breiteste Spektrum an Hardware. Zwar stehen für manche der allerneuesten Komponenten nicht sofort Treiber zur Verfügung, aber auf der anderen Seite arbeitet Linux noch mit Geräten zusammen, die Systeme wie Windows längst hinter sich gelassen haben. Ihre Investitionen in Drucker, Scanner, Grafikkarten und ähnliches werden also optimal geschützt.

Multitasking

- Linux unterstützt »präemptives Multitasking«, das heißt, mehrere Prozesse laufen – scheinbar oder bei Systemen mit mehr als einer CPU auch tatsächlich – zur selben Zeit. Dabei können diese Prozesse sich nicht in die Quere kommen oder einander ungebührlich blockieren; der Systemkern sorgt dafür, dass jeder Prozess gemäß seiner Priorität Rechenzeit zugeteilt bekommt.

  Heute ist das nichts Besonderes mehr; als Linux neu war, war es noch eher bemerkenswert.

Bei entsprechend sorgfältig konfigurierten Systemen reicht das bis in den Echtzeitbereich, und es gibt Linux-Varianten, die tatsächlich für die Steuerung von Industrieanlagen eingesetzt werden, wo »harte« Echtzeitfähigkeit, also garantierte schnelle Antwortzeiten auf extern eintretende Ereignisse, gefordert ist.

- Linux unterstützt mehrere Benutzer auf demselben Rechner, sogar gleichzeitig (über das Netz, über seriell angeschlossene Terminals oder auch mehrere Bildschirme, Tastaturen und Mäuse am selben Rechner). Für jeden Benutzer können getrennte Zugriffsrechte vergeben werden. mehrere Benutzer
- Linux kann problemlos parallel zu anderen Betriebssystemen auf demselben Rechner installiert werden, so dass man abwechselnd Linux oder ein anderes System starten kann. Über »Virtualisierung« kann ein Linux-System in unabhängige Teile aufgeteilt werden, die von aussen wie eigenständige Rechner aussehen und selbst unter Linux oder anderen Betriebssystemen laufen. Hierfür stehen verschiedene freie oder auch kommerzielle Lösungen zur Verfügung. Virtualisierung
- Linux nutzt die verfügbare Hardware effizient aus. Die heute üblichen Zweikern-Prozessoren werden genauso mit Arbeit versorgt wie die 4096 Prozessorkerne eines SGI-Altix-Servers. Linux läßt keinen Arbeitsspeicher (RAM) brachliegen, sondern benutzt ihn als Plattencache; umgekehrt wird der verfügbare Arbeitsspeicher sinnvoll eingesetzt, um Arbeitslasten zu bewältigen, die deutlich größer sind als das RAM im Rechner. Effizienz
- Linux ist auf Quellcodeebene kompatibel zu POSIX, System V und BSD und erlaubt so die Nutzung fast aller im Quellcode verfügbaren Unix-Programme. POSIX, System V und BSD
- Linux verfügt nicht nur über leistungsfähige eigene Dateisysteme mit Eigenschaften wie Journaling, Verschlüsselung und Logical Volume Management, sondern gestattet auch den Zugriff auf Dateisysteme zahlreicher anderer Betriebssysteme (etwa die FAT-, VFAT- und NTFS-Dateisysteme von Microsoft Windows) entweder auf lokalen Platten oder via Netzwerk auf entfernten Servern. Linux selbst kann als Dateiserver in Linux-, Unix- und Windows-Netzen fungieren. Dateisysteme
- Der TCP/IP-Stack von Linux ist anerkanntermaßen mit der leistungsfähigste in der Industrie (was daran liegt, dass ein großer Teil der Forschung und Entwicklung in diesem Bereich auf der Basis von Linux erbracht wird). Er unterstützt IPv4 und IPv6 und alle wichtigen Optionen und Protokolle. TCP/IP-Stack
- Linux bietet leistungsfähige und elegante Grafikumgebungen für die tägliche Arbeit und mit X11 ein sehr verbreitetes netzwerktransparentes Grafik-Basissystem. Auf den gängigen Grafikkarten wird 3D-beschleunigte Grafik unterstützt. Grafikumgebungen
- Alle wichtigen »Produktivitätsanwendungen« stehen zur Verfügung – Office-Programme, Web-Browser, Programme zum Zugriff auf elektronische Post und andere Kommunikationsmedien, Multimedia-Programme, Entwicklungsumgebungen für die verschiedensten Programmiersprachen und noch vieles mehr. Die meisten dieser Programme werden ohne Zusatzkosten mitgeliefert oder lassen sich problemlos und günstig über das Internet beziehen. Dasselbe gilt für Server für alle wichtigen Internet-Protokolle und für unterhaltsame Spiele. Software

Die Flexibilität von Linux bewirkt, dass das System nicht nur auf allen möglichen Rechnern der PC-Klasse eingesetzt werden kann (auch »alte Möhren«, auf denen kein aktuelles Windows mehr läuft, können noch im Kinderzimmer oder als Dateiserver, Router oder Mailserver gute Dienste leisten), sondern sich auch im »Embedded-Bereich«, also für fertige Geräte der Netzinfrastruktur oder Un- Embedded-Bereich

terhaltungselektronik, immer weiter verbreitet. Sie finden Linux zum Beispiel in der FRITZ!Box von AVM und ähnlichen WLAN-, DSL- und Telefoniegeräten, in diversen Set-Top-Boxen für das digitale Fernsehen, in digitalen Videorecordern, Digitalkameras, Kopierern und vielen anderen Geräten. Sogar auf Pfandflaschen-Automaten im Supermarkt hat der Autor dieser Zeilen schon Linux booten gesehen. Oft wird das nicht an die große Glocke gehängt, aber die Hersteller schätzen neben der Leistung und Bequemlichkeit von Linux an sich auch die Tatsache, dass bei Linux keine Lizenzkosten »pro verkauftem Gerät« anfallen wie bei vergleichbaren Betriebssystemen.

Sicherheitslücken

Ein weiterer Pluspunkt für Linux und freie Software ist die Weise, wie in der Szene mit Sicherheitslücken umgegangen wird. Sicherheitslücken sind in freier wie auch proprietärer Software einigermaßen unvermeidlich – jedenfalls hat noch niemand ein Programm interessanter Größe geschrieben und in Umlauf gebracht, das auf lange Sicht völlig frei von ihnen gewesen wäre. Insbesondere kann man auch nicht sagen, dass freie Software keine Sicherheitslücken hat. Die Unterschiede sind eher auf der philosophischen Ebene zu suchen:

- Ein Anbieter proprietärer Software hat in der Regel kein großes Interesse daran, Sicherheitslücken in seinem Code zu reparieren – er wird so lange wie irgend möglich versuchen, Probleme zu vertuschen und die möglichen Gefahren abzuwiegeln, denn im besten Fall bedeutet das ständige Veröffentlichen von »Patches« für Sicherheitsprobleme schlechte PR (»wo Rauch ist, ist auch Feuer«; die Konkurrenz, die gerade mal nicht im Rampenlicht steht, lacht sich ins Fäustchen), und im schlimmsten Fall hohe Kosten und jede Menge Ärger, wenn schädlicher Code im Umlauf ist, der die Sicherheitslücken ausnutzt. Außerdem gibt es wie üblich die Gefahr, bei der Korrektur eines Fehlers drei neue einzubauen, weswegen das Reparieren von Fehlern in veröffentlichter Software sich betriebswirtschaftlich nicht wirklich rechnet.

- Ein Anbieter freier Software gewinnt nichts dadurch, Informationen über Sicherheitslücken zu unterdrücken, denn der Quellcode steht allgemein zur Verfügung und jeder kann die Probleme nachvollziehen. Es ist eher eine Frage des Stolzes, bekannte Sicherheitslücken möglichst schnell in Ordnung zu bringen. Die Tatsache, dass der Quellcode allgemein zur Verfügung steht, bedeutet auch, dass Dritte es leicht haben, den Code auf Probleme zu untersuchen, die dann proaktiv repariert werden können. (Es wird gerne postuliert, dass die Verfügbarkeit des Quellcodes auch Cracker und ähnliches zwielichtiges Gesindel geradezu anlockt. Tatsache ist aber, dass solche Gestalten anscheinend kein gravierendes Problem damit haben, in proprietären Systemen wie Windows, wo der Quellcode *nicht* zur Verfügung steht, große Mengen von Sicherheitslücken zu finden. Der Unterschied ist also, wenn er überhaupt vorhanden ist, nicht gravierend.)

- Speziell im Fall von Software, die sich mit Kryptografie (also dem Verschlüsseln und Entschlüsseln vertraulicher Informationen) befasst, läßt sich argumentieren, dass die Verfügbarkeit des Quellcodes eine unabdingbare Vorbedingung dafür ist, Vertrauen aufbauen zu können, dass ein Programm tatsächlich das tut, was es soll, also das behauptete Verschlüsselungsverfahren vollständig und korrekt implementiert. Hier hat Linux eindeutig die Nase vorn.

Linux in Firmen

Linux wird heute weltweit sowohl im privaten als auch im professionellen Bereich (Firmen, Forschungseinrichtungen, Hochschulen) eingesetzt. Eine besondere Rolle spielt es dabei als System für Webserver (Apache), Mailserver (Sendmail, Postfix), Fileserver (NFS, Samba), Print-Server (LPD, CUPS), ISDN-Router, X-Terminal, Workstation usw. Linux ist aus den Rechenzentren der Industrie nicht mehr

Öffentliche Verwaltung

wegzudenken. Auch die Entscheidung von Kommunen wie der Stadt München, die Rechner der öffentlichen Verwaltung sehr weitgehend auf Linux umzustellen,

Unterstützung durch IT-Firmen

setzt Zeichen. Dazu kommt, dass namhafte IT-Firmen wie IBM, Hewlett-Packard,

Dell, Oracle, Sybase, Informix, SAP, Lotus etc. ihre Produkte auf Linux abstimmen oder in unterstützten Versionen für Linux anbieten. Ferner werden mehr und mehr Rechner von Haus aus mit Linux ausgeliefert oder zumindest vom Hersteller auf Linux-Kompatibilität getestet.

Übungen

1.6 [4] Stellen Sie sich vor, Sie sind verantwortlich für die EDV einer kleinen Firma (20–30 Mitarbeiter). Im Büro gibt es etwa 20 Büroarbeitsplätze und zwei Server (einen Datei- und Druckerserver sowie einen Mailserver bzw. Web-Proxyserver). Bisher läuft alles unter Windows. Betrachten Sie die folgenden Szenarien:

- Der Datei- und Druckerserver wird durch einen Linux-Rechner mit Samba (einem Linux/Unix-basierten Serverprogramm für Windows-Clients) ersetzt.
- Der Mailserver bzw. Web-Proxyserver wird durch einen Linux-Rechner ersetzt.
- Die 20 Büroarbeitsplätze werden durch Linux-Rechner ersetzt.

Kommentieren Sie die verschiedenen Szenarien und stellen Sie kurze Listen der Vor- und Nachteile auf.

1.6 Linux-Distributionen

Linux im engeren Sinne umfaßt nur den Betriebssystem-Kern. Um damit arbeiten zu können, benötigen Sie noch eine Vielzahl an System- und Anwendungsprogrammen, Bibliotheken, Dokumentationen usw. »Distributionen« sind nichts anderes als eine aktuelle Auswahl davon zusammen mit eigenen Programmen (insbesondere Werkzeugen zum Installieren und Administrieren), die von Firmen oder anderen Organisationen vertrieben werden, möglicherweise zusammen mit weiteren Leistungen wie Unterstützung, Dokumentation oder Aktualisierungen. Unterschiede gibt es vor allem in der Programmauswahl, den Administrationswerkzeugen, den Zusatzleistungen und dem Preis.

Distributionen

»Fedora« ist eine frei verfügbare Linux-Distribution, die unter der Federführung der amerikanischen Firma *Red Hat* entwickelt wird. Sie ist der Nachfolger der »Red Hat Linux«-Distribution; die Firma Red Hat hat sich aus dem Privatkundengeschäft zurückgezogen und zielt mit ihren Distributionen unter dem Namen »Red Hat« auf Firmenkunden. Red Hat wurde 1993 gegründet und ging im August 1999 an die Börse; das erste Red-Hat-Linux kam im Sommer 1994 heraus, das letzte (die Version 9) Ende April 2004. »Red Hat Enterprise Linux« (RHEL), das aktuelle Produkt, erschien zum ersten Mal im März 2002. Fedora ist, wie gesagt, ein frei verfügbares Angebot und dient als Entwicklungsplattform für RHEL, es ist unter dem Strich der Nachfolger von Red Hat Linux. Red Hat macht Fedora nur zum Download verfügbar; während Red Hat Linux noch als »Kiste« mit CDs und Handbüchern verkauft wurde, überläßt Red Hat das jetzt Drittanbietern.

Red Hat und Fedora

Die Firma SUSE wurde 1992 unter dem Namen »Gesellschaft für Software- und System-Entwicklung« als Unix-Beratungshaus gegründet und schrieb sich entsprechend zuerst »S.u.S.E.«. Eines ihrer Produkte war eine an den deutschen Markt angepasste Version der Linux-Distribution Slackware (von Patrick Volkerding), die ihrerseits von der ersten kompletten Linux-Distribution, *Softlanding Linux System* oder SLS, abgeleitet war. S.u.S.E. Linux 1.0 erschien 1994 und differenzierte sich langsam von Slackware, indem beispielsweise Eigenschaften von Red Hat Linux wie die RPM-Paketverwaltung oder die `/etc/sysconfig`-Datei übernommen wurden. Die

SUSE

erste S.u.S.E.-Linux-Version, die nicht mehr wie Slackware aussah, war die 4.2 von 1996. SuSE (die Punkte wurden irgendwann weggelassen) eroberte bald die Marktführerschaft im deutschsprachigen Raum und veröffentlichte SuSE Linux als »Kiste« in zwei Geschmacksrichtungen, »Personal« und »Professional«; letztere war merklich teurer und enthielt unter anderem mehr Software aus dem Server-Bereich. Wie Red Hat bot SuSE auch ein Unternehmens-Linux an, den *SuSE Linux Enterprise Server* (SLES) mit einigen Ablegern wie einem speziellen Server für Mail und Groupware (den *SuSE Linux OpenExchange Server* oder SLOX). Außerdem bemühte sich SuSE darum, ihre Distribution auch auf den MDT- und Großrechnern von IBM zur Verfügung zu stellen.

Übernahme durch Novell

Im November 2003 kündigte die US-amerikanische Softwarefirma Novell an, die SuSE für 210 Millionen Dollar übernehmen zu wollen; der Handel wurde dann im Januar 2004 perfekt gemacht. (Bei dieser Gelegenheit wurde auch das »U« groß gemacht.) Auch SUSE hat inzwischen wie Red Hat den Schritt gemacht, die »Privatkunden«-Distribution zu öffnen und als »openSUSE« frei verfügbar zu machen (die früheren Versionen erschienen immer erst mit einigen Monaten Verzögerung zum Download). Im Gegensatz zu Red Hat bietet Novell/SUSE aber nach wie vor eine »Kiste« an, die zusätzlich proprietäre Software enthält. Verkauft werden außerdem derzeit unter anderem der SLES und eine Desktop-Plattform für Unternehmen, der *SUSE Linux Enterprise Desktop* (SLED).

Attachmate
Micro Focus

Anfang 2011 wurde Novell von der Firma Attachmate übernommen, die 2014 selber von Micro Focus gekauft wurde. Beides sind Firmen, die vor allem in der Welt der traditionellen Großrechner aktiv sind und sich im Linux- und Open-Source-Bereich bisher nicht besonders hervorgetan haben. Auf SUSE an sich und deren Produkte haben diese Manöver soweit keinen großen Einfluss gehabt.

YaST

Bezeichnendes Merkmal der SUSE-Distributionen ist der »YaST«, ein umfassendes grafisch orientiertes Systemverwaltungswerkzeug.

Debian-Projekt

Im Gegensatz zu den beiden großen Linux-Distributionsfirmen Red Hat und Novell/SUSE ist das **Debian-Projekt** ein Zusammenschluss von Freiwilligen, die es sich zum Ziel gesetzt haben, eine hochwertige Linux-Distribution unter dem Namen *Debian GNU/Linux* frei zur Verfügung zu stellen. Das Debian-Projekt wurde am 16. August 1993 von Ian Murdock angekündigt; der Name ist eine Zusammensetzung seines Vornamens mit dem seiner damaligen Freundin (jetzt Ex-Frau) Debra (und wird darum »Debb-Ian« ausgesprochen). Inzwischen umfasst das Projekt über 1000 Freiwillige.

Grundlage

Grundlage von Debian sind drei Dokumente:

- Die *Debian Free Software Guidelines* (DFSG) definieren, welche Software im Sinne des Projekts als »frei« gilt. Das ist wichtig, denn nur DFSG-freie Software kann Teil der eigentlichen Debian-GNU/Linux-Distribution sein. Das Projekt vertreibt auch nichtfreie Software, diese ist jedoch auf den Distributionsservern strikt von der DFSG-freien Software getrennt: Letztere steht in einem Unterverzeichnis `main`, erstere in `non-free`. (Es gibt auch noch ein Mittelding namens `contrib`; dort findet sich Software, die für sich genommen DFSG-frei wäre, aber nicht ohne andere nichtfreie Komponenten funktioniert.)
- Der *Social Contract* (»Gesellschaftsvertrag«) beschreibt die Ziele des Projekts.
- Die *Debian Constitution* (»Verfassung«) beschreibt die Organisation des Projekts.

Versionen

Zu jedem Zeitpunkt existieren mindestens drei Versionen von Debian

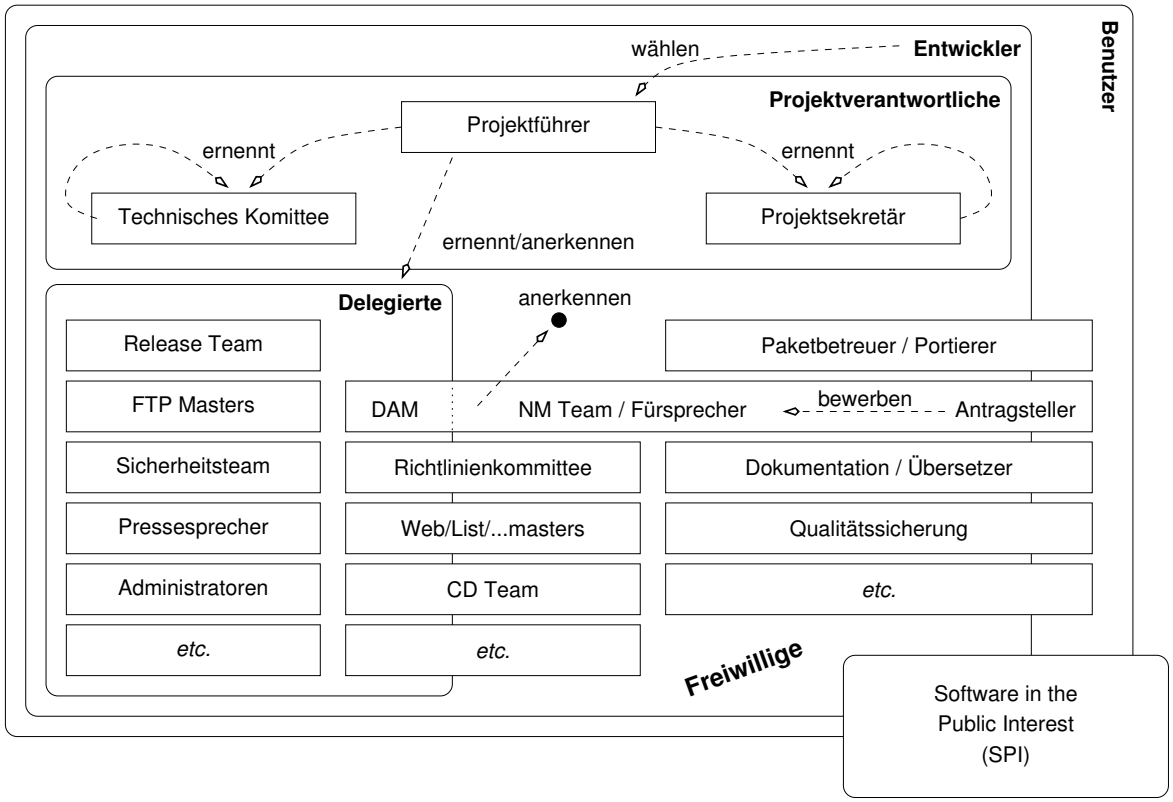


Bild 1.3: Organisationsstruktur des Debian Projekts. (Grafik von Martin F. Krafft.)

GNU/Linux: In den `unstable`-Zweig werden neue oder korrigierte Versionen von Paketen eingebracht. Tauchen in einem Paket keine gravierenden Fehler auf, wandert es nach einer gewissen Wartezeit in den `testing`-Zweig. In gewissen Abständen wird der Inhalt von `testing` »eingefroren«, gründlich getestet und schließlich als `stable` freigegeben. Ein häufig geäußerter Kritikpunkt an Debian GNU/Linux sind die langen Zeiträume zwischen `stable`-Releases; dies wird allerdings von vielen auch als Vorteil empfunden. Debian GNU/Linux wird vom Projekt ausschließlich zum Download zur Verfügung gestellt; Datenträger sind von Drittanbietern erhältlich.

Debian GNU/Linux ist durch seine Organisation, die weitgehende Abwesenheit kommerzieller Interessen und die saubere Trennung von freier und nichtfreier Software eine gute Grundlage für Ableger-Projekte. Einige populäre solche Projekte sind Knoppix (eine »Live-CD«, die es möglich macht, Linux auf einem PC zu testen, ohne es zuerst installieren zu müssen), SkoleLinux (ein speziell auf die Anforderungen von Schulen ausgerichtetes Linux) oder kommerzielle Distributionen wie Xandros. Auch Limux, das Münchner Desktop-Linux, basiert auf Debian GNU/Linux.

Ableger-Projekte

Einer der populärsten Debian-Ableger ist Ubuntu, das von der britischen Firma Canonical Ltd. des südafrikanischen Unternehmers Mark Shuttleworth angeboten wird. (»Ubuntu« ist ein Wort aus der Zulu-Sprache und steht in etwa für »Menschlichkeit gegenüber anderen«.) Das Ziel von Ubuntu ist es, auf der Basis von Debian GNU/Linux ein aktuelles, leistungsfähiges und verständliches Linux anzubieten, das in regelmäßigen Abständen erscheint. Dies wird zum Beispiel dadurch erreicht, dass Ubuntu im Gegensatz zu Debian GNU/Linux nur drei statt über zehn Rechnerarchitekturen unterstützt und sich auf eine Teilmenge der in Debian GNU/Linux angebotenen Software beschränkt. Ubuntu basiert auf dem `unstable`-Zweig von Debian GNU/Linux und verwendet in weiten Teilen dieselben Werk-

Ubuntu

Ziel von Ubuntu

zeuge etwa zur Softwareverteilung, allerdings sind Debian- und Ubuntu-Softwarepakete nicht notwendigerweise miteinander kompatibel.

Ubuntu vs. Debian

Einige Ubuntu-Entwickler sind auch im Debian-Projekt aktiv, so dass es einen gewissen Austausch gibt. Andererseits sind nicht alle Debian-Entwickler begeistert von den Abkürzungen, die Ubuntu zuweilen im Namen des Pragmatismus nimmt, wo Debian vielleicht nach tragfähigeren, aber aufwendigeren Lösungen suchen würde. Ubuntu fühlt sich auch nicht im selben Maße der Idee der freien Software verpflichtet; während alle Infrastrukturwerkzeuge von Debian (etwa das Verwaltungssystem für Fehlerberichte) als freie Software zur Verfügung stehen, ist das für die von Ubuntu nicht immer der Fall.

Ubuntu vs. SUSE/Red Hat

Ubuntu will nicht nur ein attraktives Desktop-System anbieten, sondern auch im Server-Bereich mit den etablierten Systemen wie RHEL oder SLES konkurrieren, also stabile Distributionen mit langem Lebenszyklus und guter Wartung anbieten. Es ist nicht klar, wie Canonical Ltd. auf lange Sicht Geld zu verdienen gedenkt; einstweilen wird das Projekt vor allem aus Mark Shuttleworths Schatulle unterstützt, die seit dem Verkauf seiner Internet-Zertifizierungsstelle Thawte an Verisign gut gefüllt ist …

Weitere Distributionen

Außer diesen Distributionen gibt es noch viele weitere, etwa Mageia oder Linux Mint als kleinere »allgemein nützliche« Distributionen, zahlreiche »Live-Systeme« für verschiedene Zwecke von der Firewall bis hin zur Spiele- oder Multimedia-Plattform sowie sehr kompakte Systeme, die als Router, Firewall oder Rettungssystem einsetzbar sind.

Gemeinsamkeiten

Obwohl es eine Unzahl an Distributionen gibt, verhalten sie sich in der täglichen Arbeit recht ähnlich. Das liegt zum einen daran, dass sie die gleichen grundlegenden Programme benutzen – beispielsweise ist der Kommandozeileninterpreter fast immer die `bash`. Zum anderen gibt es Standards, die dem Wildwuchs entgegenwirken. Zu nennen sind hier der *Filesystem Hierarchy Standard* (FHS) oder die *Linux Standard Base* (LSB).

Übungen

1.7 [2] Einige Linux-Hardwareplattformen wurden oben aufgezählt. Für welche Plattformen sind tatsächlich Linux-Distributionen auf dem Markt? (Tipp: `http://www.distrowatch.org/`)

Zusammenfassung

- Linux ist ein Unix-artiges Betriebssystem.
- Die erste Version von Linux wurde von Linus Torvalds entwickelt und als »freie Software« im Internet zur Verfügung gestellt. Heute wirken Hunderte von Entwicklern weltweit an der Aktualisierung und Erweiterung des Systems mit.
- Die GPL ist die bekannteste Freie-Software-Lizenz. Sie strebt danach, dass die Empfänger von Software diese modifizieren und weiterverteilen dürfen, diese »Freiheiten« dabei aber auch für künftige Empfänger erhalten bleiben. Verkaufen darf man GPL-Software trotzdem.
- »Open Source« bedeutet für den Anwender im Wesentlichen dasselbe wie »Freie Software«.
- Neben der GPL gibt es auch andere freie Lizenzen. Software kann vom Urheber auch unter mehreren verschiedenen Lizenzen gleichzeitig verteilt werden.
- Linux ist eigentlich nur der Betriebssystemkern. Man unterscheidet »stabile« und »Entwicklerkerne«. Stabile Kerne sind für Endanwender gedacht, während Entwicklerkerne nicht funktionieren müssen und Interimsversionen der Linux-Weiterentwicklung darstellen.
- Es gibt zahlreiche Linux-Distributionen, die einen Linux-Kern und zusätzliche Software vereinigen, dokumentieren und bequem installier- und administrierbar machen sollen.

Literaturverzeichnis

DFSG »Debian Free Software Guidelines«. `http://www.debian.org/social_contract`

GPL-Urteil06 Landgericht Frankfurt am Main. »Urteil 2-6 0 224/06«, Juli 2006. `http://www.jbb.de/urteil_lg_frankfurt_gpl.pdf`

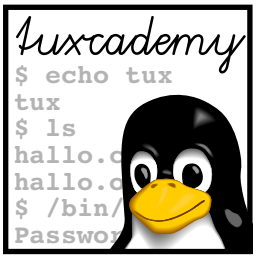
GPL91 Free Software Foundation, Inc. »GNU General Public License, Version 2«, Juni 1991. `http://www.gnu.org/licenses/gpl.html`

LR89 Don Libes, Sandy Ressler. *Life with UNIX: A Guide for Everyone*. Prentice-Hall, 1989. ISBN 0-13-536657-7.

Rit84 Dennis M. Ritchie. »The Evolution of the Unix Time-sharing System«. *AT&T Bell Laboratories Technical Journal*, Oktober 1984. **63**(6p2):1577–93. `http://cm.bell-labs.com/cm/cs/who/dmr/hist.html`

RT74 Dennis M. Ritchie, Ken Thompson. »The Unix Time-sharing System«. *Communications of the ACM*, Juli 1974. **17**(7):365–73. Der klassische Artikel über Unix.

TD01 Linus Torvalds, David Diamond (Hg.) *Just for Fun: Wie ein Freak die Computerwelt revolutionierte*. Hanser Fachbuch, 2001. ISBN 3-446-21684-7.



2

Die Bedienung des Linux-Systems

Inhalt

2.1 Anmelden und Abmelden. 34
2.2 An- und Ausschalten 36
2.3 Der Systemadministrator 36

Lernziele

- Sich beim System anmelden und abmelden können
- Den Unterschied zwischen normalen Benutzerkonten und dem Systemadministrator kennen

Vorkenntnisse

- Grundlegende Kenntnisse im Umgang mit Computern sind hilfreich

Bild 2.1: Die Anmeldebildschirme einiger gängiger Linux-Distributionen

2.1 Anmelden und Abmelden

Zugangsberechtigung

Das Linux-System unterscheidet verschiedene Benutzer. Konsequenterweise können Sie deswegen nach Einschalten des Rechners mitunter nicht sofort loslegen. Zuerst müssen Sie dem Rechner mitteilen, wer Sie sind – Sie müssen sich anmelden (oder, neudeutsch, »einloggen«). Mit dieser Kenntnis kann das System entscheiden, was Sie dürfen (oder nicht dürfen). Natürlich müssen Sie eine Zugangsberechtigung zum System haben – der Systemverwalter muss Sie als Benutzer eingetragen und Ihnen einen Benutzernamen (zum Beispiel `hugo`) und ein Kennwort (zum Beispiel `geheim`) zugeordnet haben. Das Kennwort soll sicherstellen, dass nur Sie diese Zugangsberechtigung nutzen können; Sie müssen es geheimhalten und sollten es niemandem sonst zugänglich machen. Wer Ihren Benutzernamen und Ihr Kennwort kennt, kann sich gegenüber dem System als Sie ausgeben, alle Ihre Dateien lesen (oder löschen), in Ihrem Namen elektronische Post verschicken und alle möglichen anderen Arten von Schindluder treiben.

Modernere Linux-Distributionen möchten es Ihnen leicht machen und erlauben es, auf einem Rechner, den sowieso nur Sie benutzen, den Anmeldevorgang zu überspringen. Wenn Sie so ein System verwenden, dann müssen Sie sich nicht explizit anmelden, sondern der Rechner startet direkt in Ihre Benutzersitzung. Sie sollten das natürlich nur ausnutzen, wenn Sie nicht damit rechnen müssen, dass Dritte Zugang zu Ihrem Rechner haben; verkneifen Sie sich das vor allem auf Notebooks und anderen mobilen Systemen, die gerne mal verloren gehen oder gestohlen werden.

Anmelden in einer grafischen Umgebung Linux-Arbeitsplatzrechner bieten Ihnen heutzutage, wie sich das gehört, eine grafische Umgebung an, und auch der Anmeldevorgang spielt sich normalerweise auf dem Grafikbildschirm ab. Ihr Rechner präsentiert Ihnen einen Dialog, wo Sie Ihren Benutzernamen und Ihr Kennwort eingeben können (Bild 2.1 zeigt einige repräsentative Beispiele.)

Wundern Sie sich nicht, wenn Sie beim Eingeben Ihres Kennworts nur Sternchen sehen. Das bedeutet nicht, dass Ihr Rechner Ihre Eingabe missversteht, sondern dass er es Leuten schwerer machen möchte, die Ihnen beim Tippen über die Schulter schauen, um Ihr Kennwort zu erhaschen.

Nach dem Anmelden baut Ihr Rechner eine grafische Sitzung für Sie auf, in der Sie über Menüs und Icons (Sinnbilder auf dem Bildschirmhintergrund) bequemen Zugang zu Ihren Anwendungsprogrammen bekommen. Die meisten grafischen Umgebungen für Linux unterstützen eine »Sitzungsverwaltung«, die dafür sorgt, dass beim Anmelden so weit wie möglich der Zustand wieder hergestellt wird, den Ihre Sitzung hatte, als Sie sich zuletzt abgemeldet haben. Auf diese Weise müssen Sie sich nicht merken, welche Programme Sie laufen hatten, wo deren Fenster auf dem Bildschirm platziert waren, und welche Dateien Sie gerade bearbeitet haben.

Abmelden in einer grafischen Umgebung Wenn Sie mit der Arbeit fertig sind oder den Rechner für einen anderen Benutzer frei machen wollen, müssen Sie sich abmelden. Das ist auch wichtig, damit die Sitzungsverwaltung Ihre aktuelle Sitzung für das nächste Mal sichern kann. Wie das Abmelden im Detail funktioniert, hängt von Ihrer grafischen Umgebung ab, aber in der Regel gibt es irgendwo einen Menüeintrag, der alles für Sie erledigt. Konsultieren Sie im Zweifel die Dokumentation oder fragen Sie Ihren Systemadministrator (oder sich gut auskennenden Kumpel).

Anmelden auf der Textkonsole Im Gegensatz zu Arbeitsplatzrechnern haben Serversysteme oft nur eine Textkonsole oder stehen in zugigen, lauten Rechnerräumen, wo man sich nicht länger aufhalten möchte als nötig, so dass man sich lieber über das Netz anmeldet, um auf dem Rechner zu arbeiten. In beiden Fällen bekommen Sie keinen grafischen Anmeldebildschirm, sondern der Rechner fragt Sie direkt nach Ihrem Benutzernamen und Kennwort. Zum Beispiel könnten Sie einfach etwas sehen wie

```
rechner login: _
```

(wenn wir mal annehmen, dass der betreffende Rechner »`rechner`« heißt). Hier müssen Sie Ihren Benutzernamen eingeben und mit der Eingabetaste abschließen. Daraufhin erscheint in ähnlicher Form die Frage nach dem Kennwort:

```
Password: _
```

Hier müssen Sie Ihr Kennwort eingeben. (Hier erscheinen normalerweise nicht einmal Sternchen, sondern einfach gar nichts.) Wenn Sie sowohl Benutzername als auch Kennwort korrekt angegeben haben, sollte das System die Anmeldung akzeptieren. Der Kommandozeileninterpreter (die *Shell*) wird gestartet, und Sie können über die Tastatur Kommandos eingeben und Programme aufrufen. Nach der Anmeldung befinden Sie sich automatisch in Ihrem »Heimatverzeichnis«, wo Sie Ihre Dateien finden können.

Wenn Sie zum Beispiel die »Secure Shell« verwenden, um sich auf einem anderen Rechner über das Netz anzumelden, entfällt in der Regel die Frage nach Ihrem Benutzernamen, da das System, wenn Sie nicht ausdrücklich etwas anderes angeben, davon ausgeht, dass Sie auf dem entfernten Rechner denselben Benutzernamen haben wie auf dem Rechner, von dem aus Sie die

Sitzung aufbauen. Die Details würden hier aber zu weit führen; die Secure Shell wird in der Linup-Front-Schulungsunterlage *Linux-Administration II* ausführlich besprochen.

Abmelden auf der Textkonsole Auf der Textkonsole können Sie sich zum Beispiel mit dem Befehl `logout` abmelden:

```
$ logout
```

Auf einer Textkonsole zeigt das System nach dem Abmelden wieder die Startmeldung und eine Anmeldeaufforderung für den nächsten Benutzer. Bei einer Sitzung mit der »Secure Shell« über das Netz bekommen Sie einfach wieder die Eingabeaufforderung Ihres lokalen Rechners.

Übungen

2.1 [!1] Versuchen Sie sich beim System anzumelden. Melden Sie sich anschließend wieder ab. (Einen Benutzernamen und ein Kennwort verrät Ihnen die Dokumentation Ihres Systems oder – im Schulungszentrum – Ihr Trainer.)

2.2 [!2] Was passiert, wenn Sie (a) einen nicht existierenden Benutzernamen, (b) ein falsches Kennwort angeben? Fällt Ihnen etwas auf? Welchen Grund könnte es dafür geben, dass das System sich so verhält, wie es sich verhält?

2.2 An- und Ausschalten

Rechner ausschalten

Einschalten darf einen Linux-Rechner normalerweise jeder, der an den Schalter kommt (lokale Gepflogenheiten können anders aussehen). Allerdings sollten Sie einen Linux-Rechner nicht einfach so ausschalten – es könnten noch Daten im Hauptspeicher stehen, die eigentlich auf die Festplatte gehören und so verlorengehen, oder – was schlimmer wäre – die Daten auf der Festplatte könnten völlig durcheinanderkommen. Außerdem könnten andere Benutzer über das Netz auf dem Rechner angemeldet sein, durch das plötzliche Abschalten überrascht werden und wertvolle Arbeit verlieren. Aus diesem Grund werden wichtige Rechner normalerweise nur vom Systemverwalter »heruntergefahren«. Arbeitsplatzrechner für eine einzige Person dagegen können Sie heutzutage meist auch über die grafische Oberfläche sauber herunterfahren; je nach Systemeinstellung genügen dazu die Privilegien eines normalen Benutzers, oder Sie müssen das Kennwort des Systemverwalters eingeben.

Übungen

2.3 [2] Prüfen Sie, ob Sie Ihr System als normaler Benutzer sauber herunterfahren dürfen, und probieren Sie es gegebenenfalls aus.

2.3 Der Systemadministrator

`root`

Als normaler Benutzer sind Ihre Rechte im System beschränkt. Sie dürfen zum Beispiel in manche Dateien nicht schreiben (eigentlich die meisten, nämlich alle außer Ihren eigenen) und manche Dateien nicht einmal lesen (etwa die Datei, in der die verschlüsselten Kennwörter aller Benutzer stehen). Für Systemverwaltungszwecke gibt es allerdings eine Benutzerkennung, für die diese Einschränkung nicht gilt – der Benutzer »`root`« darf alle Dateien lesen und schreiben und auch sonst diverse Dinge tun, die den anderen Benutzern nicht offenstehen.

Administrator- oder, wie man auch sagt, root-Rechte zu haben ist ein Privileg, andererseits aber auch eine Gefahr – deswegen sollten Sie sich nur als root anmelden, wenn Sie tatsächlich Administrationsaufgaben ausführen müssen, und nicht zum E-Mail-Lesen oder Internet-Surfen.

Stellen Sie sich einfach vor, Sie seien Spider-Man: »Mit großer Macht kommt große Verantwortung.« Auch Spider-Man trägt seinen Elastan-Anzug nur, wenn er muss …

Grafische Oberfläche als root: riskant

Vor allem sollten Sie es vermeiden, sich am grafischen Login als root anzumelden, da dann die gesamte grafische Oberfläche mit root-Rechten läuft, was ein Sicherheitsrisiko darstellt – Grafikoberflächen wie KDE enthalten riesige Mengen Code, die in der Regel nicht so gründlich auf mögliche Sicherheitslücken abgeklopft werden wie die im Vergleich relativ kompakte textorientierte Shell. Üblicherweise können Sie mit dem Befehl »/bin/su -« auf der Kommandozeile die Identität (und damit die Rechte) von root annehmen. su fragt nach dem root-Kennwort und startet dann eine neue Shell, in der Sie so arbeiten können, als hätten Sie sich als root angemeldet. Diese Shell können Sie später mit dem Kommando exit verlassen.

Identität von root annehmen

Sie sollten sich angewöhnen, das Programm su immer über seinen vollständigen Pfadnamen aufzurufen – »/bin/su -«. Ansonsten könnte es sein, dass ein Benutzer Sie foppt, indem er Sie wegen irgendeines unerklärlichen Fehlers an seinen Rechner holt und Sie dazu verleitet, in einem seiner Fenster »su« zu sagen und das root-Kennwort einzugeben. Was Sie in dem Moment nicht wissen, ist, dass der clevere Benutzer selber ein »trojanisches« su-Programm geschrieben hat, das nichts tut außer das eingegebene Kennwort in eine Datei zu schreiben, die Fehlermeldung für vertippte Kennwörter auszugeben und sich selbst zu löschen. Wenn Sie es dann zähneknirschend nochmal probieren, dann bekommen Sie das echte su – und Ihr Benutzer ist fortan im Besitz der begehrten Administratorprivilegien …

Eingabeaufforderung für root

Dass Sie tatsächlich über Systemverwalterprivilegien verfügen, erkennen Sie meist daran, dass die Eingabeaufforderung mit dem Zeichen »#« aufhört. Die Eingabeaufforderung für normale Benutzer endet gemeinhin auf »$« oder »>«.

In Ubuntu können Sie sich standardmäßig gar nicht als root anmelden. Statt dessen erlaubt das System dem bei der Installation als erstes angelegten Benutzer, Kommandos mit Administratorrechten auszuführen, indem er das Kommando sudo davorsetzt. Mit

```
$ sudo chown hugo datei.txt
```

könnte er zum Beispiel die Datei datei.txt dem Benutzer hugo überschreiben – eine Operation, die dem Administrator vorbehalten ist.

Neue Versionen von Debian GNU/Linux bieten ein ähnliches Arrangement an wie Ubuntu.

root und KDE

In der KDE-Oberfläche haben Sie es übrigens ganz einfach, beliebige Programme als root auszuführen: Wählen Sie im »KDE«-Menü – normalerweise der Eintrag ganz links in der Leiste, da wo ein Windows-Rechner das »Start«-Menü hat – den Punkt »Befehl ausführen …«. In dem dann erscheinenden Dialogfenster können Sie einen Befehl eingeben. Bevor Sie ihn ausführen, klicken Sie auf die Schaltfläche »Einstellungen«; es öffnet sich ein Bereich mit zusätzlichen Einstellungen, wo Sie »Mit anderer Benutzerkennung« anklicken (hilfreicherweise ist root der Standardwert). Sie müssen dann nur noch das root-Kennwort im vorgesehenen Feld eingeben (Bild 2.2).

kdesu

Alternativ dazu können Sie im selben Dialogfenster im Kommandofeld auch »kdesu« vor das eigentliche Kommando schreiben. Dann werden Sie im Anschluss nach dem root-Kennwort gefragt.

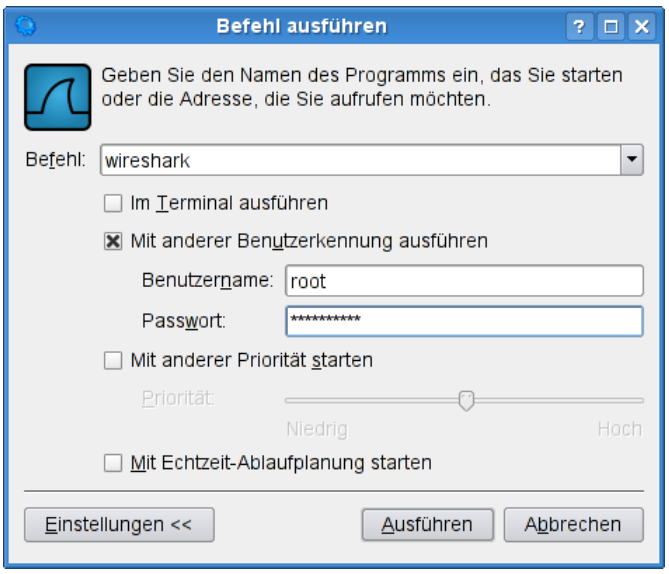


Bild 2.2: Programme als anderer Benutzer ausführen in KDE

Übungen

2.4 [!1] Verwenden Sie das Programm `su`, um Systemverwalterprivilegien zu erlangen. Rufen Sie das Kommando `id` auf, um sich zu überzeugen, dass Sie tatsächlich `root` sind – es sollte etwas ausgeben wie

```
# id
uid=0(root) gid=0(root) groups=0(root)
```

– und wechseln Sie zurück zu Ihrer normalen Benutzerkennung.

2.5 [5] (Für Programmierer.) Schreiben Sie ein überzeugendes »trojanisches« `su`-Programm. Versuchen Sie, Ihren Systemverwalter damit aufs Glatteis zu führen.

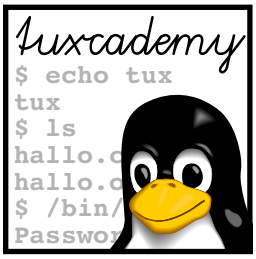
2.6 [2] Versuchen Sie, mit dem »Befehl ausführen …«-Dialog von KDE das Programm `id` in einer Terminalsitzung auszuführen. Kreuzen Sie dazu das entsprechende Feld in den erweiterten Einstellungen an.

Kommandos in diesem Kapitel

exit	Beendet eine Shell	bash(1)	37
id	Gibt UID und GIDs eines Benutzers aus	id(1)	38
kdesu	Startet unter KDE ein Programm als anderer Benutzer	KDE: help:/kdesu	37
logout	Beendet eine Sitzung („Abmelden“)	bash(1)	36
su	Startet eine Shell unter der Identität eines anderen Benutzers	su(1)	37
sudo	Erlaubt normalen Benutzern das Aufrufen bestimmter Kommandos mit Administratorprivilegien	sudo(8)	37

Zusammenfassung

- Vor dem Benutzen eines Linux-Rechners müssen Sie sich (meistens) mit Benutzername und Kennwort anmelden und hinterher wieder abmelden.
- Für den Benutzer `root` gelten die normalen Zugriffsrechte nicht – er darf (tendenziell) alles. Diese Privilegien sollten so sparsam wie möglich eingesetzt werden.
- Sie sollten sich nicht als `root` in der grafischen Oberfläche anmelden, sondern lieber bei Bedarf zum Beispiel mit `su` die Identität des Administrators annehmen.



3

Keine Angst vor der Shell

Inhalt

3.1 Warum? 42
3.2 Was ist die Shell? 42
3.3 Kommandos 44
3.3.1 Wozu Kommandos? 44
3.3.2 Wie sind Kommandos aufgebaut? 44
3.3.3 Arten von Kommandos 45
3.3.4 Noch mehr Spielregeln 46

Lernziele

- Die Vorteile einer textorientierten Kommandooberfläche bewerten können
- Gängige Linux-Shells kennen
- Mit der Bourne-Again-Shell (Bash) arbeiten
- Struktur von Linux-Kommandos verstehen

Vorkenntnisse

- Grundlegende Kenntnisse im Umgang mit Computern sind hilfreich

3.1 Warum?

Mehr als andere moderne Betriebssysteme basiert Linux (wie Unix) auf der Idee, textuelle Kommandos über die Tastatur einzugeben. Manch einem mag das vorsintflutlich vorkommen, vor allem wenn man Systeme wie Windows gewöhnt ist, die dem Publikum seit 15 Jahren oder so einzureden versuchen, dass grafische Benutzungsoberflächen das A und O darstellen. Für viele, die von Windows zu Linux kommen, ist der Fokus auf die Kommandooberfläche zunächst ein »Kulturschock« vergleichbar dem, den ein Mensch des 21. Jahrhunderts erlitte, wenn er plötzlich an den Hof von Kaiser Barbarossa versetzt würde: Kein Internet, schlechte Tischmanieren und furchtbare Zahnärzte!

Allerdings muss man das heute nicht mehr so krass sehen. Zum einen gibt es für Linux grafische Oberflächen, die dem, was Windows oder MacOS X dem Benutzer bieten, in wenig bis nichts nachstehen oder sie in manchen Punkten an Bequemlichkeit und Leistung sogar noch übertreffen. Zum anderen schließen die grafischen Oberflächen und die textorientierte Kommandooberfläche sich ja nicht aus, sondern ergänzen sich hervorragend (gemäß der Philosophie »Das richtige Werkzeug für jede Aufgabe«).

Unter dem Strich bedeutet das nur, dass Sie als angehender Linux-Anwender gut daran tun, sich *auch* mit der textorientierten Kommandooberfläche, bekannt als »Shell«, vertraut zu machen. Natürlich will Sie niemand davon abhalten, eine grafische Oberfläche für alles zu benutzen, was Ihnen einfällt. Die Shell ist aber eine Abkürzung zu zahlreichen extrem mächtigen Operationen, die sich grafisch einfach nicht gut ausdrücken lassen. Die Shell abzulehnen ist äquivalent dazu, sich in der Fahrschule gegen die Verwendung aller Gänge außer des ersten zu sträuben: Sicher, auch im ersten Gang kommt man letztendlich ans Ziel, aber nur vergleichsweise langsam und mit heulendem Motor. Warum also nicht lernen, wie Sie mit Linux so richtig auf die Tube drücken können? Und wenn Sie gut aufpassen, haben wir noch den einen oder anderen Trick für Sie auf Lager.

3.2 Was ist die Shell?

Für den Anwender ist eine direkte Kommunikation mit dem Linux-Betriebssystemkern nicht möglich. Das geht nur über Programme, die ihn über »Systemaufrufe« ansprechen. Irgendwie müssen Sie aber solche Programme starten können. Diese Aufgabe übernimmt die Shell, ein besonderes Anwendungsprogramm, das (meistens) Kommandos von der Tastatur liest und diese – zum Beispiel – als Programmaufrufe interpretiert und ausführt. Die Shell stellt damit eine Art »Oberfläche« für den Computer dar, die das eigentliche Betriebssystem wie eine Nußschale (engl. *shell* – daher der Name) umschließt, das dadurch nicht direkt sichtbar ist. Natürlich ist die Shell nur ein Programm unter vielen, die auf das Betriebssystem zugreifen.

Auch die grafischen Oberflächen heutiger Systeme wie KDE kann man als »Shells« ansehen. Anstatt dass Sie textuelle Kommandos über die Tastatur eingeben, geben Sie eben grafische Kommandos mit der Maus ein – aber so wie die Textkommandos einer bestimmten »Grammatik« folgen, tun die Mauskommandos das auch. Zum Beispiel wählen Sie Objekte durch Anklicken aus und legen dann fest, was mit ihnen gemacht werden soll: Öffnen, Kopieren, Löschen, …

Schon das allererste Unix – Ende der 1960er Jahre – hatte eine Shell. Die älteste Shell, die heute noch außerhalb von Museen zu finden ist, wurde Mitte der 70er Jahre für »Unix Version 7« von Stephen L. Bourne entwickelt. Diese nach ihm benannte Bourne-Shell enthält alle grundlegenden Funktionen und erfreute sich einer weiten Verbreitung, ist heute aber nur mehr sehr selten in ihrer ursprünglichen Form anzutreffen. Daneben zählen die C-Shell, an der Berkeley-Universi-

Bourne-Shell

C-Shell

tät für BSD entwickelt und (extrem vage) an die Programmiersprache C angelehnt, sowie die zur Bourne-Shell weitgehend kompatible, aber mit einem größeren Funktionsumfang ausgestattete Korn-Shell (von David Korn, ebenfalls bei AT&T) zu den klassischen Unix-Shells. Korn-Shell

Standard für Linux-Systeme ist die Bourne-Again-Shell, kurz `bash`. Diese wurde im Rahmen des GNU-Projektes der *Free Software Foundation* von Brian Fox und Chet Ramey entwickelt und vereinigt Funktionen der Korn- und der C-Shell. Bourne-Again-Shell

Über die genannten Shells hinaus gibt es noch zahlreiche andere. Eine Shell unter Unix ist ein Anwendungsprogramm wie jedes andere auch, und Sie brauchen keine besonderen Privilegien, um eine schreiben zu können – Sie müssen sich nur an die »Spielregeln« halten, die bestimmen, wie eine Shell mit anderen Programmen kommuniziert. Shells: normale Programme

Shells können interaktiv aufgerufen werden und akzeptieren dann Kommandos vom Benutzer (typischerweise auf einem irgendwie gearteten »Terminal«). Die allermeisten Shells können ihre Kommandos aber auch aus Dateien lesen, die vorgekochte Befehlsfolgen enthalten. Solche Dateien heißen Shellskripte. Shellskripte

Die Shell übernimmt dabei im Einzelnen folgende Aufgaben:

1. Ein Kommando von der Tastatur (oder aus einer Datei) einlesen.
2. Das eingegebene Kommando überprüfen.
3. Das Kommando direkt ausführen oder das korrespondierende Programm starten.
4. Das Resultat auf dem Bildschirm (oder anderswohin) ausgeben.
5. Weiter bei 1.

Neben dieser Standardfunktion umfasst eine Shell meist noch weitere Elemente wie z. B. eine Programmiersprache. Mit Schleifen, Bedingungen und Variablen sind damit komplexe Befehlsstrukturen realisierbar (normalerweise in Shellskripten, seltener im interaktiven Gebrauch). Weiterhin kann die Shell durch eine ausgefeilte Verwaltung früherer Kommandos dem Anwender das Leben erleichtern. Programmiersprache

Eine Shell können Sie mit dem Kommando `exit` wieder verlassen. Das gilt auch für die Shell, die Sie gleich nach dem Anmelden bekommen haben. Shell verlassen

Obwohl es, wie erwähnt, diverse Shells gibt, konzentrieren wir uns hier auf die Bash als Standard-Shell bei den meisten Linux-Distributionen. Auch die Prüfungen des LPI beziehen sich ausschließlich auf die Bash.

Wenn im System mehrere verschiedene Shells zur Verfügung stehen (der Regelfall), können Sie mit den folgenden Befehlen zwischen diesen umschalten: Wechseln zwischen Shells

`sh` für die klassische Bourne-Shell (falls vorhanden – auf den meisten Linux-Systemen ist auch `sh` eine Bash).

`bash` für die »Bourne-Again-Shell« (Bash).

`ksh` für die Korn-Shell.

`csh` für die C-Shell.

`tcsh` für die »Tenex-C-Shell«, eine erweiterte und verbesserte Version der gewöhnlichen C-Shell. Auf vielen Linux-Systemen ist das Programm `csh` in Wirklichkeit ein Verweis auf die `tcsh`.

Für den Fall, dass Sie nicht mehr wissen sollten, mit welcher Shell Sie gerade arbeiten, hilft die Eingabe von »`echo $0`«, die so ziemlich überall funktioniert und als Ausgabe die Bezeichnung der Shell liefert.

Übungen

3.1 [2] Wie viele verschiedene Shells sind auf Ihrem Rechner installiert und welche? (*Tipp:* Prüfen Sie die Datei `/etc/shells`.)

3.2 [2] Melden Sie sich ab und wieder an und prüfen Sie die Ausgabe des Kommandos »`echo $0`« in der »Loginshell«. Starten Sie mit dem Kommando »`bash`« eine neue Shell und geben Sie wiederum das Kommando »`echo $0`«. Vergleichen Sie die Ausgabe der beiden Kommandos. Fällt Ihnen etwas auf?

3.3 Kommandos

3.3.1 Wozu Kommandos?

Die Arbeitsweise eines Rechners, ganz egal, welches Betriebssystem sich darauf befindet, lässt sich allgemein in drei Schritten beschreiben:

1. Der Rechner wartet auf eine Eingabe durch den Benutzer.
2. Der Benutzer legt ein Kommando fest und gibt dieses per Tastatur oder per Maus ein.
3. Der Rechner führt die erhaltene Anweisung aus.

In einem Linux-System zeigt die Shell eine Eingabeaufforderung (engl. *prompt*) auf dem Textbildschirm oder in einem grafischen Terminalprogramm wie `xterm` oder `konsole`. an. Das bedeutet, dass sie dazu bereit ist, einen Befehl entgegenzunehmen. Diese Eingabeaufforderung setzt sich oft aus Benutzer- und Rechnernamen, dem aktuellen Verzeichnis und einem abschließenden Zeichen zusammen.

```
hugo@red:/home > _
```

In diesem Beispiel befindet sich also der Benutzer `hugo` auf dem Rechner `red` im Verzeichnis `/home`. (Aus Platzgründen und um den Text nicht zu sehr auf eine spezielle Linux-Distribution auszurichten, verwenden wir in diesen Unterlagen die neutrale, traditionelle Eingabeaufforderung »`$ `«.)

3.3.2 Wie sind Kommandos aufgebaut?

Ein Kommando in der Shell ist grundsätzlich eine Folge von Zeichen, die durch die Eingabetaste (»Return«) abgeschlossen und danach von der Shell ausgewertet wird. Diese Kommandos sind zumeist vage der englischen Sprache entlehnt und ergeben in ihrer Gesamtheit eine eigene »Kommandosprache«. Kommandos in dieser Sprache müssen gewissen Regeln, einer Syntax, gehorchen, damit sie von der Shell verstanden werden können.

Syntax

Um eine Eingabezeile zu interpretieren, versucht die Shell zunächst, die Eingabezeile in einzelne Wörter aufzulösen. Als Trennelement dient dabei, genau wie im richtigen Leben, das Leerzeichen. Bei dem ersten Wort in der Zeile handelt es sich normalerweise um das eigentliche Kommando. Alle weiteren Wörter in der Kommandozeile sind Parameter, die das Kommando genauer spezifizieren.

Wörter

Erstes Wort: Kommando

Parameter

Für DOS- und Windows-Anwender ergibt sich hier ein möglicher Stolperstein, da die Shell zwischen Groß- und Kleinschreibung unterscheidet. Linux-Kommandos werden in der Regel komplett klein geschrieben (Ausnahmen bestätigen die Regel) und nicht anders verstanden. Siehe hierzu auch Abschnitt 3.3.4.

Groß- und Kleinschreibung

Beim Trennen von Wörtern in einem Kommando ist ein einzelnes Leerzeichen so gut wie viele – die Shell macht da keinen Unterschied. Genaugenommen besteht die Shell nicht mal auf Leerzeichen; Tabulatorzeichen sind

auch erlaubt, was allerdings vor allem beim Lesen von Kommandos aus Dateien von Bedeutung ist, denn die Bash läßt Sie nicht ohne weiteres Tabulatorzeichen eintippen.

Sie können sogar den Zeilentrenner (↩) verwenden, um ein langes Kommando auf mehrere Eingabezeilen zu verteilen, aber Sie müssen direkt davor ein »\« setzen, damit die Shell das Kommando nicht vorzeitig für vollständig hält.

Die an ein Kommando übergebenen Parameter können grob in zwei Klassen eingeteilt werden:

- Parameter, die mit einem Minuszeichen »-« beginnen, werden Optionen genannt. Diese können angegeben werden, aber müssen meistens nicht – die Details hängen vom jeweiligen Kommando ab. Bildlich gesprochen handelt es sich hierbei um »Schalter«, mit denen Sie Aspekte des jeweiligen Kommandos ein- oder ausschalten können. Möchten Sie mehrere Optionen übergeben, können diese (meistens) hinter einem einzigen Minuszeichen zusammengefasst werden, d. h. die Parameterfolge »`-a -l -F`« entspricht »`-alF`«. Viele Programme haben mehr Optionen, als sich leicht merkbar auf Buchstaben abbilden lassen, oder unterstützen aus Gründen der besseren Lesbarkeit »lange Optionen« (oft zusätzlich zu »normalen« Optionen, die dasselbe tun). Lange Optionen werden meist mit zwei Minuszeichen eingeleitet und können nicht zusammengefasst werden: »`bla --fasel --blubb`«. Optionen; lange Optionen
- Parameter ohne einleitendes Minuszeichen nennen wir »Argumente«. Dies sind oft die Namen der Dateien, die mit dem Kommando bearbeitet werden sollen. Argumente

Die allgemeine Befehlsstruktur lässt sich also folgendermaßen darstellen: Befehlsstruktur

- Kommando – »Was wird gemacht?«
- Optionen – »Wie wird es gemacht?«
- Argumente – »Womit wird es gemacht?«

Üblicherweise stehen hinter einem Kommando erst die Optionen und dann kommen die Argumente. Allerdings bestehen nicht alle Kommandos darauf; bei manchen können Sie Argumente und Optionen beliebig mischen, und sie benehmen sich so, als stünden alle Optionen direkt hinter dem Kommando. Bei anderen dagegen werden Optionen erst in dem Moment beachtet, wo das Kommando beim Abarbeiten der Kommandozeile auf sie stößt.

Die Kommandostruktur heutiger Unix-Systeme (einschließlich Linux) ist über fast 40 Jahre historisch gewachsen und weist daher diverse Inkonsistenzen und kleine Überraschungen auf. Wir finden auch, dass man da mal gründlich aufräumen müßte, aber Shellskripte im Werte von 30 Jahren lassen sich leider nicht völlig ignorieren … Seien Sie also gefasst darauf, sich hin und wieder mit gewissen Merkwürdigkeiten anfreunden zu müssen.

3.3.3 Arten von Kommandos

In Shells gibt es prinzipiell zwei Arten von Kommandos:

Interne Kommandos Diese Befehle werden von der Shell selbst zur Verfügung gestellt. Zu dieser Gruppe gehören bei der Bash etwa 30 Kommandos, die den Vorteil haben, dass sie besonders schnell ausgeführt werden können. Einige Kommandos (etwa `exit` oder `cd`) können nur als interne Kommandos realisiert werden, weil sie den Zustand der Shell selbst beeinflussen.

Externe Kommandos Diese Kommandos führt die Shell nicht selber aus, sondern startet dafür ausführbare Dateien, die im Dateisystem üblicherweise in Verzeichnissen wie `/bin` oder `/usr/bin` abgelegt sind. Sie können als Benutzer Ihre eigenen Programme der Shell bekannt machen, so dass diese wie alle anderen externen Kommandos ausgeführt werden können.

Extern oder intern? Um die Art eines Kommandos heraus zu finden, können Sie das Kommando `type` benutzen. Übergeben Sie hier den Befehlsnamen als Argument, wird die Art des Kommandos oder der entsprechende Verzeichnispfad ausgegeben, etwa

```
$ type echo
echo is a shell builtin
$ type date
date is /bin/date
```

(`echo` ist ein nettes Kommando, das einfach nur seine Parameter ausgibt:

```
$ echo Ha, wackrer Tell! Das gleicht dem Waidgesellen!
Ha, wackrer Tell! Das gleicht dem Waidgesellen!
```

`date` liefert das aktuelle Datum und die Uhrzeit, gegebenenfalls angepasst an die aktuelle Zeitzone und Spracheinstellung:

```
$ date
Mo 12. Mär 09:57:34 CET 2012
```

Mehr über `echo` und `date` erfahren Sie in Kapitel 8.)

Hilfe Hilfe für die internen Kommandos der Bash erhalten Sie übrigens über das Kommando `help`:

```
$ help type
type: type [-afptP] name [name ...]
    For each NAME, indicate how it would be interpreted if used as a
    command name.

    If the -t option is used, `type' outputs a single word which is one of
    `alias', `keyword', `function', `builtin', `file' or `', if NAME is an
⋖⋖⋖⋖⋖
```

Übungen

3.3 [2] Welche der folgenden Kommandos sind bei der Bash extern und welche intern realisiert: `alias`, `echo`, `rm`, `test`?

3.3.4 Noch mehr Spielregeln

Wie schon weiter oben erwähnt, unterscheidet die Shell bei der Kommandoeingabe Groß- und Kleinschreibung. Das gilt nicht nur für die Kommandos selbst, sondern konsequenterweise auch für Optionen und Parameter (typischerweise Dateinamen).

Leerzeichen Außerdem sollten Sie beachten, dass die Shell bestimmte Zeichen in der Eingabe besonders interpretiert. An erster Stelle ist hier das schon erwähnte Leerzeichen zu nennen, das als Trennzeichen für Wörter auf der Kommandozeile dient. Weitere Zeichen mit Sonderbedeutung sind

```
$&;(){}[]*?!<>«
```

Wenn Sie eines dieser Zeichen verwenden wollen, ohne dass es von der Shell in seiner besonderen Bedeutung interpretiert werden soll, müssen Sie es **maskieren**. Hierfür können Sie den Rückstrich »\« zum Maskieren eines einzelnen Sonderzeichens oder einfache oder doppelte Anführungszeichen (`'…'`, `"…"`) zum Maskieren mehrerer Sonderzeichen verwenden. Zum Beispiel:

Maskierung

```
$ touch 'Neue Datei'
```

Durch die Anführungszeichen bezieht dieses Kommando sich auf eine Datei namens `Neue Datei`. Ohne Anführungszeichen wären zwei Dateien namens `Neue` und `Datei` gemeint.

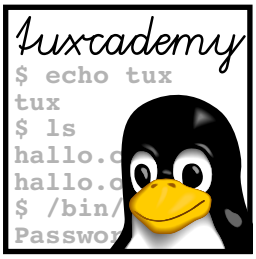
Die anderen Sonderzeichen zu erklären würde an dieser Stelle zu weit führen. Die meisten tauchen anderswo in dieser Schulungsunterlage auf – oder beziehen Sie sich auf die Bash-Dokumentation.

Kommandos in diesem Kapitel

bash	Die „Bourne-Again-Shell“, ein interaktiver Kommandointerpreter	bash(1)	43
csh	Die „C-Shell“, ein interaktiver Kommandointerpreter	csh(1)	43
date	Gibt Datum und Uhrzeit aus	date(1)	46
echo	Gibt alle seine Parameter durch Leerzeichen getrennt auf der Standardausgabe aus	bash(1), echo(1)	46
help	Zeigt Hilfe für bash-Kommandos	bash(1)	46
konsole	Ein „Terminalemulator“ für KDE	KDE: help:/konsole	44
ksh	Die „Korn-Shell“, ein interaktiver Kommandointerpreter	ksh(1)	43
sh	Die „Bourne-Shell“, ein interaktiver Kommandointerpreter	sh(1)	43
tcsh	Die „Tenex-C-Shell“, ein interaktiver Kommandointerpreter	tcsh(1)	43
type	Bestimmt die Art eines Kommandos (intern, extern, Alias)	bash(1)	46
xterm	Ein „Terminalemulator“ für das X-Window-System	xterm(1)	44

Zusammenfassung

- Die Shell liest Benutzerkommandos und führt sie aus.
- Die meisten Shells haben die Eigenschaften von Programmiersprachen und erlauben das Erstellen von Shellskripten, also vorgekochten Kommandofolgen, die in Dateien abgelegt werden.
- Kommandos haben Optionen und Argumente. Die Optionen geben an, *wie* das Kommando wirkt und die Argumente *worauf*.
- Shells unterscheiden zwischen *internen* Kommandos, die in der Shell selbst implementiert sind, und *externen* Kommandos, für die andere Programme als Hintergrundprozesse gestartet werden.



4

Hilfe

Inhalt

4.1 Hilfe zur Selbsthilfe 50
4.2 Der `help`-Befehl und die `--help`-Option 50
4.3 Die Handbuchseiten 51
4.3.1 Überblick 51
4.3.2 Struktur 51
4.3.3 Kapitel 52
4.3.4 Handbuchseiten anzeigen 52
4.4 Die Info-Seiten 53
4.5 Die HOWTOs 54
4.6 Weitere Informationsquellen 54

Lernziele

- Mit Handbuch- und Infoseiten umgehen können
- HOWTOs kennen und finden können
- Die wichtigsten anderen Informationsquellen kennen

Vorkenntnisse

- Linux-Überblick
- Kenntnisse über Linux auf der Kommandozeile (etwa aus den vorangegangenen Kapiteln)

4.1 Hilfe zur Selbsthilfe

Linux ist ein leistungsfähiges und vielschichtiges System, und leistungsfähige und vielschichtige Systeme sind in der Regel komplex. Ein wichtiges Hilfsmittel zur Beherrschung dieser Komplexität ist Dokumentation, und viele (leider nicht alle) Aspekte von Linux sind sehr ausführlich dokumentiert. Dieses Kapitel beschreibt einige Methoden zum Zugriff auf die Dokumentation.

»Hilfe« unter Linux bedeutet in vielen Fällen »Selbsthilfe«. Die Kultur der freien Software beinhaltet auch, dass man die Zeit und das Wohlwollen anderer Leute, die als Freiwillige in der Szene unterwegs sind, nicht unnötig mit Dingen strapaziert, die offensichtlich in den ersten paar Absätzen der Dokumentation erklärt sind. Als Linux-Anwender tun Sie gut daran, zumindest einen Überblick über die vorhandene Dokumentation und die empfohlenen Wege dafür zu haben, wo Sie notfalls Hilfe herbekommen können. Wenn Sie Ihre Hausaufgaben machen, werden Sie normalerweise erfahren, dass Ihnen aus der Klemme geholfen wird, aber die Toleranz gegenüber Leuten, die sich selber auf die faule Haut legen und erwarten, dass andere sich in ihrer Freizeit für sie in Knoten binden, ist nicht notwendigerweise besonders ausgeprägt.

Wenn Sie gerne möchten, dass Ihnen auch für die nicht so gründlich selber recherchierten Fragen und Probleme rund um die Uhr, sieben Tage die Woche, ein offenes Ohr zur Verfügung steht, müssen Sie auf eines der zahlreichen »kommerziellen« Support-Angebote zurückgreifen. Diese stehen für alle namhaften Distributionen zur Verfügung, teils vom Anbieter der Distribution selbst und teils von Drittfirmen. Vergleichen Sie die verschiedenen Dienstleister und suchen Sie sich einen davon aus, dessen Dienstgüteversprechen und Preis Ihnen zusagen.

4.2 Der `help`-Befehl und die `--help`-Option

Interne `bash`-Kommandos

Die in der Shell integrierten Kommandos werden durch den Befehl `help` mit dem entsprechenden Befehlsnamen als Argument genauer beschrieben:

```
$ help exit
exit: exit [n]
    Exit the shell with a status of N.
    If N is omitted, the exit status
    is that of the last command executed.
$ _
```

Ausführliche Erklärungen finden Sie in den Handbuchseiten und der Info-Dokumentation der Shell. Auf diese Informationsquellen gehen wir später in diesem Kapitel ein.

Option `--help` bei externen Kommandos

Viele externe Kommandos (Programme) unterstützen statt dessen die Option `--help`. Daraufhin erscheint bei den meisten Befehlen eine kurze Angabe, die Parameter und Syntax des Kommandos angibt.

Nicht jedes Kommando reagiert auf `--help`; oft heißt die Option `-h` oder `-?`, oder die Hilfe wird ausgegeben, wenn Sie irgendeine ungültige Option oder ansonsten illegale Kommandozeile angeben. Leider gibt es da keine einheitliche Konvention.

Tabelle 4.1: Gliederung der Handbuchseiten

Abschnitt	Inhalt
NAME	Kommandoname mit knapper Funktionsbeschreibung
SYNOPSIS	Beschreibung der Befehlssyntax
DESCRIPTION	Ausführliche Beschreibung der Kommandowirkung
OPTIONS	Die möglichen Optionen
ARGUMENTS	Die möglichen Argumente
FILES	Benötigte bzw. bearbeitete Dateien
EXAMPLES	Beispiele zur Anwendung
SEE ALSO	Querverweise auf verwandte Themen
DIAGNOSTICS	Fehlermeldungen des Kommandos
COPYRIGHT	Autor(en) des Kommandos
BUGS	Bekannte Fehler des Kommandos

4.3 Die Handbuchseiten

4.3.1 Überblick

Zu fast jedem kommandozeilenorientierten Programm gibt es eine »Handbuchseite« (engl. *manual page* oder kurz *manpage*), genau wie für viele Konfigurationsdateien, Systemaufrufe und so weiter. Diese Texte werden in der Regel bei der Installation einer Software mit installiert und können mit dem Kommando »`man` ⟨*Name*⟩« eingesehen werden. ⟨*Name*⟩ ist dabei der Kommando- oder Dateiname, den Sie erklärt haben möchten. »`man bash`« zum Beispiel liefert unter anderem eine Auflistung der oben erwähnten internen Kommandos der Shell.

Kommando `man`

Die Handbuchseiten haben für den Anwender allerdings einige Nachteile: Zum einen liegen viele davon nur in englischer Sprache vor. Lediglich einige Distributionen enthalten deutsche Übersetzungen, die allerdings oftmals recht knapp gehalten sind. Zum anderen sind die Texte oft sehr komplex. Jedes einzelne Wort kann bedeutsam sein, was dem Einsteiger den Zugang natürlich erschwert. Ferner ist gerade bei langen Texten die Aufteilung recht unübersichtlich. Dennoch ist der Wert dieser Dokumentation nicht zu unterschätzen. Statt den Anwender mit einer Unmenge Papier zu überhäufen, ist die Hilfe immer vor Ort verfügbar.

Viele Linux-Distributionen verfolgen die Philosophie, dass es zu jedem auf der Kommandozeile aufrufbaren Programm auch eine Handbuchseite geben muss. Dies gilt leider nicht im selben Umfang für Programme, die zu den grafischen Arbeitsumgebungen KDE und GNOME gehören, von denen viele nicht nur keine Handbuchseite haben, sondern die auch innerhalb der grafischen Umgebung überaus schlecht dokumentiert sind. Der Umstand, dass viele dieser Programme von Freiwilligen erstellt wurden, bietet hierfür nur eine schwache Entschuldigung.

4.3.2 Struktur

Der Aufbau der Handbuchseiten folgt lose der in Tabelle 4.1 angegebenen Gliederung. Allerdings enthält nicht jede Handbuchseite alle Punkte; vor allem die EXAMPLES kommen oft zu kurz.

Gliederung von Handbuchseiten

Die Überschrift BUGS wird gerne missverstanden: Echte *Fehler* in der Implementierung gehören natürlich repariert; was hier dokumentiert wird, sind in der Regel Einschränkungen, die aus dem *Ansatz* des Kommandos folgen, nicht mit vertretbarem Aufwand zu beheben sind und über die Sie als Anwender Bescheid wissen sollten. Beispielsweise wird in der Dokumentation zum Kommando `grep` darauf hingewiesen, dass bestimmte Konstrukte im

Tabelle 4.2: Themenbereiche der Handbuchseiten

Nr.	Themenbereich
1	Benutzerkommandos
2	Systemaufrufe
3	Funktionen der Programmiersprache C
4	Gerätedateien und Treiber
5	Konfigurationsdateien und Dateiformate
6	Spiele
7	Diverses (z. B. `groff`-Makros, ASCII-Tabelle, …)
8	Kommandos zur Systemadministration
9	Kernel-Funktionen
n	»Neue« Kommandos

zu suchenden regulären Ausdruck dazu führen können, dass ein `grep`-Prozess sehr viel Speicher braucht. Dies ist eine Konsequenz daraus, wie `grep` das Suchen implementiert, und kein trivialer, leicht zu behebender Fehler.

Handbuchseiten werden in einem speziellen Format geschrieben, das mit dem Programm `groff` für die Anzeige in einem Textterminal oder den Ausdruck aufbereitet werden kann. Die Quelltexte für die Handbuchseiten liegen im Verzeichnis `/usr/share/man` in Unterverzeichnissen der Form `man`*n*, wobei *n* eine der Kapitelnummern aus Tabelle 4.2 ist.

Handbuchseiten in anderen Verzeichnissen können Sie integrieren, indem Sie die Umgebungsvariable `MANPATH` setzen, die die von `man` durchsuchten Verzeichnisse und deren Reihenfolge benennt. Das Kommando `manpath` gibt Tipps für die `MANPATH`-Einstellung.

4.3.3 Kapitel

Kapitel

Jede Handbuchseite gehört zu einem »Kapitel« im konzeptuellen Gesamthandbuch (Tabelle 4.2). Wichtig sind vor allem die Kapitel 1, 5 und 8. Sie könnnen im `man`-Kommando eine Kapitelnummer angeben, um die Suche einzuschränken. So zeigt zum Beispiel »`man 1 crontab`« die Handbuchseite zum `crontab`-Kommando und »`man 5 crontab`« die Handbuchseite, die das Format von `crontab`-Dateien erklärt. Wenn man auf Handbuchseiten verweist, wird gerne das Kapitel in Klammern angehängt; wir unterscheiden also zwischen `crontab`(1), der Anleitung für das `crontab`-Kommando, und `crontab`(5), der Beschreibung des Dateiformats.

`man -a`

Mit dem Parameter `-a` zeigt `man` die zum Suchbegriff gehörenden Handbuchseiten aller Kapitel nacheinander an, ohne diesen Schalter wird nur der erste gefundene Text, also meist der im Kapitel 1, dargestellt.

4.3.4 Handbuchseiten anzeigen

Anzeigeprogramm

Als Anzeigeprogramm für das Kommando `man` ist in der Regel `less` voreingestellt, das noch ausführlich besprochen wird. Wichtig ist zu diesem Zeitpunkt nur, dass Sie sich mit den Cursortasten ↑ und ↓ durch den Handbuchtext bewegen können. Innerhalb des Textes lässt sich nach Drücken der Taste / ein Stichwort suchen. Nachdem das Wort eingetippt und abschließend die Eingabetaste betätigt wurde, springt der Cursor zum gesuchten Wort – sofern es denn im aktuellen Text vorkommt. Wenn Ihr Wissensdurst gestillt ist, können Sie die Anzeige durch Drücken der Taste q beenden und zur Shell zurückkehren.

Mit dem KDE-Webbrowser Konqueror ist es bequem möglich, ansprechend formatierte Handbuchseiten angezeigt zu bekommen. Geben Sie in der Adresszeile des Browsers den URL »`man:/`⟨*Name*⟩« oder einfach nur »`#`⟨*Name*⟩« an. Dasselbe funktioniert auch in der KDE-Befehlszeile.

Bevor Sie sich planlos durch die unzähligen Dokumentationen arbeiten, ist es oft sinnvoll, allgemeine Informationen zu einem Stichwort mittels `apropos` abzurufen. Dieses Kommando funktioniert genauso wie »`man -k`«. Beide suchen in den »NAME«-Abschnitten aller Handbuchseiten nach dem entsprechenden Stichwort. Als Ausgabe erhalten Sie eine Liste mit allen Handbuchseiten, die einen Eintrag zu dem eingegebenen Thema enthalten. Stichwortsuche

Verwandt ist das Kommando `whatis` (engl. »was ist«). Auch dieses sucht in allen Handbuchseiten, allerdings nicht nach einem Stichwort, sondern nach den eingegebenen Namen. Auf diese Weise erscheint eine kurze Beschreibung des mit dem Namen verbundenen Kommandos, Systemaufrufs o. ä. – eben der zweite Teil des »NAME«-Abschnitts der gefundenen Handbuchseite(n). `whatis` ist äquivalent zu »`man -f`«. `whatis` Namenssuche

Übungen

4.1 [!1] Schauen Sie sich die Handbuchseite zum Programm `ls` an. Benutzen Sie dafür das textbasierte Programm `man` und – falls vorhanden – den Konqueror-Browser.

4.2 [2] Welche Handbuchseiten auf Ihrem System beschäftigen sich (jedenfalls ihrem NAME-Abschnitt nach) mit Prozessen?

4.3 [5] (Für Fortgeschrittene.) Verwenden Sie einen Editor, um eine Handbuchseite für ein hypothetisches Kommando zu schreiben. Lesen Sie dazu vorher die Handbuchseite `man`(7). Prüfen Sie das Aussehen der Handbuchseite auf dem Bildschirm (mit »`groff -Tascii -man` ⟨*Datei*⟩ `| less`«) und in der Druckansicht (mit etwas wie »`groff -Tps -man` ⟨*Datei*⟩ `| gv -`«).

4.4 Die Info-Seiten

Für einige oft komplexere Kommandos existieren ausschließlich oder zusätzlich zur Handbuchseite so genannte Info-Seiten. Sie sind meist ausführlicher und basieren auf den Prinzipien von Hypertext, ähnlich zum World-Wide Web. Hypertext

Die Idee der Info-Seiten stammt aus dem GNU-Projekt, man findet sie daher vor allem bei Software, die von der FSF veröffentlicht wird oder anderweitig zum GNU-Projekt gehört. Ursprünglich sollte es im »GNU-System« *nur* Info-Dokumentation geben; da GNU aber auch zahlreiche Software integriert,

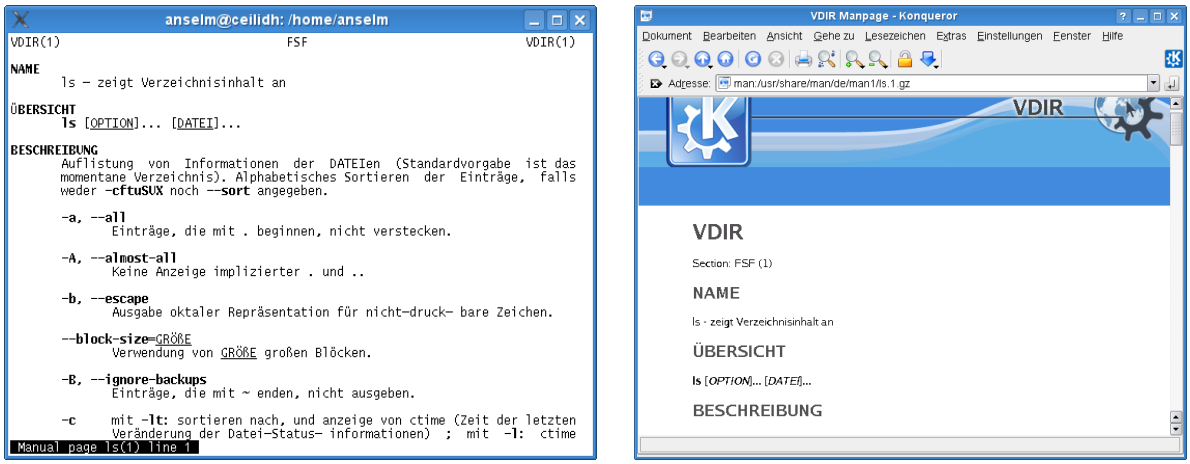


Bild 4.1: Eine Handbuchseite im Textterminal (links) und im Konqueror (rechts)

die nicht unter der Ägide der FSF erstellt wurde, und die GNU-Werkzeuge auch auf Systemen eingesetzt werden, die einen konservativeren Ansatz pflegen, ist die FSF in vielen Fällen weich geworden.

Info-Seiten werden analog zu Handbuchseiten mit dem Befehl »`info` ⟨*Kommando*⟩« aufgerufen (das Paket mit dem `info`-Programm muss möglicherweise zusätzlich installiert werden). Daneben kann man die Infoseiten auch mit dem Editor `emacs` betrachten oder im KDE-Browser Konqueror über URLs der Form »`info:/`⟨*Kommando*⟩« aufrufen.

Ein Vorteil von Info-Seiten ist, dass man sie (ähnlich wie bei den Handbuchseiten) in einem Quellformat schreibt, das bequem automatisch für die Bildschirmanzeige und das Ausdrucken von Handbüchern im PostScript- oder PDF-Format aufbereitet werden kann. Statt `groff` wird hier zur Aufbereitung das Programm TEX verwendet.

Übungen

4.4 [!1] Schauen Sie sich die Info-Seite zum Programm `ls` an. Probieren Sie das textbasierte `info` und, falls vorhanden, den Konqueror-Browser aus.

4.5 [2] Info-Dateien realisieren eine primitive (?) Form von Hypertext, ähnlich den HTML-Seiten im World-Wide Web. Warum werden die Info-Dateien nicht gleich in HTML geschrieben?

4.5 Die HOWTOs

Die Handbuch- und Info-Seiten haben das Problem, dass man im Grunde schon wissen muss, wie das Kommando heißt, das man benutzen möchte. Auch die Suche mit `apropos` ist oft nicht mehr als ein Glücksspiel. Außerdem läßt sich nicht jedes Problem auf ein spezielles Kommando reduzieren. Es besteht also Bedarf für »problemorientierte« statt »kommandoorientierte« Dokumentation. Dafür gibt es die **HOWTOs**.

Problemorientierte Dokumentation

HOWTOs

Die HOWTOs sind umfassendere Dokumente, die nicht nur einzelne Kommandos behandeln, sondern komplette Problemlösungen. So gibt es beispielsweise ein »DSL HOWTO«, das detailliert beschreibt, wie man einen Linuxrechner per DSL ans Internet anbindet, oder ein »Astronomy HOWTO«, das Astronomiesoftware für Linux diskutiert. Viele HOWTOs gibt es auch auf Deutsch; bei Übersetzungen kann es aber sein, dass die deutsche Version hinter dem Original herhinkt. (Einige HOWTOs sind von Anfang an auf Deutsch geschrieben.)

HOWTOs als Paket

Die meisten Linux-Distributionen ermöglichen es, die HOWTOs (oder eine signifikante Teilmenge davon) als Paket zu installieren. Sie befinden sich dann in einem distributionsabhängigen Verzeichnis – bei den SUSE-Distributionen `/usr/share/doc/howto`, bei Debian GNU/Linux `/usr/share/doc/HOWTO` –, typischerweise entweder als einfache Textdokumente oder im HTML-Format. Aktuelle Versionen der HOWTOs und Fassungen in anderen Formaten wie PostScript oder PDF sind im WWW von den Seiten des ***Linux Documentation Project*** (`http://www.tldp.org`) zu beziehen, wo auch andere Linux-Dokumentation zu finden ist.

HOWTOs im Netz

Linux Documentation Project

4.6 Weitere Informationsquellen

Zu (fast) jedem installierten Softwarepaket finden Sie auf Ihrem Rechner unter (typischerweise) `/usr/share/doc` oder `/usr/share/doc/packages` weitere Dokumentation und Beispieldateien. Außerdem gibt es für Programme unter der grafischen Oberfläche (z. B. KDE oder GNOME) entsprechende Hilfemenüs. Viele Distributionen bieten auch spezielle Hilfe-Center an, die bequemen Zugang auf die Dokumentation der verschiedenen Pakete gestatten.

weitere Dokumentation

Unabhängig vom lokalen Rechner gibt es im Internet Unmengen an Dokumentation, unter anderem im WWW und in USENET-Archiven. WWW USENET

Einige interessante Seiten für Linux sind die folgenden:

http://www.tldp.org/ Die Webseiten des »Linux Documentation Project«, das u. a. Handbuchseiten und HOWTOs betreut.

http://www.linux.org/ Ein allgemeines »Portal« für Linux-Interessenten. (Englisch)

http://www.linuxwiki.de/ Eine »Freiform-Text-Informationsdatenbank für alles, was mit GNU/Linux zusammenhängt.«

http://lwn.net/ *Linux Weekly News* – die vermutlich beste Web-Präsenz für Linux-Neuigkeiten aller Art. Neben einer täglichen Übersicht über die neuesten Entwicklungen, Produkte, Sicherheitslücken, Äußerungen pro und contra Linux in der Presse u. ä. erscheint jeden Donnerstag eine umfassende Online-Zeitschrift mit gründlich recherchierten Hintergrundberichten rund um die Vorkommnisse der Woche davor. Die täglichen Neuigkeiten sind frei zugänglich, während die wöchentliche Ausgabe kostenpflichtig abonniert werden muss (verschiedene Preisstufen ab US-$ 5 pro Monat). Eine Woche nach Erscheinen kann man auch auf die wöchentlichen Ausgaben kostenfrei zugreifen. (Englisch)

http://freecode.com/ Hier erscheinen Ankündigungen neuer (vornehmlich freier) Softwarepakete, die meist auch unter Linux zur Verfügung stehen. Daneben gibt es eine Datenbank, in der man nach interessanten Projekten oder Softwarepaketen recherchieren kann. (Englisch)

http://www.linux-knowledge-portal.de/ Eine Seite, die »Schlagzeilen« anderer interessanter Linux-Seiten zusammenträgt (darunter auch *Linux Weekly News* oder *Freshmeat*). (Deutsch/Englisch)

Wenn im WWW oder in USENET-Archiven nichts zu finden ist, gibt es die Möglichkeit, in Mailinglisten oder USENET-Newsgruppen Fragen zu stellen. Dabei sollten Sie jedoch beachten, dass viele Benutzer solcher Foren verärgert reagieren, wenn Sie Fragen posten, deren Antworten auch offensichtlich im Handbuch oder in **FAQ**-Sammlungen (engl. *frequently answered questions*) zu finden sind. FAQ Versuchen Sie, eine Problembeschreibung gründlich vorzubereiten und zum Beispiel mit relevanten Auszügen aus Protokolldateien zu untermauern, ohne die eine »Ferndiagnose« eines komplexen Problems nicht möglich ist (und die nicht komplexen Probleme können Sie ja sicher selber lösen …).

Zugriff auf ein *Newsarchiv* finden Sie u. a. bei `http://groups.google.de/` (ehemals Deja-News)

Interessante *Newsgroups* für Linux finden Sie in englischer Sprache in der `comp.os.linux.*`- und auf Deutsch in der `de.comp.os.unix.linux.*`-Hierarchie. Für viele Linux-Themen sind auch die entsprechenden Unix-Gruppen passend; eine Frage über die Shell steht besser in `de.comp.os.unix.shell` als in `de.comp.os.unix.linux.misc`, weil Shells zumeist keine Linux-spezifischen Programme sind.

Linux-orientierte Mailinglisten gibt es unter anderem bei `majordomo@vger.kernel.org`. Dabei handelt es sich um eine E-Mail-Adresse, an die Sie eine Nachricht mit dem Text »`subscribe LISTE`« schicken müssen, um eine Liste namens LISTE zu abonnieren. Eine kommentierte Aufstellung aller angebotenen Listen finden Sie unter `http://vger.kernel.org/vger-lists.html`. Mailinglisten

Eine probate Strategie zum Umgang mit scheinbar unerklärlichen Problemen besteht darin, die betreffende Fehlermeldung bei Google (oder einer anderen Suchmaschine Ihres Vertrauens) einzugeben. Wenn Sie nicht gleich ein hilfreiches Ergebnis erzielen, dann lassen Sie bei der Anfrage Teile weg, Suchmaschinen

die von Ihrer speziellen Situation abhängen (etwa Dateinamen, die es nur auf Ihrem System gibt). Der Vorteil ist, dass Google nicht nur die gängigen Webseiten indiziert, sondern auch viele Archive von Mailinglisten, und die Chancen gut stehen, dass Sie auf einen Dialog stoßen, wo jemand anderes ein sehr ähnliches Problem hatte wie Sie.

Freie Dokumentation

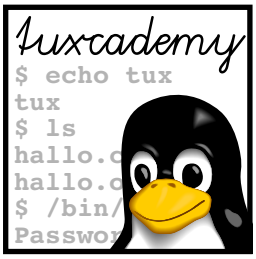
Der große Vorteil an Open-Source-Software ist übrigens nicht nur die riesige Menge an Dokumentation, sondern auch, dass die meisten Dokumente ähnlich wenig Restriktionen unterliegen wie die Software selbst. Dadurch ist eine umfassendere Zusammenarbeit von Softwareentwicklern und Dokumentationsautoren möglich, und auch die Übersetzung von Dokumentation in andere Sprachen ist einfacher. In der Tat gibt es genug Gelegenheit für Nichtprogrammierer, freien Entwicklungsprojekten zuzuarbeiten, indem sie zum Beispiel dabei helfen, gute Dokumentation zu erstellen. Die Freie-Software-Szene sollte versuchen, Dokumentationsautoren dieselbe Achtung entgegenzubringen wie Programmierern – ein Umdenkprozess, der zwar eingesetzt hat, aber noch bei weitem nicht abgeschlossen ist.

Kommandos in diesem Kapitel

apropos	Zeigt alle Handbuchseiten mit dem angegebenen Stichwort im „NAME“-Abschnitt	apropos(1) 52
groff	Programm zur druckreifen Aufbereitung von Texten	groff(1) 52
help	Zeigt Hilfe für bash-Kommandos	bash(1) 50
info	Zeigt GNU-Info-Seiten auf einem Textterminal an	info(1) 54
less	Zeigt Texte (etwa Handbuchseiten) seitenweise an	less(1) 52
man	Zeigt Handbuchseiten des Systems an	man(1) 50
manpath	Bestimmt den Suchpfad für Handbuchseiten	manpath(1) 52
whatis	Sucht Handbuchseiten mit dem gegebenen Stichwort in der Beschreibung	whatis(1) 53

Zusammenfassung

- Interne Kommandos der Bash erklärt »`help` ⟨*Kommando*⟩«. Externe Kommandos unterstützen oft eine Option `--help`.
- Für die meisten Programme gibt es mit `man` abrufbare Handbuchseiten. `apropos` sucht in allen Handbuchseiten nach Stichwörtern, `whatis` nach den Namen von Kommandos.
- Info-Seiten sind für manche Programme eine Alternative zu Handbuchseiten.
- HOWTOs stellen eine problemorientierte Form der Dokumentation dar.
- Es gibt eine Vielzahl interessanter Linux-Ressourcen im World-Wide Web und USENET.



5

Der Editor `vi`

Inhalt

5.1 Editoren 58
5.2 Der Standard – vi 58
5.2.1 Überblick. 58
5.2.2 Grundlegende Funktionen 59
5.2.3 Erweiterte Funktionen 63
5.3 Andere Editoren 65

Lernziele

- Den Editor `vi` kennenlernen
- Textdateien anlegen und verändern können

Vorkenntnisse

- Arbeit mit der Shell (siehe Kapitel 2)

5.1 Editoren

Um Texte komfortabel erfassen oder verändern zu können, stehen auf allen Betriebssystemen entsprechende Hilfsprogramme zur Verfügung. Entsprechend seiner Aufgabe »Textbearbeitung« wird ein derartiges Programm als Editor (lat. *edire*, »bearbeiten«) bezeichnet.

Was tut ein Editor?

Zunächst sollte der Funktionsumfang eines Texteditors genauer umrissen werden. Mit dem einfachen Bearbeiten einzelner Zeichen sind komplexere Aufgaben nur schwer zu meistern. Gute Editoren stellen daher Funktionen zur Verfügung, mit denen gleich ganze Wörter oder Zeilen ausgeschnitten, kopiert oder eingefügt werden können. Für lange Dateien ist ferner ein Mechanismus sinnvoll, welcher den Text nach bestimmten Zeichenfolgen durchforstet. In Erweiterung dazu können ausgefeilte »Suchen und Ersetzen«-Befehle mühsame Aufgaben wie »Ersetze jedes `x` durch ein `u`« spürbar vereinfachen. Je nach Editor stehen noch wesentlich mächtigere Hilfsfunktionen zum Bearbeiten von Texten zur Verfügung.

Unterschied zu Textverarbeitungen

Im Unterschied zu Textverarbeitungen wie OpenOffice.org Writer oder Microsoft Word stellen Editoren jedoch normalerweise keine gestalterischen Elemente wie verschiedene Schriftschnitte (Times, Helvetica, Courier, …), Schriftattribute (fett, kursiv, unterstrichen, …), Satzvorlagen (Blocksatz, …) oder ähnliches zur Verfügung – sie sind vor allem zum Eingeben und Ändern von reinen Textdateien gedacht, wo so etwas eher stören würde.

Natürlich spricht nichts dagegen, mit einem Texteditor Eingabedateien für Textsatzsysteme wie `groff` oder LaTeX zu schreiben, die diese Elemente aus dem FF beherrschen. Nur sehen Sie davon mitunter nichts in der ursprünglichen Eingabe – was auch ein Vorteil sein kann: Schließlich lenken viele dieser Elemente beim Schreiben eher ab, und die Autoren werden verleitet, bei der Texteingabe am Aussehen des Textes herumzubasteln, statt sich auf den Inhalt zu konzentrieren.

Syntaxhervorhebung

Was die meisten aktuellen Texteditoren heutzutage unterstützen, ist Syntaxhervorhebung, also die Kenntlichmachung bestimmter Elemente in einem Programmtext – Kommentare, Variablennamen, Schlüsselwörter, Zeichenketten – durch Farbe oder besondere Schriftarten. Dies sieht auf jeden Fall gefällig aus, wenn auch die Frage, ob das wirklich beim Programmieren hilft, noch nicht durch entsprechende psychologische Studien entschieden werden konnte.

In den nachfolgenden Abschnitten nehmen wir den vielleicht wichtigsten Linux-Editor, `vi`, unter die Lupe. Allerdings beschränkt sich die Betrachtung dabei auf die grundlegenden Funktionsweisen, denn im Prinzip könnte man zu jedem dieser Programme eine mehrtägige Schulung durchführen. Ebenso wie bei den Shells ist es letztendlich dem persönlichen Geschmack des Benutzers überlassen, mit welchem Editor er seine Texte manipuliert.

Übungen

5.1 [2] Welche Texteditoren sind auf Ihrem System installiert? Wie können Sie das herausfinden?

5.2 Der Standard – `vi`

5.2.1 Überblick

Der einzige Editor, der wohl in allen Linux-Systemen installiert ist, heißt `vi` (engl. *visual*, »sichtbar«). Aus praktischen Gründen handelt es sich dabei oftmals nicht um das Originalprogramm (das aus BSD stammt und inzwischen deutlich in die Jahre gekommen ist), sondern um Ableger wie `vim` (engl. *vi improved*, »verbessert«)

`vi`: heute meist Klon

oder elvis; diese Editoren sind jedoch so weit kompatibel zu vi, dass wir sie der Einfachheit halber mit dem Original in einen Topf werfen.

Einer der ersten Bildschirmeditoren

vi, ursprünglich entwickelt von Bill Joy für BSD, war einer der ersten gebräuchlichen »bildschirmorientierten« Editoren für Unix. Das bedeutet, dass nicht mehr nur einzelne Zeilen bearbeitet werden können, sondern der gesamte Bildschirm als Arbeitsfläche zur Verfügung steht. Was heutzutage eine Selbstverständlichkeit darstellt, war damals eine Innovation – was nicht daran liegt, dass die Programmierer vorher unfähig gewesen wären, sondern dass Textterminals mit freier Ansteuerung von beliebigen Zeichenpositionen auf dem Bildschirm, eine zwingende Voraussetzung für Programme wie den vi, erst zu jener Zeit zu erschwinglichen Preisen auf den Markt kamen. Aus Rücksichtnahme auf die damals älteren Systeme mit Fernschreibern oder *glass ttys*, also Terminals, die fernschreibermäßig nur unten am Bildschirmrand neue Zeichen schreiben konnten, unterstützt der vi unter dem Namen ex noch einen zeilenorientierten Modus.

Niedrige Anforderungen an Tastatur

Man konnte sich selbst bei den fortschrittlichen Terminals der damaligen Zeit allerdings nicht darauf verlassen, dass die Tastatur über das normale Buchstabenfeld hinaus noch weitere Funktionstasten etwa zur Cursorsteuerung enthielt – die heute üblichen PC-Standardtastaturen wären damals durchaus als luxuriös, wenn nicht gar überladen empfunden worden. Diese Tatsache rechtfertigt das eigentümliche Bedienungskonzept des vi, das man heute mit Recht als vorsintflutlich ansehen kann. Man kann es niemandem übelnehmen, wenn er oder sie den vi deswegen ablehnt. Trotzdem kann es nicht schaden, wenn Sie zumindest rudimentäre vi-Kenntnisse haben, selbst wenn Sie sich für Ihre tägliche Arbeit einen anderen Editor aussuchen – was Sie unbedingt tun sollten, wenn der vi Ihnen nicht zusagt. Es ist ja nicht so, dass es keine Auswahl gäbe, und auf kindische Spielchen wie »Wer den vi nicht benutzt, ist kein richtiger Linux-Anwender« lassen wir uns nicht ein. Die heute üblichen grafischen Oberflächen wie KDE enthalten zum Beispiel auch sehr schöne und leistungsfähige Editoren.

Es gibt in der Tat noch einen primitiveren Editor als den vi – das Programm ed. Eigentlich gebührt der Titel »einziger Editor, der garantiert auf jedem Unix-System vorhanden ist«, dem ed viel eher als dem vi, aber der ed als reiner Zeileneditor mit Benutzungsoberfläche aus der Fernschreiberzeit ist selbst Hardcore-Unix-Verfechtern zu ungenießbar. (Von der Idee her ist ed zu vergleichen mit dem DOS-Programm EDLIN, ed ist aber ungleich leistungsfähiger als das Programm aus Redmond.) Der Grund dafür, dass ed trotz der Existenz von Dutzenden bequemerer Texteditoren immer noch vorhanden ist, ist etwas unoffensichtlich, aber sehr Unix-artig: ed akzeptiert Kommandos über seine Standardeingabe und kann darum in Shellskripten benutzt werden, um Textdateien automatisch zu verändern. Der ed erlaubt Editieroperationen auf der gesamten Datei und kann damit prinzipiell mehr als sein Kollege, der »Stromeditor« sed, der nur seine Standardeingabe auf seine Standardausgabe kopiert und dabei gewisse Veränderungen vornehmen kann; normalerweise würde man den sed verwenden und nur in Ausnahmefällen auf den ed zurückgreifen, aber der ist trotzdem hin und wieder nützlich.

5.2.2 Grundlegende Funktionen

Puffer

Das Pufferkonzept Die Arbeit von vi erfolgt mit Hilfe sogenannter **Puffer**. Rufen Sie vi mit einem Dateinamen als Argument auf, so wird der Inhalt dieser Datei in einen Puffer gelesen. Falls keine Datei entsprechenden Namens existiert, wird ein leerer Puffer angelegt.

Alle im Editor vorgenommen Änderungen am Text werden zunächst nur innerhalb des Puffers durchgeführt. Um diese Korrekturen dauerhaft auf dem Datenträger zu verewigen, muss der Pufferinhalt explizit in die Datei zurückgeschrieben werden. Sollen stattdessen die bisher durchgeführten Änderungen verworfen werden, beenden Sie vi einfach, ohne vorher den Pufferinhalt abzuspeichern – die Datei auf dem Datenträger bleibt unverändert.

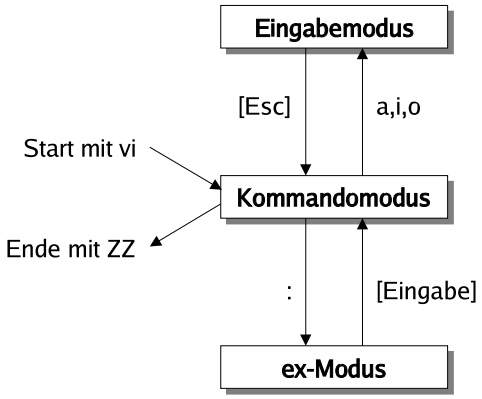


Bild 5.1: Arbeitsmodi von vi

Neben einem Dateinamen als Argument können vi beim Aufruf wie gewohnt auch Optionen anvertraut werden. Aufgrund derer relativ geringen Bedeutung sei hierzu auf die zugehörige Dokumentation verwiesen.

Arbeitsmodi

Arbeitsmodi Wie eingangs erwähnt, zeichnet sich vi durch seine ausgefallene Bedienung aus. Dieser Editor kennt nämlich drei verschiedene **Arbeitsmodi**:

Kommandomodus Alle Tastatureingaben stellen Kommandos dar, die nicht auf dem Bildschirm erscheinen und zumeist nicht durch Drücken der Eingabetaste bestätigt werden müssen.

In diesem Modus befinden Sie sich direkt nach dem Start. Seien Sie vorsichtig: Jeder Tastendruck könnte eine Aktion auslösen.

Eingabemodus Alle Tastatureingaben werden als Text behandelt und sind auf dem Monitor zu sehen. vi verhält sich hier wie ein heute gebräuchlicher Editor, allerdings mit eingeschränkten Manövrier- und Korrektur-Möglichkeiten.

Kommandozeilenmodus Im Kommandozeilenmodus können längere Befehle eingegeben werden. Diese werden üblicherweise durch einen Doppelpunkt eingeleitet und mit der Eingabetaste abgeschlossen.

Im Eingabemodus sind nahezu alle Kommandos zur Positionierung der Schreibmarke oder zur Textkorrektur unzulässig, was einen häufigen Wechsel zwischen Eingabe- und Kommandomodus erforderlich macht. Da es zudem – je nach der tatsächlich benutzten vi-Implementierung und deren Konfiguration – schwierig ist, zu erkennen, in welchem Modus sich vi gerade befindet, wird dem Einsteiger die Arbeit nicht gerade erleichtert. Eine Übersicht über die Arbeitsmodi von vi gibt Bild 5.2.

Denken Sie daran: Der vi fing an zu einer Zeit, als man nur den »Buchstabenblock« der Tastatur hatte (127 ASCII-Zeichen). An einer Mehrfachbelegung der Tasten führte darum kein Weg vorbei.

Kommandomodus

Nach dem Aufruf mit vi ohne Dateinamen gelangen Sie zunächst in den Kommandomodus. Im Unterschied zu den meisten anderen Editoren ist also nach dem Start keine direkte Texteingabe möglich. Auf dem Bildschirm blinkt links oben der Cursor, darunter befindet sich eine Spalte mit Tilden. In der letzten Zeile, auch als Statuszeile bekannt, werden neben Informationen zum momentanen Zustand des Editors (wenn die Implementierung das so macht) oder der aktuellen Datei noch die Cursorposition angezeigt (Bild 5.2).

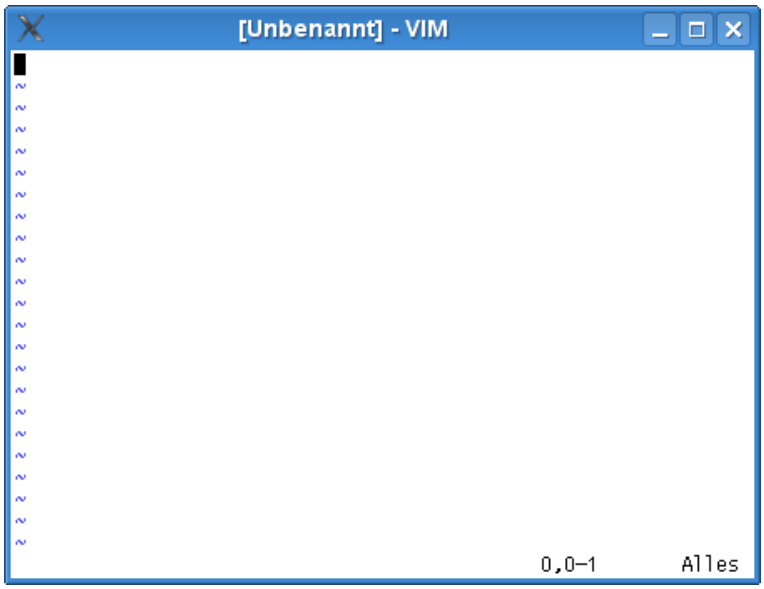


Bild 5.2: Der Editor vi (direkt nach dem Start)

Tabelle 5.1: Tastaturbefehle für den Eingabemodus von vi

Tastaturkommando	Wirkung
a	hängt neuen Text hinter dem aktuellen Zeichen an
A	hängt neuen Text am Zeilenende an
i	fügt neuen Text vor dem aktuellen Zeichen ein
I	fügt neuen Text am Zeilenanfang ein
o	fügt eine neue Zeile unter der aktuellen ein
O	fügt eine neue Zeile über der aktuellen ein

Tabelle 5.2: Tastaturbefehle zur Cursorpositionierung in vi

Tastaturkommando	Wirkung
h oder ←	ein Zeichen nach links
l oder →	ein Zeichen nach rechts
k oder ↑	ein Zeichen nach oben
j oder ↓	ein Zeichen nach unten
0	zum Zeilenanfang springen
$	zum Zeilenende springen
w	zum nächsten Wort springen
b	zum vorherigen Wort springen
f ⟨*Zeichen*⟩	zum nächsten ⟨*Zeichen*⟩ in der Zeile springen
Strg + f	eine Bildschirmseite vorblättern
Strg + b	eine Bildschirmseite zurückblättern
G	zur letzten Zeile der Datei springen
⟨*n*⟩ G	zur Zeile Nummer ⟨*n*⟩ springen

Wenn Ihr vi keine Statusinformationen anzeigt, versuchen Sie Ihr Glück einmal mit Esc `:set showmode` ↩.

Eingabemodus

Erst durch ein Kommando wie a (engl. *append*, »Anhängen«), i (engl. *insert*, »Einfügen«) oder o (engl. *open*, »Öffnen«) wechselt vi in den Eingabemodus. In der Statuszeile erscheint etwas wie »`-- INSERT --`«, und Tastatureingaben werden als Text entgegengenommen.

Die möglichen Kommandos zum Übergang in den Eingabemodus sind in Tabelle 5.1 aufgelistet, wobei zwischen Groß- und Kleinschreibung unterschieden wird. Um den Eingabemodus wieder zu verlassen, genügt ein Druck auf die Esc -Taste, und Sie finden sich im Kommandomodus wieder. Dort schreibt vi durch Eingabe von Z Z den Inhalt des Puffers auf den Datenträger und beendet sich anschließend.

Wollen Sie die vorgenommenen Änderungen lieber verwerfen, müssen Sie den Editor beenden, ohne dabei den Pufferinhalt abzuspeichern. Dies geschieht mit dem Befehl : `q!` ↩ (engl. *quit*, »aufhören«). Der einleitende Doppelpunkt zeigt, dass es sich hierbei um einen Befehl im Kommandozeilenmodus handelt.

Kommandozeilenmodus

Durch Eingabe von : im Kommandomodus wechselt vi in den Kommandozeilenmodus. Dies ist daran zu erkennen, dass in der untersten Zeile der Doppelpunkt und dahinter der Cursor erscheint. Alle weiteren Tastatureingaben werden nun an den Doppelpunkt angehängt, bis der Befehl durch die Eingabetaste (↩) abgeschlossen wird und vi in den Kommandomodus zurückkehrt. Im Kommandozeilenmodus verarbeitet vi die zeilenorientierten Kommandos seines *alter ego*, des Zeileneditors ex.

Auch zum zwischenzeitlichen Abspeichern des Pufferinhalts existiert ein ex-Kommando, dieses lautet : `w` (engl. *write*, »schreiben«). Mit dem Befehl : `x` oder der Kombination : `wq` wird der Pufferinhalt abgespeichert und danach der Editor beendet, beide Möglichkeiten sind also identisch mit dem Kommando Z Z.

Bewegung im Text Im Eingabemodus werden neu eingegebene Zeichen in die aktuelle Zeile aufgenommen, durch die Eingabetaste wird eine neue Zeile angelegt. Zwar ist mit den Pfeiltasten eine Bewegung im Text möglich, aber nur innerhalb der aktuellen Zeile können Sie die zuletzt getippten Buchstaben mit der Taste ⇐ wieder entfernen – ein Erbe der zeilenorientierten Vorgänger von vi. Weitergehende Bewegungen im Text sind nur im Kommandomodus möglich (Tabelle 5.2).

Haben Sie den Cursor nun an die richtige Textstelle dirigiert, können Sie mit der Korrektur beginnen, die ebenfalls im Kommandomodus erfolgen muss.

Tabelle 5.3: Tastaturbefehle zur Textkorrektur in vi

Tastaturkommando	Wirkung
x	löscht das Zeichen unter dem Cursor
X	löscht das Zeichen vor dem Cursor
r ⟨*Zeichen*⟩	ersetzt das Zeichen unter dem Cursor durch ⟨*Zeichen*⟩
d w	löscht ab Cursor bis Wortende
d $	löscht ab Cursor bis Zeilenende
d 0	löscht ab Cursor bis Zeilenanfang
d f ⟨*Zeichen*⟩	löscht ab Cursor bis zum nächsten ⟨*Zeichen*⟩ in der Zeile
d d	löscht aktuelle Zeile
d G	löscht ab aktueller Zeile bis zum Textende
d 1 G	löscht ab aktueller Zeile bis zum Textanfang

Tabelle 5.4: Tastaturbefehle zur Textersetzung in vi

Tastaturkommando	Wirkung
c w	ab Cursor bis Wortende ersetzen
c $	ab Cursor bis Zeilenende ersetzen
c 0	ab Cursor bis Zeilenanfang ersetzen
c f ⟨*Zeichen*⟩	ab Cursor bis zum nächsten ⟨*Zeichen*⟩ in der Zeile ersetzen
c / abc	ab Cursor bis zur nächsten Zeichenfolge `abc` ersetzen

Löschen von Zeichen Zum Löschen von Zeichen existiert der Befehl d (engl. *delete*, »Löschen«), der immer durch ein Folgezeichen konkretisiert werden muss (Tabelle 5.3). Um Korrekturen zu erleichtern, können Sie den aufgelisteten Befehlen einen Wiederholungsfaktor voranstellen, zum Beispiel löscht die Eingabe 3 x die nächsten drei Zeichen.

Wiederholungsfaktor

Falls Sie im Eifer des Gefechtes zu viel ver(schlimm)bessert haben, können Sie mit u (*undo*, »ungeschehen machen«) je nach Systemeinstellungen nur die letzte oder gar alle vorgenommenen Änderungen nacheinander zurücknehmen.

undo

Überschreiben von Zeichen Das Kommando c (engl. *change*, »wechseln«) dient zum Überschreiben eines ausgewählten Textbereichs. Genau wie d ist c ein Kombinationskommando, verlangt also eine zusätzliche Definition. Nach der Befehlseingabe wird der angegebene Textausschnitt entfernt und automatisch in den Eingabemodus gewechselt. Dort können Sie den neue Text einfügen und mit Esc wieder in den Kommandomodus zurückkehren (Tabelle 5.4).

Überschreiben

5.2.3 Erweiterte Funktionen

Textteile ausschneiden, kopieren und einfügen Ein häufiges Problem bei der Textbearbeitung ist, dass eine bereits existierende Passage an eine andere Position verschoben oder kopiert werden soll. Auch hierfür hat vi passende Kombinationsbefehle parat, für die die für c aufgelisteten Zusätze gültig sind. y (engl. *yank*, »zerren«) kopiert einen Text in den Zwischenspeicher, ohne den Ursprungstext zu verändern. d hingegen verschiebt den ausgewählten Text in den Zwischenspeicher, der Textbereich wird also aus seiner ursprünglichen Position ausgeschnitten und ist nur noch im Zwischenspeicher vorhanden. (Wir haben das schon weiter

oben als »Löschen« vorgestellt.)

Natürlich gibt es auch einen Befehl, um Textbausteine aus dem Zwischenspeicher wieder einzufügen. Dies erfolgt mit p hinter bzw. mit P vor der momentanen Cursorposition (engl. *paste*, »aufkleben«).

26 Zwischenspeicher

Eine Besonderheit von vi ist, dass er für derartige Operationen nicht nur einen, sondern gleich 26 Zwischenspeicher anbietet. Das ergibt Sinn, wenn mehrere unterschiedliche Textblöcke häufiger in den Text eingebaut werden sollen. Diese Zwischenspeicher sind fortlaufend von »a« bis »z« benannt und werden mit einer Kombination aus Anführungszeichen und Speichernamen aufgerufen. Die Befehlssequenz " c y 4 w kopiert also die nächsten vier Wörter in den Zwischenspeicher namens c; mit " g p wird der Inhalt des Zwischenspeichers g hinter der momentanen Cursorposition eingefügt.

Textsuche mit regulären Ausdrücken Wie es sich für einen ordentlichen Editor gehört, stellt vi ausgeklügelte Suchfunktionen zur Verfügung. Neben exakten Zeichenfolgen können mit Hilfe von »regulären Ausdrücken« auch ganze Mengen von Zeichenfolgen aufgespürt werden. Um den Suchvorgang einzuleiten, müssen Sie im Kommandomodus einen Schrägstrich / eingeben. Dieser erscheint daraufhin gefolgt vom Cursor in der letzten Zeile. Nachdem der Suchbegriff eingetippt worden ist, startet die Eingabetaste den Suchvorgang. Dieser wird von der aktuellen Cursorposition bis zum Textende, also textabwärts, durchgeführt. Soll stattdessen textaufwärts gesucht werden, muss die Suchfunktion nicht mit / , sondern mit ? aufgerufen werden. Sobald vi eine passende Zeichenfolge gefunden hat, wird die Suche beendet und der Cursor auf den ersten Buchstaben dieser Sequenz gesetzt. Mit n wird die Suche textabwärts, mit N hingegen textaufwärts fortgesetzt.

Suchen und Ersetzen Mit dem Auffinden von Zeichenketten alleine ist es oftmals nicht getan, darum bietet vi zusätzlich die Möglichkeit, die gefundenen Textteile durch andere zu ersetzen. Diese Aufgabe erledigt ein ex-Kommando mit der Syntax:

: [⟨*Startzeile*⟩,⟨*Endzeile*⟩]s/⟨*reg. Ausdruck*⟩/⟨*Ersetzung*⟩[/g]

Die in eckigen Klammern stehenden Parameter sind optional, müssen also nicht angeführt werden. Was bedeuten nun die einzelnen Bestandteile des Befehls?

Textbereich

⟨*Startzeile*⟩ und ⟨*Endzeile*⟩ legen den Bereich fest, in dem die Ersetzung durchgeführt werden soll. Ohne diese Angabe wird nur die aktuelle Zeile bearbeitet! Außer Zeilennummern sind zum Beispiel auch ein Punkt als Symbol für die aktuelle sowie ein Dollarzeichen für die letzte Zeile zulässig :

```
:5,$s/rot/blau/
```

ersetzt in jeder Zeile das erste Vorkommen von rot durch blau, wobei die ersten vier Zeilen nicht berücksichtigt werden.

```
:5,$s/rot/blau/g
```

ersetzt in denselben Zeilen *jedes* Vorkommen von rot durch blau. (Passen Sie auf: Auch die Brotmaschine wird zur Bblaumaschine.)

Statt Zeilennummern, ».« und »$« sind als Start- und Endangaben auch reguläre Ausdrücke in Schrägstrichen erlaubt:

```
:/^ANFANG/,/^ENDE/s/rot/blau/g
```

ersetzt rot durch blau nur in Zeilen, die zwischen einer Zeile mit ANFANG am Anfang und einer mit ENDE am Anfang stehen.

Tabelle 5.5: ex-Kommandos von vi

Kommando	Wirkung
: w ⟨*Dateiname*⟩	schreibt den kompletten Pufferinhalt in die angegebene Datei
: w! ⟨*Dateiname*⟩	schreibt den Pufferinhalt auch in eine schreibgeschützte Datei
: e ⟨*Dateiname*⟩	liest die angegebene Datei in den Puffer
: e #	liest die letzte Datei in den Puffer
: r ⟨*Dateiname*⟩	fügt den Inhalt der angegebenen Datei hinter der aktuellen Zeile ein
: ! ⟨*Shellkommando*⟩	führt das angegebene Shellkommando aus und kehrt danach zu vi zurück
: r! ⟨*Shellkommando*⟩	fügt die Ausgabe des Kommandos hinter der aktuellen Zeile ein
: s/⟨*Muster*⟩/⟨*Ersatz*⟩	sucht nach dem Suchmuster ⟨*Muster*⟩ und ersetzt es durch ⟨*Ersatz*⟩
: q	beendet vi
: q!	beendet vi ohne Rückfrage
: x oder : wq	sichert den Pufferinhalt und beendet danach vi

Nach dem Kommandonamen s und einem Schrägstrich müssen Sie den gesuchten regulären Ausdruck angeben. Nach einem weiteren Schrägstrich steht unter ⟨*Ersetzung*⟩ die Zeichenfolge, mit welcher der ursprüngliche Text überschrieben werden soll.

Rückbezug auf Gefundenes

Für dieses Argument existiert eine spezielle Sonderfunktion: mit einem & kann auf die Zeichenkette verwiesen werden, auf die der gesuchte reguläre Ausdruck im konkreten Fall gepasst hat, d. h. »: s/Zebra/&fink« verwandelt jedes Zebra im Suchbereich in einen Zebrafinken – eine Aufgabe, an der die Gentechnologie noch einige Zeit scheitern dürfte.

Befehle im Kommandozeilenmodus Im bisherigen Text haben wir schon einige Befehle des Kommandozeilenmodus, auch als ex-Modus bekannt, beschrieben. Neben diesen existieren noch einige andere, die alle aus dem Kommandomodus heraus mit einem Doppelpunkt eingeleitet und der Eingabetaste abgeschlossen werden müssen (Tabelle 5.5).

Übungen

5.2 [5] (Für Systeme mit vim, z. B. die SUSE-Distributionen.) Finden Sie heraus, wie die interaktive vim-Einführung aufgerufen wird, und arbeiten Sie sie durch.

5.3 Andere Editoren

Wir haben schon durchblicken lassen, dass Ihre Editor-Auswahl ebensoviel mit Ihren persönlichen Vorlieben zu tun hat und wohl auch ebensoviel über Sie als Benutzer aussagt wie die Wahl Ihres Autos: Fahren Sie einen polierten BMW oder reicht Ihnen ein verbeulter Golf? Oder muss es doch ein Land-Rover sein? Was die Wahlmöglichkeiten angeht, so steht der Editor-Markt dem Automarkt wohl kaum nach. Wir haben Ihnen den vielleicht wichtigsten Linux-Editor präsentiert, aber es gibt natürlich auch noch einige andere. kate unter KDE und gedit unter GNOME

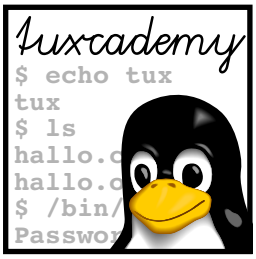
zum Beispiel sind übersichtliche und leicht erlernbare Editoren mit einer grafischen Oberfläche, die für die Aufgaben eines »normalen Benutzers« sicher ausreichen. Der GNU Emacs dagegen ist ein extrem leistungsfähiger und anpassbarer Editor für Programmierer und Autoren, dessen umfangreiches »Ökosystem« von Erweiterungen kaum Wünsche offen läßt. Durchstöbern Sie ruhig einmal die Paketlisten Ihrer Distribution und schauen Sie, ob Sie dort den Editor Ihrer Träume finden können.

Kommandos in diesem Kapitel

ed	Primitiver zeilenorientierter Editor	ed(1)	59
elvis	Populärer „Klon“ des vi-Editors	elvis(1)	58
ex	Leistungsfähiger zeilenorientierter Editor (eigentlich vi)	vi(1)	59
sed	Datenstromorientierter Texteditor, kopiert Eingabe unter Änderungen auf Ausgabe	sed(1)	59
vi	Bildschirmorientierter Texteditor	vi(1)	58
vim	Populärer „Klon“ des vi-Editors	vim(1)	58

Zusammenfassung

- Texteditoren sind wichtig zum Ändern von Konfigurationsdateien und zum Programmieren. Sie bieten oft besondere Eigenschaften, die diese Aufgaben erleichtern.
- Der vi ist ein traditioneller, sehr verbreiteter und leistungsfähiger Editor mit eigenwilliger Bedienung.



6

Dateien: Aufzucht und Pflege

Inhalt

6.1 Datei- und Pfadnamen 68
6.1.1 Dateinamen 68
6.1.2 Verzeichnisse 70
6.1.3 Absolute und relative Pfadnamen 70
6.2 Kommandos für Verzeichnisse 71
6.2.1 Das aktuelle Verzeichnis: `cd` & Co. 71
6.2.2 Dateien und Verzeichnisse auflisten – `ls` 72
6.2.3 Verzeichnisse anlegen und löschen: `mkdir` und `rmdir` 74
6.3 Suchmuster für Dateien 75
6.3.1 Einfache Suchmuster 75
6.3.2 Zeichenklassen 77
6.3.3 Geschweifte Klammern 78
6.4 Umgang mit Dateien 79
6.4.1 Kopieren, Verschieben und Löschen – `cp` und Verwandte 79
6.4.2 Dateien verknüpfen – `ln` und `ln -s` 81
6.4.3 Dateiinhalte anzeigen – `more` und `less` 86
6.4.4 Dateien suchen – `find` 86
6.4.5 Dateien schnell finden – `locate` und `slocate` 90

Lernziele

- Die Linux-Konventionen über Datei- und Verzeichnisnamen kennen
- Die wichtigsten Kommandos zum Umgang mit Dateien und Verzeichnissen beherrschen
- Mit Suchmustern in der Shell umgehen können

Vorkenntnisse

- Arbeit mit der Shell (siehe Kapitel 2)
- Umgang mit einem Texteditor (siehe Kapitel 5)

6.1 Datei- und Pfadnamen

6.1.1 Dateinamen

Dateien

Eine der wesentlichen Dienste eines Betriebssystems wie Linux besteht darin, Daten auf dauerhaften Speichermedien wie Festplatten oder USB-Sticks speichern und später wiederfinden zu können. Um das für Menschen erträglich zu gestalten, werden zusammengehörende Daten normalerweise zu »Dateien« zusammengefasst, die auf dem Speichermedium unter einem Namen gespeichert werden.

Auch wenn Ihnen das trivial vorkommen mag: Selbstverständlich ist das nicht. Betriebssysteme in früheren Zeiten machten es mitunter nötig, dass man Gräuel wie die passenden Spurnummern auf Platten kennen musste, wenn man seine Daten wieder finden wollte.

Bevor wir Ihnen erklären können, wie Sie mit Dateien umgehen, müssen wir Ihnen also erklären, wie Linux Dateien *benennt*.

Erlaubte Zeichen

In Linux-Dateinamen sind grundsätzlich alle Zeichen zugelassen, die der Computer darstellen kann (und noch ein paar mehr). Da einige dieser Zeichen aber eine besondere Bedeutung haben, raten wir von deren Verwendung in Dateinamen aber ab. Komplett verboten innerhalb von Dateinamen sind nur zwei Zeichen, der Schrägstrich und das Nullbyte (das Zeichen mit dem ASCII-Wert 0). Andere Zeichen wie Leerzeichen, Umlaute oder Dollarzeichen können verwendet werden, müssen dann aber auf der Kommandozeile meist durch einen Rückstrich oder Anführungszeichen maskiert werden, um Fehlinterpretationen durch die Shell zu vermeiden.

Groß- und Kleinschreibung

Eine Stolperfalle für Umsteiger ist, dass Linux in Dateinamen zwischen Groß- und Kleinschreibung unterscheidet. Im Gegensatz zu Windows, wo Groß- und Kleinbuchstaben in Dateinamen zwar dargestellt, aber identisch behandelt werden, interpretiert Linux `x-files` und `X-Files` tatsächlich als zwei verschiedene Dateinamen.

Dateinamen dürfen unter Linux »ziemlich lang« sein – eine feste allgemeinverbindliche Grenze gibt es nicht, da das Maximum vom »Dateisystem« abhängt, also der Methode, wie genau die Bytes auf der Platte arrangiert sind (davon gibt es unter Linux mehrere). Eine typische Obergrenze sind aber 255 Zeichen – und da so ein Name auf einem normalen Textterminal gut drei Zeilen einnehmen würde, sollte das Ihren Stil nicht übermäßig einschränken.

Endungen

Eine weitere Abweichung zu DOS- bzw. Windows-Rechnern besteht darin, dass bei Linux Dateien nicht durch entsprechende Endungen charakterisiert werden müssen. Der Punkt ist in einem Dateinamen also ein ganz gewöhnliches Zeichen. Ein Text kann daher als `blabla.txt` abgespeichert werden, aber einfach `blabla` wäre grundsätzlich ebenso zulässig. Was nicht heißt, dass Sie keine »Dateiendungen« benutzen sollten – immerhin erleichtern solche Kennzeichnungen die Identifizierung des Dateiinhaltes.

Manche Programme bestehen darauf, dass ihre Eingabedateien bestimmte Endungen haben. Der C-Übersetzer `gcc` zum Beispiel betrachtet Dateien mit Namen, die auf »`.c`« enden, als C-Quelltext, solche mit Namen auf »`.s`« als Assemblerquelltext und solche mit Namen auf »`.o`« als vorübersetzte Objektdateien.

Sonderzeichen

Umlaute und andere Sonderzeichen können Sie in Dateinamen grundsätzlich ungeniert verwenden. Sollen aber Dateien gleichermaßen auf anderen Systemen benutzt werden, ist es besser, auf Sonderzeichen in Dateinamen zu verzichten, da nicht garantiert ist, dass sie auf anderen Systemen als dieselben Zeichen erscheinen.

Was mit Sonderzeichen passiert, hängt auch von der Ortseinstellung ab, da es keinen allgemein üblichen Standard dafür gibt, Zeichen darzustellen, die den Zeichenvorrat des ASCII (128 Zeichen, die im wesentlichen die englische Sprache, Ziffern und die gängigsten Sonderzeichen abdecken) überschreiten. In weitem Gebrauch sind zum Beispiel die Zeichentabellen ISO 8859-1 und ISO 8859-15 (vulgo ISO-Latin-1 und ISO-Latin-9 … fragen Sie nicht) sowie ISO 10646, gerne salopp und nicht ganz korrekt als »Unicode« bezeichnet und in der Regel als »UTF-8« codiert. Dateinamen mit Sonderzeichen, die Sie anlegen, während Zeichentabelle *X* aktiv ist, können völlig anders aussehen, wenn Sie sich das Verzeichnis gemäß Zeichentabelle *Y* anschauen. Das ganze Thema ist nichts, worüber man beim Essen nachdenken möchte.

Ortseinstellung

Sollten Sie jemals in die Verlegenheit kommen, einen Berg Dateien vor sich zu haben, deren Namen allesamt gemäß einer verkehrten Zeichentabelle codiert sind, kann Ihnen vielleicht das Programm `convmv` helfen, das Dateinamen zwischen verschiedenen Zeichentabellen konvertieren kann. (Sie werden es vermutlich selber installieren müssen, da es sich normalerweise nicht im Standardumfang gängiger Distributionen befindet.) Allerdings sollten Sie sich damit erst befassen, wenn Sie den Rest dieses Kapitels durchgearbeitet haben, denn wir haben Ihnen noch nicht einmal das reguläre `mv` erklärt …

`convmv`

Ohne Bedenken lassen sich alle Zeichen aus

Portable Dateinamen

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789+-._
```

in Dateinamen benutzen. Allerdings sollten Sie die folgenden Tipps berücksichtigen:

- Um den Datenaustausch mit älteren Unix-Systemen zu ermöglichen, sollte die Länge eines Dateinamens gegebenenfalls maximal 14 Zeichen betragen.
- Dateinamen sollten stets mit einem der genannten Buchstaben oder einer Ziffer beginnen, die vier anderen Zeichen sind nur innerhalb eines Dateinamens problemlos verwendbar.

Diese Konventionen lassen sich am leichtesten mit ein paar Beispielen verstehen. Zulässige Dateinamen wären etwa:

```
X-files
bla.txt.bak
17.Juni
7_of_9
```

Schwierigkeiten wären dagegen möglich (wenn nicht gar wahrscheinlich oder sogar sicher) mit:

```
-20°C                    Beginnt mit »-«, Sonderzeichen
.profile                 Wird versteckt
3/4-Takt                 Enthält verbotenes Zeichen
Müll                     Enthält Umlaut
```

Als weitere Besonderheit werden Dateinamen, die mit einem Punkt (».«) anfangen, an einigen Stellen (etwa beim Auflisten von Verzeichnisinhalten) übersprungen – Dateien mit solchen Namen gelten als »versteckt«. Diese Eigenschaft verwendet man unter anderem gerne für Dateien, die Voreinstellungen für Programme enthalten und die in Dateinamenslisten nicht von wichtigeren Dateien ablenken sollen.

Versteckte Dateien


Für DOS- und Windows-Experten: Diese Systeme erlauben »versteckte« Dateien über ein unabhängig vom Namen zu setzendes »Dateiattribut«. Unter Linux bzw. Unix gibt es so etwas nicht.

6.1.2 Verzeichnisse

Da auf demselben Linux-System potentiell viele Benutzer arbeiten können, wäre es ein Problem, wenn es jeden Dateinamen nur einmal geben dürfte. Dem Benutzer Hugo wäre wohl nur schwer klar zu machen, dass er keine Datei namens `brief.txt` anlegen kann, weil die Benutzerin Susi schon eine Datei hat, die so heißt. Außerdem muss irgendwie (bequem) geregelt werden können, dass Hugo nicht alle Dateien von Susi lesen kann und umgekehrt.

Aus diesem Grund unterstützt Linux das Konzept hierarchischer »Verzeichnisse«, die zum Gruppieren von Dateien dienen. Dateinamen müssen nicht über das ganze System hinweg eindeutig sein, sondern nur im selben Verzeichnis. Das bedeutet insbesondere, dass Hugo und Susi vom System verschiedene Verzeichnisse zugeordnet bekommen können und darin dann jeweils ihre Dateien so nennen dürfen, wie sie mögen, ohne aufeinander Rücksicht nehmen zu müssen. Ferner kann man Hugo den Zugriff auf Susis *Verzeichnis* verbieten (und umgekehrt) und muss sich dann nicht mehr um die einzelnen Dateien kümmern, die dort jeweils stehen.

Verzeichnisse sind bei Linux auch nur Dateien, wenn auch Dateien, die Sie nicht mit denselben Mitteln bearbeiten können wie »gewöhnliche« Dateien. Das bedeutet aber, dass für die Namen von Verzeichnissen dieselben Regeln gelten wie für die Namen von Dateien (siehe voriger Abschnitt). Sie müssen nur noch lernen, dass der Schrägstrich (»/«) dazu dient, Dateinamen von Verzeichnisnamen und Verzeichnisnamen voneinander zu trennen. `hugo/brief.txt` wäre also die Datei `brief.txt` im Verzeichnis `hugo`.

Schrägstrich

Verzeichnisse können andere Verzeichnisse enthalten (das ist das »hierarchisch« von weiter vorne), wodurch sich eine baumartige Struktur ergibt (naheliegenderweise gerne »Verzeichnisbaum« genannt). Ein Linux-System hat ein spezielles Verzeichnis, das die Wurzel des Baums bildet und deswegen »Wurzelverzeichnis« oder neudeutsch *root directory* genannt wird. Es hat nach Konvention den Namen »/« (Schrägstrich).

Verzeichnisbaum


Trotz seiner englischen Bezeichnung hat das Wurzelverzeichnis nichts mit dem Systemverwalter `root` zu tun. Die beiden heißen einfach nur so ähnlich.


Der Schrägstrich spielt hier eine Doppelrolle – er dient sowohl als Name des Wurzelverzeichnisses als auch als Trennzeichen zwischen anderen Verzeichnissen. Hierzu gleich mehr.

Die Grundinstallation gängiger Linux-Distributionen enthält normalerweise Zehntausende von Dateien in einer größtenteils durch gewisse Konventionen vorstrukturierten Verzeichnishierarchie. Über diese Verzeichnishierarchie erfahren Sie mehr in Kapitel 9.

6.1.3 Absolute und relative Pfadnamen

Jede Datei in einem Linux-System wird durch einen Namen beschrieben, der sich ergibt, indem man ausgehend vom Wurzelverzeichnis jedes Verzeichnis nennt bis zu dem, in dem die Datei steht, gefolgt vom Namen der Datei selbst. Beispielsweise benennt `/home/hugo/brief.txt` die Datei `brief.txt`, die im Verzeichnis `hugo` steht, das wiederum im Verzeichnis `home` zu finden ist. Das Verzeichnis `home` steht direkt im Wurzelverzeichnis. Solche Namen, die vom Wurzelverzeichnis ausgehen, heißen »absolute Pfadnamen« (engl. *absolute path names*) – wir sprechen von »Pfadnamen«, weil der Name einen »Pfad« durch den Verzeichnisbaum beschreibt, in dem die Namen von Verzeichnissen und Dateien vorkommen können (es handelt sich also um einen Sammelbegriff).

absolute Pfadnamen

Jeder Prozess in einem Linux-System hat ein »aktuelles Verzeichnis« (engl. *current directory*, auch *working directory*, »Arbeitsverzeichnis«). Dateinamen werden in diesem Verzeichnis gesucht; `brief.txt` ist also eine bequeme Abkürzung für »die Datei `brief.txt` im aktuellen Verzeichnis«, und `susi/brief.txt`« steht für »die Datei `brief.txt` im Verzeichnis `susi`, das im aktuellen Verzeichnis steht«. Solche Namen, die vom aktuellen Verzeichnis ausgehen, nennen wir »relative Pfadnamen« (engl. *relative path names*). aktuelles Verzeichnis relative Pfadnamen

Sie können ganz einfach absolute von relativen Pfadnamen unterscheiden: Ein Pfadname, der mit »/« anfängt, ist absolut; alle anderen Pfadnamen sind relativ.

Das aktuelle Verzeichnis »vererbt« sich vom Elterprozess auf den Kindprozess. Wenn Sie also aus einer Shell heraus eine neue Shell (oder ein anderes Programm) starten, dann hat diese neue Shell dasselbe aktuelle Verzeichnis wie die Shell, aus der Sie sie gestartet haben. Sie können innerhalb der neuen Shell mit `cd` in ein anderes Verzeichnis wechseln, aber das aktuelle Verzeichnis der alten Shell ändert sich dadurch nicht – wenn Sie die neue Shell verlassen, dann sind Sie wieder im (unveränderten) aktuellen Verzeichnis der alten Shell.

In relativen (oder auch absoluten) Pfadnamen gibt es zwei bequeme Abkürzungen: Der Name »`..`« steht für das dem jeweiligen Verzeichnis im Verzeichnisbaum *übergeordnete* Verzeichnis – im Falle von `/home/hugo` also beispielsweise `/home`. Das erlaubt Ihnen oftmals, bequemer als über absolute Pfadnamen auf Dateien Bezug zu nehmen, die aus der Sicht des aktuellen Verzeichnisses in einem »Seitenast« des Verzeichnisbaums zu finden sind. Nehmen wir an, `/home/hugo` hätte die Unterverzeichnisse `briefe` und `romane`. Mit `briefe` als aktuellem Verzeichnis könnten Sie sich über den relativen Pfadnamen `../romane/werther.txt` auf die Datei `werther.txt` im Verzeichnis `romane` beziehen, ohne den umständlichen absoluten Namen `/home/hugo/romane/werther.txt` angeben zu müssen. Abkürzungen

Die zweite Abkürzung ergibt keinen ganz so offensichtlichen Sinn: Der Name »`.`« in einem Verzeichnis steht immer für das Verzeichnis selbst. Es leuchtet nicht unmittelbar ein, wofür man eine Methode braucht, auf ein Verzeichnis zu verweisen, in dem man sich bereits befindet, aber es gibt Situationen, wo das sehr nützlich ist. Zum Beispiel wissen Sie vielleicht schon (oder können in Kapitel 8 nachschauen), dass die Shell die Programmdateien für externe Kommandos in den Verzeichnissen sucht, die in der Umgebungsvariablen `PATH` aufgelistet sind. Wenn Sie als Softwareentwickler ein Programm, nennen wir es mal `prog`, aufrufen wollen, das (a) in einer Datei im aktuellen Verzeichnis steht und (b) dieses aktuelle Verzeichnis *nicht* in `PATH` zu finden ist (aus Sicherheitsgründen immer eine gute Idee), dann können Sie mit

```
$ ./prog
```

die Shell trotzdem dazu bringen, Ihre Datei als Programm zu starten, ohne dass Sie einen absoluten Pfadnamen angeben müssen.

Als Linux-Benutzer haben Sie ein »Heimatverzeichnis«, in dem Sie direkt nach dem Anmelden am System landen. Welches das ist, bestimmt der Systemadministrator beim Anlegen Ihres Benutzerkontos, aber es heißt normalerweise so wie Ihr Benutzername und ist unterhalb von `/home` zu finden – etwas wie `/home/hugo` für den Benutzer `hugo`.

6.2 Kommandos für Verzeichnisse

6.2.1 Das aktuelle Verzeichnis: `cd` & Co.

In der Shell können Sie mit dem Kommando `cd` das aktuelle Verzeichnis wechseln: Geben Sie einfach das gewünschte Verzeichnis als Parameter an. Verzeichnis wechseln

Tabelle 6.1: Einige Dateitypenkennzeichnungen in `ls`

Dateityp	Farbe	Zeichen (`ls -F`)	Kennung (`ls -l`)
gewöhnliche Datei	schwarz	keins	-
ausführbare Datei	grün	*	-
Verzeichnis	blau	/	d
Link	cyan	@	l

```
$ cd briefe                    Wechseln ins Verzeichnis briefe
$ cd ..                        Wechseln ins übergeordnete Verzeichnis
```

Wenn Sie keinen Parameter angeben, landen Sie in Ihrem Heimatverzeichnis:

```
$ cd
$ pwd
/home/hugo
```

Aktuellen Verzeichnisnamen ausgeben

Mit dem Kommando `pwd` (engl. *print working directory*) können Sie, wie im Beispiel zu sehen, den absoluten Pfadnamen des aktuellen Verzeichnisses ausgeben.

Eingabeaufforderung

Möglicherweise sehen Sie das aktuelle Verzeichnis auch an Ihrer Eingabeaufforderung; je nach der Voreinstellung Ihres Systems könnte da etwas erscheinen wie

```
hugo@red:~/briefe> _
```

Dabei ist `~/briefe` eine Abkürzung für `/home/hugo/briefe`; die Tilde (»~«) steht für das Heimatverzeichnis des aktuellen Benutzers.

Das Kommando »`cd -`« wechselt in das Verzeichnis, das vor dem letzten `cd`-Kommando aktuell war. Damit können Sie bequem zwischen zwei Verzeichnissen hin und her wechseln.

Übungen

6.1 [2] Ist `cd` ein externes Kommando oder als internes Kommando in die Shell eingebaut? Warum?

6.2 [3] Lesen Sie in der Handbuchseite der `bash` über die Kommandos `pushd`, `popd` und `dirs` nach. Überzeugen Sie sich, dass diese Kommandos funktionieren.

6.2.2 Dateien und Verzeichnisse auflisten – `ls`

Zur Orientierung im Verzeichnisbaum ist es wichtig, herausfinden zu können, welche Dateien und Verzeichnisse in einem Verzeichnis stehen. Hierzu dient das Kommando `ls` (*list*, »auflisten«).

Tabellenformat

Ohne Übergabe von Optionen werden diese Informationen als mehrspaltige, nach Dateinamen sortierte Tabelle ausgegeben. Da Farbbildschirme heutzutage keine Besonderheit mehr darstellen, hat es sich eingebürgert, Dateien verschiedenen Typs in verschiedenen Farben darzustellen. (Über Dateitypen haben wir noch nicht geredet; dieses Thema kommt im Kapitel 9 zur Sprache.)

Erfreulicherweise sind die meisten Distributionen sich über die Farben inzwischen weitgehend einig. Tabelle 6.1 nennt die verbreitetste Zuordnung.

Tabelle 6.2: Einige Optionen für `ls`

Option	Wirkung
`-a` oder `--all`	zeigt auch versteckte Dateien an
`-i` oder `--inode`	gibt die eindeutige Dateinummer (Inode-Nr.) aus
`-l` oder `--format=long`	liefert zusätzliche Informationen
`--color=no`	verzichtet auf die Farbcodierung
`-p` oder `-F`	markiert Dateityp durch angehängte Sonderzeichen
`-r` oder `--reverse`	kehrt Sortierreihenfolge um
`-R` oder `--recursive`	durchsucht auch Unterverzeichnisse (DOS: `DIR/S`)
`-S` oder `--sort=size`	sortiert Dateien nach Größe (größte zuerst)
`-t` oder `--sort=time`	sortiert Dateien nach Zeit (neueste zuerst)
`-X` oder `--sort=extension`	sortiert Dateien nach Dateityp

Auf Monochrom-Monitoren – auch die gibt es noch – bietet sich zur Unterscheidung die Angabe der Optionen `-F` oder `-p` an. Hierdurch werden zur Charakterisierung Sonderzeichen an die Dateinamen angehängt. Eine Auswahl der Zeichen zeigt Tabelle 6.1.

Versteckte Dateien, deren Name mit einem Punkt beginnt, können Sie mit dem Schalter `-a` (engl. *all*, »alle«) anzeigen. Sehr nützlich ist weiterhin der Parameter `-l` (das ist ein kleines »L« und steht für engl. *long*, »lang«). Damit werden nicht nur die Dateinamen, sondern auch noch diverse Zusatzinformationen ausgegeben.

Versteckte Dateien

Zusatzinformationen

In manchen Linux-Distributionen sind für gängige Kombination dieser hilfreichen Optionen Abkürzungen voreingestellt; in den SUSE-Distributionen steht etwa ein einfaches `l` für das Kommando »`ls -alF`«. Auch »`ll`« und »`la`« sind Abkürzungen für `ls`-Varianten.

Abkürzungen

Hier sehen Sie `ls` ohne und mit `-l`:

```
$ ls
datei.txt
datei2.dat
$ ls -l
-rw-r--r--  1  hugo users 4711  Oct 4 11:11 datei.txt
-rw-r--r--  1  hugo users  333  Oct 2 13:21 datei2.dat
```

Im ersten Fall werden alle sichtbaren Dateien des Verzeichnisses tabellarisch aufgelistet, im zweiten Fall kommen die Zusatzinformationen dazu.

Die einzelnen Teile der langen Darstellung haben folgende Bedeutung: Die erste Spalte gibt den Dateityp (siehe Kapitel 9) an; normale Dateien haben ein »-«, Verzeichnisse ein »`d`« und so weiter (»Kennbuchstabe« in Tabelle 6.1). Die nächsten neun Zeichen informieren über die Zugriffsrechte. Danach folgen ein Referenzzähler, der Eigentümer der Datei, hier `hugo`, und die Gruppenzugehörigkeit der Datei zur Gruppe `users`. Nach der Dateigröße in Bytes sind Datum und Uhrzeit der letzten Änderung zu sehen. Abgeschlossen wird die Zeile durch den Dateinamen.

Format der langen Darstellung

Je nachdem, welche Sprache Sie eingestellt haben, kann insbesondere die Datums- und Uhrzeitangabe ganz anders aussehen als in unserem Beispiel (das wir mit der Minimal-Sprachumgebung »`C`« erzeugt haben). Das ist im interaktiven Gebrauch normalerweise kein Problem, kann aber zu Ärger führen, wenn Sie in einem Shellskript versuchen, die Ausgabe von »`ls -l`« auseinander zu pflücken. (Ohne hier der Schulungsunterlage *Linux für Fortgeschrittene* vorgreifen zu wollen, empfehlen wir, in Shellskripten die Sprachumgebung auf einen definierten Wert zu setzen.)

Wenn Sie die Zusatzinformationen für ein Verzeichnis (etwa `/tmp`) sehen wollen, hilft »`ls -l /tmp`« Ihnen nicht weiter, denn `ls` listet dann die Daten für alle Dateien in `/tmp` auf. Mit der Option `-d` können Sie das unterdrücken und bekommen die Informationen über `/tmp` selbst.

Neben den besprochenen erlaubt `ls` noch eine Vielzahl weiterer Optionen, von denen die Wichtigsten in Tabelle 6.2 dargestellt sind.

In den LPI-Prüfungen *Linux Essentials* und LPI-101 erwartet niemand von Ihnen, dass Sie alle 57 Geschmacksrichtungen von `ls`-Optionen rauf und runter beten können. Das wichtigste gute halbe Dutzend – den Inhalt von Tabelle 6.2 oder so – sollten Sie aber schon auswendig parat haben.

Übungen

6.3 [1] Welche Dateien stehen im Verzeichnis `/boot`? Hat das Verzeichnis Unterverzeichnisse und, wenn ja, welche?

6.4 [2] Erklären Sie den Unterschied zwischen `ls` mit einem Dateinamen als Argument und `ls` mit einem Verzeichnisnamen als Argument.

6.5 [2] Wie können Sie `ls` dazu bringen, bei einem Verzeichnisnamen als Argument Informationen über das benannte *Verzeichnis* selbst anstatt über die darin enthaltenen *Dateien* zu liefern?

6.2.3 Verzeichnisse anlegen und löschen: `mkdir` und `rmdir`

Damit Sie Ihre eigenen Dateien möglichst überschaubar halten können, ist es sinnvoll, selbst Verzeichnisse anzulegen. In diesen »Ordnern« können Sie Ihre Dateien dann zum Beispiel nach Themengebieten sortiert aufbewahren. Zur weiteren Strukturierung können Sie natürlich auch in solchen Ordnern wiederum neue Unterverzeichnisse anlegen – hier sind Ihrem Gliederungseifer kaum Grenzen gesetzt.

Verzeichnisse anlegen

Zum Anlegen neuer Verzeichnisse steht das Kommando `mkdir` (engl. *make directory*, »erstelle Verzeichnis«) zur Verfügung. `mkdir` erwartet zwingend einen oder mehrere Verzeichnisnamen als Argument, andernfalls erhalten Sie statt eines neuen Verzeichnisses lediglich eine Fehlermeldung. Um verschachtelte Verzeichnisse in einem Schritt neu anzulegen, können Sie die Option `-p` angeben, ansonsten wird angenommen, dass alle Verzeichnisse bis auf das letzte im Namen schon existieren. Zum Beispiel:

```
$ mkdir bilder/urlaub
mkdir: cannot create directory `bilder/urlaub': No such file or directory
$ mkdir -p bilder/urlaub
$ cd bilder
$ ls -F
urlaub/
```

Verzeichnisse löschen

Es kann vorkommen, dass ein Verzeichnis nicht mehr benötigt wird. Zur besseren Übersicht können Sie es dann mit dem Kommando `rmdir` (engl. *remove directory*, »entferne Verzeichnis«) wieder entfernen.

In Analogie zum Kommando `mkdir` müssen Sie auch hier den Name mindestens eines zu löschenden Verzeichnisses angeben. Außerdem müssen die Verzeichnisse leer sein, dürfen also keinerlei Einträge (Dateien oder Unterverzeichnisse) mehr enthalten. Auch hier wird nur das letzte Verzeichnis in jedem übergebenen Namen entfernt:

```
$ rmdir bilder/urlaub
$ ls -F
⊲⊲⊲⊲⊲
```

```
bilder/
◁◁◁◁◁
```

Durch Angabe der Option `-p` können Sie von rechts her alle leeren Unterverzeichnisse, die im Namen mit aufgeführt wurden, in einem Schritt mitbeseitigen.

```
$ mkdir -p bilder/urlaub/sommer
$ rmdir bilder/urlaub/sommer
$ ls -F bilder
bilder/urlaub/
$ rmdir -p bilder/urlaub
$ ls -F bilder
ls: bilder: No such file or directory
```

Übungen

6.6 [!2] Legen Sie in Ihrem Heimatverzeichnis ein Verzeichnis `grd1-test` mit den Unterverzeichnissen `dir1`, `dir2` und `dir3` an. Wechseln Sie ins Verzeichnis `grd1-test/dir1` und legen Sie (etwa mit einem Texteditor) eine Datei namens `hallo` mit dem Inhalt »`hallo`« an. Legen Sie in `grd1-test/dir2` eine Datei namens `huhu` mit dem Inhalt »`huhu`« an. Vergewissern Sie sich, dass diese Dateien angelegt wurden. Löschen Sie das Unterverzeichnis `dir3` mit `rmdir`. Versuchen Sie anschließend, das Unterverzeichnis `dir2` mit `rmdir` zu löschen. Was passiert und warum?

6.3 Suchmuster für Dateien

6.3.1 Einfache Suchmuster

Oft kommt es vor, dass Sie ein Kommando auf mehrere Dateien gleichzeitig ausführen wollen. Wenn Sie zum Beispiel alle Dateien, deren Namen mit »`p`« anfangen und mit »`.c`« enden, vom Verzeichnis `prog1` ins Verzeichnis `prog2` kopieren wollten, wäre es höchst ärgerlich, jede Datei explizit benennen zu müssen – jedenfalls wenn es sich um mehr als zwei oder drei Dateien handelt! Viel bequemer ist es, die Suchmuster der Shell einzusetzen.

Suchmuster

Wenn Sie auf der Kommandozeile der Shell einen Parameter angeben, der einen Stern enthält – etwa wie

Stern

```
prog1/p*.c
```

–, dann ersetzt die Shell diesen Parameter im tatsächlichen Programmaufruf durch eine sortierte Liste aller Dateinamen, die auf den Parameter »passen«. »Passen« dürfen Sie dabei so verstehen, dass im tatsächlichen Dateinamen statt des Sterns eine beliebig lange Folge von beliebigen Zeichen stehen darf. In Frage kommen zum Beispiel

```
prog1/p1.c
prog1/paulchen.c
prog1/pop-rock.c
prog1/p.c
```

(beachten Sie vor allem den letzten Namen im Beispiel – »beliebig lang« heißt auch »Länge Null«!). Das einzige Zeichen, auf das der Stern *nicht* passt, ist – na, fällt's Ihnen ein? – der Schrägstrich; es ist normalerweise besser, ein Suchmuster wie den Stern auf das aktuelle Verzeichnis zu beschränken.

Testen können Sie diese Suchmuster bequem mit `echo`. Ein

```
$ echo prog1/p*.c
```

gibt Ihnen unverbindlich und ohne weitere Konsequenzen jedweder Art die passenden Dateinamen aus.

Wenn Sie wirklich ein Kommando auf alle Dateien im *Verzeichnisbaum* ab einem bestimmten Verzeichnis ausführen wollen: Auch dafür gibt es Mittel und Wege. Wir reden darüber in Abschnitt 6.4.4.

Alle Dateien

Das Suchmuster »*« beschreibt »alle Dateien im aktuellen Verzeichnis« – mit Ausnahme der versteckten Dateien, deren Name mit einem Punkt anfängt. Um möglicherweise unangenehme Überraschungen auszuschließen, werden die versteckten Dateien nämlich von Suchmustern konsequent ignoriert, solange Sie nicht explizit mit etwas wie ».*« veranlassen, dass sie einbezogen werden.

Sie kennen den Stern vielleicht von der Kommandooberfläche von Betriebssystemen wie DOS oder Windows[1] und sind von dort gewöhnt, ein Muster wie »*.*« anzugeben, um alle Dateien in einem Verzeichnis zu beschreiben. Unter Linux ist das nicht richtig – das Muster »*.*« passt auf »alle Dateien mit einem Punkt im Namen«, aber der Punkt ist ja nicht vorgeschrieben. Das Linux-Äquvalent ist, wie erwähnt, »*«.

Fragezeichen

Ein Fragezeichen steht als Suchmuster für genau ein beliebiges Zeichen (wieder ohne den Schrägstrich). Ein Muster wie

```
p?.c
```

passt also auf die Namen

```
p1.c
pa.c
p-.c
p..c
```

(unter anderem). Beachten Sie, dass es aber ein Zeichen sein muss – die Option »gar nichts« besteht hier nicht.

Sie sollten sich einen ganz wichtigen Umstand sehr gründlich merken: *Die Expansion von Suchmustern ist eine Leistung der Shell!* Die aufgerufenen Kommandos wissen normalerweise nichts über Suchmuster und interessieren sich auch nicht die Bohne dafür. Alles, was sie zu sehen bekommen, sind Listen von Pfadnamen, ohne dass kommuniziert wird, wo diese Pfadnamen herkommen – also ob sie direkt eingetippt wurden oder über Suchmuster entstanden sind.

Es sagt übrigens niemand, dass die Ergebnisse der Suchmuster immer als Pfadnamen interpretiert werden. Wenn Sie zum Beispiel in einem Verzeichnis eine Datei namens »`-l`« haben, dann wird ein »`ls *`« in diesem Verzeichnis ein interessantes und vielleicht überraschendes Ergebnis liefern (siehe Übung 6.9).

Was passiert, wenn die Shell keine Datei findet, die auf das Suchmuster passt? In diesem Fall bekommt das betreffende Kommando das Suchmuster direkt als solches übergeben; was es damit anfängt, ist seine eigene Sache. Typischerweise werden solche Suchmuster als Dateinamen interpretiert, die betreffende »Datei« wird aber nicht gefunden und es gibt eine Fehlermeldung. Es gibt aber auch Kommandos, die mit direkt übergebenen Suchmustern Vernünftiges anfangen können – hier ist die Herausforderung dann eher, dafür zu sorgen, dass die Shell, die das Kommando aufruft, sich nicht mit ihrer eigenen Expansion vordrängelt. (Stichwort: Anführungszeichen)

[1] Für CP/M sind Sie wahrscheinlich zu jung.

6.3.2 Zeichenklassen

Eine etwas genauere Einschränkung der passenden Zeichen in einem Suchmuster bieten »Zeichenklassen«: In einem Suchmuster der Form

```
prog[123].c
```

passen die eckigen Klammern auf genau die Zeichen, die darin aufgezählt werden (keine anderen). Das Muster im Beispiel passt also auf

```
prog1.c
prog2.c
prog3.c
```

aber nicht

`prog.c`	*Ein Zeichen muss es schon sein*
`prog4.c`	*Die 4 ist nicht in der Aufzählung*
`proga.c`	*Das a auch nicht*
`prog12.c`	*Genau ein Zeichen, bitte!*

Bereiche

Als Schreibvereinfachung können Sie Bereiche angeben wie in

```
prog[1-9].c
[A-Z]brakadabra.txt
```

Die eckigen Klammern in der ersten Zeile passen auf alle Ziffern, die in der zweiten Zeile auf alle Großbuchstaben.

Denken Sie daran, dass in den gängigen Zeichencode-Tabellen die Buchstaben nicht lückenlos hintereinander liegen: Ein Muster wie

```
prog[A-z].c
```

passt nicht nur auf `progQ.c` und `progx.c`, sondern zum Beispiel auch auf `prog_.c`. (Schauen Sie in einer ASCII-Tabelle nach, etwa mit »`man ascii`«.) Wenn Sie nur »Groß- und Kleinbuchstaben« haben wollen, müssen Sie

```
prog[A-Za-z].c
```

schreiben.

Selbst von einer Konstruktion wie

```
prog[A-Za-z].c
```

werden Umlaute nicht erfasst, obwohl die verdächtig aussehen wie Buchstaben.

Komplement

Als weitere Schreibvereinfachung können Sie Zeichenklassen angeben, die als »alle Zeichen außer diesen« interpretiert werden: Etwas wie

```
prog[!A-Za-z].c
```

passt auf alle Namen, bei denen das Zeichen zwischen »`g`« und »`.`« *kein* Buchstabe ist. Ausgenommen ist wie üblich der Schrägstrich.

6.3.3 Geschweifte Klammern

Gerne in einem Atemzug mit Shell-Suchmustern erwähnt, auch wenn sie eigentlich eine nur entfernte Verwandte ist, wird die Expansion geschweifter Klammern in der Form

```
{rot,gelb,blau}.txt
```

– die Shell ersetzt dies durch

```
rot.txt gelb.txt blau.txt
```

Allgemein gilt, dass ein Wort auf der Kommandozeile, das einige durch Kommas getrennte Textstücke innerhalb von geschweiften Klammern enthält, durch so viele Wörter ersetzt wird, wie Textstücke zwischen den Klammern stehen, wobei in jedem dieser Wörter der komplette Klammerausdruck durch eins der Textstücke ersetzt wird. *Dieser Ersetzungsvorgang findet einzig und allein auf textueller Basis statt und ist völlig unabhängig von der Existenz oder Nichtexistenz irgendwelcher Dateien oder Verzeichnisse* – dies im Gegensatz zu Suchmustern, die immer nur solche Namen produzieren, die tatsächlich als Pfadnamen im System vorkommen.

kartesisches Produkt

Sie können auch mehr als einen Klammerausdruck in einem Wort haben, dann erhalten Sie als Ergebnis das kartesische Produkt, einfacher gesagt alle möglichen Kombinationen:

```
{a,b,c}{1,2,3}.dat
```

ergibt

```
a1.dat a2.dat a3.dat b1.dat b2.dat b3.dat c1.dat c2.dat c3.dat
```

Nützlich ist das zum Beispiel, um systematisch neue Verzeichnisse anzulegen; die normalen Suchmuster helfen da nicht, da sie ja nur bereits Existierendes finden können:

```
$ mkdir -p umsatz/200{8,9}/q{1,2,3,4}
```

Übungen

6.7 [!1] Im aktuellen Verzeichnis stehen die Dateien

```
prog.c   prog1.c  prog2.c  progabc.c  prog
p.txt    p1.txt   p21.txt  p22.txt    p22.dat
```

Auf welche dieser Dateinamen passen die Suchmuster (a) `prog*.c`, (b) `prog?.c`, (c) `p?*.txt`, (d) `p[12]*`, (e) `p*`, (f) `*.*`?

6.8 [!2] Was ist der Unterschied zwischen »`ls`« und »`ls *`«? (*Tipp:* Probieren Sie beides in einem Verzeichnis aus, das Unterverzeichnisse enthält.)

6.9 [2] Erklären Sie, warum das folgende Kommando zur angezeigten Ausgabe führt:

```
$ ls
-l  datei1  datei2  datei3
$ ls *
-rw-r--r-- 1 hugo users 0 Dec 19 11:24 datei1
-rw-r--r-- 1 hugo users 0 Dec 19 11:24 datei2
-rw-r--r-- 1 hugo users 0 Dec 19 11:24 datei3
```

6.10 [2] Warum ist es sinnvoll, dass »*« nicht auf Dateinamen mit Punkt am Anfang passt?

Tabelle 6.3: Optionen für `cp`

Option		Wirkung
`-b`	(*backup*)	Legt von existierenden Zieldateien Sicherungskopien an, indem an den Namen eine Tilde angehängt wird
`-f`	(*force*)	Überschreibt bereits existierende Zieldateien ohne Rückfrage
`-i`	(*interactive*)	fragt nach, ob bereits bestehende Dateien überschrieben werden sollen
`-p`	(*preserve*)	Behält möglichst alle Dateiattribute der Quelldatei bei
`-R`	(*recursive*)	Kopiert Verzeichnisse mit allen Unterverzeichnissen und allen darin enthaltenen Dateien
`-u`	(*update*)	Kopiert nur, wenn die Quelldatei neuer als die Zieldatei ist (oder noch keine Zieldatei existiert)
`-v`	(*verbose*)	Zeigt alle Aktivitäten auf dem Bildschirm

6.4 Umgang mit Dateien

6.4.1 Kopieren, Verschieben und Löschen – `cp` und Verwandte

Dateien kopieren

Mit dem Kommando `cp` (engl. *copy*, »kopieren«) können Sie beliebige Dateien kopieren. Dabei gibt es zwei Vorgehensweisen:

1 : 1-Kopie

Nennen Sie `cp` die Namen von Quell- und Zieldatei als Argumente, dann wird unter dem Zieldateinamen eine 1 : 1-Kopie des Inhalts der Quelldatei abgelegt. Standardmäßig fragt `cp` nicht nach, ob es eine bereits existierende Zieldatei überschreiben soll, sondern tut dies einfach – hier ist also Vorsicht (oder die Option `-i`) angebracht.

Zielverzeichnis

Statt eines Zieldateinamens können Sie auch ein Zielverzeichnis angeben. Die Quelldatei wird dann unter ihrem alten Namen in das Verzeichnis hineinkopiert.

```
$ cp liste liste2
$ cp /etc/passwd .
$ ls -l
-rw-r--r--  1  hugo  users   2500  Oct 4 11:11 liste
-rw-r--r--  1  hugo  users   2500  Oct 4 11:25 liste2
-rw-r--r--  1  hugo  users   8765  Oct 4 11:26 passwd
```

Im Beispiel haben wir mit `cp` zunächst eine exakte Kopie der Datei `liste` unter dem Namen `liste2` erzeugt. Anschließend haben wir noch die Datei `/etc/passwd` ins aktuelle Verzeichnis (repräsentiert durch den Punkt als Zielverzeichnis) kopiert. Die wichtigsten der zahlreichen `cp`-Optionen sind in Tabelle 6.3 aufgelistet.

Liste von Quelldateien

Statt einer einzigen Quelldatei sind auch eine längere Liste von Quelldateien oder die Verwendung von Shell-Suchmustern zulässig. Allerdings lassen sich auf diese Art nicht mehrere Dateien in einem Schritt kopieren und umbenennen, sondern nur in ein anderes Verzeichnis kopieren. Während in der DOS- und Windows-Welt mit »`COPY *.TXT *.BAK`« von jeder Datei mit der Endung `TXT` eine Sicherungskopie mit der Endung `BAK` angelegt wird, versagt unter Linux der Befehl »`cp *.txt *.bak`« normalerweise unter Ausgabe einer Fehlermeldung.

Um dies zu verstehen, müssen Sie sich vergegenwärtigen, wie die Shell diesen Befehl verarbeitet. Sie versucht zunächst, alle Suchmuster durch die passenden Dateinamen zu ersetzen, also zum Beispiel `*.txt` durch `brief1.txt` und `brief2.txt`. Was mit `*.bak` passiert, hängt davon ab, auf wieviele Namen `*.txt` gepasst hat und ob es im aktuellen Verzeichnis Namen gibt, die auf `*.bak` passen – allerdings wird fast nie das passieren, was ein DOS-Anwender erwarten würde! Normalerweise wird es wohl darauf hinauslaufen, dass die Shell dem `cp`-Kommando das unersetzte Suchmuster `*.bak` als letztes einer Reihe von Argumenten übergibt, und das geht aus der Sicht von `cp`

schief, da es sich dabei nicht um einen gültigen Verzeichnisnamen handelt (handeln dürfte).

Datei verschieben/umbenennen

Während das Kommando `cp` eine exakte Kopie einer Datei anlegt, also die Datei physikalisch auf dem Datenträger verdoppelt oder auf einem anderen Datenträger identisch anlegt, ohne das Original zu verändern, dient der Befehl `mv` (engl. *move*, »bewegen«) dazu, eine Datei entweder an einen anderen Ort zu verschieben oder deren Namen zu verändern. Dieser Vorgang verändert lediglich Verzeichnisse, es sei denn, die Datei wird auf ein anderes Dateisystem verlagert – etwa von einer Partition auf der Platte auf einen USB-Stick. Dabei wird dann tatsächlich ein physikalisches Verschieben (Kopieren an den neuen Platz und anschließendes Löschen am alten) notwendig.

Syntax und Regeln von `mv` entsprechen denen von `cp` – auch hier können Sie statt einer einzigen eine ganze Liste von Quelldateien angeben, woraufhin das Kommando als letzte Angabe ein Zielverzeichnis erwartet. Einziger Unterschied: Neben gewöhnlichen Dateien können Sie auch Verzeichnisse umbenennen.

Die Optionen `-b`, `-f`, `-i`, `-u` und `-v` entsprechen für `mv` in ihrer Funktion den bei `cp` beschriebenen Parametern.

```
$ mv passwd liste2
$ ls -l
-rw-r--r--  1  hugo  users   2500  Oct 4 11:11 liste
-rw-r--r--  1  hugo  users   8765  Oct 4 11:26 liste2
```

Im Beispiel ist die ursprüngliche Datei `liste2` durch die umbenannte Datei `passwd` ersetzt worden. Ebenso wie `cp` fragt auch `mv` nicht nach, wenn die angegebene Zieldatei bereits existiert, sondern überschreibt diese gnadenlos.

Löschen von Dateien

Der Befehl zum Löschen von Dateien lautet `rm` (engl. *remove*, »entfernen«). Um eine Datei löschen zu dürfen, müssen Sie im entsprechenden Verzeichnis Schreibrechte besitzen. Daher sind Sie im eigenen Heimatverzeichnis der uneingeschränkte Herr im Haus, dort dürfen Sie, gegebenenfalls nach einer entsprechenden Rückfrage, auch fremde Dateien löschen (falls es welche gibt).

Das Schreibrecht auf die *Datei* selbst ist dagegen zum Löschen völlig irrelevant, genau wie die Frage, welchem Benutzer oder welcher Gruppe die Datei gehört.

Löschen ist endgültig!

`rm` geht bei seiner Arbeit genauso konsequent vor wie `cp` oder `mv` – die angegebenen Dateien werden ohne Rückfrage oder -meldung unwiederbringlich aus dem Dateisystem getilgt. Besonders bei der Verwendung von Platzhaltern sollten Sie darum nur mit großer Vorsicht vorgehen. Im Unterschied zur DOS-Welt ist der Punkt innerhalb von Linux-Dateinamen ein Zeichen ohne besondere Bedeutung. Aus diesem Grund löscht das Kommando »`rm *`« unerbittlich alle nicht versteckten Dateien im aktuellen Verzeichnis. Zwar bleiben dabei wenigstens die Unterverzeichnisse unbehelligt; mit »`rm -r *`« müssen aber auch diese daran glauben.

Als Systemadministrator können Sie mit unüberlegten Kommandos wie »`rm -rf /`« im Wurzelverzeichnis das ganze System demolieren; hier ist allergrößte Aufmerksamkeit angebracht! Leicht tippt man einmal »`rm -rf bla *`« statt »`rm -rf bla*`«.

Wo `rm` ohne Rücksicht alle Dateien löscht, die als Parameter angegeben wurden, geht »`rm -i`« etwas behutsamer vor:

```
$ rm -i lis*
rm: remove 'liste'? n
rm: remove 'liste2'? y
$ ls -l
-rw-r--r--  1  hugo  users   2500  Oct 4 11:11 liste
```

Wie das Beispiel zeigt, wird für jede zum Suchmuster passende Datei einzeln nachgefragt, ob sie gelöscht werden soll (»y« für engl. *yes* = »ja«) oder nicht (»n« für engl. *no* = »nein«).

In der deutschen Sprachumgebung können Sie auch »j« für »ja« verwenden. »y« funktioniert trotzdem.

Arbeitsumgebungen wie KDE unterstützen meistens die Verwendung eines »Papierkorbs«, in dem über den Dateimanager gelöschte Dateien erst einmal landen und aus dem Sie sie bei Bedarf wieder hervorholen können. Es gibt äquivalente Ansätze auch für die Kommandozeile, wobei Sie auch da auf eine Umdefinition von `rm` verzichten sollten.

Neben den bereits beschriebenen Optionen `-i` und `-r` sind ferner die bei `cp` beschriebenen Optionen `-v` und `-f` mit vergleichbarer Wirkung für `rm` zulässig.

Übungen

6.11 [!2] Legen Sie in Ihrem Heimatverzeichnis eine Kopie der Datei `/etc/services` unter dem Namen `myservices` an. Benennen Sie sie um in `srv.dat` und kopieren Sie sie unter demselben Namen ins Verzeichnis `/tmp`. Löschen Sie anschließend beide Kopien der Datei.

6.12 [1] Warum hat `mv` keine `-R`-Option wie `cp`?

6.13 [!2] Angenommen, in einem Ihrer Verzeichnisse steht eine Datei namens »`-file`« (mit einem Minuszeichen am Anfang des Namens). Wie würden Sie diese Datei entfernen?

6.14 [2] Wenn Sie ein Verzeichnis haben, in dem Sie nicht versehentlich einem »`rm *`« zum Opfer fallen wollen, können Sie dort eine Datei namens »`-i`« anlegen, etwa mit

```
$ > -i
```

(wird in Kapitel 7 genauer erklärt). Was passiert, wenn Sie jetzt das Kommando »`rm *`« ausführen, und warum?

6.4.2 Dateien verknüpfen – `ln` und `ln -s`

In Linux können Sie Verweise auf Dateien, sogenannte *links*, anlegen und so einzelnen Dateien mehrere Namen geben. Aber wofür ist das gut? Die Anwendungsfälle reichen von Schreibvereinfachungen bei Datei- und Verzeichnisnamen über ein »Sicherheitsnetz« gegen ungewollte Löschoperationen oder Bequemlichkeiten beim Programmieren bis zum Platzsparen bei großen Dateihierarchien, die in mehreren Versionen mit nur kleinen Änderungen vorliegen sollen.

Mit dem Befehl `ln` (engl. *link*, »verknüpfen«) können Sie einer Datei neben dem ursprünglichen Namen (erstes Argument) einen weiteren (zweites Argument) zuweisen:

```
$ ln liste liste2
$ ls -l
-rw-r--r--  2  hugo  users   2500  Oct 4 11:11 liste
-rw-r--r--  2  hugo  users   2500  Oct 4 11:11 liste2
```

Eine Datei mit mehreren Namen

Das Verzeichnis enthält nun scheinbar eine neue Datei `liste2`. In Wahrheit handelt es sich hier jedoch nur um zwei Verweise auf ein und dieselbe Datei. Einen Hinweis auf diesen Sachverhalt liefert der **Referenzzähler**, der in der zweiten Spalte der Ausgabe von »`ls -l`« zu sehen ist. Dessen Wert 2 gibt an, dass diese Datei zwei Namen hat. Ob es sich bei den beiden Dateinamen nun wirklich um Verweise auf

Referenzzähler

Inode-Nummern

dieselbe Datei handelt, ist nur mit Hilfe von »`ls -i`« eindeutig zu entscheiden. In diesem Fall muss die in der ersten Spalte angezeigte Dateinummer für beide Dateien identisch sein. Dateinummern, auch **Inode-Nummern** genannt, identifizieren Dateien eindeutig auf ihrem Dateisystem:

```
$ ls -i
876543 liste  876543 liste2
```

»Inode« ist kurz für *indirection node*, also etwa »Indirektionsknoten« oder »Weiterleitungsknoten«. In den Inodes sind alle Informationen gespeichert, die das System über eine Datei hat, bis auf den Namen. Jede Datei hat genau ein Inode.

Wenn Sie den Inhalt einer der beiden Dateien verändern, ändert sich der Inhalt der anderen ebenfalls, da in Wirklichkeit ja nur eine einzige Datei mit der eindeutigen Nummer 876543 existiert. Wir haben der Datei mit der Nummer 876543 lediglich einen weiteren Namen gegeben.

Verzeichnisse sind nichts anderes als Tabellen, in denen Dateinamen Inode-Nummern zugeordnet werden. Offensichtlich kann es in einer Tabelle mehrere Einträge mit verschiedenen Namen und derselben Inode-Nummer geben. Ein Verzeichniseintrag mit einem Namen und einer Inode-Nummer heißt »Link«.

Sie sollten sich klar machen, dass es bei einer Datei mit zwei Links nicht möglich ist, festzustellen, welcher Name »das Original« ist, also der erste Parameter im `ln`-Kommando. Beide Namen sind aus der Sicht des Systems absolut gleichwertig und ununterscheidbar.

Links auf Verzeichnisse sind unter Linux übrigens nicht erlaubt. Ausgenommen davon sind »`.`« und »`..`«, die vom System für jedes Verzeichnis verwaltet werden. Da auch Verzeichnisse Dateien sind und demnach Inode-Nummern haben, können Sie so verfolgen, wie das Dateisystem intern zusammenhängt. (Siehe hierzu auch Übung 6.20)

Wenn Sie eine der Dateien mit `rm` »löschen«, reduziert das zunächst nur die Anzahl der Namen für Datei Nummer 876543 (der Referenzzähler wird entsprechend angepasst). Erst wenn der Referenzzähler durch das Entfernen eines Namens den Wert 0 annimmt, wird die Datei tatsächlich gelöscht.

```
$ rm liste
$ ls -li
876543 -rw-r--r--  1  hugo  users   2500  Oct 4 11:11 liste2
```

Da Inode-Nummern nur auf demselben physikalischen Dateisystem (Partition, USB-Stick, …) eindeutig sind, sind solche Verknüpfungen nur in demselben Dateisystem möglich, wo auch die Datei liegt.

Ganz stimmt es nicht mit dem Löschen der Dateidaten: Wenn der letzte Dateiname entfernt wird, ist die Datei nicht mehr zugänglich, aber wenn ein Prozess noch mit ihr arbeitet, darf er sie weiterbenutzen, bis er sie explizit schließt oder sich beendet. In Unix-Programmen ist das ein gängiges Idiom für den Umgang mit temporären Dateien, die beim Programmende verschwinden sollen: Sie erzeugen sie mit Schreib- und Lesezugriff und »löschen« sie dann gleich wieder, ohne sie jedoch gleich zu schließen. Anschließend können Sie Daten hinein schreiben und später an den Dateianfang zurückspringen, um sie wieder zu lesen.

`ln` können Sie außer mit zwei Argumenten auch mit einem oder vielen aufrufen. Im ersten Fall wird im aktuellen Verzeichnis ein Link mit demselben Namen angelegt wie die Ursprungsdatei (die sinnvollerweise nicht im aktuellen Verzeichnis liegen sollte), im zweiten Fall werden alle benannten Dateien unter ihrem Namen in das durch das letzte Argument gegebene Verzeichnis »gelinkt« (denken Sie an `mv`).

Mit dem Kommando »`cp -l`« können Sie eine »Link-Farm« anlegen. Das heißt, die angegebenen Dateien werden nicht (wie sonst üblich) an den Zielort kopiert, sondern es werden Links auf die Originale angelegt: Link-Farm

```
$ mkdir prog-1.0.1                                    Neues Verzeichnis
$ cp -l prog-1.0/* prog-1.0.1
```

Der Vorteil dieses Ansatzes ist, dass die Dateien nach wie vor nur einmal auf der Platte stehen und damit auch nur einmal Platz verbrauchen. Bei den heutigen Preisen für Plattenplatz ist das vielleicht nicht zwingend nötig – aber eine gängige Anwendung dieser Idee besteht zum Beispiel darin, regelmäßige Sicherheitskopien von großen Dateihierarchien anzulegen, die auf dem Archivmedium (Platte oder entfernter Rechner) dann als separate, datierte Dateibäume erscheinen sollen. Erfahrungsgemäß ändern sich die meisten Dateien nur selten, und wenn man diese Dateien dann nur einmal abspeichern muss statt immer und immer wieder, dann addiert sich das mit der Zeit schon auf. Außerdem muss man die Dateien dann nicht immer wieder in die Sicherheitskopie schreiben, was mitunter massiv Zeit spart.

Backup-Pakete, die diese Idee aufgreifen, sind zum Beispiel Rsnapshot (`http://www.rsnapshot.org/`) oder Dirvish (`http://www.dirvish.org/`).

Dieser Ansatz ist mitunter mit Vorsicht zu genießen: Zwar können identische Dateien über Links »dedupliziert« werden, Verzeichnisse aber nicht. Das heißt, für jeden datierten Verzeichnisbaum auf dem Archivmedium müssen sämtliche Verzeichnisse neu angelegt werden, selbst wenn in den Verzeichnissen wiederum nur Links auf existierende Dateien stehen. Das kann zu sehr komplexen Verzeichnisstrukturen führen und im Extremfall dazu, dass eine Konsistenzprüfung des Archivmediums fehlschlägt, weil der Rechner nicht genug Hauptspeicher hat, um die Verzeichnishierarchie zu prüfen.

Sie müssen natürlich auch aufpassen, wenn Sie etwa – wie im Beispiel angedeutet – eine »Kopie« eines Programmquellcodes als Link-Farm machen (was sich zum Beispiel für den Linux-Quellcode durchaus plattenplatzmäßig rentieren könnte): Bevor Sie eine Datei in Ihrer neu angelegten Version ändern können, müssen Sie dafür sorgen, dass es sich tatsächlich um eine eigenständige Datei handelt und nicht nur um ein Link auf das Original (das Sie ja höchstwahrscheinlich nicht ändern wollen). Das heißt, Sie müssen das Link auf die Datei entweder manuell durch eine tatsächliche Kopie ersetzen oder einen Editor benutzen, der geänderte Versionen automatisch als eigenständige Datei schreibt[2].

Das ist noch nicht alles: In Linux-Systemen gibt es zwei Arten von Links. Das oben stehende Beispiel entspricht dem Standard des Kommandos `ln` und wird als *hard link* (engl. »feste Verknüpfung«) bezeichnet. Zur Identifikation der Datei wird dabei die Dateinummer gespeichert. **Symbolische Links** (oder in Anlehnung an die *hard links* von eben auch *soft links*, »weiche Verknüpfungen«) sind selbst eigentlich Dateien, in denen aber nur der Name der »Zieldatei« des Links steht; gleichzeitig wird vermerkt, dass es sich um ein symbolisches Link handelt und Symbolische Links

[2] Wenn Sie den Vim (alias `vi`) verwenden, können Sie in die Datei `.vimrc` in Ihrem Heimatverzeichnis das Kommando »`set backupcopy=auto,breakhardlink`« schreiben.

bei Zugriffen auf das Link in Wirklichkeit die Zieldatei gemeint ist. Anders als bei harten Links »weiß« die Zieldatei nicht, dass ein symbolisches Link auf sie existiert. Das Anlegen oder Löschen eines symbolischen Links hat keine Auswirkungen auf die Zieldatei; wird dagegen die Zieldatei entfernt, weist der symbolische Link ins Leere (Zugriffe führen zu einer Fehlermeldung).

Verweise auf Verzeichnisse

Im Gegensatz zu harten Links lassen symbolische Links auch Verweise auf Verzeichnisse zu sowie auf Dateien, die sich im Verzeichnisbaum auf anderen physikalischen Dateisystemen befinden. In der Praxis werden symbolische Links oft bevorzugt, da sich die Verknüpfungen anhand des Pfadnamens leichter nachvollziehen lassen.

Symbolische Links werden gerne verwendet, wenn sich Datei- oder Verzeichnisnamen ändern, aber eine gewisse Rückwärtskompatibilität erwünscht ist. Beispielsweise hat man sich geeinigt, die Postfächer von Systembenutzern (wo deren ungelesene E-Mail steht) im Verzeichnis `/var/mail` abzulegen. Traditionell hieß das Verzeichnis `/var/spool/mail`, und viele Programme haben diesen Namen fest einprogrammiert. Um eine Umstellung auf `/var/mail` zu erleichtern, kann eine Distribution also ein symbolisches Link unter dem Namen `/var/spool/mail` einrichten, das auf `/var/mail` verweist. (Mit harten Links wäre das nicht möglich, weil harte Links auf Verzeichnisse nicht erlaubt sind.)

Um ein symbolisches Link zu erzeugen, müssen Sie dem Kommando `ln` die Option `-s` übergeben:

```
$ ln -s /var/log kurz
$ ls -l
-rw-r--r--  1  hugo  users   2500  Oct 4 11:11 liste2
lrwxrwxrwx  1  hugo  users     14  Oct 4 11:40 kurz -> /var/log
$ cd kurz
$ pwd -P
/var/log
```

Neben der Option `-s` zur Erstellung von »soft links« unterstützt das Kommando `ln` unter anderem die bereits bei `cp` besprochenen Optionen `-b`, `-f`, `-i` und `-v`.

Um nicht mehr benötigte symbolische Links wieder zu entfernen, können Sie sie einfach wie gewöhnliche Dateien mit dem Kommando `rm` löschen. *Diese* Operation wirkt auf das Link und nicht das Ziel des Links:

```
$ cd
$ rm kurz
$ ls
liste2
```

Wie Sie weiter oben gesehen haben, zeigt »`ls -l`« für symbolische Links auch die Datei an, auf die das Link zeigt. Mit den Optionen `-L` und `-H` können Sie `ls` dazu bringen, symbolische Links direkt aufzulösen:

```
$ mkdir dir
$ echo XXXXXXXXX >dir/datei
$ ln -s datei dir/symlink
$ ls -l dir
insgesamt 4
-rw-r--r-- 1 hugo users 11 Jan 21 12:29 datei
lrwxrwxrwx 1 hugo users  5 Jan 21 12:29 symlink -> datei
$ ls -lL dir
-rw-r--r-- 1 hugo users 11 Jan 21 12:29 datei
-rw-r--r-- 1 hugo users 11 Jan 21 12:29 symlink
$ ls -lH dir
```

```
-rw-r--r-- 1 hugo users 11 Jan 21 12:29 datei
lrwxrwxrwx 1 hugo users  5 Jan 21 12:29 symlink -> datei
$ ls -l dir/symlink
lrwxrwxrwx 1 hugo users 5 Jan 21 12:29 dir/symlink -> datei
$ ls -lH dir/symlink
-rw-r--r-- 1 hugo users 11 Jan 21 12:29 dir/symlink
```

Der Unterschied zwischen `-L` und `-H` ist, dass die Option `-L` symbolische Links *immer* auflöst und die Informationen über die eigentliche Datei anzeigt (der angezeigte Name ist trotzdem immer der des Links). Die Option `-H` tut das, wie die letzten drei Kommandos im Beispiel illustrieren, nur für direkt auf der Kommandozeile angegebene Links.

`cp` und symbolische Links

In Analogie zu »`cp -l`« erstellt »`cp -s`« Link-Farmen auf der Basis von symbolischen Links. Die sind allerdings nicht ganz so nützlich wie die oben gezeigten mit harten Links. »`cp -a`« kopiert Dateihierarchien so, wie sie sind, unter Beibehaltung symbolischer Links als solche; »`cp -L`« sorgt beim Kopieren dafür, dass symbolische Links durch die Dateien ersetzt werden, auf die sie zeigen, und »`cp -P`« schließt das aus.

Übungen

6.15 [!2] Erzeugen Sie eine Datei mit beliebigem Inhalt in Ihrem Heimatverzeichnis (etwa mit »`echo Hallo >~/hallo`« oder einem Texteditor). Legen Sie unter dem Namen `link` ein hartes Link auf die Datei an. Überzeugen Sie sich, dass nun zwei Namen für die Datei existieren. Versuchen Sie den Dateiinhalt mit einem Editor zu ändern. Was passiert?

6.16 [!2] Legen Sie unter dem Namen `~/symlink` ein symbolisches Link auf die Datei aus der vorigen Aufgabe an. Prüfen Sie, ob der Zugriff funktioniert. Was passiert, wenn Sie die Zieldatei des Links löschen?

6.17 [!2] Auf welches Verzeichnis zeigt das `..`-Link im Verzeichnis »`/`«?

6.18 [3] Betrachten Sie das folgende Kommando und seine Ausgabe:

```
$ ls -ai /
     2 .      330211 etc         1 proc   4303 var
     2 ..          2 home    65153 root
  4833 bin    244322 lib    313777 sbin
228033 boot   460935 mnt    244321 tmp
330625 dev    460940 opt    390938 usr
```

Offensichtlich haben die Verzeichnisse `/` und `/home` dieselbe Inodenummer. Da es sich dabei offensichtlich nicht wirklich um dasselbe Verzeichnis handeln kann – können Sie dieses Phänomen erklären?

6.19 [2] Betrachten Sie die Inode-Nummern der Links »`.`« und »`..`« im Wurzelverzeichnis (`/`) und in einigen beliebigen anderen Verzeichnissen. Was fällt Ihnen auf?

6.20 [3] Wir haben gesagt, dass harte Links auf Verzeichnisse nicht erlaubt sind. Welchen Grund könnte es dafür geben?

6.21 [3] Woran erkennt man in der Ausgabe von »`ls -l ~`«, dass ein *Unterverzeichnis* von `~` keine weiteren Unterverzeichnisse hat?

6.22 [2] Wie verhalten sich »`ls -lH`« und »`ls -lL`«, wenn ein symbolisches Link auf ein anderes symbolisches Link zeigt?

Tabelle 6.4: Tastaturbefehle für `more`

Taste	Wirkung
↩	zeigt nächste Zeile an
␣	zeigt nächste Seite an
b	zeigt vorherige Seite an
h	gibt einen Hilfstext aus
q	beendet `more`
/ ⟨*Wort*⟩ ↩	sucht nach ⟨*Wort*⟩
! ⟨*Kommando*⟩ ↩	führt ⟨*Kommando*⟩ in Sub-Shell aus
v	ruft Editor (`vi`) auf
Strg + l	zeichnet Bildschirm neu

6.23 [3] Wie lang darf eine »Kette« von symbolischen Links maximal sein? (Mit anderen Worten, wenn Sie mit einem symbolischen Link auf eine Datei anfangen, wie oft können Sie ein symbolisches Link anlegen, das auf das jeweils vorige symbolische Link verweist?)

6.24 [4] (Nachdenk- und Rechercheaufgabe:) Was braucht mehr Platz auf der Platte, ein hartes oder ein symbolisches Link? Warum?

6.4.3 Dateiinhalte anzeigen – `more` und `less`

Darstellung von Textdateien

Eine komfortable Darstellung von Textdateien am Bildschirm ist mittels `more` (engl. für »mehr«) möglich. Mit diesem Kommando können Sie längere Texte seitenweise betrachten. Die Anzeige wird dabei nach einer Bildschirmseite automatisch angehalten und in der letzten Zeile »`--More--`«, gegebenenfalls ergänzt durch eine Prozentangabe, ausgegeben. Der Prozentwert zeigt, welcher Anteil der Datei bereits dargestellt wurde. Auf Tastendruck wird die Ausgabe des restlichen Textes fortgesetzt. Welche Tasten eine Funktion besitzen, zeigt Tabelle 6.4.

Optionen

`more` erlaubt natürlich auch einige Optionen. Durch `-s` (engl. *squeeze*, »quetschen«) werden mehrere aufeinanderfolgende Leerzeilen zu einer einzigen zusammengefasst, Seitenvorschübe (repräsentiert durch das Symbol »^L«) werden bei Übergabe von `-l` ignoriert. Die Zeilenzahl des Bildschirms kann mit `-n` ⟨*Zahl*⟩ auf ⟨*Zahl*⟩ Zeilen eingestellt werden, andernfalls liest `more` den aktuellen Wert aus der Variablen `TERM`.

Einschränkungen

Die Textdarstellung von `more` ist noch immer gravierenden Einschränkungen unterworfen. Etwa fehlt eine allgemein mögliche Rückwartsbewegung in der Ausgabe. Aus diesem Grund ist heute meist die verbesserte Version `less` (engl. für »weniger«[3]) im Einsatz. Nun können Sie sich mit den Cursortasten wie gewohnt vertikal durch den Text bewegen. Ferner wurden die Suchroutinen erweitert und ermöglichen die Suche sowohl textauf- als auch -abwärts. Tabelle 6.5 bietet einen Überblick über die wichtigsten Tastaturbefehle.

`less`

Wie bereits in Kapitel 4 erwähnt, ist `less` üblicherweise als Anzeigeprogramm für `man` voreingestellt. Alle diese Tastaturkommandos greifen daher auch bei der Anzeige der Online-Hilfe via `man`.

6.4.4 Dateien suchen – `find`

Welcher Anwender kennt das nicht: »Da gab's doch mal eine Datei `sowieso`, aber wo war die gleich?« Natürlich können Sie alle Verzeichnisse mühsam von Hand nach der gesuchten Datei durchkämmen. Aber Linux wäre nicht Linux, wenn es Ihnen nicht hilfreich zur Seite stünde.

Das Kommando `find` sucht den Verzeichnisbaum rekursiv nach Dateien ab, die den angegebenen Kriterien entsprechen. »Rekursiv« bedeutet, dass es auch

[3] Ein schwaches Wortspiel – denken Sie an »weniger ist mehr«.

Tabelle 6.5: Tastaturbefehle für `less`

Taste	Wirkung
↓ oder e oder j oder ↩	zeigt nächste Zeile an
f oder Leertaste	zeigt nächste Seite an
↑ oder y oder k	zeigt vorherige Zeile an
b	zeigt vorherige Seite an
Pos1 oder g	an den Textanfang springen
Ende oder ⇑+g	ans Textende springen
p ⟨*Zahl*⟩ ↩	springt an Position ⟨*Zahl*⟩ (in %) des Textes
h	gibt einen Hilfetext aus
q	beendet `less`
/ ⟨*Wort*⟩ ↩	sucht abwärts nach ⟨*Wort*⟩
n	setzt Suche abwärts fort
? ⟨*Wort*⟩ ↩	sucht aufwärts nach ⟨*Wort*⟩
⇑+n	setzt Suche aufwärts fort
! ⟨*Kommando*⟩ ↩	führt ⟨*Kommando*⟩ in Sub-Shell aus
v	ruft Editor (`vi`) auf
r oder Strg+l	zeichnet Bildschirm neu

alle im Startverzeichnis enthaltenen Unterverzeichnisse, deren Unterverzeichnisse usw. mit erfasst. Als Suchergebnis liefert `find` die Pfadnamen der gefundenen Dateien, die dann an andere Programme weitergeleitet werden können. Zur Verdeutlichung der Befehlsstruktur ein Beispiel:

```
$ find . -user hugo -print
./liste
```

Hier wird also das aktuelle Verzeichnis inklusive aller Unterverzeichnisse nach Dateien durchsucht, die dem Benutzer `hugo` gehören. Die Anweisung `-print` dient dazu, das Suchergebnis, im Beispiel nur eine einzige Datei, auf dem Bildschirm auszugeben. Zur Arbeitserleichterung wird daher die Angabe `-print` automatisch ergänzt, wenn Sie nicht ausdrücklich gesagt haben, wie mit den gefundenen Dateinamen zu verfahren ist.

Wie anhand des Beispiels zu sehen ist, benötigt `find` einige Angaben, um seine Aufgabe ordnungsgemäß erledigen zu können.

Startverzeichnis Die Auswahl des Startverzeichnisses sollte mit Bedacht erfolgen. Wenn Sie das Wurzelverzeichnis wählen, wird die gesuchte Datei – sofern sie denn existiert – mit Sicherheit gefunden werden, der Suchvorgang kann jedoch viel Zeit in Anspruch nehmen. Natürlich dürfen Sie nur in denjenigen Verzeichnissen suchen, auf die Sie angemessene Zugriffsrechte haben.

Die Angabe eines absoluten Pfadnamens für das Startverzeichnis liefert als Suchergebnis absolute Pfadnamen, ein relativ angegebenes Startverzeichnis ergibt entsprechend relative Resultate.

Ausgabe absolut oder relativ?

Statt eines einzelnen Startverzeichnisses können Sie auch eine Liste von Verzeichnisnamen angeben, die dann der Reihe nach durchsucht werden.

Verzeichnisliste

Auswahlkriterien Mit diesen Angaben können Sie die Merkmale der zu findenden Dateien genauer festlegen. Tabelle 6.6 enthält eine Auswahl zulässiger Angaben, weitere stehen in der Dokumentation.

Tabelle 6.6: Testkriterien von `find`

Test	Beschreibung
`-name`	Sucht nach passenden Dateinamen. Hierbei sind alle Sonderzeichen nutzbar, die von der Shell zur Verfügung gestellt werden. Mit `-iname` werden dabei Groß- und Kleinbuchstaben gleich behandelt.
`-user`	Sucht nach Dateien, die dem angegebenen Benutzer gehören. Dabei ist statt des Anwendernamens auch die Angabe der eindeutigen Benutzernummer, der UID, möglich.
`-group`	Sucht nach Dateien, die zu der angegebenen Gruppe gehören. Wie bei `-user` ist hier statt des Gruppennamens auch die Angabe der eindeutigen Gruppennummer, der GID, zulässig.
`-type`	Ermöglicht die Suche nach verschiedenen Dateitypen (siehe Abschnitt 9.2). Dabei werden unterschieden: `b` blockorientierte Gerätedatei `c` zeichenorientierte Gerätedatei `d` Verzeichnis `f` normale Datei `l` Symbolisches Link `p` FIFO (*named pipe*) `s` Unix-Domain-Socket
`-size`	Sucht nach Dateien bestimmter Größe. Zahlenwerte werden dabei als 512-Byte-Blöcke interpretiert, durch ein nachgestelltes `c` sind Größenangaben in Byte, durch `k` in Kibibyte erlaubt. Vorangestellte Plus- oder Minuszeichen entsprechen Unter- und Obergrenzen, womit ein Größenbereich abgedeckt werden kann. Das Kriterium `-size +10k` trifft z. B. für alle Dateien zu, die größer als 10 KiB sind.
`-atime`	(engl. *access*, »Zugriff«) sucht Dateien nach dem Zeitpunkt des letzten Zugriffs. Hier sowie für die beiden nächsten Auswahlkriterien wird der Zeitraum in Tagen angegeben; …`min` statt …`time` ermöglicht eine minutengenaue Suche.
`-mtime`	(engl. *modification*, »Veränderung«) wählt passende Dateien über den Zeitpunkt der letzten Veränderung aus.
`-ctime`	(engl. *change*, »Änderung«) sucht Dateien anhand der letzten Änderung der Inodes (durch Zugriff auf den Inhalt, Rechteänderung, Umbenennen etc.)
`-perm`	Findet nur Dateien, deren Zugriffsrechte genau mit den angegebenen übereinstimmen. Die Festlegung erfolgt mittels einer Oktalzahl, die beim Befehl `chmod` beschrieben wird. Möchte man nur nach einem bestimmten Recht suchen, muss der Oktalzahl ein Minuszeichen vorangestellt werden, z. B. berücksichtigt `-perm -20` alle Dateien, die Gruppenschreibrechte besitzen, unabhängig von deren übrigen Zugriffsrechten.
`-links`	Sucht nach Dateien, deren Referenzzähler den angegebenen Zahlenwert hat.
`-inum`	Findet Verweise auf die Datei mit der angegebenen Inode-Nummer.

mehrere Auswahlkriterien

Wenn Sie mehrere Auswahlkriterien gleichzeitig angeben, werden diese automatisch Und-verknüpft, es müssen also alle erfüllt werden. Daneben versteht `find` noch andere logische Operationen (siehe Tabelle 6.7).

Kompliziertere logische Verknüpfungen von Auswahlkriterien können (bzw. sollten) in runde Klammern eingeschlossen werden, um fehlerhafte Auswertungen zu vermeiden. Die Klammern müssen Sie natürlich vor der Shell verstecken:

```
$ find . \( -type d -o -name "A*" \) -print
```

```
./.
./..
./bilder
./Abfall
$ _
```

Das Beispiel listet alle Dateien auf dem Bildschirm auf, die entweder Verzeichnisse repräsentieren oder deren Name mit einem »A« beginnt oder beides.

Aktionen Die Ausgabe der Suchergebnisse auf dem Bildschirm geschieht, wie eingangs erwähnt, mit der Kommandooption `-print`. Daneben existieren mit `-exec` (engl. *execute*, »ausführen«) und `-ok` noch zwei weitere Optionen, die es ermöglichen, Folgekommandos mit den gefundenen Dateien auszuführen. Hierbei unterscheidet sich `-ok` nur dadurch von `-exec` dass vor der Ausführung des Folgekommandos der Benutzer um Zustimmung gebeten wird; `-exec` setzt diese stillschweigend voraus. Im weiteren Text steht daher `-exec` stellvertretend für beide Varianten.

Kommando ausführen

Zur Anwendung von `-exec` gibt es einige allgemeingültige Regeln:

- Das Kommando hinter `-exec` ist mit einem Strichpunkt (»;«) abzuschließen. Da dieser in üblichen Shells eine Sonderbedeutung hat, muss er maskiert werden (etwa als »\;« oder mit Anführungszeichen), damit `find` ihn überhaupt zu sehen bekommt.
- Zwei geschweifte Klammern (»{}«) werden in dem Kommando durch den gefundenen Dateinamen ersetzt. Am besten setzen Sie die geschweiften Klammern in Anführungszeichen, um Probleme mit Leerzeichen in Dateinamen zu vermeiden.

Zum Beispiel:

```
$ find . -user hugo -exec ls -l '{}' \;
-rw-r--r--  1  hugo  users   4711  Oct 4 11:11 datei.txt
$ _
```

Das obenstehende Beispiel sucht nach allen Dateien innerhalb des aktuellen Verzeichnisses, die dem Anwender `hugo` gehören und führt für diese das Kommando »`ls -l`« aus. Mehr Sinn ergibt da schon folgendes:

```
$ find . -atime +13 -exec rm -i '{}' \;
```

Hiermit werden alle Dateien im aktuellen Verzeichnis und darunter interaktiv gelöscht, auf die seit zwei Wochen oder länger nicht mehr zugegriffen wurde.

Mitunter – etwa im letzten Beispiel – ist es sehr ineffizient, für jeden einzelnen Dateinamen extra einen neuen Prozess mit dem `-exec`-Kommando zu starten. In diesem Fall hilft oft das Kommando `xargs`, das so viele Dateinamen wie möglich sammelt, bevor es damit tatsächlich ein Kommando ausführt:

Tabelle 6.7: Logische Operatoren für `find`

Zeichen	Operator	Bedeutung
`!`	Nicht	die folgende Bedingung darf nicht zutreffen
`-a`	*and* (Und)	die Bedingungen links und rechts vom `-a` müssen zutreffen
`-o`	*or* (Oder)	von den Bedingungen links und rechts vom `-o` muss mindestens eine zutreffen

```
$ find . -atime +13 | xargs -r rm -i
```

`xargs` liest seine Standardeingabe bis zu einem bestimmten (konfigurierbaren) Maximum von Zeichen oder Zeilen und verwendet das Gelesene als Argumente für das angegebene Kommando (hier `rm`). Als Trennzeichen für die Argumente gelten dabei Leerzeichen (die mit Anführungszeichen oder »\« maskiert werden können) oder Zeilentrenner. Das Kommando wird nur so oft wie nötig aufgerufen, um die Eingabe zu verbrauchen. – Die Option »`-r`« von `xargs` sorgt dafür, dass das Kommando `rm` nur ausgeführt wird, wenn `find` wirklich einen Dateinamen schickt – ansonsten würde es zumindest einmal gestartet.

Die `find`/`xargs`-Kombination kommt bei eigenartigen Dateinamen in Schwierigkeiten, etwa solchen, die Leerzeichen oder gar Zeilentrenner enthalten, welche dann als Trennzeichen fehlinterpretiert werden. Die todsichere Abhilfe dagegen besteht darin, die `find`-Option »`-print0`« zu benutzen, die wie `-print` die Namen der gefundenen Dateien ausgibt, diese aber nicht durch Zeilentrenner, sondern durch Nullbytes voneinander trennt. Da das Nullbyte in Dateinamen nicht auftauchen kann, ist keine Verwechslung mehr möglich. `xargs` muss mit der Option »`-0`« aufgerufen werden, um diese Form der Eingabe zu verstehen:

```
$ find . -atime +13 -print0 | xargs -0r rm -i
```

Übungen

6.25 [!2] Finden Sie alle Dateien in Ihrem System, die größer als 1 MiB sind, und lassen Sie deren Namen ausgeben.

6.26 [2] Wie können Sie `find` benutzen, um eine Datei zu löschen, die einen merkwürdigen Namen hat (etwa mit unsichtbaren Kontrollzeichen oder mit Umlauten, die von älteren Shells nicht verstanden werden)?

6.27 [3] (Beim zweiten Durcharbeiten.) Wie würden Sie beim Abmelden dafür sorgen, dass etwaige Dateien in `/tmp`, die Ihnen gehören, automatisch gelöscht werden?

6.4.5 Dateien schnell finden – `locate` und `slocate`

Das Kommando `find` erlaubt die Suche nach Dateien gemäß einer Vielzahl von Kriterien, muss aber die komplette Dateihierarchie unterhalb des Startverzeichnisses ablaufen. Je nach deren Umfang kann es also durchaus eine Weile dauern, bis die Suche abgeschlossen ist. Für einen typischen Anwendungsfall – die Suche nach Dateien mit einem bestimmten Namen – gibt es darum ein beschleunigtes Verfahren.

Das Kommando `locate` gibt alle Dateien aus, deren Namen auf ein bestimmtes Shell-Suchmuster passen. Im einfachsten Fall ist das eine simple Zeichenkette:

```
$ locate brief.txt
/home/hugo/Briefe/brief.txt
/home/hugo/Briefe/omabrief.txt
/home/hugo/Briefe/omabrief.txt~
◁◁◁◁◁
```

Obwohl es sich bei `locate` um einen ziemlich grundlegenden Dienst handelt (wie die Tatsache unterstreicht, dass das Programm zum LPIC-1-Stoff gehört), gehört es nicht bei allen Linux-Systemen zur Standardinstallation.


Wenn Sie zum Beispiel eine SUSE-Distribution verwenden, müssen Sie das Paket `findutils-locate` explizit nachinstallieren, bevor Sie `locate` benutzen können.

Die Sonderzeichen »*«, »?« und »[…]« funktionieren bei `locate` wie bei Suchmustern in der Shell. Während eine Anfrage *ohne* Suchmuster-Sonderzeichen jedoch alle Namen liefert, in denen der Suchbegriff irgendwo auftaucht, gibt eine Anfrage *mit* Suchmuster-Sonderzeichen nur diejenigen Namen aus, die *komplett* – von Anfang bis Ende – vom Suchbegriff beschrieben werden. Aus diesem Grund beginnen Suchmuster-Anfragen mit `locate` meistens mit »*«:

```
$ locate "*/brief.t*"
/home/hugo/Briefe/brief.txt
/home/hugo/Briefe/brief.tab
<<<<<
```


Am besten stellen Sie `locate`-Anfragen mit Sonderzeichen in Anführungszeichen, damit die Shell sie nicht zu expandieren versucht.

Der Schrägstrich (»/«) erfährt keine Sonderbehandlung:

```
$ locate Briefe/oma
/home/hugo/Briefe/omabrief.txt
/home/hugo/Briefe/omabrief.txt~
```

`locate` ist so schnell, weil es nicht das Dateisystem absucht, sondern in einer »Datenbank« von Dateinamen nachschaut, die irgendwann vorher mit dem Programm `updatedb` aufgebaut wurde. Das bedeutet natürlich, dass es Dateien nicht erwischt, die seit der letzten Aktualisierung der Datenbank dazu gekommen sind, und umgekehrt die Namen von Dateien liefern kann, die seitdem gelöscht wurden.


Mit der Option »`-e`« können Sie `locate` dazu bringen, nur tatsächlich existierende Dateinamen zu liefern, aber den Geschwindigkeitsvorteil des Programms machen Sie damit zunichte.

Das Programm `updatedb` baut die Datenbank für `locate` auf. Da dieser Vorgang einige Zeit dauern kann, sorgt Ihr Systemverwalter meist dafür, dass das passiert, wenn das System sonst nicht viel zu tun hat, auf ständig eingeschalteten Serversystemen zum Beispiel nachts.


Der dafür nötige Systemdienst `cron` wird in der Unterlage *Linux für Fortgeschrittene* ausführlich besprochen. Für jetzt mag genügen, dass die meisten Linux-Distributionen einen Mechanismus mitliefern, der dafür sorgt, dass `updatedb` regelmäßig aufgerufen wird.

Als Systemverwalter können Sie konfigurieren, welche Verzeichnisse `updatedb` beim Aufstellen der Datenbank beachtet. Wie das im Detail passiert, ist distributionsabhängig: `updatedb` selbst liest keine Konfigurationsdatei, sondern übernimmt seine Konfigurationseinstellungen über Kommandozeilenargumente oder (zum Teil) Umgebungsvariable. Allerdings rufen die meisten Distributionen `updatedb` über ein Shellskript auf, das vorher eine Datei wie `/etc/updatedb.conf` oder `/etc/sysconfig/locate` einliest, in der zum Beispiel geeignete Umgebungsvariable gesetzt werden können.


Dieses Shellskript finden Sie zum Beispiel in `/etc/cron.daily` (Details sind distributionsabhängig).

Sie können `updatedb` zum Beispiel mitteilen, welche Verzeichnisse es durchsuchen und welche es auslassen soll; das Programm erlaubt auch die Definiton von »Netz-Dateisystemen«, die von verschiedenen Rechnern verwendet werden und in deren Wurzelverzeichnissen Datenbanken geführt werden sollen, damit nicht jeder Rechner dafür seine eigene Datenbank aufbauen muss.

Eine wichtige Konfigurationseinstellung ist die des Benutzers, mit dessen Identität `updatedb` läuft. Dafür gibt es im wesentlichen zwei Möglichkeiten:

- `updatedb` läuft mit den Rechten von `root` und kann so jede Datei in die Datenbank aufnehmen. Das heißt aber auch, dass Benutzer Dateinamen in Verzeichnissen ausspähen können, für die sie eigentlich gar keine Zugriffsrechte haben, zum Beispiel die Heimatverzeichnisse anderer Benutzer.
- `updatedb` läuft mit eingeschränkten Rechten, etwa denen des Benutzers `nobody`. In diesem Fall landen nur diejenigen Dateinamen in der Datenbank, die in Verzeichnissen stehen, deren Inhalt `nobody` auflisten darf.

Das Programm `slocate` – eine Alternative zum gewöhnlichen `locate` – umgeht dieses Problem, indem es außer dem Namen einer Datei auch noch Eigentümer, Gruppenzugehörigkeit und Zugriffsrechte in der Datenbank speichert und einen Dateinamen nur dann ausgibt, wenn der Benutzer, der `slocate` aufgerufen hat, tatsächlich Zugriff auf die betreffende Datei hat. Auch `slocate` verfügt über ein `updatedb`-Programm, das allerdings nur ein anderer Name für `slocate` selber ist.

In vielen Fällen ist `slocate` so installiert, dass es auch unter dem Namen `locate` aufgerufen werden kann.

Übungen

6.28 [!1] `README` ist ein sehr populärer Dateiname. Geben Sie die absoluten Pfadnamen aller Dateien auf Ihrem System an, die `README` heißen.

6.29 [2] Legen Sie eine neue Datei in Ihrem Heimatverzeichnis an und überzeugen Sie sich durch einen `locate`-Aufruf, dass diese Datei nicht gefunden wird (wählen Sie gegebenenfalls einen hinreichend ausgefallenen Dateinamen). Rufen Sie dann (mit Administratorrechten) das Programm `updatedb` auf. Wird Ihre neue Datei danach mit `locate` gefunden? Löschen Sie die Datei wieder und wiederholen Sie die vorigen Schritte.

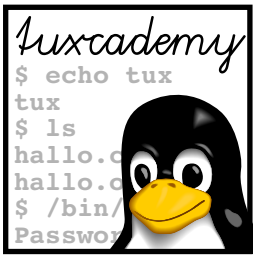
6.30 [1] Überzeugen Sie sich, dass `slocate` funktioniert, indem Sie als gewöhnlicher Benutzer nach Dateien wie `/etc/shadow` suchen.

Kommandos in diesem Kapitel

cd	Wechselt das aktuelle Arbeitsverzeichnis der Shell	bash(1)	71
convmv	Konvertiert Dateinamen zwischen Zeichenkodierungen	convmv(1)	69
cp	Kopiert Dateien	cp(1)	78
find	Sucht nach Dateien, die bestimmte Kriterien erfüllen	find(1), Info: find	86
less	Zeigt Texte (etwa Handbuchseiten) seitenweise an	less(1)	86
ln	Stellt („harte“ oder symbolische) Links her	ln(1)	81
locate	Sucht Dateien über ihren Namen in einer Dateinamensdatenbank	locate(1)	90
ls	Listet Dateien oder den Inhalt von Verzeichnissen auf	ls(1)	72
mkdir	Legt neue Verzeichnisse an	mkdir(1)	74
more	Zeigt Textdaten seitenweise an	more(1)	86
mv	Verschiebt Dateien in andere Verzeichnisse oder benennt sie um	mv(1)	80
pwd	Gibt den Namen des aktuellen Arbeitsverzeichnisses aus	pwd(1), bash(1)	72
rm	Löscht Dateien oder Verzeichnisse	rm(1)	80
rmdir	Entfernt (leere) Verzeichnisse	rmdir(1)	74
slocate	Sucht Dateien über ihren Namen in einer Datenbank und beachtet dabei deren Zugriffsrechte	slocate(1)	92
updatedb	Erstellt die Dateinamensdatenbank für locate	updatedb(1)	91
xargs	Konstruiert Kommandozeilen aus seiner Standardeingabe	xargs(1), Info: find	89

Zusammenfassung

- In Dateinamen dürfen fast alle möglichen Zeichen auftauchen. Im Interesse der Portabilität sollte man sich aber auf Buchstaben, Ziffern und einige Sonderzeichen beschränken.
- Linux unterscheidet Groß- und Kleinschreibung in Dateinamen.
- Absolute Pfade beginnen immer mit einem Schrägstrich und benennen alle Verzeichnisse vom Wurzelverzeichnis des Verzeichnisbaums bis zum betreffenden Verzeichnis bzw. der Datei. Relative Pfade beziehen sich auf das »aktuelle Verzeichnis«.
- Mit cd können Sie das aktuelle Verzeichnis der Shell ändern, mit pwd können Sie seinen Namen anzeigen.
- ls gibt Informationen über Dateien und Verzeichnisse aus.
- Mit mkdir und rmdir können Sie Verzeichnisse anlegen oder entfernen.
- Die Kommandos cp, mv und rm kopieren, verschieben und löschen Dateien und Verzeichnisse.
- Mit ln können Sie »harte« und »symbolische« Links anlegen.
- more und less dienen zum seitenweisen Anzeigen von Dateien auf dem Terminal.
- find sucht nach Dateien oder Verzeichnissen, die bestimmte Kriterien erfüllen.



7

Standardkanäle und Filterkommandos

Inhalt

7.1 Ein-/Ausgabeumlenkung und Kommandopipelines 96
7.1.1 Die Standardkanäle 96
7.1.2 Standardkanäle umleiten 97
7.1.3 Kommando-Pipelines 101
7.2 Filterkommandos 102
7.3 Dateien lesen und ausgeben 103
7.3.1 Textdateien ausgeben und aneinanderhängen – `cat` und `tac` . . 103
7.3.2 Anfang und Ende von Dateien – `head` und `tail` 105
7.3.3 Mit der Lupe – `od` und `hexdump` 106
7.4 Textbearbeitung 109
7.4.1 Zeichen für Zeichen – `tr`, `expand` und `unexpand` 109
7.4.2 Zeile für Zeile – `fmt`, `pr` und so weiter 112
7.5 Datenverwaltung 117
7.5.1 Sortierte Dateien – `sort` und `uniq` 117
7.5.2 Spalten und Felder – `cut`, `paste` & Co. 122

Lernziele

- Die Ein- und Ausgabeumlenkung der Shell beherrschen
- Die wichtigsten Filterkommandos kennen

Vorkenntnisse

- Arbeit mit der Shell (Kapitel 2)
- Umgang mit einem Texteditor (Kapitel 5)
- Umgang mit Dateien und Verzeichnissen (Kapitel 6)

Tabelle 7.1: Standardkanäle unter Linux

Kanal	Bezeichnung	Kürzel	Gerät	Zweck
0	Standard-Eingabe	`stdin`	Tastatur	Programme erhalten Eingaben
1	Standard-Ausgabe	`stdout`	Bildschirm	Programme senden Ausgaben
2	Standard-Fehlerausgabe	`stderr`	Bildschirm	Programme senden Fehlermeldungen

7.1 Ein-/Ausgabeumlenkung und Kommandopipelines

7.1.1 Die Standardkanäle

Viele Linux-Kommandos – beispielsweise `grep` & Co. – sind so gebaut, dass sie Eingabedaten lesen, diese irgendwie manipulieren und das Ergebnis dieser Manipulationen wieder ausgeben. Wenn Sie zum Beispiel einfach

```
$ grep xyz
```

eingeben, dann können Sie anschließend Textzeilen auf der Tastatur tippen, und `grep` wird nur diejenigen durchlassen, die die Zeichenfolge »xyz« enthalten:

```
$ grep xyz
abc def
xyz 123
xyz 123
aaa bbb
YYYxyzZZZ
YYYxyzZZZ
Strg+d
```

(Die Tastenkombination am Schluss läßt `grep` wissen, dass die Eingabe zu Ende ist.)

Standard-Eingabe
Standard-Ausgabe
Standard-Fehlerausgabe

Man sagt, `grep` liest Daten von der »Standard-Eingabe« – hier der Tastatur – und schreibt sie auf die »Standard-Ausgabe« – hier den Konsolenbildschirm oder, wahrscheinlicher, ein Terminalprogramm in einer grafischen Arbeitsumgebung. Als dritten dieser »Standardkanäle« gibt es noch die »Standard-Fehlerausgabe«; während auf die Standard-Ausgabe die »Nutzdaten« geschrieben werden, die `grep` produziert, landen auf dieser allfällige Fehlermeldungen (etwa weil eine Eingabedatei nicht existiert oder der reguläre Ausdruck einen Syntaxfehler hat).

In diesem Kapitel werden Sie lernen, wie Sie die Standard-Ausgabe eines Programms zum Beispiel in eine Datei umlenken oder die Standard-Eingabe einer Datei entnehmen können. Noch wichtiger werden Sie lernen, wie Sie die Ausgabe eines Programms direkt (ohne Umweg über eine Datei) als Eingabe an ein anderes Programm verfüttern können. Dies öffnet Ihnen die Tür dazu, die für sich genommen alle recht simplen Linux-Kommandos im Baukastenprinzip zu Anwendungen zu verketten, die sehr komplexe Dinge tun können.

Ganz erschöpfend werden wir dieses Thema in diesem Kapitel nicht behandeln können. Freuen Sie sich auf die Linup-Front-Schulungsunterlage *Linux für Fortgeschrittene*, wo die Erstellung von Shellskripten mit den Kommandos des Linux-Baukastens eine sehr wichtige Rolle spielt! Aber hier lernen Sie die sehr wichtigen Grundlagen dafür, schon auf der Kommandozeile Linux-Kommandos geschickt zu kombinieren.

Standardkanäle

Die **Standardkanäle** werden in Bild 7.1 noch einmal zusammengefasst. Sie werden im Jargon meist nur mit den englischen Kürzeln benannt – `stdin`, `stdout` und `stderr` für die Standard-Eingabe, Standard-Ausgabe und Standard-Fehlerausgabe.

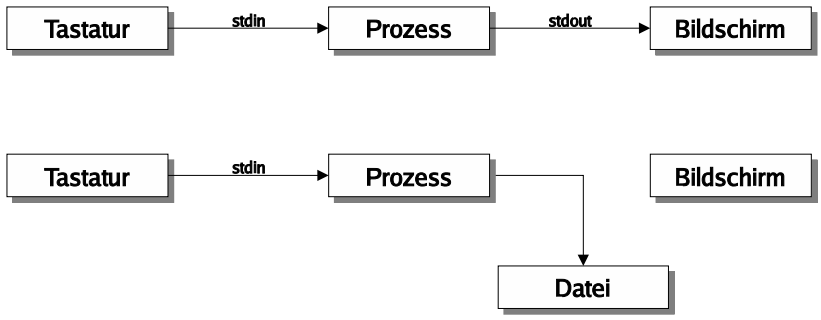


Bild 7.1: Standardkanäle unter Linux

Diesen Kanälen sind respektive auch die Nummern 0, 1 und 2 zugeordnet, was wir gleich noch brauchen werden.

Standardkanäle umleiten

Die Shell kann für einzelne Kommandos diese Standardkanäle auf andere Ziele umleiten, ohne dass die betroffenen Programme davon etwas mitbekommen. Diese benutzen immer die Standardkanäle, lediglich gelangen etwa die Ausgabedaten gegebenenfalls nicht mehr auf den Bildschirm bzw. ins Terminal-Fenster, sondern in eine beliebige andere Datei. Diese kann ein anderes Gerät sein, etwa der Drucker – es ist aber auch möglich, zum Beispiel eine Textdatei anzugeben, in der die Ausgabedaten abgelegt werden. Diese muss nicht einmal vorher existieren, sondern wird bei Bedarf neu erzeugt.

Auf die gleiche Art und Weise können Sie auch den Standard-Eingabe-Kanal umleiten. Ein Programm erhält seine Informationen dann nicht von der Tastatur, sondern entnimmt sie der angegebenen Datei, die wiederum für ein Gerät stehen oder eine Datei im eigentlichen Sinne sein kann.

Tastatur und Bildschirm des »Terminals«, an dem Sie arbeiten (egal ob die Textkonsole eines Linux-Rechners, ein »echtes« seriell angeschlossenes Terminal, ein Terminalfenster in einer grafischen Umgebung oder eine Sitzung über das Netz etwa mit der Secure Shell), können Sie über die »Datei« `/dev/tty` ansprechen – wenn Sie Daten lesen wollen, meint dies die Tastatur, bei der Ausgabe den Bildschirm (umgekehrt wäre ziemlich albern). Der Aufruf

```
$ grep xyz /dev/tty
```

wäre zum Beispiel äquivalent zu unserem Beispiel weiter oben in diesem Abschnitt. Mehr über solche »besonderen Dateien« erzählt Kapitel 9.)

7.1.2 Standardkanäle umleiten

Standard-Ausgabe umleiten

Den Standard-Ausgabe-Kanal können Sie mit dem Operator »>«, also dem »Größer-Als«-Zeichen, umleiten. So wird im folgenden Beispiel die Ausgabe von »`ls -laF`« in eine Datei namens `inhalt` umgelenkt; auf dem Bildschirm erscheint dabei lediglich

```
$ ls -laF >inhalt
$ _
```

Falls die Datei `inhalt` noch nicht existiert, wird sie neu angelegt. Sollte hingegen bereits eine Datei dieses Namens vorhanden sein, wird deren Inhalt überschrieben. Das ganze arrangiert die Shell, noch bevor das gewünschte Programm überhaupt gestartet wird – die Ausgabedatei wird also auch dann angelegt, wenn der eigentliche Programmaufruf falsch eingetippt wurde oder das Programm überhaupt keine Ausgabe liefert (die Datei `inhalt` ist dann anschließend leer).

Existierende Dateien schützen

Wenn Sie verhindern wollen, dass die Shell-Ausgabeumlenkung schon existierende Dateien leert, können Sie in der Bash das Kommando »`set -o noclobber`« geben. In diesem Fall bleibt eine schon existierende Datei, die Ziel einer Ausgabeumlenkung ist, unverändert. Statt dessen erscheint eine Fehlermeldung.

Die Textdatei `inhalt` können Sie nun wie üblich anschauen, z. B. mit `less`:

```
$ less inhalt
total 7
drwxr-xr-x  12 hugo   users     1024 Aug 26 18:55 ./
drwxr-xr-x   5 root   root      1024 Aug 13 12:52 ../
drwxr-xr-x   3 hugo   users     1024 Aug 20 12:30 fotos/
-rw-r--r--   1 hugo   users        0 Sep  6 13:50 inhalt
-rw-r--r--   1 hugo   users    15811 Aug 13 12:33 pingu.gif
-rw-r--r--   1 hugo   users    14373 Aug 13 12:33 hobby.txt
-rw-r--r--   2 hugo   users     3316 Aug 20 15:14 chemie.txt
```

Wenn Sie den Inhalt von `inhalt` genau betrachten, sehen Sie einen Verzeichniseintrag für `inhalt` mit der Dateigröße 0. Das liegt an der Arbeitsweise der Shell: Bei der Bearbeitung der Kommandozeile wird zunächst die Ausgabeumlenkung erkannt und eine neue Datei `inhalt` angelegt bzw. deren Inhalt gelöscht. Danach führt die Shell das Kommando, hier `ls`, aus, wobei sie die Standardausgabe von `ls` mit der Datei `inhalt` statt dem Bildschirm verbindet.

Die Datei hat in der `ls`-Ausgabe die Länge 0, weil das `ls`-Kommando die Dateiinformationen für `inhalt` abgerufen hat, bevor tatsächlich etwas in die Datei geschrieben wurde – obwohl vor dem betreffenden Eintrag eigentlich drei andere Zeilen stehen! Das liegt daran, dass `ls` erst sämtliche Verzeichniseinträge liest, sie alphabetisch sortiert und erst dann die Ausgabe zu schreiben beginnt. `ls` sieht also die von der Shell neu angelegte oder gerade geleerte Datei `inhalt` ohne Inhalt.

Standardausgabe an Datei anhängen

Wenn Sie die Ausgabe eines Programms ans Ende einer bestehenden Datei anhängen wollen, ohne dass deren bisheriger Inhalt ersetzt wird, können Sie den Operator `>>` benutzen. Wenn diese Datei noch nicht existiert, wird sie auch hier neu angelegt:

```
$ date >> inhalt
$ less inhalt
total 7
drwxr-xr-x  12 hugo   users     1024 Aug 26 18:55 ./
drwxr-xr-x   5 root   root      1024 Aug 13 12:52 ../
drwxr-xr-x   3 hugo   users     1024 Aug 20 12:30 fotos/
-rw-r--r--   1 hugo   users        0 Sep  6 13:50 inhalt
-rw-r--r--   1 hugo   users    15811 Aug 13 12:33 pingu.gif
-rw-r--r--   1 hugo   users    14373 Aug 13 12:33 hobby.txt
-rw-r--r--   2 hugo   users     3316 Aug 20 15:14 chemie.txt
Wed Oct 22 12:31:29 CEST 2003
```

Im Beispiel wurde das aktuelle Datum ans Ende der Datei `inhalt` angefügt.

Kommandosubstitution

Eine andere Möglichkeit zur Umleitung der Standardausgabe eines Programms bieten die »verkehrten« Anführungszeichen `` `…` ``. Man spricht auch von **Kommandosubstitution**: Die Standardausgabe eines Kommandos, das in verkehrten Anführungszeichen steht, wird anstelle des Kommandoaufrufs in die Befehlszeile eingebaut; ausgeführt wird dann das, was sich durch diese Ersetzung ergibt. Zum Beispiel:

```
$ cat termine                                    Unser kleiner Terminkalender
22.12. Geschenke besorgen
```

```
23.12. Christbaum besorgen
24.12. Heiligabend
$ date +%d.%m.                                    Welches Datum haben wir?
23.12.
$ grep ^`date +%d.%m.` termine                    Was liegt an?
23.12. Christbaum besorgen
```

Eine unter Umständen bequemere, aber nur von modernen Shells wie der Bash unterstützte Syntax für »`date`« ist »$(date)«. Damit ist es einfacher möglich, solche Aufrufe zu verschachteln.

Standard-Eingabe umleiten

Mit <, dem »Kleiner-Als«-Zeichen, können Sie den Standard-Eingabe-Kanal umleiten. Nun wird statt der Tastatureingabe der Inhalt der angegebenen Datei ausgewertet:

```
$ wc -w <frosch.txt
1255
```

Im Beispiel zählt das Filterkommando wc die in der Datei frosch.txt vorkommenden Wörter.

Einen <<-Umleitungsoperator zur Verkettung mehrerer Eingabedateien gibt es nicht; um den Inhalt mehrerer Dateien als Eingabe an ein Kommando zu leiten, müssen Sie cat benutzen:

```
$ cat datei1 datei2 datei3 | wc -w
```

(Zum »|«-Operator steht mehr im nächsten Abschnitt.) Die meisten Programme akzeptieren allerdings einen oder mehrere Dateinamen als Argumente auf der Kommandozeile.

Mit dem <<-Operator können Sie allerdings Eingabedaten für ein Programm direkt aus den Zeilen übernehmen, die dem Programmaufruf in der Shell folgen. Dies ist für den interaktiven Einsatz weniger interessant als für den Gebrauch in Shellskripten, soll aber der Vollständigkeit halber trotzdem hier erwähnt werden. Man spricht von »Hier-Dokumenten« (engl. *here documents*). Im Beispiel

```
$ grep Linux <<ENDE
Rosen sind rot,
Veilchen sind blau,
Linux ist super,
das weiß ich genau.
ENDE
```

besteht die Eingabe von grep aus den Folgezeilen bis zu der Zeile mit dem Inhalt »ENDE«. Die Ausgabe des Kommandos ist

```
Linux ist super,
```

Wird bei einem Hier-Dokument die Zeichenkette, die das Ende des Hier-Dokuments bezeichnet, ohne Anführungszeichen angegeben, so werden in den Zeilen des Hier-Dokuments Shellvariable ersetzt und Kommandosubstitution (mit `…` oder $(…)) vorgenommen. Steht die End-Zeichenkette allerdings in Anführungszeichen (egal ob einfachen oder doppelten), bleiben die Zeilen des Hier-Dokuments so, wie sie sind. Vergleichen Sie die Ausgabe von

```
$ cat <<EOF
Heutiges Datum: `date`
EOF
```

mit der von

```
$ cat <<"EOF"
Heutiges Datum: `date`
EOF
```

Zu guter Letzt: Wird das Hier-Dokument nicht mit »<<<«, sondern mit »<<-« eingeleitet, so werden alle Tabulatorzeichen am Anfang jeder Zeile des Hier-Dokuments entfernt. Dies macht es möglich, Hier-Dokumente in Shellskripten sinnvoll einzurücken.

kombinierte Umleitung

Selbstverständlich ist auch die kombinierte Umleitung von Ein- und Ausgabekanal zulässig. Die Ausgabe des Wortzähl-Beispiels wird hier in eine Datei namens `wortzahl` geschrieben:

```
$ wc -w <frosch.txt >wortzahl
$ cat wortzahl
1255
```

Standard-Fehler-Kanal

Neben den Standard-Eingabe- und -Ausgabe-Kanälen existiert noch der Standard-Fehler-Kanal. Sollte ein Programm während seiner Arbeit auf Probleme stoßen, werden die zugehörigen Fehlermeldungen auf diesem Kanal ausgegeben. So bekommen Sie sie zu sehen, auch wenn die Standardausgabe in eine Datei umgeleitet wird. Wenn Sie auch die Standardfehlerausgabe in eine Datei umleiten wollen, müssen Sie beim Umleitungsoperator die Kanalnummer angeben – bei `stdin` (`0<`) und `stdout` (`1>`) ist diese optional, für `stderr` (`2>`) jedoch zwingend erforderlich.

Mit dem Operator `>&` können Sie einen Kanal in einen anderen umleiten:

```
make >make.log 2>&1
```

leitet die Standard-Ausgabe *und* die Standard-Fehlerausgabe des Kommandos `make` in die Datei `make.log`.

Passen Sie auf: Hier kommt es auf die Reihenfolge an! Die beiden Kommandos

```
make >make.log 2>&1
make 2>&1 >make.log
```

führen zu sehr unterschiedlichen Resultaten. Im zweiten Fall wird erst die Standard-Fehlerausgabe dorthin geleitet, wohin die Standard-Ausgabe geht (`/dev/tty`, wo sie sowieso schon hingeht) und anschließend die Standard-Ausgabe in die Datei `make.log` geschickt, was an dem Ziel der Standard-Fehlerausgabe aber nichts mehr ändert.

Übungen

7.1 [2] Mit der Option `-U` können Sie `ls` dazu bringen, die Verzeichniseinträge unsortiert auszugeben. Trotzdem hat nach dem Kommando »`ls -laU >inhalt`« der Eintrag für `inhalt` in der Ausgabedatei immer noch die Länge Null. Woran könnte das liegen?

7.2 [!2] Vergleichen Sie die Ausgabe der Kommandos »`ls /tmp`« und »`ls /tmp >ls-tmp.txt`« (wobei wir mit »Ausgabe« im zweiten Fall den Inhalt der Datei `ls-tmp.txt` meinen). Fällt Ihnen etwas auf? Falls ja, wie könnte man das Phänomen erklären?

7.3 [!2] Warum ist es nicht möglich, eine Datei in einem Schritt durch eine neue Version zu ersetzen, etwa mit »`grep xyz datei >datei`«?

7.4 [!1] Und was ist das Problem mit »`cat bla >>bla`« (eine nichtleere Datei `bla` vorausgesetzt)?

7.5 [2] Wie würden Sie in der Shell eine Meldung so ausgeben, dass sie auf der Standard-Fehlerausgabe landet?

7.1.3 Kommando-Pipelines

Oft dient die Ausgabeumleitung nur dazu, das Ergebnis eines Programms abzuspeichern, um es dann mit einem anderen Kommando weiterzubearbeiten. Diese Art von Zwischenspeicherung ist jedoch nicht nur recht mühsam, sondern Sie müssen auch noch daran denken, die nicht mehr benötigten Dateien wieder zu löschen. Linux bietet daher eine direkte Verknüpfung von Kommandos durch **Pipes** an: Die Ausgabe eines Programms wird automatisch zur Eingabe des nächsten Programms.

Pipes

Die direkte Verknüpfung mehrerer Befehle zu einer solchen Kommando-***Pipeline*** (»Rohrleitung«) erfolgt mit dem Zeichen `|`. Statt also wie im ersten Beispiel dieses Kapitels zunächst den Verzeichnisinhalt mit dem Kommando »`ls -laF`« in eine Datei `inhalt` umzulenken und diese dann mit `less` anzusehen, können Sie diesen Vorgang auch in einem Schritt ohne Zwischenspeicherung in einer Datei ausführen:

direkte Verknüpfung mehrerer Befehle

Pipeline

```
$ ls -laF | less
total 7
drwxr-xr-x  12 hugo   users     1024 Aug 26 18:55 ./
drwxr-xr-x   5 root   root      1024 Aug 13 12:52 ../
drwxr-xr-x   3 hugo   users     1024 Aug 20 12:30 fotos/
-rw-r--r--   1 hugo   users      449 Sep  6 13:50 inhalt
-rw-r--r--   1 hugo   users    15811 Aug 13 12:33 pingu.gif
-rw-r--r--   1 hugo   users    14373 Aug 13 12:33 hobby.txt
-rw-r--r--   2 hugo   users     3316 Aug 20 15:14 chemie.txt
```

Derartige Befehlsfolgen können praktisch beliebig lang werden, auch ist eine abschließende Ausgabeumleitung in eine Datei möglich:

```
$ cut -d: -f1 /etc/passwd | sort | pr -2 >userlst
```

Diese Kommando-Pipeline entnimmt zunächst der Systemdatei `/etc/passwd` alle Benutzernamen (die in der ersten durch »:« getrennten Spalte stehen), sortiert diese alphabetisch und schreibt sie dann zweispaltig in die Datei `userlst`. Die hier benutzten Kommandos werden übrigens im Rest dieses Kapitels genauer beschrieben.

Manchmal ist es sinnvoll, den Datenstrom innerhalb einer Kommando-Pipeline an einer bestimmten Stelle abzuspeichern, etwa weil das dortige Zwischenergebnis auch für andere Arbeiten von Interesse ist. Der Befehl `tee` führt eine Verzweigung des Datenstroms herbei, indem dieser verdoppelt und je ein Strom in eine Datei sowie an das nächste Kommando geleitet wird. Der Name dieses Kommandos (lautmalerisch für den Buchstaben »T«) lässt sich leicht aus Bild 7.2 herleiten – oder Sie denken an ein »T-Stück« in der »Pipeline« …

Zwischenergebnis abspeichern

Die Anweisung `tee` ohne Parameter legt die gewünschte Datei neu an oder überschreibt eine vorhandene; mit `-a` (engl. *append*, »anhängen«) lässt sich die Ausgabe an das Ende einer bestehenden Datei anfügen.

```
$ ls -laF | tee liste | less
total 7
drwxr-xr-x  12 hugo   users     1024 Aug 26 18:55 ./
```

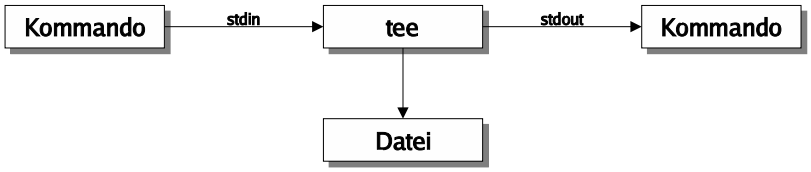


Bild 7.2: Das Kommando `tee`

```
drwxr-xr-x   5 root   root      1024 Aug 13 12:52 ../
drwxr-xr-x   3 hugo   users     1024 Aug 20 12:30 fotos/
-rw-r--r--   1 hugo   users      449 Sep  6 13:50 inhalt
-rw-r--r--   1 hugo   users    15811 Aug 13 12:33 pingu.gif
-rw-r--r--   1 hugo   users    14373 Aug 13 12:33 hobby.txt
-rw-r--r--   2 hugo   users     3316 Aug 20 15:14 chemie.txt
```

In diesem Beispiel wird der Inhalt des aktuellen Verzeichnisses sowohl in die Datei `liste` geschrieben als auch auf dem Bildschirm ausgegeben. (Die Datei `liste` ist noch nicht in der Ausgabe von `ls` zu sehen, da sie erst später von `tee` angelegt wird – ein Unterschied zur Ausgabeumlenkung der Shell.)

Übungen

7.6 [!2] Wie würden Sie dasselbe Zwischenergebnis gleichzeitig in mehrere Dateien schreiben?

7.2 Filterkommandos

Werkzeugkastenprinzip

Eine der Grundlagen von Unix – und damit Linux – ist das »Werkzeugkastenprinzip«. Das System verfügt über eine große Menge von Systemprogrammen, die jeweils eine (konzeptuell) simple Aufgabe erledigen. Diese Programme können dann von anderen Programmen als »Bausteine« verwendet werden und ersparen es den Autoren jener Programme, die entsprechende Funktionalität selbst zu entwickeln. So hat z. B. nicht jedes Programm eine eigene Sortierfunktion, sondern viele Programme greifen auf das Kommando `sort` zurück. Dieser modulare Aufbau hat mehrere Vorteile:

- Eine Vereinfachung für die Programmierer, die nicht ständig neue Sortierroutinen schreiben oder zumindest in ihre Programme einbauen müssen.
- Bei einer Fehlerkorrektur oder Optimierung von `sort` profitieren alle Programme, die darauf zugreifen, ebenfalls, und das ohne explizite Anpassung (meistens jedenfalls).

Filterkommandos

Programme, die ihre Eingabe von der Standard-Eingabe lesen und ihre Ausgabe auf die Standard-Ausgabe schreiben, heißen **Filterkommandos** oder kurz »Filter«. Ohne Eingabeumleitung lesen Filter also von der Tastatur. Zum Beenden der Standard-Eingabe eines solchen Kommandos müssen Sie die Tastenkombination Strg + d eingeben, die der Terminaltreiber als »Datei-Ende« interpretiert.

Tabelle 7.2: Optionen für `cat` (Auswahl)

Option	Wirkung
`-b`	(engl. *number non-blank lines*) Numeriert alle nichtleeren Zeilen der Ausgabe, beginnend mit 1.
`-E`	(engl. *end-of-line*) Zeigt am Ende jeder Zeile der Ausgabe ein `$` an (gut zum Aufspüren von ansonsten nicht sichtbaren Leerzeichen).
`-n`	(engl. *number*) Numeriert alle Zeilen der Ausgabe, beginnend mit 1.
`-s`	(engl. *squeeze*) Ersetzt Folgen von Leerzeilen durch je eine.
`-T`	(engl. *tabs*) Stellt Tabulatorzeichen als »`^I`« dar.
`-v`	(engl. *visible*) Macht Steuerzeichen c als »`^`c« und Zeichen α mit Zeichencodes größer als 127 als »`M-`α« sichtbar.
`-A`	(engl. *show all*) Äquivalent zu `-vET`.

Das gilt wohlgemerkt *nur* für die Eingabe von Inhalten über die Tastatur. Textdateien auf der Platte dürfen natürlich das Strg+d-Zeichen (ASCII 4) enthalten, ohne dass das System glaubt, die Datei wäre an dieser Stelle zu Ende. Dies im Gegensatz zu einem gewissen anderen sehr populären Betriebssystem, das traditionell etwas eigenwillige Vorstellungen von der Bedeutung des Zeichens Strg+z (ASCII 26) hat, selbst wenn es in einer Textdatei steht …

Auch viele »gewöhnliche« Kommandos, zum Beispiel das schon gezeigte `grep`, verhalten sich wie Filter, wenn Sie keine Dateinamen zur Bearbeitung angeben.

Eine Auswahl der wichtigsten dieser Befehle besprechen wir im Rest des Kapitels. Einige haben sich dabei eingeschlichen, die keine waschechten Filter-Kommandos sind, aber für alle gilt, dass sie wichtige Bausteine von Pipelines bilden.

7.3 Dateien lesen und ausgeben

7.3.1 Textdateien ausgeben und aneinanderhängen – `cat` und `tac`

Dateien verketten

Der Befehl `cat` (engl. *concatenate*, »verketten«) dient eigentlich dazu, mehrere auf der Kommandozeile benannte Dateien zu einer einzigen zusammenzufügen. Übergeben Sie jedoch nur einen Dateinamen als Argument, wird der Inhalt der betreffenden Datei auf der Standardausgabe ausgegeben. Wenn Sie überhaupt keinen Dateinamen übergeben, liest `cat` seine Standardeingabe – dies scheint nutzlos, aber `cat` bietet über Optionen die Möglichkeit, die gelesene Eingabe mit Zeilennummern zu verbrämen, Zeilenenden und Sonderzeichen sichtbar zu machen oder Folgen von Leerzeilen zu einer zu komprimieren (Tabelle 7.2).

Textdateien

Es versteht sich, dass mit `cat` nur Textdateien eine vernünftige Bildschirmausgabe liefern. Wenn Sie das Kommando auf andere Dateitypen wie etwa die Binärdatei `/bin/cat` anwenden, ist es zumindest auf einem Textterminal sehr wahrscheinlich, dass nach Ende der Ausgabe die Eingabeaufforderung aus unleserlichen Zeichenkombinationen besteht. In diesem Fall können Sie durch (blinde) Eingabe von `reset` den Bildschirmzeichensatz wiederherstellen. Beim Umlenken der `cat`-Ausgabe in eine Datei ist das natürlich kein Problem.

Der *useless use of `cat` award* (Preis für den überflüssigen Gebrauch von `cat`) wird Leuten verliehen, die `cat` benutzen, wo es überhaupt nicht nötig ist. In den meisten Fällen akzeptieren Kommandos Dateinamen und lesen nicht nur ihre Standardeingabe, so dass `cat` nicht erforderlich ist, nur um ihnen eine einzige Datei auf der Standardeingabe zu verfüttern. Ein Aufruf wie »`cat`

Tabelle 7.3: Optionen für `tac` (Auswahl)

Option	Wirkung
`-b`	(engl. *before*) Die Eingabe wird so interpretiert, dass der Trenner jeweils *vor* einem Teil steht (und ausgegeben wird), nicht dahinter.
`-r`	(engl. *regular expression*) Der Trenner wird als regulärer Ausdruck interpretiert.
`-s` *s*	(engl. *separator*) Gibt einen anderen Trenner *s* (statt `\n`) an. Der Trenner darf mehrere Zeichen lang sein.

`data.txt | grep bla`« ist unnötig, wenn man genausogut »`grep bla data.txt`« schreiben kann. Selbst wenn `grep` nur seine Standardeingabe lesen könnte, wäre »`grep bla <data.txt`« kürzer und würde keinen zusätzlichen `cat`-Prozess involvieren. Die ganze Angelegenheit ist aber etwas subtiler; siehe hierzu Übung 7.21.

Zeilen in umgekehrter Reihenfolge ausgeben

Das Kommando `tac` ist vom Namen her »`cat` rückwärts« und funktioniert auch so: Es liest als Parameter benannte Dateien oder seine Standardeingabe und gibt die gelesenen Zeilen in *umgekehrter* Reihenfolge wieder aus:

```
$ tac <<ENDE
Alpha
Beta
Gamma
ENDE
Gamma
Beta
Alpha
```

Trenner

Da hört die Ähnlichkeit aber auch schon auf: `tac` unterstützt nicht dieselben Optionen wie `cat`, sondern hat ein paar eigene (Tabelle 7.3). Zum Beispiel können Sie mit der Option `-s` einen alternativen Trenner einstellen, an dem `tac` sich beim Umdrehen der Eingabe orientiert – normal ist das Trennzeichen der Zeilentrenner, so dass zeilenweise umgedreht wird. Siehe etwa:

```
$ echo A:B:C:D | tac -s :
D
C:B:A:$ _
```

(mit der neuen Eingabeaufforderung direkt an die letzte Zeile geklebt). Diese auf den ersten Blick total eigenartige Ausgabe erklärt sich wie folgt: Die Eingabe besteht aus den vier Teilen »`A:`«, »`B:`«, »`C:`« und »`D\n`« (der Trenner, hier »`:`«, gilt als an den jeweils vorhergehenden Teil angehängt, und der Zeilentrenner `\n` wird von `echo` beigesteuert). Diese Teile werden in umgekehrter Reihenfolge ausgegeben, das heißt, zuerst kommt »`D\n`« und dann die anderen drei, jeweils ohne irgendein zusätzliches Trennzeichen dazwischen (es ist ja schon ein völlig zufriedenstellender Trenner vorhanden); die nächste Eingabeaufforderung der Shell wird direkt (ohne Schaltung in die nächste Zeile) an die Ausgabe angehängt. Die Option `-b` betrachtet den Trenner nicht als an den vorhergehenden Teil angehängt, sondern als vor den nachfolgenden Teil gestellt; unser Beispiel würde unter »`tac -s : -b`« also die Ausgabe

```
:D
:C:BA$ _
```

liefern (denken Sie's durch!).

Übungen

7.7 [2] Wie können Sie prüfen, ob in einem Verzeichnis Dateien mit »merkwürdigen« Namen enthalten sind, etwa solche mit Leerzeichen am Schluss oder mit unsichtbaren Steuerzeichen in der Mitte?

7.3.2 Anfang und Ende von Dateien – `head` und `tail`

Mitunter interessiert Sie nur ein Teil einer Datei: Die ersten paar Zeilen, um zu sehen, ob es die richtige Datei ist, oder vor allem bei einer Protokolldatei die letzten Einträge. Die Kommandos `head` und `tail` liefern genau das – standardmäßig respektive die ersten oder die letzten 10 Zeilen jeder Datei, deren Name sie als Argument übergeben bekommen (ersatzweise wie üblich die ersten oder letzten 10 Zeilen der Standard-Eingabe). Die Option `-n` erlaubt es, eine andere Anzahl von Zeilen auszugeben: »`head -n 20`« liefert die ersten 20 Zeilen der Standard-Eingabe, »`tail -n 5 daten.txt`« die letzten 5 Zeilen der Datei `daten.txt`.

Traditionell können Sie die Anzahl *n* der gewünschten Zeilen auch direkt in der Form »*-n*« angeben. Offiziell ist das nicht mehr erlaubt, aber die Linux-Versionen von `head` und `tail` unterstützen es noch.

Mit der Option `-c` können Sie angeben, dass nicht Zeilen, sondern Bytes gezählt werden sollen: »`head -c 20`« zeigt die ersten 20 Bytes der Standard-Eingabe, egal wie viele Zeilen das sind. Wenn Sie an die Zahl ein »`b`«, »`k`« oder »`m`« (»Blöcke«, »Kibibyte«, »Mebibyte«) anhängen, wird sie mit 512, 1024 bzw. 1048576 multipliziert.

`head` erlaubt ferner den Einsatz eines Minuszeichens: »`head -c -20`« zeigt die ganze Standard-Eingabe *bis auf* die letzten 20 Bytes.

Aus Fairnessgründen kann `tail` auch etwas, was `head` nicht kann: Wenn die Zeilenzahl mit »+« anfängt, zeigt es alles *ab* der betreffenden Zeile:

```
$ tail -n +3 datei                                   Alles ab Zeile 3
```

Das Programm `tail` unterstützt außerdem die wichtige Option `-f`. Sie führt dazu, dass `tail` nach der Ausgabe des aktuellen Dateiendes wartet und später hinzukommende Daten am Dateiende auch noch ausgibt. Dies ist sehr nützlich zur Beobachtung von Protokolldateien. Wenn Sie `tail -f` mehrere Dateinamen übergeben, schreibt es vor jeder neuen Ausgabe eine Kopfzeile, die angibt, in welcher Datei jetzt wieder neue Daten angekommen sind.

Übungen

7.8 [!2] Wie würden Sie gezielt nur die 13. Zeile der Eingabe ausgeben?

7.9 [3] Probieren Sie »`tail -f`« aus: Legen Sie eine Datei an und rufen Sie »`tail -f`« für diese Datei auf. Hängen Sie dann in einem anderen Fenster bzw. auf einer anderen virtuellen Konsole z. B. mit »`echo >>…`« etwas an die Datei an und beobachten Sie die `tail`-Ausgabe. Wie sieht das ganze aus, wenn `tail` mehrere Dateien gleichzeitig beobachtet?

7.10 [3] Was passiert mit »`tail -f`«, wenn die beobachtete Datei kürzer wird?

7.11 [3] Erklären Sie die Ausgabe der folgenden Kommandos

```
$ echo Hallo >/tmp/hallo
$ echo "Huhu Welt" >/tmp/hallo
```

Tabelle 7.4: Optionen für od (Auszug)

Option	Wirkung
`-A` *r*	Basis der Positionsangabe am Zeilenanfang. Gültige Werte sind: `d` (dezimal), `o` (oktal), `x` (hexadezimal), `n` (keine Positionsangabe ausgeben).
`-j` *o*	Am Anfang der Eingabe werden *o* Bytes übersprungen und erst dann mit der Ausgabe begonnen.
`-N` *n*	Maximal *n* Bytes ausgeben.
`-t` *t*	Ausgabe gemäß Typangabe *t*. Es können mehrere `-t`-Optionen auftreten, dann wird für jede der `-t`-Optionen eine Zeile im entsprechenden Format ausgegeben. Möglichkeiten für *t*: `a` (benanntes Zeichen), `c` (ASCII-Zeichen), `d` (Dezimalzahl mit Vorzeichen), `f` (Gleitkommazahl), `o` (Oktalzahl), `u` (vorzeichenlose Dezimalzahl), `x` (Hexadezimalzahl). An die Optionen außer `a` und `c` können Sie eine Ziffer anhängen, die angibt, wie viele Bytes der Eingabe jeweils gemeinsam interpretiert werden sollen. Details hierfür und für Breitenangaben durch Buchstaben finden sich in od(1). Hängen Sie an eine Option ein `z` an, werden rechts auf der Zeile die druckbaren Zeichen der Zeile ausgegeben.
`-v`	Gibt auch doppelt vorkommende Zeilen komplett aus.
`-w` *w*	Gibt *w* Bytes pro Zeile aus; Standardwert ist 16.

wenn Sie nach dem ersten `echo` in einem anderen Fenster das Kommando

```
$ tail -f /tmp/hallo
```

gestartet haben.

7.3.3 Mit der Lupe – od und hexdump

`cat`, `tac`, `head` und `tail` funktionieren am besten mit Textdateien: Beliebige Binärdaten können zwar prinzipiell verarbeitet werden, aber gerade die letzteren drei Programme gehen standardmäßig am liebsten mit Dateien um, die erkennbare Zeilen haben. Trotzdem ist es ab und zu nützlich, sich vergewissern zu können, welche Daten nun ganz genau in einer Datei stehen. Ein Hilfsmittel dafür ist das

od

Kommando `od` (kurz für *octal dump*), das beliebige Daten in unterschiedlichen Formaten anzeigen kann. Binärdaten lassen sich zum Beispiel byteweise oder wortweise in oktaler, hexadezimaler, dezimaler oder ASCII-Darstellung präsentieren. Die Standard-Darstellungsweise von `od` ist wie folgt:

```
$ od /etc/passwd | head -3
0000000 067562 072157 074072 030072 030072 071072 067557 035164
0000020 071057 067557 035164 061057 067151 061057 071541 005150
0000040 060563 064163 067562 072157 074072 030072 030072 071072
$ _
```

Zeilenformat

Ganz links steht die (oktale) Position in der Datei, an der die Ausgabezeile beginnt. Die acht folgenden Zahlen entsprechen jeweils zwei Bytes aus der Datei, als Oktalzahl interpretiert. Nützlich ist das nur unter ganz speziellen Umständen.

Zum Glück unterstützt `od` Optionen, mit denen Sie das Format der Ausgabe in

Option `-t`

sehr weiten Grenzen ändern können (Tabelle 7.4). Am wichtigsten ist die Option `-t`, die angibt, wie die `od`-Datenzeilen aussehen sollen. Für byteweise hexadezimale Ausgabe können Sie zum Beispiel

```
$ od -txC /etc/passwd
0000000 72 6f 6f 74 3a 78 3a 30 3a 30 3a 72 6f 6f 74 3a
0000020 2f 72 6f 6f 74 3a 2f 62 69 6e 2f 62 61 73 68 0a
```

```
0000040 73 61 73 68 72 6f 6f 74 3a 78 3a 30 3a 30 3a 72
◁◁◁◁◁
```

verwenden (die Positionsangabe bleibt oktal). Dabei steht das `x` für »hexadezimal« und das `C` für »byteweise«. Wenn Sie außer den Hexadezimalzahlen noch die entsprechenden Zeichen sehen wollen, können Sie ein `z` anhängen:

```
$ od -txCz /etc/passwd
0000000 72 6f 6f 74 3a 78 3a 30 3a 30 3a 72 6f 6f 74 3a  >root:x:0:0:root:<
0000020 2f 72 6f 6f 74 3a 2f 62 69 6e 2f 62 61 73 68 0a  >/root:/bin/bash.<
0000040 73 61 73 68 72 6f 6f 74 3a 78 3a 30 3a 30 3a 72  >sashroot:x:0:0:r<
◁◁◁◁◁
```

Nicht druckbare Zeichen (hier das `0a` – ein Zeilentrenner – am Ende der zweiten Zeile) werden durch ».« vertreten.

Sie können auch mehrere Typangaben hintereinandergehängt oder in separaten `-t`-Optionen angeben. Dann bekommen Sie eine Zeile pro Typangabe: mehrere Typangaben

```
$ od -txCc /etc/passwd
0000000 72 6f 6f 74 3a 78 3a 30 3a 30 3a 72 6f 6f 74 3a
          r   o   o   t   :   x   :   0   :   0   :   r   o   o   t   :
0000020 2f 72 6f 6f 74 3a 2f 62 69 6e 2f 62 61 73 68 0a
          /   r   o   o   t   :   /   b   i   n   /   b   a   s   h  \n
0000040 73 61 73 68 72 6f 6f 74 3a 78 3a 30 3a 30 3a 72
          s   a   s   h   r   o   o   t   :   x   :   0   :   0   :   r
◁◁◁◁◁
```

(was dasselbe wäre wie »`od -txC -tc /etc/passwd`«).

Eine Folge von Zeilen, die in der Ausgabe identisch zu der unmittelbar davorstehenden Zeile wären, wird durch einen Stern (»*«) am linken Rand abgekürzt: identische Ausgabezeilen

```
$ od -tx -N 64 /dev/zero
0000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
*
0000100
```

(`/dev/zero` liefert Nullen in beliebigen Mengen, und die `od`-Option `-N` beschränkt die Ausgabe auf 64 davon.) Mit der Option `-v` findet diese Abkürzung nicht statt:

```
$ od -tx -N 64 -v /dev/zero
0000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000100
```

Eine ganz ähnliche Leistung erbringt das Programm `hexdump` (oder kurz `hd`). Zunächst unterstützt es Ausgabeformate, die denen von `od` sehr ähneln, auch wenn die dafür nötigen Optionen völlig anders heißen. So entspricht das Kommando `hexdump`

```
$ hexdump -o -s 446 -n 64 /etc/passwd
```

im Wesentlichen dem ersten Beispiel für `od` weiter oben. Die meisten Optionen von `od` haben ziemlich direkte Äquivalente in `hexdump`.

Ein wesentlicher Unterschied zwischen `hexdump` und `od` ist `hexdumps` Unterstützung für Ausgabeformate. Hiermit können Sie wesentlich detaillierter festlegen, wie die Ausgabe aussehen soll, als mit `od`. Betrachten Sie das folgende Beispiel: Ausgabeformate

```
$ cat hexdump.txt
0123456789ABC◁◁◁◁◁ XYZabc◁◁◁◁◁ xyz
$ hexdump -e '"%x-« hexdump.txt
33323130-37363534-◁◁◁◁◁-a7a79-$
```

Die folgenden Punkte sind bemerkenswert:

- Das Ausgabeformat »`"%x"`« gibt 4 Bytes der Eingabedatei in hexadezimaler Form aus – »`30`« ist die hexadezimale Darstellung von 48, dem numerischen Wert des Zeichens »`0`« im ASCII. Das angehängte »`-`« wird so ausgegeben, wie es ist.
- Die 4 Bytes werden in umgekehrter Reihenfolge ausgegeben, das ist ein Artefakt der Intel-Prozessorarchitektur.
- Die doppelten Anführungszeichen sind Bestandteil der Syntax von `hexdump` und müssen darum durch einfache Anführungszeichen (oder etwas Äquivalentes) davor geschützt werden, dass die Shell sie entfernt.
- Das `$` am Schluss ist die nächste Eingabeaufforderung der Shell; von sich aus gibt `hexdump` keinen Zeilentrenner aus.

Die möglichen Ausgabeformate richten sich (bequemerweise für Programmierer) nach denen, die von der Funktion `printf`(3) in Programmiersprachen wie C, Perl, `awk` usw. verwendet werden (auch die Bash unterstützt ein `printf`-Kommando). Entnehmen Sie die Details der Dokumentation!

`hexdump`-Ausgabeformate sind deutlich vielschichtiger als das eben gezeigte simple Beispiel. Wie bei `printf` üblich können Sie etwa eine »Feldbreite« für die Ausgabe bestimmen:

```
$ hexdump -e '"%10x-"' hexdump.txt
  33323130-  37363534-  ◁◁◁◁◁ a7a79-$
```

(Hier erscheint jede Sequenz von Hexadezimalzahlen – acht Zeichen lang – rechtsbündig in einem zehn Zeichen breiten Feld.)

Wiederholungszahl

Sie können auch angeben, wie oft ein Format »ausgeführt« wird:

```
$ hexdump -e '4 "%x-" "\n"' hexdump.txt
33323130-37363534-42413938-46454443-
4a494847-4e4d4c4b-5251504f-56555453-
5a595857-64636261-68676665-6c6b6a69-
706f6e6d-74737271-78777675-a7a79-
```

Die Kommandos dieses Abschnitts vorangestellte `4` gibt an, dass das Format »`"%x"-`« viermal angewendet werden soll. Danach geht es mit dem nächsten Format – »`"\n"`« – weiter, das einen Zeilentrenner erzeugt. Anschließend fängt `hexdump` wieder vorne an.

Zeilentrenner

Byteanzahl

Ferner können Sie sagen, wie viele Bytes ein Format verarbeiten soll (bei den numerischen Formaten haben Sie meist die Wahl zwischen 1, 2 und 4):

```
$ hexdump -e '/2 "%x-" "\n"' hexdump.txt
3130-
3332-
◁◁◁◁◁
7a79-
a-
```

(»`/2`« ist hier eine Abkürzung für »`1/2`«, darum erscheint jedes `%x`-Format pro Zeile nur einmal.) Wiederholungs- und Byteanzahl lassen sich kombinieren:

Tabelle 7.5: Optionen für `tr`

Option	Wirkung
`-c` (*complement*)	ersetzt alle Zeichen, die *nicht* in $\langle s_1 \rangle$ stehen, durch Zeichen aus $\langle s_2 \rangle$
`-d` (*delete*)	entfernt alle Zeichen, die in $\langle s_1 \rangle$ stehen, ohne Ersatz
`-s` (*squeeze*)	mehrere gleiche, aufeinander folgende Zeichen aus $\langle s_2 \rangle$ werden zusammengefasst

```
$ hexdump -e '4/2 "%x-" "\n"' hexdump.txt
3130-3332-3534-3736-
3938-4241-4443-4645-
<<<<<
7675-7877-7a79-a-
```

Sie können auch verschiedene Ausgabeformate mischen:

```
$ hexdump -e '"%2_ad" "%2.2s" 3/2 " %x" " %1.1s" "\n"'▷
 ◁ hexdump.txt
0 01 3332 3534 3736 8
9 9A 4342 4544 4746 H
<<<<<
```

Hier geben wir die ersten beiden Zeichen der Datei als Zeichen (nicht als numerische Codes) aus (»`"%2.2s"`«), dann die Codes von dreimal zwei Zeichen in hexadezimaler Schreibweise (»`3/2 " %x"`«), gefolgt von einem weiteren Zeichen als Zeichen (»`"%1.1s"`«) und einem Zeilentrenner. Danach geht es mit dem Format wieder von vorne los. Das »`"%2_ad"`« am Zeilenanfang liefert die aktuelle Position in der Datei (gezählt in Bytes vom Dateianfang) in dezimaler Schreibweise in einem zwei Zeichen breiten Feld.

Übungen

7.12 [2] Was ist der Unterschied zwischen den Typangaben »`a`« und »`c`« von `od`?

7.13 [3] Das »Gerät« `/dev/random` liefert zufällige Bytes (siehe Abschnitt 9.3). Verwenden Sie `od` mit `/dev/random`, um der Shellvariablen `r` eine dezimale Zufallszahl zwischen 0 und 65535 (einschließlich) zuzuweisen.

7.4 Textbearbeitung

7.4.1 Zeichen für Zeichen – `tr`, `expand` und `unexpand`

Der Befehl `tr` dient dazu, innerhalb eines Textes einzelne Zeichen gegen andere auszutauschen oder komplett zu löschen. `tr` ist ein reines Filterkommando, es kann also nicht auf benannte Dateien angewendet werden, sondern arbeitet ausschließlich mit den Standardkanälen.

Austauschoperationen

Für Austauschoperationen lautet die Syntax dieses Kommandos »`tr` $\langle s_1 \rangle$ $\langle s_2 \rangle$«. Die beiden Parameter sind Zeichenketten, die die Austauschoperation beschreiben: Im einfachsten Fall wird das erste Zeichen von $\langle s_1 \rangle$ durch das erste Zeichen von $\langle s_2 \rangle$ ersetzt, das zweite Zeichen von $\langle s_1 \rangle$ durch das zweite Zeichen von $\langle s_2 \rangle$ und so weiter. Ist $\langle s_1 \rangle$ länger als $\langle s_2 \rangle$, so wird für die »überstehenden« Zeichen in $\langle s_1 \rangle$ das letzte Zeichen von $\langle s_2 \rangle$ benutzt; ist $\langle s_2 \rangle$ länger als $\langle s_1 \rangle$, werden die zusätzlichen Zeichen in $\langle s_2 \rangle$ ignoriert.

Ein kleines Beispiel zur Illustration:

Tabelle 7.6: Zeichen und Zeichenklassen für `tr`

Klasse	Bedeutung
`\a`	Control-G (ASCII 7), Glockenton
`\b`	Control-H (ASCII 8), »*backspace*«
`\f`	Control-L (ASCII 12), »*formfeed*«
`\n`	Control-J (ASCII 10), »*linefeed*«
`\r`	Control-M (ASCII 13), »*carriage return*«
`\t`	Control-I (ASCII 9), Tabulatorzeichen
`\v`	Control-K (ASCII 11), vertikaler Tabulator
`\`*kkk*	das durch die drei Oktalziffern *kkk* benannte Zeichen
`\\`	ein Rückstrich
`[`*c*`*`*n*`]`	in $\langle s_2 \rangle$: das Zeichen *c* *n*-mal
`[`*c*`*]`	in $\langle s_2 \rangle$: das Zeichen *c* so oft, dass $\langle s_2 \rangle$ so lang ist wie $\langle s_1 \rangle$
`[:alnum:]`	alle Buchstaben und Ziffern
`[:alpha:]`	alle Buchstaben
`[:blank:]`	aller horizontaler Freiplatz
`[:cntrl:]`	alle Kontrollzeichen
`[:digit:]`	alle Ziffern
`[:graph:]`	alle druckbaren Zeichen (ohne Leerzeichen)
`[:lower:]`	alle Kleinbuchstaben
`[:print:]`	alle druckbaren Zeichen (inklusive Leerzeichen)
`[:punct:]`	alle Satzzeichen
`[:space:]`	aller horizontaler oder vertikaler Freiplatz
`[:upper:]`	alle Großbuchstaben
`[:xdigit:]`	alle Hexadezimalziffern (`0`–`9`, `A`–`F`, `a`–`f`)
`[:`*c*`:]`	alle Zeichen, die äquivalent zu *c* sind (im Moment nur *c* selbst)

```
$ tr AEiü aey <bsp.txt >neu1.txt
```

Mit diesem Kommando werden in der Datei `bsp.txt` alle »A« durch »a«, alle »E« durch »e« sowie alle »i« bzw. »ü« durch »y« ersetzt und das Resultat als `neu1.txt` gespeichert.

Bereiche

Es ist erlaubt, Folgen von Zeichen durch Bereiche der Form »*m*-*n*« auszudrücken. Dabei muss *m* in der Sortierreihenfolge vor *n* stehen. Mit der Option `-c` wird

Komplement

nicht der Inhalt von $\langle s_1 \rangle$ ersetzt, sondern dessen »Komplement«, also alle Zeichen, die *nicht* in $\langle s_1 \rangle$ enthalten sind. Das Kommando

```
$ tr -c A-Za-z ' ' <bsp.txt >neu1.txt
```

ersetzt alle Nichtbuchstaben in `bsp.txt` durch Leerzeichen.

Zeichenklassen

Es ist auch möglich, Zeichenklassen der Form `[:`*k*`:]` zu benutzen (die möglichen Klassennamen ergeben sich aus Tabelle 7.6); in vielen Fällen ist das sinnvoll, um Befehle zu konstruieren, die auch in anderen Sprachumgebungen funktionieren. In einer deutschsprachigen Umgebung enthält die Zeichenklasse »`[:alpha:]`« zum Beispiel auch die Umlaute, eine »handgestrickte« Klasse der Form »`A-Za-z`«, die für Englisch stimmt, dagegen nicht. Für Zeichenklassen gelten noch einige weitere Einschränkungen, die Sie am besten der Dokumentation für `tr` entnehmen (siehe »`info tr`«).

Löschen von Zeichen

Zum Löschen von Zeichen müssen Sie nur $\langle s_1 \rangle$ angeben: Das Kommando

```
$ tr -d a-z <bsp.txt >neu2.txt
```

löscht alle Kleinbuchstaben aus der Datei `bsp.txt`. Außerdem können Sie Folgen gleicher Zeichen in der Eingabe durch ein einziges Zeichen in der Ausgabe ersetzen: Das Kommando

```
$ tr -s '\n' <bsp.txt >neu3.txt
```

entfernt Leerzeilen, indem Folgen von Zeilenendezeichen durch jeweils eins ersetzt werden.

»*squeeze*«

Mit der Option `-s` (»*squeeze*«) ist es ferner möglich, beim Ersetzen zwei verschiedene Ursprungszeichen durch zwei gleiche zu ersetzen und diese dann (wie bei `-s` mit nur einem Argument) zu einem zusammenzufassen. Aus allen »A«, »E« sowie Folgen dieser Zeichen in der Datei `bsp.txt` wird ein »Ä« in `neu3.txt`:

```
$ tr -s AE Ä <bsp.txt >neu3.txt
```

Tabulator

Der »Tabulator« – bekannt von der guten alten Schreibmaschine – ist eine bequeme Möglichkeit, beim Programmieren oder allgemein bei der Eingabe von Texten Einrückungen zu realisieren. Nach Konvention sind in verschiedenen Spalten (normalerweise alle 8 Spalten, also in den Positionen 8, 16, 24, …) »Tabulatorstopps« gesetzt, und die gängigen Editoren rücken beim Druck auf die Tab-Taste nach rechts an den nächsten Tabulatorstopp – wenn Sie die Tab-Taste drücken, während die Eingabemarke in Spalte 11 auf dem Bildschirm steht, zum Beispiel an Spalte 16. Trotzdem werden die aus dem Druck auf Tab resultierenden »Tabulatorzeichen« als solche in die Datei geschrieben, und manche Programme können sie nicht richtig interpretieren. Das Kommando `expand` hilft hier: Es liest die durch die übergebenen Parameter bezeichneten Dateien (ersatzweise – Sie wissen schon – seine Standard-Eingabe) und gibt sie auf seiner Standard-Ausgabe aus, wobei alle Tabulatorzeichen durch die richtige Anzahl von Leerzeichen ersetzt werden, damit die Tabulatorstopps alle 8 Zeichen erhalten bleiben. Mit der Option `-t` können Sie einen anderen »Multiplikator« für die Tabulatorstopps angeben; gängig ist zum Beispiel »`-t 4`«, was die Stopps in die Spalten 4, 8, 12, 16, … setzt.

Tabulatorzeichen expandieren

Wenn Sie bei `-t` mehrere durch Komma getrennte Zahlen angeben, werden Tabulatorstopps genau auf die benannten Spalten gesetzt: »`expand -t 4,12,32`« setzt Tabulatorstopps in die Spalten 4, 12 und 32. Weitere Tabulatorzeichen in einer Zeile werden durch einzelne Leerzeichen ersetzt.

Die Option `-i` (engl. *initial*) bewirkt, dass nur Tabulatorzeichen am Zeilenanfang zu Leerzeichen expandiert werden.

Leerzeichen durch Tabulatorzeichen ersetzen

Das Kommando `unexpand` bewirkt im Großen und Ganzen das Gegenteil von `expand`: Alle Folgen von Tabulator- und Leerzeichen am Anfang von Zeilen in der Eingabe (wie üblich benannte Dateien oder die Standard-Eingabe) werden durch die minimale Folge von Tabulator- und Leerzeichen ersetzt, die die gewünschte Einrückung realisieren. Eine Zeile mit einem Tabulatorzeichen, zwei Leerzeichen, einem weiteren Tabulatorzeichen und neun Leerzeichen am Anfang zum Beispiel wird – Standard-Tabulatorstopps in jeder achten Spalte vorausgesetzt – durch eine Zeile ersetzt, die mit drei Tabulatorzeichen und einem Leerzeichen anfängt. Die Option `-a` (engl. *all*) bewirkt, dass *alle* Folgen von zwei oder mehr Tabulator- oder Leerzeichen »optimiert« werden und nicht nur die am Zeilenanfang.

Übungen

7.14 [!2] Der berühmte römische Feldherr Julius Caesar verwendete angeblich für die Übermittlung geheimer Botschaften das folgende Verschlüsselungsverfahren: Der Buchstabe »A« wurde ersetzt durch »D«, »B« durch »E« und so weiter; »X« entsprechend durch »A«, »Y« durch »B« und »Z« durch »C« (wenn wir mal die heute üblichen 26 Buchstaben zugrundelegen und den Umstand ignorieren, dass die alten Römer kein J, kein K, kein W und

kein Y hatten). Stellen Sie Sich vor, Sie sind ein Programmierer in Caesars Legion. Wie lauten die `tr`-Aufrufe, um eine Botschaft des Feldherrn zu verschlüsseln und wieder zu entschlüsseln?

7.15 [3] Mit welchem `tr`-Kommando können Sie alle Vokale in einem Text durch einen gegebenen einzigen ersetzen? Denken Sie an das Kinderlied:

```
DREI CHINESEN MIT DEM KONTRABASS
DRAA CHANASAN MAT DAM KANTRABASS
```

7.16 [3] Wie würden Sie eine Textdatei so umformen, dass alle Satzzeichen entfernt und jedes Wort auf eine einzelne Zeile gestellt wird?

7.17 [2] Geben Sie ein `tr`-Kommando an, mit dem die Zeichen »`a`«, »`z`«, und »`-`« aus der Standardeingabe entfernt werden können.

7.18 [1] Wie würden Sie sich davon überzeugen, dass `unexpand` tatsächlich Leer- und Tabulatorzeichen durch eine »optimale« Folge ersetzt?

7.4.2 Zeile für Zeile – `fmt`, `pr` und so weiter

Während die Kommandos aus dem vorigen Abschnitt die Eingabezeichen einzeln oder in kleinen Gruppen betrachtet haben, gibt es in Linux auch viele Kommandos, die ganze Eingabezeilen bearbeiten. Einige davon stellen wir im folgenden vor.

Zeilenumbruch

Das Programm `fmt` umbricht die Zeilen der Eingabe (wie gewöhnlich den auf der Kommandozeile benannten Dateien oder der Standard-Eingabe entnommen) so, dass sie maximal eine bestimmte vorgegebene Länge haben – 75 Zeichen, wenn nicht mit der Option `-w` anders festgelegt. Dabei bemüht es sich durchaus um gut aussehende Ausgabe.

Betrachten wir einige Beispiele für `fmt` (die Datei `frosch0.txt` entspricht der Datei `frosch.txt`, bis darauf, dass die erste Zeile in jedem Absatz um zwei Zeichen eingerückt ist):

```
$ head frosch0.txt
Der Froschkönig oder der eiserne Heinrich

  In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
schön, daß die Sonne selber, die doch so vieles schon gesehen hat,
sich verwunderte, sooft sie ihr ins Gesicht schien.

  Nahe bei dem Schlosse war ein großer, dunkler Wald, und mitten darin,
unter einer alten Linde, war ein Brunnen. Wenn nun der Tag recht heiß war,
ging die jüngste Prinzessin hinaus in den Wald und setzte sich an den Rand
```

Im ersten Beispiel reduzieren wir die Zeilenlänge auf 40 Zeichen:

```
$ fmt -w 40 frosch0.txt
Der Froschkönig oder der eiserne
Heinrich

  In alten Zeiten, als das Wünschen
  noch geholfen hat, lebte einmal ein
König, der hatte wunderschöne
Töchter. Die jüngste von ihnen war
so schön, daß die Sonne selber, die
doch so vieles schon gesehen hat,
sich verwunderte, sooft sie ihr ins
⊲⊲⊲⊲
```

Beachten Sie, dass die zweite Zeile des ersten Absatzes um zwei Leerzeichen eingerückt ist, ohne dass unmittelbar einleuchtet, warum. Die Lösung des Rätsels ist, dass `fmt` normalerweise nur Folgen von Zeilen mit derselben Einrückung als Absatz betrachtet, der für einen »gemeinsamen« Zeilenumbruch in Frage kommt. Die eingerückte erste Zeile des ersten Absatzes der Beispieldatei gilt also als eigener »Absatz«, wird für sich umbrochen und alle resultierenden Zeilen mit der Einrückung der ersten (und einzigen) Zeile des »Absatzes« in der Eingabe versehen. Die zweite und die folgenden Zeilen in der Eingabe gelten als unabhängiger weiterer »Absatz« und werden ebenfalls separat umbrochen (mit der Einrückung der zweiten Zeile).

`fmt` versucht Leerzeilen, Wortzwischenraum und Einrückung in der Eingabe beizubehalten. Zeilenumbrüche macht es am liebsten am Satzende und ungern nach dem ersten oder vor dem letzten Wort im Satz. Ein »Satzende« aus der Sicht von `fmt` ist entweder das Ende eines Absatzes oder ein Wort, das mit »`.`«, »`?`« oder »`!`«, gefolgt von zwei (!) Leerzeichen, aufhört.

Mehr Details über die Funktionsweise von `fmt` stehen in der Info-Seite des Programms (Tipp zum Finden: `fmt` ist Bestandteil der GNU-Coreutils).

Im nächsten Beispiel verwenden wir die Option `-c` (engl. *crown-margin mode*), um das gerade gezeigte Phänomen zu vermeiden:

```
$ fmt -c -w 40 frosch0.txt
Der Froschkönig oder der eiserne
Heinrich

  In alten Zeiten, als das Wünschen
noch geholfen hat, lebte einmal
ein König, der hatte wunderschöne
Töchter. Die jüngste von ihnen war
so schön, daß die Sonne selber, die
doch so vieles schon gesehen hat,
sich verwunderte, sooft sie ihr ins
⋖⋘⋘⋖
```

Hier orientiert die Einrückung des (kompletten) Absatzes sich an den ersten zwei Zeilen der Eingabe; die Einrückung dieser beiden Zeilen wird erhalten, und die weiteren Zeilen der Eingabe folgen der Einrückung der zweiten Zeile.

Zum Schluss noch ein Beispiel mit langen Zeilen:

```
$ fmt -w 100 frosch0.txt
Der Froschkönig oder der eiserne Heinrich

  In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so schön, daß die Sonne selber,
die doch so vieles schon gesehen hat, sich verwunderte, sooft sie ihr ins Gesicht schien.

  Nahe bei dem Schlosse war ein großer, dunkler Wald, und mitten darin,
unter einer alten Linde, war ein Brunnen. Wenn nun der Tag recht heiß war, ging die jüngste
⋖⋘⋘⋖
```

Hier hätten wir ebenfalls `-c` angeben können, um die »kurze« erste Zeile der Absätze zu vermeiden. Ohne diese Option gilt sie wieder als eigenständiger Absatz und wird nicht mit den Folgezeilen verschmolzen.

Der Name des Kommandos `pr` (engl. *print*, »drucken«) mag zunächst in die Irre führen. Es gibt Dateien nicht, wie vielleicht naheliegend, auf einem Drucker aus – hierzu dient `lpr`. Statt dessen ermöglicht `pr` die Formatierung eines Textes, der entweder aus einer Datei (bzw. Dateien) oder der Standard-Eingabe stammt, mit

Tabelle 7.7: Optionen von `pr`

Option	Wirkung
`-`*n*	ergibt eine Darstellung in *n* Spalten (zwar sind beliebige Werte erlaubt, aber nur Zahlen von 2 bis 5 sind normalerweise sinnvoll)
`-h` *t*	(engl. *header*) Gibt statt des Dateinamens den Text *t* in jedem Seitenkopf aus
`-l` *n*	(engl. *length*) Legt die Anzahl der Zeilen pro Seite fest, voreingestellt sind 66
`-n`	(engl. *number*) Versieht jede Zeile mit einer fünfstelligen Zeilennummer, die durch ein Tabulatorzeichen abgetrennt ist
`-o` *n*	(engl. *offset*) Rückt den Text *n* Zeichen vom linken Rand ein
`-t`	(engl. *omit*) unterdrückt die je fünf Kopf- und Fußzeilen
`-w` *n*	(engl. *width*) Legt die Anzahl von Zeichen pro Zeile fest, voreingestellt ist 72

Seitenumbrüchen, Einrückung und Kopf- und Fußzeilen. Auch hier sind natürlich umfangreiche Kontrollmöglichkeiten gegeben. Die wichtigsten Optionen für `pr` finden Sie in Tabelle 7.7.

Zur Veranschaulichung der Arbeitsweise von `pr` ein etwas komplexeres Beispiel:

```
$ fmt -w 34 frosch.txt | pr -h "Grimms Märchen" -2


2003-10-20 21:03                Grimms Märchen                  Seite 1


Der Froschkönig oder der eiserne    antwortete der Frosch, >>ich kann
Heinrich                            wohl Rat schaffen. Aber was gibst
                                    du mir, wenn ich dein Spielzeug
In alten Zeiten, als das Wünschen   wieder heraufhole?«
noch geholfen hat, lebte einmal
ein König, der hatte wunderschöne   >>Was du haben willst, lieber
Töchter. Die jüngste von ihnen war  Frosch«, sagte sie, >>meine
so schön, daß die Sonne selber,     Kleider, meine Perlen und
die doch so vieles schon gesehen    Edelsteine, auch noch die goldene
hat, sich verwunderte, sooft sie    Krone, die ich trage.«
ihr ins Gesicht schien.
◁◁◁◁◁
```

Hier verwenden wir `fmt`, um den Text des »Froschkönigs« in einer langen, schmalen Spalte zu formatieren, und `pr`, um den Text dann zweispaltig zu Papier (bzw. zu Bildschirm) zu bringen.

Zeilennumerierung

Auf Zeilennumerierung spezialisiert ist das Kommando `nl`. Wenn nichts anderes angegeben wurde, numeriert es die nichtleeren Zeilen der Eingabe (die wie gewöhnlich aus benannten Dateien oder von der Standard-Eingabe kommen darf) fortlaufend durch:

```
$ nl frosch.txt
     1  Der Froschkönig oder der eiserne Heinrich

     2  In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
     3  König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
     4  schön, daß die Sonne selber, die doch so vieles schon gesehen hat,
     5  sich verwunderte, sooft sie ihr ins Gesicht schien.

     6  Nahe bei dem Schlosse war ein großer, dunkler Wald, und mitten darin,
◁◁◁◁◁
```

Tabelle 7.8: Optionen für `nl` (Auswahl)

Option		Wirkung
`-b` *s*	(*body style*)	Numeriert die Zeilen des Körpers gemäß *s*. Mögliche Werte für *s* sind `a` (alle Zeilen numerieren), `t` (nur nichtleere Zeilen numerieren), `n` (gar keine Zeilen numerieren) und `p`⟨*Ausdruck*⟩ (nur die Zeilen numerieren, die auf den regulären Ausdruck ⟨*Ausdruck*⟩ passen). Standardwert ist `t`.
`-d` *p*[*q*]	(*delimiter*)	Verwendet die beiden Zeichen *pq* statt »`\:`« in Trennzeilen. Wurde nur *p* angegeben, so ist *q* weiterhin »`:`«.
`-f` *s*	(*footer style*)	Formatiert die Fußzeilen gemäß *s*. Die möglichen Werte für *s* entsprechen denen bei `-b`; Standardwert ist `n`.
`-h` *s*	(*header style*)	Analog zu `-f`, für Kopfzeilen.
`-i` *n*	(*increment*)	Erhöht die Zeilennummer pro Zeile um *n*.
`-n` *f*	(*number format*)	Bestimmt das Format für die Zeilennummern. Mögliche Werte für *f*: `ln` (linksbündig ohne führende Nullen), `rn` (rechtsbündig ohne führende Nullen), `rz` (rechtsbündig mit führenden Nullen).
`-p`	(*page*)	Setzt die Zeilennummer zwischen logischen Seiten *nicht* auf den Anfangswert zurück.
`-v` *n*		Beginnt die Numerierung bei Zeilennummer *n*.
`-w` *n*	(*width*)	Die Zeilennummer wird *n* Zeichen breit ausgegeben.

Das ist zunächst einmal nichts, was Sie nicht auch mit »`cat -b`« hinkriegen würden. Zum einen erlaubt `nl` aber die genauere Steuerung der Numerierung:

```
$ nl -b a -n rz -w 5 -v 1000 -i 10 frosch.txt
01000   Der Froschkönig oder der eiserne Heinrich
01010
01020   In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
01030   König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
01040   schön, daß die Sonne selber, die doch so vieles schon gesehen hat,
01050   sich verwunderte, sooft sie ihr ins Gesicht schien.
01060
01070   Nahe bei dem Schlosse war ein großer, dunkler Wald, und mitten darin,
01080   unter einer alten Linde, war ein Brunnen. Wenn nun der Tag recht heiß war,
01090   ging die jüngste Prinzessin hinaus in den Wald und setzte sich an den Rand
◁◁◁◁◁
```

Im Einzelnen stehen die Optionen für das Folgende (siehe auch Tabelle 7.8): »`-b a`« sorgt dafür, dass alle Zeilen numeriert werden und nicht – wie im vorigen Beispiel – nur die nichtleeren. »`-n rz`« formatiert die Zeilennummern rechtsbündig mit führenden Nullen, »`-w 5`« sorgt für eine Breite von fünf Spalten. »`-v 1000`« lässt die Numerierung bei 1000 beginnen, und »`-i 10`« erhöht die Zeilennummer pro Zeile um 10 (nicht, wie sonst, 1).

Zeilennummern pro Seite

Zum anderen kann `nl` auch mit einer logischen Seiteneinteilung der Eingabe umgehen. Dazu dienen die »magischen« Zeichenketten »`\:\:\:`«, »`\:\:`« und »`\:`« wie im folgenden Beispiel gezeigt:

```
$ cat nl-test
\:\:\:
Kopf der ersten Seite
\:\:
Erste Zeile der ersten Seite
Zweite Zeile der ersten Seite
Letzte Zeile der ersten Seite
\:
```

Tabelle 7.9: Optionen für `wc` (Auswahl)

Option	Wirkung
`-l` (*lines*)	gibt Anzahl der Zeilen aus
`-w` (*words*)	gibt Anzahl der Wörter aus
`-c` (*characters*)	gibt Anzahl der Zeichen aus

```
Fuß der ersten Seite
\:\:\:
Kopf der zweiten Seite
(Zwei Zeilen hoch)
\:\:
Erste Zeile der zweiten Seite
Zweite Zeile der zweiten Seite
Vorletzte Zeile der zweiten Seite
Letzte Zeile der zweiten Seite
\:
Fuß der zweiten Seite
(Zwei Zeilen hoch)
```

Kopf- und Fußbereiche

Jede (logische) Seite hat einen Kopf- und einen Fußbereich sowie einen »Körper« (engl. *body*), wo der eigentliche Text steht. Der Kopfbereich wird mit »`\:\:\:`« eingeleitet und mit »`\:\:`« vom Körper getrennt. Der Körper wiederum endet bei einer »`\:`«-Zeile. Kopf und Fuß können auch wegfallen.

In der Standardeinstellung numeriert `nl` die Zeilen jeder Seite beginnend bei 1; Kopf- und Fußzeilen werden nicht numeriert:

```
$ nl nl-test

       Kopf der ersten Seite

     1  Erste Zeile der ersten Seite
     2  Zweite Zeile der ersten Seite
     3  Letzte Zeile der ersten Seite

       Fuß der ersten Seite

       Kopf der zweiten Seite
       (Zwei Zeilen hoch)

     1  Erste Zeile der zweiten Seite
     2  Zweite Zeile der zweiten Seite
     3  Vorletzte Zeile der zweiten Seite
     4  Letzte Zeile der zweiten Seite

       Fuß der zweiten Seite
       (Zwei Zeilen hoch)
```

Die »`\:…`«-Trennzeilen werden in der Ausgabe jeweils durch eine Leerzeile ersetzt.

Zeilen, Wörter, Zeichen zählen

Der Name des Kommandos `wc` stellt die Abkürzung für *»word count«* (engl. für »Wörter zählen«) dar. Entgegen dieser Bezeichnung lässt sich nicht nur die Anzahl der Wörter, sondern auch der Zeichen überhaupt und der Zeilen der Eingabe (Dateien, Standard-Eingabe) ermitteln. Dies geschieht mit in Tabelle 7.9 aufgeführten Optionen. Ein »Wort« aus der Sicht von `wc` ist eine Folge von einem oder mehreren Buchstaben. Ohne Option werden alle drei Werte in der Reihenfolge aufgelistet, wie sie in Tabelle 7.9 erscheinen:

```
$ wc frosch.txt
 159 1255 7406 frosch.txt
```

Mit den Optionen in Tabelle 7.9 können Sie die Ausgabe von `wc` auf bestimmte der Werte beschränken:

```
$ ls | wc -l
     13
$ _
```

Das Beispiel zeigt, wie mit `wc` durch Zählen der Ausgabezeilen von `ls` die Anzahl der Einträge im aktuellen Verzeichnis bestimmt werden kann.

Übungen

7.19 [1] Numerieren Sie die Zeilen der Datei `frosch.txt` in Zweierschritten beginnend bei 100.

7.20 [3] Wie können Sie die Zeilen einer Datei in *umgekehrter* Reihenfolge numerieren, also etwa so:

```
159  Der Froschkönig oder der eiserne Heinrich
158
157  In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
156  König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
◁◁◁◁◁
```

(*Tipp:* Zweimal umgedreht gibt wieder das Original).

7.21 [!2] Wie unterscheidet sich die Ausgabe des Kommandos »`wc a.txt b.txt c.txt`« von der des Kommandos »`cat a.txt b.txt c.txt | wc`«?

7.5 Datenverwaltung

7.5.1 Sortierte Dateien – `sort` und `uniq`

Mit dem Kommando `sort` können Sie die Zeilen von Textdateien nach vorgegebenen Kriterien sortieren. Die Standardeinstellung ist eine aufsteigende Sortierung, also von A nach Z, anhand der ASCII-Werte[1] der ersten Zeichen in einer Zeile. Dies ist auch der Grund dafür, dass deutsche Sonderzeichen fehlerhaft einsortiert werden. Beispielsweise beträgt der ASCII-Code von »Ä« 143, dieser Buchstabe wird also weit hinter »Z« mit dem ASCII-Code 91 eingeordnet. Auch der Kleinbuchstabe »a« gilt als »größer« als der Großbuchstabe »Z«.

Standardeinstellung

Selbstverständlich kann `sort` sich auch nach deutschen Gepflogenheiten richten. Setzen Sie dazu eine der Umgebungsvariablen `LANG`, `LC_ALL` oder `LC_COLLATE` auf einen Wert wie »`de`«, »`de_DE`« oder »`de_DE.UTF-8`« (der genaue Wert hängt von Ihrer Distribution und Systemumgebung ab). Wenn Sie diese Einstellung nur für das `sort`-Kommando haben wollen, dann ist eine Angabe der Form

```
$ … | LC_COLLATE=de_DE.UTF-8 sort
```

[1] Bekanntlich geht der ASCII nur bis 127. Wirklich gemeint ist hier der ASCII zusammen mit der gerade aktuellen Erweiterung für die Zeichen mit den Codes ab 128, also zum Beispiel ISO-8859-1, auch bekannt als ISO-Latin-1.

möglich. Der Wert von `LC_ALL` hat Vorrang vor dem Wert von `LC_COLLATE` und dieser wiederum Vorrang vor dem Wert von `LANG`. Als Nebeneffekt sorgt die deutsche Sortierreihenfolge übrigens dafür, dass Groß- und Kleinschreibung ignoriert werden.

Wenn Sie nichts anderes angeben, wird »lexikographisch« unter Betrachtung der kompletten Zeile sortiert, das heißt, wenn die ersten Zeichen zweier Zeilen gleich sind, entscheidet das erste verschiedene Zeichen weiter hinten in der Zeile über deren relative Anordnung. Natürlich kann `sort` Zeilen nicht nur nach der ganzen

Sortieren in Feldern

Zeile, sondern auch »gezielt« nach den »Spalten« oder Feldern einer (zumindest konzeptuellen) Tabelle sortieren. Die Felder werden beginnend mit 1 numeriert; mit der Option »`-k 2`« würde das erste Feld ignoriert und das zweite Feld jeder Zeile zum Sortieren herangezogen. Sind die Werte zweier Zeilen im zweiten Feld gleich, wird der Rest der Zeile angeschaut, wenn Sie nicht mit etwas wie »`-k 2,3`« das letzte anzuschauende Feld angeben. »`-k 2,2`« sortiert *nur* nach dem Wert der zweiten Spalte. Es ist übrigens auch erlaubt, im selben Kommando mehrere `-k`-Optionen anzugeben.

`sort` unterstützt außerdem noch eine antiquierte Form der Positionsangabe: Hier werden die Felder beginnend mit 0 numeriert, und das Startfeld wird mit »+*m*« und das Stoppfeld mit »-*n*« angegeben. Um die Differenzen zur modernen Form komplett zu machen, ist die Angabe des Stoppfelds auch noch »exklusive« – benannt wird das erste Feld, nach dem *nicht mehr* sortiert werden soll. Die Beispiele von oben wären also respektive »`+1`«, »`+1 -3`« und »`+1 -2`«.

Trennmarkierung

Als Trennmarkierung zwischen den verschiedenen Feldern dient das Leerzeichen. Folgen mehrere Leerzeichen aufeinander, wird nur das erste als Trennzeichen interpretiert, die restlichen werden dem Inhalt des folgenden Feldes zugeordnet. Dazu ein kleines Beispiel, namentlich die Meldeliste für den alljährlichen Marathonlauf des TSV Lahmhausen. Ganz zu Anfang stellen wir sicher, dass wir die Standardsprachumgebung des Systems (»`POSIX`«) verwenden, indem wir die entsprechenden Umgebungsvariablen zurücksetzen. (Die vierte Spalte ist übrigens die Startnummer.)

```
$ unset LANG LC_ALL LC_COLLATE
$ cat teilnehmer.dat
Schulz        Hugo      SV Schnaufenberg     123 Herren
Schleicher    Detlef    TSV Lahmhausen       13  Herren
Flöttmann     Fritz     Sportfreunde Renntal 217 Herren
Springinsfeld Karlheinz TV Jahnstein         154 Herren
von Traben    Gesine    TV Jahnstein         26  Damen
Rasbichel     Ulla      TSV Lahmhausen       117 Damen
Schwitz       Sieglinde Sportfreunde Renntal 93  Damen
Rasbichel     Katja     TSV Lahmhausen       119 Damen
Langbein      Leni      SV Schnaufenberg     55  Damen
Zielinger     Hannes    TV Jahnstein         45  Herren
Fluschinsky   Käthe     Sportfreunde Renntal 57  Damen
```

Versuchen wir uns zunächst an einer alphabetisch nach dem Nachnamen sortierten Liste. Das ist prinzipiell einfach, weil die Nachnamen ganz vorne in der Zeile stehen:

```
$ sort teilnehmer.dat
Fluschinsky   Käthe     Sportfreunde Renntal 57  Damen
Flöttmann     Fritz     Sportfreunde Renntal 217 Herren
Langbein      Leni      SV Schnaufenberg     55  Damen
Rasbichel     Katja     TSV Lahmhausen       119 Damen
Rasbichel     Ulla      TSV Lahmhausen       117 Damen
Schleicher    Detlef    TSV Lahmhausen       13  Herren
```

```
Schulz        Hugo      SV Schnaufenberg     123 Herren
Schwitz       Sieglinde Sportfreunde Renntal 93  Damen
Springinsfeld Karlheinz TV Jahnstein         154 Herren
Zielinger     Hannes    TV Jahnstein         45  Herren
von Traben    Gesine    TV Jahnstein         26  Damen
```

Sie sehen bestimmt die zwei kleinen Probleme in dieser Liste: Einerseits sollte »Flöttmann« vor »Fluschinsky« einsortiert werden, andererseits »von Traben« vor »Zielinger«. Beide verschwinden, wenn wir darauf achten, die deutschen Sortierregeln einzuhalten:

```
$ LC_COLLATE=de_DE sort teilnehmer.dat
Flöttmann     Fritz     Sportfreunde Renntal 217 Herren
Fluschinsky   Käthe     Sportfreunde Renntal 57  Damen
Langbein      Leni      SV Schnaufenberg     55  Damen
Rasbichel     Katja     TSV Lahmhausen       119 Damen
Rasbichel     Ulla      TSV Lahmhausen       117 Damen
Schleicher    Detlef    TSV Lahmhausen       13  Herren
Schulz        Hugo      SV Schnaufenberg     123 Herren
Schwitz       Sieglinde Sportfreunde Renntal 93  Damen
Springinsfeld Karlheinz TV Jahnstein         154 Herren
von Traben    Gesine    TV Jahnstein         26  Damen
Zielinger     Hannes    TV Jahnstein         45  Herren
```

Als nächstes sortieren wir nach dem Vornamen:

```
$ sort -k 2,2 teilnehmer.dat
Schulz        Hugo      SV Schnaufenberg     123 Herren
Schwitz       Sieglinde Sportfreunde Renntal 93  Damen
Langbein      Leni      SV Schnaufenberg     55  Damen
Flöttmann     Fritz     Sportfreunde Renntal 217 Herren
Zielinger     Hannes    TV Jahnstein         45  Herren
Rasbichel     Katja     TSV Lahmhausen       119 Damen
Rasbichel     Ulla      TSV Lahmhausen       117 Damen
Schleicher    Detlef    TSV Lahmhausen       13  Herren
Fluschinsky   Käthe     Sportfreunde Renntal 57  Damen
Springinsfeld Karlheinz TV Jahnstein         154 Herren
von Traben    Gesine    TV Jahnstein         26  Damen
```

Hier kommt die oben erwähnte Eigenschaft von `sort` zum Tragen, das erste einer Folge von Leerzeichen als Trenner zu interpretieren und die folgenden dem Anfang des nächsten Feldes zuzuschlagen. Wie Sie sehen, sind zwar die Vornamen alphabetisch sortiert, aber immer nur innerhalb der jeweils gleich langen Nachnamen. Dies können Sie durch die Option `-b` beheben, die Folgen von Leerzeichen so behandelt wie ein einziges:

```
$ sort -b -k 2,2 teilnehmer.dat
Schleicher    Detlef    TSV Lahmhausen       13  Herren
Flöttmann     Fritz     Sportfreunde Renntal 217 Herren
Zielinger     Hannes    TV Jahnstein         45  Herren
Schulz        Hugo      SV Schnaufenberg     123 Herren
Springinsfeld Karlheinz TV Jahnstein         154 Herren
Rasbichel     Katja     TSV Lahmhausen       119 Damen
Fluschinsky   Käthe     Sportfreunde Renntal 57  Damen
Langbein      Leni      SV Schnaufenberg     55  Damen
Schwitz       Sieglinde Sportfreunde Renntal 93  Damen
von Traben    Gesine    TV Jahnstein         26  Damen
Rasbichel     Ulla      TSV Lahmhausen       117 Damen
```

Tabelle 7.10: Optionen für `sort` (Auswahl)

Option		Wirkung
`-b`	(*blank*)	ignoriert führende Leerzeichen im Feldinhalt
`-d`	(*dictionary*)	sortiert nach Wörterbuch-Kriterien, d. h. nur Buchstaben, Ziffern und Leerzeichen werden berücksichtigt
`-f`	(*fold*)	keine Unterscheidung von Groß- und Kleinbuchstaben
`-i`	(*ignore*)	nicht druckbare Zeichen werden ignoriert
`-k` ⟨*Feld*⟩`[,`⟨*Feld'*⟩`]`	(*key*)	Sortiere gemäß ⟨*Feld*⟩ (bis einschließlich ⟨*Feld'*⟩)
`-n`	(*numeric*)	betrachtet Feldinhalt als Zahl und sortiert nach dem numerischen Wert, führende Leerzeichen werden ignoriert
`-o datei`	(*output*)	schreibt Arbeitsergebnis in eine Datei, deren Name hier mit der Ursprungsdatei übereinstimmen darf!
`-r`	(*reverse*)	sortiert absteigend, also von Z nach A
`-t`⟨*Zeichen*⟩	(*terminate*)	das ⟨*Zeichen*⟩ dient als Feldtrennzeichen
`-u`	(*unique*)	gibt nur die erste einer Folge von identischen Zeilen aus

Die korrekte Sortierung von »Karlheinz«, »Katja« und »Käthe« erreichen Sie natürlich durch die Verwendung der deutschen Sprachumgebung, wobei Sie feststellen werden, dass diese auch die `-b`-Option impliziert. Die sortierte Liste enthält dann immer noch einen Schönheitsfehler; siehe hierzu Übung 7.24.

genauere Feldbestimmung

Das zu sortierende Feld können Sie noch genauer bestimmen, wie das folgende Beispiel zeigt:

```
$ sort -br -k 2.2 teilnehmer.dat
Fluschinsky   Käthe     Sportfreunde Renntal 57  Damen
Schulz        Hugo      SV Schnaufenberg     123 Herren
Flöttmann     Fritz     Sportfreunde Renntal 217 Herren
von Traben    Gesine    TV Jahnstein         26  Damen
Rasbichel     Ulla      TSV Lahmhausen       117 Damen
Schwitz       Sieglinde Sportfreunde Renntal 93  Damen
Schleicher    Detlef    TSV Lahmhausen       13  Herren
Langbein      Leni      SV Schnaufenberg     55  Damen
Rasbichel     Katja     TSV Lahmhausen       119 Damen
Springinsfeld Karlheinz TV Jahnstein         154 Herren
Zielinger     Hannes    TV Jahnstein         45  Herren
```

Hier wird die Datei `teilnehmer.dat` absteigend (`-r`) nach dem zweiten Zeichen der zweiten Tabellenspalte, also dem zweiten Buchstaben des Vornamens, sortiert (sehr sinnvoll!). Auch in diesem Fall ist es erforderlich, führende Leerzeichen mit `-b` zu ignorieren. (Der Schönheitsfehler aus Übung 7.24 manifestiert sich auch hier noch.)

andere Trennzeichen

Mit der Option `-t` (engl. *terminate*, »begrenzen«) können Sie statt des Leerzeichens beliebige andere Trennzeichen festlegen. Dies ist fundamental eine gute Idee, weil die zu sortierenden Felder dann Leerzeichen enthalten dürfen. Hier ist eine bequemer zu verwendende (wenn auch schwerer zu lesende) Fassung unserer Beispieldatei:

```
Schulz:Hugo:SV Schnaufenberg:123:Herren
Schleicher:Detlef:TSV Lahmhausen:13:Herren
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
```

```
Langbein:Leni:SV Schnaufenberg:55:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
```

Die Sortierung nach dem Vornamen liefert nun mit »`LC_COLLATE=de_DE sort -t: -k2,2`« korrekte Ergebnisse. Auch wird es leichter, zum Beispiel nach der Startnummer (nun Feld 4, unabhängig von der Anzahl der Leerzeichen im Vereinsnamen) zu sortieren:

```
$ sort -t: -k4 teilnehmer0.dat
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Schulz:Hugo:SV Schnaufenberg:123:Herren
Schleicher:Detlef:TSV Lahmhausen:13:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
Langbein:Leni:SV Schnaufenberg:55:Damen
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
```

Natürlich ist auch die Sortierung nach der Startnummer, wenn Sie nichts anderes sagen, lexikographisch – »117« und »123« stehen vor »13« und das wiederum vor »154«. Dies läßt sich ändern, indem Sie die Option `-n` angeben und dadurch einen numerischen Vergleich erzwingen:

numerischer Vergleich

```
$ sort -t: -k4 -n teilnehmer0.dat
Schleicher:Detlef:TSV Lahmhausen:13:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
Langbein:Leni:SV Schnaufenberg:55:Damen
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Schulz:Hugo:SV Schnaufenberg:123:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
```

Diese und die wichtigsten anderen Optionen für `sort` finden sich in Tabelle 7.10; es empfiehlt sich in jedem Fall, die Dokumentation des Programms genau zu studieren. `sort` ist ein vielseitiges und leistungsfähiges Programm, mit dem Sie sich jede Menge Arbeit sparen können.

Kommando `uniq`

Das Kommando `uniq` hat die wichtige Funktion, von unmittelbar aufeinanderfolgenden »gleichen« Zeilen in der Eingabe nur eine durchzulassen. Wann zwei Zeilen als »gleich« gelten, lässt sich – wie üblich – durch Optionen einstellen. `uniq` unterscheidet sich von den meisten der bisher gesehenen Programme dadurch, dass es keine beliebige Anzahl von benannten Eingabedateien akzeptiert, sondern höchstens eine; ein etwaiger zweiter Dateiname wird als Name für die Ausgabedatei angesehen (ersatzweise die Standard-Ausgabe). Wird keine Datei im `uniq`-Aufruf benannt, liest `uniq`, so wie es sich gehört, seine Standard-Eingabe.

`uniq` funktioniert am besten, wenn die Eingabezeilen sortiert sind, so dass *alle* gleichen Zeilen hintereinander stehen. Ist das nicht der Fall, so ist eben nicht gesagt, dass jede Zeile in der Ausgabe nur einmal vorkommt:

```
$ cat uniq-test
Hipp
Hopp
```

```
Hopp
Hipp
Hipp
Hopp
$ uniq uniq-test
Hipp
Hopp
Hipp
Hopp
```

Vergleichen Sie das mit der Ausgabe von »`sort -u`«:

```
$ sort -u uniq-test
Hipp
Hopp
```

Übungen

7.22 [!2] Sortieren Sie die Teilnehmerliste in `teilnehmer0.dat` (der Datei mit Doppelpunkten als Feldtrenner) nach den Vereinsnamen und innerhalb der Vereine nach den Nach- und Vornamen der Spieler (in dieser Reihenfolge).

7.23 [3] Wie können Sie die Teilnehmerliste aufsteigend nach den Vereinsnamen und innerhalb der Vereine absteigend nach der Startnummer sortieren? (*Tipp:* Dokumentation lesen!)

7.24 [!2] Was ist der »Schönheitsfehler«, von dem in den Beispielen die Rede ist, und warum tritt er auf?

7.25 [2] Ein Verzeichnis enthält Dateien mit den folgenden Namen:

```
01-2002.txt  01-2003.txt  02-2002.txt  02-2003.txt
03-2002.txt  03-2003.txt  04-2002.txt  04-2003.txt
<<<<<
11-2002.txt  11-2003.txt  12-2002.txt  12-2003.txt
```

Geben Sie ein `sort`-Kommando an, mit dem Sie die Ausgabe von `ls` in die »chronologisch richtige« Reihenfolge

```
01-2002.txt
02-2002.txt
<<<<<
12-2002.txt
01-2003.txt
<<<<<
12-2003.txt
```

bringen können.

7.26 [3] Wie können Sie eine sortierte Liste aller Wörter in einer Textdatei aufstellen? Jedes Wort soll in der Liste nur einmal erscheinen. (*Tipp:* Übung 7.16)

7.5.2 Spalten und Felder – `cut`, `paste` & Co.

Spalten ausschneiden

Während Sie mit dem Kommando `grep` Zeilen einer Textdatei durchsuchen und ausschneiden können, arbeitet sich `cut` (engl. für »schneiden«) gewissermaßen vertikal durch einen Text. Dies kann auf zwei Arten erfolgen:

absolute Spalten

Eine Möglichkeit ist die absolute Bearbeitung von Spalten. Diese Spalten ent-

sprechen einzelnen Zeichen einer Zeile. Um solche Spalten auszuschneiden, muss nach der Option `-c` (engl. *column*, »Spalte«) die Spaltennummer angegeben werden. Sollen mehrere Spalten in einem Schritt ausgeschnitten werden, können diese als kommaseparierte Liste festgelegt werden. Auch die Angabe von Spaltenbereichen ist zulässig.

```
$ cut -c 15,1-5 teilnehmer.dat
SchulH
SchleD
FlöttF
SprinK
von TG
```

Im Beispiel werden der Anfangsbuchstabe des Vornamens sowie die ersten fünf Zeichen des Nachnamens ausgeschnitten. Es illustriert auch gleich die bemerkenswerte Tatsache, dass die Ausgabe immer in der Reihenfolge erfolgt, wie die ausgeschnittenen Spalten in der Eingabe stehen. Auch wenn die ausgewählten Spaltenbereiche sich überlappen, wird jedes Zeichen der Eingabe höchstens einmal ausgegeben:

```
$ cut -c 1-5,2-6,3-7 teilnehmer.dat
Schulz
Schleic
Flöttma
Springi
von Tra
```

Felder ausschneiden

Die zweite Möglichkeit ist, relativ in Feldern auszuschneiden. Diese Felder werden durch Trennzeichen abgegrenzt. Möchten Sie feldweise ausschneiden, benötigt `cut` die Option `-f` (engl. *field*, »Feld«) und die gewünschte Feldnummer. Für diese gelten die gleichen Regeln wie für die Spaltennummern. Übrigens schließen sich die Optionen `-c` und `-f` gegenseitig aus.

Trenner

Als Trenner ist das Tabulatorzeichen voreingestellt, andere Trenner lassen sich mit der Option `-d` (engl. *delimiter*, »Trennzeichen«) vorgeben:

```
$ cut -d: -f 1,4 teilnehmer0.dat
Schulz:123
Schleicher:13
Flöttmann:217
Springinsfeld:154
von Traben:26
Rasbichel:117
```

Auf diese Weise werden der Meldeliste die Nachnamen der Teilnehmer (Spalte 1) sowie die Benutzernummern (Spalte 4) entnommen, Trennzeichen ist der Doppelpunkt. Aus Gründen der Übersichtlichkeit ist hier nur eine verkürzte Ausgabe abgebildet.

Sie können übrigens mit der Option `--output-delimiter` ein anderes Trennzeichen für die Ausgabe festlegen als das, welches für die Eingabe verwendet wird:

```
$ cut -d: --output-delimiter=': ' -f 1,4 teilnehmer0.dat
Schulz: 123
Schleicher: 13
Flöttmann: 217
```

```
Springinsfeld: 154
von Traben: 26
Rasbichel: 117
◁◁◁◁◁
```

Wenn Sie tatsächlich Spalten oder Felder umsortieren wollen, ist schwereres Geschütz angesagt, etwa die Programme `awk` oder `perl`. Notfalls geht es auch mit dem gleich vorgestellten Kommando `paste`, das ist aber etwas mühselig.

Unterdrückung von Zeilen ohne Felder

Bei der feldweisen Bearbeitung von Textdateien ist `-s` (engl. *separator*, »Trenner«) eine sinnvolle Option. Findet »`cut -f`« Zeilen, die kein Trennzeichen enthalten, werden diese üblicherweise komplett ausgegeben; `-s` verhindert diese Ausgabe.

Dateien zeilenweise zusammenfügen

Das Kommando `paste` (engl. für »zusammenkleben«) fügt die angegebenen Dateien zeilenweise zusammen, es wird daher oft in Verbindung mit `cut` benutzt. Wie Sie sicher sofort bemerkt haben, ist `paste` eigentlich kein Filterkommando. Geben Sie jedoch für einen der Dateinamen ein Minuszeichen an, so registriert `paste`, dass dieser Text aus der Standard-Eingabe gelesen werden soll. Die Ausgabe erfolgt stets auf der Standard-Ausgabe.

Dateien parallel durchlaufen

Wie erwähnt arbeitet `paste` zeilenweise. Bei der Angabe von zwei Dateinamen werden die erste Zeile aus der ersten Datei und die erste aus der zweiten Datei, durch ein Tabulatorzeichen getrennt, zur ersten Zeile der Ausgabe verbunden, entsprechend wird mit allen weiteren Zeilen verfahren. Wenn Sie statt des Tabulatorzeichens ein anderes Trennzeichen verwenden wollen, kann dies mit der Option `-d` festgelegt werden.

Trennzeichen

Zum Beispiel können wir eine Version der Marathon-Meldeliste herstellen, bei der die Startnummer vorne steht:

```
$ cut -d: -f4 teilnehmer0.dat >startnr.dat
$ cut -d: -f1-3,5 teilnehmer0.dat \
>   | paste -d: startnr.dat - >tn-startnr.dat
$ cat tn-startnr.dat
123:Schulz:Hugo:SV Schnaufenberg:Herren
13:Schleicher:Detlef:TSV Lahmhausen:Herren
217:Flöttmann:Fritz:Sportfreunde Renntal:Herren
154:Springinsfeld:Karlheinz:TV Jahnstein:Herren
26:von Traben:Gesine:TV Jahnstein:Damen
117:Rasbichel:Ulla:TSV Lahmhausen:Damen
93:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
119:Rasbichel:Katja:TSV Lahmhausen:Damen
55:Langbein:Leni:SV Schnaufenberg:Damen
45:Zielinger:Hannes:TV Jahnstein:Herren
57:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
```

Diese Datei kann jetzt bequem mit »`sort -n tn-startnr.dat`« nach dem numerischen Wert der Startnummer sortiert werden.

Dateien nacheinander durchlaufen

Durch `-s` (engl. *serial*, »nacheinander«) werden die angegebenen Dateien nacheinander durchlaufen. Zunächst werden alle Zeilen der ersten Datei mit Trennzeichen zu einer Zeile zusammengefasst, anschließend alle Zeilen aus der zweiten Datei in der zweiten Zeile usw.

```
$ cat liste1
Hund
Katze
Maus
$ cat liste2
Ei
Blut
Kakao
```

Tabelle 7.11: Optionen für `join` (Auswahl)

Option		Wirkung
`-j1` *n*		Verwendet Feld *n* der ersten Datei als »Vereinigungsfeld« ($n \geqslant 1$). Synonym: `-1` *n*.
`-j2` *n*		Verwendet Feld *n* der zweiten Datei als »Vereinigungsfeld« ($n \geqslant 1$). Synonym: `-2` *n*.
`-j` *n*	(*join*)	Abkürzung für »`-j1` *n* `-j2` *n*«
`-o` *f*	(*output*)	Spezifikation für die Ausgabezeilen. *f* ist eine durch Komma getrennte Folge von Feldspezifikationen, wobei jede entweder die Ziffer »`0`« oder eine Feldnummer *m.n* ist. Dabei steht die »`0`« für das »Vereinigungsfeld«, *m* ist 1 oder 2, und *n* ist eine Feldnummer in der ersten bzw. zweiten Datei.
`-t` *c*		Das Zeichen *c* wird als Feldtrenner für Ein- und Ausgabe verwendet.

```
$ paste -s liste*
Hund      Katze     Maus
Ei        Blut      Kakao
```

Alle Dateien, deren Name dem Suchmuster `liste*` entspricht, hier also lediglich `liste1` und `liste2`, werden von `paste` zusammengesetzt. Die Angabe von `-s` bewirkt, dass jede Zeile dieser Dateien eine Spalte der Ausgabe ergibt.

»Relationales« Zusammenfügen von Dateien

Auch das Kommando `join` setzt Zeilen aus zwei Dateien zusammen, arbeitet aber wesentlich komplizierter als `paste`. Statt einfach die ersten Zeilen, zweiten Zeilen, … aneinanderzuhängen, orientiert es sich an jeweils einem ausgezeichneten Feld der Zeilen aus beiden Dateien und »vereinigt« zwei Zeilen nur, wenn die jeweiligen Werte in diesem Feld übereinstimmen. `join` implementiert also den gleichnamigen Operator aus der Relationenalgebra, so wie man ihn von SQL-Datenbanken her kennt – auch wenn die tatsächliche Ausführung deutlich primitiver und ineffizienter ist, als eine echte Datenbank das ermöglichen würde.

Beispiel

Trotzdem kann `join` durchaus nützlich werden. Stellen wir uns vor, der große Tag ist gekommen und der Marathonlauf des TSV Lahmhausen hat stattgefunden. Die Schiedsrichter waren fleißig und haben nicht nur alle Zeiten gestoppt, sondern diese auch in eine Datei `zeiten.dat` eingetragen. Die erste Spalte ist dabei immer die Startnummer, die zweite die erreichte Zeit (der Einfachheit halber in ganzen Sekunden):

```
$ cat zeiten.dat
45:8445
123:8517
217:8533
93:8641
154:8772
119:8830
13:8832
117:8954
57:9111
26:9129
```

Wir wollen nun diese Datei mit der Teilnehmerliste zusammenbringen, damit wir jedem Sportler bzw. jeder Sportlerin seine bzw. ihre erreichte Zeit zuordnen können. Dazu müssen wir die Ergebnisdatei zunächst nach den Startnummern sortieren:

```
$ sort -n zeiten.dat >zeiten-s.dat
```

Anschließend können wir die Zeilen der Datei `zeiten-s.dat` mit dem Programm `join` mit den entsprechenden Zeilen der modifizierten Teilnehmerdatei aus dem

paste-Beispiel zusammenbringen – join setzt standardmäßig voraus, dass seine Eingabedateien zeilenweise nach dem »Vereinigungsfeld« sortiert sind, und dass das »Vereinigungsfeld« das erste Feld auf der Zeile ist.

```
$ cat tn-startnr.dat
123:Schulz:Hugo:SV Schnaufenberg:Herren
13:Schleicher:Detlef:TSV Lahmhausen:Herren
217:Flöttmann:Fritz:Sportfreunde Renntal:Herren
154:Springinsfeld:Karlheinz:TV Jahnstein:Herren
26:von Traben:Gesine:TV Jahnstein:Damen
117:Rasbichel:Ulla:TSV Lahmhausen:Damen
93:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
119:Rasbichel:Katja:TSV Lahmhausen:Damen
55:Langbein:Leni:SV Schnaufenberg:Damen
45:Zielinger:Hannes:TV Jahnstein:Herren
57:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
$ sort -n tn-startnr.dat \
>  | join -t: zeiten-s.dat - >tn-zeiten.dat
$ cat tn-zeiten.dat
13:8832:Schleicher:Detlef:TSV Lahmhausen:Herren
26:9129:von Traben:Gesine:TV Jahnstein:Damen
45:8445:Zielinger:Hannes:TV Jahnstein:Herren
57:9111:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
93:8641:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
117:8954:Rasbichel:Ulla:TSV Lahmhausen:Damen
119:8830:Rasbichel:Katja:TSV Lahmhausen:Damen
123:8517:Schulz:Hugo:SV Schnaufenberg:Herren
154:8772:Springinsfeld:Karlheinz:TV Jahnstein:Herren
217:8533:Flöttmann:Fritz:Sportfreunde Renntal:Herren
```

Die resultierende Datei tn-zeiten.dat muss jetzt nur noch nach den Zeiten sortiert werden:

```
$ sort -t: -k2,2 tn-zeiten.dat
45:8445:Zielinger:Hannes:TV Jahnstein:Herren
123:8517:Schulz:Hugo:SV Schnaufenberg:Herren
217:8533:Flöttmann:Fritz:Sportfreunde Renntal:Herren
93:8641:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
154:8772:Springinsfeld:Karlheinz:TV Jahnstein:Herren
119:8830:Rasbichel:Katja:TSV Lahmhausen:Damen
13:8832:Schleicher:Detlef:TSV Lahmhausen:Herren
117:8954:Rasbichel:Ulla:TSV Lahmhausen:Damen
57:9111:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
26:9129:von Traben:Gesine:TV Jahnstein:Damen
```

Dies ist ein schönes Beispiel dafür, dass mit Linux-»Bordmitteln« auch durchaus komplexe Text- und Datenverarbeitung möglich ist. Im »wirklichen Leben« würden Sie die gezeigten Befehlsfolgen natürlich als Shellskripte vorbereiten und die entsprechenden Schritte auf diese Weise automatisieren.

Übungen

7.27 [!2] Generieren Sie eine neue Version der Datei teilnehmer.dat (der mit der festen Spaltenbreite), in der die Startnummer und die Vereinszugehörigkeit nicht auftauchen.

7.28 [!2] Generieren Sie eine neue Version der Datei teilnehmer0.dat (der mit den durch Doppelpunkt getrennten Feldern), in der die Startnummer und die Vereinszugehörigkeit nicht auftauchen.

7.29 [3] Erzeugen Sie eine Version der Datei `teilnehmer0.dat`, bei der die Felder nicht durch Doppelpunkte, sondern durch die Zeichenkette »,␣« (Komma gefolgt von einem Leerzeichen) getrennt sind.

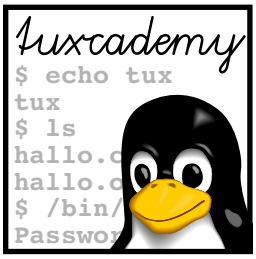
7.30 [3] Wie viele verschiedene Gruppen werden von Benutzern auf Ihrem System als primäre Gruppen benutzt? (Die primäre Gruppe eines Benutzers ist das vierte Feld in der Datei `/etc/passwd`.)

Kommandos in diesem Kapitel

cat	Hängt Dateien aneinander	cat(1)	103
cut	Extrahiert Felder oder Spalten aus seiner Eingabe	cut(1)	122
expand	Ersetzt Tabulatorzeichen in der Eingabe durch äquivalente Leerzeichen	expand(1)	111
fmt	Umbricht die Zeilen der Eingabe auf eine bestimmte Breite	fmt(1)	112
hd	Abkürzung für `hexdump`	hexdump(1)	107
head	Zeigt den Anfang einer Datei an	head(1)	105
hexdump	Gibt Dateiinhalte in hexadezimaler (oktaler, …) Form aus	hexdump(1)	107
join	Führt die Zeilen zweier Dateien „relational“ zusammen	join(1)	125
nl	Numeriert die Zeilen der Eingabe	nl(1)	114
od	Zeigt die Bytes einer Datei in dezimaler, oktaler, hexadezimaler, ... Darstellung	od(1)	106
paste	Fügt verschiedene Eingabedateien zeilenweise aneinander	paste(1)	124
pr	Bereitet seine Eingabe zum Drucken auf – mit Kopfzeilen, Fußzeilen usw.	pr(1)	113
reset	Setzt den Bildschirmzeichensatz auf einen „vernünftigen“ Wert	tset(1)	103
sort	Sortiert die Zeilen seiner Eingabe	sort(1)	117
tac	Zeigt eine Datei von hinten nach vorne an	tac(1)	104
tail	Zeigt das Ende einer Datei an	tail(1)	105
tee	Kopiert die Standardeingabe in die Standardausgabe und außerdem in Dateien	tee(1)	101
tr	Tauscht Zeichen in der Standardeingabe gegen andere aus oder löscht sie	tr(1)	109
unexpand	„Optimiert“ Tabulator- und Leerzeichen in der Eingabe	unexpand(1)	111
uniq	Ersetzt Folgen von gleichen Zeilen in der Eingabe durch die erste solche	uniq(1)	121
wc	Zählt Zeilen, Wörter und Zeichen in seiner Eingabe	wc(1)	116

Zusammenfassung

- Jedes Linux-Programm unterstützt die Standard-Ein- und -Ausgabe-Kanäle `stdin`, `stdout` und `stderr`.
- Die Standard-Ausgabe und Standard-Fehlerausgabe können mit den Operatoren > und >>, die Standard-Eingabe mit dem Operator < umgeleitet werden.
- Über Pipelines lassen sich die Standard-Aus- und -Eingabe von Programmen direkt (ohne Zwischendateien) miteinander verbinden.
- Mit dem Kommando `tee` können Zwischenergebnisse einer Pipeline in Dateien gespeichert werden.
- Filterkommandos (oder »Filter«) lesen ihre Standardeingabe, manipulieren sie und schreiben die Ergebnisse auf die Standardausgabe.
- Das Kommando `tr` dient zum Austauschen oder Löschen einzelner Zeichen. `expand` und `unexpand` konvertieren Tabulatorzeichen in Leerzeichen und umgekehrt.
- Mit `pr` kann man Daten zum Drucken aufbereiten (nicht drucken).
- Mit `wc` kann man die Zeilen, Wörter und Zeichen der Standardeingabe (oder der angegebenen Datei(en)) zählen.
- `sort` ist ein vielseitiges Sortierprogramm.
- Das Kommando `cut` schneidet bestimmte Spaltenbereiche oder Felder jeder Zeile der Eingabe aus.
- Mit `paste` können die Zeilen von Dateien aneinandergehängt werden.



8

Mehr über die Shell

Inhalt

8.1 `sleep`, `echo` und `date` 130
8.2 Shell-Variable und die Umgebung 131
8.3 Arten von Kommandos – die zweite 133
8.4 Die Shell als komfortables Werkzeug 135
8.5 Kommandos aus einer Datei 138
8.6 Die Shell als Programmiersprache 139
8.7 Vorder- und Hintergrundprozesse. 143

Lernziele

- Shell- und Umgebungsvariable kennenlernen
- Mit Vordergrund- und Hintergrund-Prozessen umgehen können

Vorkenntnisse

- Shell-Grundkenntnisse (Kapitel 3)
- Dateiverwaltung und einfache Filterkommandos (Kapitel 6, Kapitel 7)
- Umgang mit einem Texteditor (Kapitel 5)

8.1 `sleep`, `echo` und `date`

Bevor wir uns mit der eigentlichen Shell beschäftigen, müssen wir Ihnen noch ein bisschen mehr Material für Experimente geben. Dazu erklären wir Ihnen hier noch einige ganz einfache Kommandos:

sleep Dieses Kommando tut für die als Parameter angegebene Anzahl von Sekunden gar nichts. Sie können es benutzen, wenn Sie wollen, dass Ihre Shell eine »Kunstpause« einlegt:

```
$ sleep 10
                                        Ca. 10 Sekunden lang passiert nichts
$ _
```

Argumente ausgeben

echo Das Kommando `echo` gibt seine Argumente aus (sonst nichts), und zwar durch Leerzeichen getrennt. Es ist aber doch interessant und nützlich, da die Shell vorher Variablenbezüge (siehe Abschnitt 8.2) und ähnliches ersetzt:

```
$ w=Welt
$ echo Hallo $w
Hallo Welt
$ echo Hallo ${w}enbummler
Hallo Weltenbummler
```

(Das zweite `echo` illustriert, was Sie machen können, wenn direkt etwas an den Ausgabewert einer Variable angehängt werden soll.)

Wird `echo` mit der Option `-n` aufgerufen, so schreibt es am Ende seiner Ausgabe keinen Zeilentrenner:

```
$ echo -n Hallo
Hallo$
```

Datum und Uhrzeit anzeigen

date Das Kommando `date` (engl. »Datum«) zeigt das aktuelle Datum sowie die Uhrzeit an. Sie können in weiten Grenzen bestimmen, wie diese Ausgabe aussehen soll – rufen Sie »`date --help`« auf oder lesen Sie mit »`man date`« die Online-Dokumentation.

(Beim zweiten Durcharbeiten dieses Kapitels:) Insbesondere betätigt `date` sich als Weltzeituhr, wenn Sie vorher die Umgebungsvariable `TZ` auf den Namen einer Zeitzone oder wichtigen Stadt (typischerweise Hauptstadt) setzen:

```
$ date
Thu Oct  5 14:26:07 CEST 2006
$ export TZ=Asia/Tokyo
$ date
Tue Oct  5 21:26:19 JST 2006
$ unset TZ
```

Die gültigen Zeitzonen- und Städtenamen können Sie herausfinden, indem Sie in `/usr/share/zoneinfo` stöbern.

Systemzeit stellen

Während jeder Anwender die Systemzeit abfragen darf, ist es nur dem Systemadministrator `root` erlaubt, mit dem Befehl `date` und einem Argument in der Form `MMTThhmm` die Systemzeit zu verändern. Im Argument stehen dabei `MM` für Monat, `TT` für Tag, `hh` für Stunde und `mm` für Minute. Optional dazu können noch jeweils zwei

Stellen für die Jahreszahl (plus möglicherweise zwei für das Jahrhundert) sowie die Sekunden (mit einem Punkt davor) angegeben werden, was aber nur in den seltensten Fällen notwendig sein dürfte.

```
$ date
Thu Oct  5 14:28:13 CEST 2006
$ date 08181715
date: cannot set date: Operation not permitted
Fri Aug 18 17:15:00 CEST 2006
```

Mit dem `date`-Kommando wird nur die interne Zeit des Linux-Systems geändert. Diese Zeit wird nicht notwendigerweise in die CMOS-Uhr auf der Hauptplatine des Rechners übertragen, so dass es notwendig sein kann, diese mit einem speziellen Kommando zu ändern. Viele Distributionen machen das automatisch, wenn das System heruntergefahren wird.

Übungen

8.1 [!3] Angenommen, jetzt ist der 22. Oktober 2003, 12:34 Uhr und 56 Sekunden. Studieren Sie die Dokumentation von `date` und geben Sie die Formatierungsanweisungen an, mit denen Sie die folgenden Ausgaben erreichen können:

1. `22-10-2003`
2. `03-294 (KW43)` (Zweistellige Jahreszahl, fortlaufende Nummer des Tags im Jahr, Kalenderwoche)
3. `12h34m56s`

8.2 [!2] Wie spät ist es gerade in Los Angeles?

8.2 Shell-Variable und die Umgebung

Die Bash hat – wie die meisten gängigen Shells – Eigenschaften, die man sonst in Programmiersprachen findet. Zum Beispiel ist es möglich, Text oder numerische Werte in Variablen abzulegen und später wieder hervorzuholen. Variable steuern auch verschiedene Aspekte der Funktionsweise der Shell selbst.

In der Shell wird eine Variable durch ein Kommando wie »`bla=fasel`« gesetzt (dieses Kommando setzt die Variable `bla` auf den textuellen Wert `fasel`). Achten Sie darauf, dass vor und hinter dem Gleichheitszeichen *keine* Leerzeichen stehen! Verwenden können Sie den Wert der Variablen, indem Sie den Variablennamen mit einem vorgesetzten Dollarzeichen benutzen:

Variable setzen

```
$ bla=fasel
$ echo bla
bla
$ echo $bla
fasel
```

(achten Sie auf den Unterschied).

Wir unterscheiden **Umgebungsvariable** und **Shellvariable**. Shellvariable sind nur in der betreffenden Shell sichtbar. Im Gegensatz dazu werden die Umgebungsvariablen beim Starten eines externen Kommandos an den Kindprozess weitergegeben und können auch dort benutzt werden. (Der Kindprozess muss nicht unbedingt eine Shell sein; jeder Linux-Prozess hat Umgebungsvariable.) Alle Umgebungsvariablen einer Shell sind gleichzeitig auch Shellvariable, aber umgekehrt ist es nicht so.

Umgebungsvariable
Shellvariable

Mit dem Kommando `export` können Sie eine existierende Shellvariable zur Umgebungsvariable erklären:

`export`

Tabelle 8.1: Wichtige Variable der Shell

Variable	Bedeutung
PWD	Name des aktuellen Verzeichnisses
PS1	enthält die Zeichenkette, die am Beginn jeder Eingabezeile angezeigt wird (den Prompt)
UID	enthält die Benutzerkennung
HOME	Pfad des Heimatverzeichnisses des Benutzers
PATH	Liste von Verzeichnissen, die von der Shell automatisch als Suchpfade für Befehle verwendet werden
LOGNAME	enthält den Benutzernamen

```
$ bla=fasel                                   bla ist jetzt Shellvariable
$ export bla                                  bla ist jetzt Umgebungsvariable
```

Oder Sie definieren eine neue Variable gleich als Shell- und Umgebungsvariable:

```
$ export bla=fasel
```

`export` funktioniert auch für mehrere Variable auf einmal:

```
$ export bla blubb
$ export bla=fasel blubb=bli
```

Variable auflisten

Die Umgebungsvariablen können Sie sich mit dem Befehl `export` (ohne Parameter) anzeigen lassen. Auch der Befehl `env` (ebenfalls ohne Parameter) zeigt die aktuelle Umgebung an. Alle Shellvariablen (inklusive diejenigen, die auch in der Umgebung sind) können Sie sich mit dem Befehl `set` anzeigen lassen. Die gebräuchlichsten Variablen und ihre Zuordnung sind in Tabelle 8.1 aufgelistet.

Der Befehl `set` tut noch viele andere fremdartige und wundervolle Dinge. Er wird Ihnen in der Linup-Front-Schulungsunterlage *Linux für Fortgeschrittene* wieder begegnen, wo es um Shellprogrammierung geht.

Auch `env` ist eigentlich dafür gedacht, die Prozessumgebung zu manipulieren, und nicht nur, sie anzuzeigen. Betrachten Sie das folgende Beispiel:

```
$ env bla=fasel bash                          Starte Kind-Shell mit bla
$ echo $bla
fasel
$ exit                                        Zurück in die Elter-Shell
$ echo $bla
                                              Nicht definiert
$ _
```

Zumindest in der Bash (und Verwandten) brauchen Sie `env` nicht wirklich, um Kommandos mit einer erweiterten Umgebung zu starten – ein einfaches

```
$ bla=fasel bash
```

hat dieselbe Wirkung. Allerdings erlaubt `env` Ihnen auch das temporäre Entfernen von Variablen aus der Umgebung (wie?).

Variable löschen

Wenn Sie von einer Shellvariablen genug haben, können Sie sie mit dem Befehl `unset` löschen. Damit verschwindet sie auch aus der Prozessumgebung. Wenn Sie eine Variable aus der Umgebung entfernen wollen, sie aber als Shellvariable beibehalten möchten, verwenden Sie »`export -n`«:

```
$ export bla=fasel                    bla ist Umgebungsvariable
$ export -n bla                       bla ist (nur) Shellvariable
$ unset bla                           bla ist ganz weg
```

Übungen

8.3 [!2] Überzeugen Sie sich, dass die Übergabe (oder Nichtübergabe) von Umgebungs- und Shellvariablen an Kindprozesse funktioniert wie behauptet, indem Sie die folgende Kommandosequenz durchspielen:

```
$ bla=fasel                           bla ist Shellvariable
$ bash                                Neue Shell (Kindprozess)
$ echo $bla
                                      bla ist nicht definiert
$ exit                                Zurück in die Elter-Shell
$ export bla                          bla ist Umgebungsvariable
$ bash                                Neue Shell (Kindprozess)
$ echo $bla
fasel                                 Umgebungsvariable wurde vererbt
$ exit                                Zurück in die Elter-Shell
```

8.4 [!2] Was passiert, wenn Sie eine Umgebungsvariable im Kindprozess ändern? Betrachten Sie die folgende Kommandosequenz:

```
$ export bla=fasel                    bla ist Umgebungsvariable
$ bash                                Neue Shell (Kindprozess)
$ echo $bla
fasel                                 Umgebungsvariable wurde vererbt
$ bla=blubb                           Neuer Wert
$ exit                                Zurück in die Elter-Shell
$ echo $bla                           Was kommt hier heraus??
```

8.3 Arten von Kommandos – die zweite

Steuerung der Shell

Eine Anwendung von Shellvariablen ist, wie erwähnt, die Steuerung der Shell selbst. Hierzu noch ein Beispiel: Wie in Kapitel 3 beschrieben, unterscheidet die Shell interne und externe Kommandos. Externe Kommandos entsprechen ausführbaren Programmen, die die Shell in den Verzeichnissen sucht, die in der Umgebungsvariablen `PATH` stehen. Hier ist ein typischer Wert für `PATH`:

```
$ echo $PATH
/home/hugo/bin:/usr/local/bin:/usr/bin:/bin:/usr/games
```

Die einzelnen Verzeichnisse werden in der Liste durch Doppelpunkte getrennt, die Liste im Beispiel besteht also aus fünf Verzeichnissen. Wenn Sie ein Kommando wie

```
$ ls
```

eingeben, weiß die Shell, dass das kein internes Kommando ist (sie kennt ihre internen Kommandos) und fängt darum an, die Verzeichnisse in `PATH` abzusuchen, beginnend am linken Ende. Konkret prüft sie, ob die folgenden Dateien existieren:

```
/home/hugo/bin/ls                                   Nein …
/usr/local/bin/ls                                   Immer noch nicht …
/usr/bin/ls                                         Nach wie vor nicht …
/bin/ls                                             Hah, Treffer!
                                    Das Verzeichnis /usr/games wird nicht mehr angeschaut.
```

Das heißt, zur Ausführung des Kommandos `ls` wird die Datei `/bin/ls` herangezogen.

Diese Suche ist natürlich ein relativ aufwendiger Prozess, und darum baut die Shell für die Zukunft vor: Wenn sie einmal die Datei `/bin/ls` als Implementierung des Kommandos `ls` ausgemacht hat, dann merkt sie sich diese Zuordnung bis auf weiteres. Diesen Vorgang nennt der Fachmann *hashing*, und dass er passiert ist, können Sie sehen, wenn Sie `type` auf unser Kommando `ls` anwenden:

```
$ type ls
ls is hashed (/bin/ls)
```

Das Kommando »`hash`« sagt Ihnen, welche Kommandos Ihre Bash bereits »gehasht« hat und wie oft sie seitdem aufgerufen wurden. Mit »`hash -r`« können Sie das komplette Hashing-Gedächtnis der Shell löschen. Es gibt noch ein paar andere Optionen, die Sie in der Bash-Dokumentation nachschlagen oder per »`help hash`« herausfinden können.

Die Variable `PATH` muss prinzipiell gesehen überhaupt keine Umgebungsvariable sein – für die aktuelle Shell funktioniert sie auch als Shellvariable (siehe Übung 8.5). Allerdings ist es bequem, sie als Umgebungsvariable zu definieren, damit auch die Kindprozesse der Shell (oft wieder Shells) den gewünschten Wert verwenden.

Wenn Sie wissen wollen, genau welches Programm die Shell für ein externes Kommando heranzieht, können Sie dafür das Kommando `which` verwenden:

```
$ which grep
/bin/grep
```

`which` verwendet dasselbe Verfahren wie die Shell – es beginnt beim ersten Verzeichnis in `PATH` und prüft jeweils, ob es in dem betreffenden Verzeichnis eine ausführbare Datei gibt, die so heißt wie das gesuchte Kommando.

`which` weiß nichts über die internen Kommandos der Shell; selbst wenn etwas wie »`which test`« also »`/usr/bin/test`« liefert, heißt das noch lange nicht, dass dieses Programm tatsächlich ausgeführt wird, denn interne Kommandos haben Vorrang gegenüber externen. Wenn Sie wissen wollen, was wirklich passiert, müssen Sie das Shell-Kommando »`type`« benutzen (Abschnitt 3.3.3).

Das Kommando `whereis` liefert nicht nur ausführbare Programme, sondern auch Dokumentationsdateien (Manpages), Quellcodedateien und andere interessante Dateien, die etwas mit den angegebenen Kommandos zu tun haben. Zum Beispiel:

```
$ whereis passwd
passwd: /usr/bin/passwd /etc/passwd /etc/passwd.org /usr/share/passwd▷
 ◁ /usr/share/man/man1/passwd.1.gz /usr/share/man/man1/passwd.1ssl.gz▷
 ◁ /usr/share/man/man5/passwd.5.gz
```

Dafür wird ein im Programm hartcodiertes Verfahren verwendet, das (ansatzweise) in `whereis`(1) erklärt wird.

Übungen

8.5 [2] Überzeugen Sie sich davon, dass die Kommandosuche der Shell auch funktioniert, wenn `PATH` keine Umgebungsvariable, sondern »nur« eine Shellvariable ist. Was passiert, wenn Sie `PATH` komplett entfernen?

8.6 [!1] Wie heißen auf Ihrem System die ausführbaren Programme, die zur Bearbeitung der folgenden Kommandos herangezogen werden: `fgrep`, `sort`, `mount`, `xterm`

8.7 [!1] Wie heißen auf Ihrem System die Dateien, die die Dokumentation für das Kommando »`crontab`« enthalten?

8.4 Die Shell als komfortables Werkzeug

Da für viele Linux-Anwender die Shell das am meisten genutzte Werkzeug ist, haben deren Entwickler sich große Mühe gegeben, ihre Bedienung komfortabel zu machen. Hier zeigen wir Ihnen noch ein paar nützliche Kleinigkeiten.

Kommandoeditor Sie können Kommandozeilen wie mit einem einfachen Texteditor bearbeiten. Der Cursor kann also in der Zeile hin- und herbewegt sowie Zeichen beliebig gelöscht oder hinzugefügt werden, bis die Eingabe durch Betätigen der Eingabetaste beendet wird. Das Verhalten dieses Editors kann übrigens mit »`set -o vi`« bzw. mit »`set -o emacs`« an das Verhalten der beiden bekanntesten Editoren unter Linux (Kapitel 5) angepasst werden.

Kommandoabbruch Bei den zahlreichen Linux-Kommandos kann es durchaus vorkommen, dass Sie mal einen Namen verwechseln oder einen falschen Parameter übergeben. Deshalb können Sie ein Kommando abbrechen, während es läuft. Hierzu müssen Sie nur die Tasten Strg+c gleichzeitig drücken.

Die »Vorgeschichte« Die Shell merkt sich Ihre letzten soundsovielen Kommandos als Vorgeschichte (engl. *history*), und Sie können sich mit den Cursortasten ↑ und ↓ in der Liste bewegen. Wenn Sie ein früheres Kommando finden, das Ihnen gefällt, können Sie es mit ↩ einfach, so wie es ist, erneut ausführen oder es zunächst (wie weiter oben angedeutet) abändern. Mit Strg+r können Sie die Liste »inkrementell« durchsuchen – tippen Sie einfach eine Zeichenfolge ein, und die Shell zeigt Ihnen das zuletzt ausgeführte Kommando, das die Zeichenfolge enthält. Je länger Ihre Zeichenfolge ist, desto präziser wird die Suche.

Die Vorgeschichte wird beim ordnungsgemäßen Verlassen des Systems in der versteckten Datei `~/.bash_history` gespeichert und steht nach dem nächsten Anmelden wieder zur Verfügung. (Sie können einen anderen Dateinamen verwenden, indem Sie die Variable `HISTFILE` auf den gewünschten Namen setzen.)

Eine Konsequenz der Tatsache, dass die Vorgeschichte in einer »gewöhnlichen« Datei abgespeichert wird, ist, dass Sie sie mit einem Texteditor ändern können. (Kapitel 5 erklärt Ihnen, wie das geht.) Sollten Sie also versehentlich Ihr Kennwort auf der Kommandozeile eintippen, können (und sollten!) Sie es mit der Hand aus der Vorgeschichte entfernen – vor allem, wenn Ihr System zu den eher freizügigen gehört, wo Heimatverzeichnisse standardmäßig für alle anderen Benutzer lesbar sind.

Die Shell merkt sich standardmäßig die letzten 500 Kommandos; ändern können Sie das, indem Sie die gewünschte Zahl in die Variable `HISTSIZE` schreiben. Die Variable `HISTFILESIZE` gibt an, wie viele Kommandos in die `HISTFILE`-Datei geschrieben werden – normalerweise ebenfalls 500.

Außer über die Pfeiltasten können Sie die Vorgeschichte auch über »magische« Zeichenfolgen in neuen Kommandos ansprechen. Diese Zeichenfolgen ersetzt die Shell als allererstes, direkt nach dem Einlesen der Kommandozeile. Die Ersetzung erfolgt in zwei Schritten:

- Zunächst wird bestimmt, welches Kommando aus der Vorgeschichte zur Ersetzung herangezogen wird. Die Zeichenfolge `!!` steht dabei für das unmittelbar vorige Kommando, `!-`*n* bezieht sich auf das *n*-te Kommando vor dem aktuellen (`!-2` also zum Beispiel auf das vorletzte) und `!`*n* auf das Kommando mit der Nummer *n* in der Vorgeschichte. (Das Kommando `history` gibt die komplette Vorgeschichte mit Nummern aus.) `!xyz` selektiert das jüngste Kommando, das mit `xyz` anfängt, und `!?xyz` das jüngste Kommando, das `xyz` *enthält*.

- Anschließend wird entschieden, welcher Teil des ausgewählten Kommandos »recyclet« wird und wie. Wenn Sie nichts sagen, wird das komplette Kommando eingesetzt; ansonsten gibt es einige Auswahlmöglichkeiten. Alle diese Auswahlmöglichkeiten werden durch einen Doppelpunkt (»`:`«) von der Kommandoauswahl-Zeichenfolge getrennt.

  n Wählt das *n*-te Wort. Wort 0 ist dabei der Befehl selbst.

  `^` Wählt das erste Wort (direkt nach dem Befehl).

  `$` Wählt das letzte Wort.

  m`-`*n* Wählt die Wörter *m* bis *n*.

  n`*` Wählt alle Wörter ab dem *n*-ten.

  n`-` Wählt alle Wörter ab dem *n*-ten, bis auf das letzte.

  Einige Beispiele zur Verdeutlichung:

  `!-2:$` Wählt das letzte Wort des vorletzten Kommandos.

  `!!:0-` Wählt das komplette letzte Kommando bis auf das letzte.

  `!333^` Wählt das erste Wort aus Kommando 333.

  Das letzte Beispiel ist übrigens kein Tippfehler; wenn das erste Zeichen der Teilauswahl aus der Liste `^$*-%` kommt, darf der Doppelpunkt wegfallen. – Studieren Sie, wenn Sie mögen, die Bash-Dokumentation (Abschnitt HISTORY), um herauszufinden, was die Shell noch alles Interessantes auf Lager hat. Auswendig lernen müssen Sie das von uns (und dem LPI) aus nicht.

Die Vorgeschichte ist eine der Sachen, die die Bash von der C-Shell übernommen hat, und wer die 1980er Jahre nicht selbst miterlebt hat, kann sich möglicherweise nur schwer vorstellen, wie die Welt vor der Erfindung des interaktiven Kommandozeilen-Editierens ausgesehen haben mag. (Für Windows-Anwender liegt diese Zeit noch nicht *so* lange zurück.) Damals galt die Vorgeschichte mit all ihren `!`-Selektoren und Transformationen als die beste Idee seit der Erfindung des vorgeschnittenen Brots; heute hingegen übt die Dokumentation dieselbe Sorte morbide Faszination aus wie die Gebrauchsanleitung für eine Dampfmaschine aus Kaisers Zeit.

Noch ein paar Anmerkungen zum `history`-Befehl: Ein Aufruf wie

```
$ history 33
```

(mit einer Zahl als Parameter) gibt nur die letzten so vielen Zeilen der Vorgeschichte aus. »`history -c`« leert die Vorgeschichte. Es gibt noch einige andere Optionen; schauen Sie in die Bash-Dokumentation oder probieren Sie »`help history`«.

Tabelle 8.2: Tastaturkürzel innerhalb der Bash

Tastaturkürzel	Funktion
↑ bzw. ↓	durch frühere Kommandos blättern
Strg + r	Frühere Kommandos durchsuchen
← bzw. →	Cursor in Kommandozeile bewegen
Pos1 oder Strg + a	Cursor an Zeilenanfang setzen
Ende oder Strg + e	Cursor an Zeilenende setzen
⇐ bzw. Entf	Zeichen vor bzw. nach Cursor löschen
Strg + t	die beiden Zeichen vor/unter Cursor tauschen
Strg + l	Bildschirm löschen
Strg + c	Kommando abbrechen
Strg + d	Eingabe beenden (in der Login-Shell: Abmelden)

Autovervollständigung Eine massive Bequemlichkeit ist die Fähigkeit der Bash zur automatischen Komplettierung von Kommando- bzw. Dateinamen. Wenn Sie die Tab-Taste drücken, vervollständigt die Shell eine unvollständige Eingabe, sofern die Fortsetzung eindeutig identifiziert werden kann. Dazu werden für das erste Wort des Kommandos alle ausführbaren Programme und weiter hinten die im aktuellen bzw. angegebenen Verzeichnis befindlichen Dateien herangezogen. Existieren hierbei mehrere Dateien, deren Bezeichnungen gleich beginnen, vervollständigt die Shell die Eingabe so weit wie möglich und gibt durch ein akustisches Signal zu erkennen, dass der Datei- bzw. Befehlsname noch immer unvollendet sein kann. Ein erneutes Drücken von Tab listet dann die verbleibenden Möglichkeiten auf.

Komplettierung von Kommando- bzw. Dateinamen

Wenn Sie einen einzelnen (oder ein paar) Buchstaben eingeben und dann zweimal die Tab-Taste drücken, gibt die Shell eine Liste aller verfügbaren Befehle mit dem gewünschten Anfangsbuchstaben aus. Dies ist zum Beispiel sehr hilfreich, um sich die Schreibweise selten gebrauchter Befehle ins Gedächtnis zurück zu rufen. Der gleiche Effekt ist auch mit der Tastenkombination Esc ? zu erzielen. Soll ein Dateiname vervollständigt werden, kann dies mit Esc / erfolgen.

Befehlsliste

Es ist möglich, die Vervollständigung der Shell an bestimmte Programme anzupassen. Zum Beispiel könnte sie auf der Kommandozeile eines FTP-Programms statt Dateinamen die Namen bereits besuchter Rechner anbieten. Näheres steht in der Bash-Dokumentation.

Tabelle 8.2 gibt eine Übersicht über die in der Bash möglichen Tastaturkürzel.

Mehrere Kommandos auf einer Zeile Sie können durchaus mehrere Kommandos auf derselben Eingabezeile angeben. Sie müssen sie dazu nur mit dem Semikolon trennen:

```
$ echo Heute ist; date
Heute ist
Fr 5. Dez 12:12:47 CET 2008
```

In diesem Fall wird das zweite Kommando ausgeführt, sobald das erste fertig ist.

Bedingte Ausführung Manchmal nützlich ist es, die Ausführung des zweiten Kommandos davon abhängig zu machen, ob das erste korrekt ausgeführt wurde oder nicht. Jeder Unix-Prozess liefert einen **Rückgabewert**, der angibt, ob er korrekt ausgeführt wurde oder ob irgendwelche Fehler aufgetreten sind. Im ersteren Fall ist der Rückgabewert 0, im letzteren von 0 verschieden.

Rückgabewert

Sie können den Rückgabewert eines Kindprozesses Ihrer Shell herausfinden, indem Sie die Variable `$?` anschauen:

```
$ bash                                          Eine Kind-Shell starten …
$ exit 33                                       … und gleich wieder beenden
exit
$ echo $?
33                                              Der Wert aus unserem exit oben
$ _
```

Aber für das Folgende ist das eigentlich egal.

Mit && als »Trennzeichen« zwischen zwei Kommandos (da, wo sonst ein Semikolon stünde) wird das zweite Kommando nur dann ausgeführt, wenn das erste erfolgreich beendet wurde. Um Ihnen das zu demonstrieren, benutzen wir die `-c`-Option der Bash, mit der Sie der Kind-Shell ein Kommando auf der Kommandozeile übergeben können (beeindruckend, was?):

```
$ bash -c "exit 0" && echo "Erfolgreich"
Erfolgreich
$ bash -c "exit 33" && echo "Erfolgreich"
                                                Nichts - 33 ist kein Erfolg!
```

Umgekehrt wird mit || als »Trennzeichen« das zweite Kommando nur dann ausgeführt, wenn das erste *nicht* erfolgreich beendet wurde:

```
$ bash -c "exit 0" || echo "Nicht erfolgreich"
$ bash -c "exit 33" || echo "Nicht erfolgreich"
Nicht erfolgreich
```

Übungen

8.8 [3] Was ist das Problem mit dem Kommando »`echo "Hallo!"`«? (*Tipp:* Experimentieren Sie mit Kommandos der Form »`!-2`« oder »`!ls`«.)

8.5 Kommandos aus einer Datei

Sie können Shell-Kommandos in einer Datei ablegen und *en bloc* ausführen. (Wie Sie bequem eine Datei anlegen können, lernen Sie in Kapitel 5.) Dazu müssen Sie nur die Shell aufrufen und den Namen der Datei als Parameter übergeben:

```
$ bash meine-kommandos
```

Shellskript

Eine solche Datei bezeichnet man auch als **Shellskript**, und die Shell hat umfangreiche Möglichkeiten zur Programmierung, die wir hier nur grob umreißen können. (Die Linup-Front-Schulungsunterlage *Linux für Fortgeschrittene* erklärt die Programmierung von Shellskripten sehr ausführlich.)

Sie können sich das vorgesetzte `bash` sparen, indem Sie als erste Zeile der Skriptdatei die magische Anrufung

```
#!/bin/bash
```

einfügen und die Skriptdatei »ausführbar« machen:

```
$ chmod +x meine-kommandos
```

(Mehr über `chmod` und Zugriffsrechte finden Sie in Kapitel 12.) Anschließend genügt der Aufruf

```
$ ./meine-kommandos
```

Wenn Sie ein Shellskript wie oben gezeigt aufrufen – ob mit vorgesetztem `bash` oder als ausführbare Datei –, wird es in einer Sub-Shell ausgeführt, also einer Shell, die ein Kindprozess der aktuellen Shell ist. Das bedeutet, dass Änderungen zum Beispiel an Shell- oder Umgebungsvariablen die aktuelle Shell nicht beeinflussen. Nehmen wir einmal an, die Datei `zuweisung` enthält die Zeile

Sub-Shell

```
bla=fasel
```

Betrachten Sie die folgende Kommandosequenz:

```
$ bla=blubb
$ bash zuweisung                                   Enthält bla=fasel
$ echo $bla
blubb                          Keine Änderung; Zuweisung war nur in Sub-Shell
```

In der Regel wird das als Vorteil empfunden, aber hin und wieder wäre es schon wünschenswert, Kommandos aus einer Datei auf die *aktuelle* Shell wirken zu lassen. Auch das funktioniert: Das Kommando `source` liest die Zeilen einer Datei so ein, als ob Sie sie direkt in die aktuelle Shell tippen würden – alle Änderungen von Variablen (unter anderem) wirken also auf Ihre aktuelle Shell:

```
$ bla=blubb
$ source zuweisung                                 Enthält bla=fasel
$ echo $bla
fasel                                          Variable wurde geändert!
```

Ein anderer Name für das Kommando `source` ist übrigens ».« (Sie haben richtig gelesen – Punkt!) Statt

```
$ source zuweisung
```

funktioniert also auch

```
$ . zuweisung
```

Wie die Programmdateien für externe Kommandos werden auch die Dateien, die mit `source` bzw. . gelesen werden sollen, in den Verzeichnissen gesucht, die die Variable `PATH` angibt.

8.6 Die Shell als Programmiersprache

Shellkommandos aus einer Datei ausführen zu können ist zweifellos eine gute Sache. Noch besser ist es aber, diese Shellkommandos so zu strukturieren, dass sie nicht jedesmal dasselbe tun müssen, sondern zum Beispiel Parameter von der Kommandozeile lesen können. Die Vorteile liegen auf der Hand: Bei oft gebrauchten Vorgängen wird dröge Tipperei eingespart, und bei selten gebrauchten Vorgängen vermeiden Sie Fehler, die sich einschleichen können, weil Sie irgendeinen wichtigen Schritt versehentlich auslassen. Der Platz reicht hier nicht für eine komplette Erklärung der Shell als Programmiersprache, aber für ein paar kurze Beispiele ist zum Glück Raum.

Einzelne Parameter

Parameter von der Kommandozeile Die Parameter von der Kommandozeile eines Shellskript-Aufrufs stellt die Shell in den Variablen `$1`, `$2`, …zur Verfügung. Betrachten Sie das folgende Beispiel:

```
$ cat hallo
#!/bin/bash
echo Hallo $1, was machst Du $2?
$ ./hallo Hugo heute
Hallo Hugo, was machst Du heute?
$ ./hallo Susi morgen
Hallo Susi, was machst Du morgen?
```

Alle Parameter

Die Variable `$*` enthält alle Parameter auf einmal, und in `$#` steht die Anzahl der Parameter:

```
$ cat parameter
#!/bin/bash
echo $# Parameter: $*
$ ./parameter
0 Parameter:
$ ./parameter Hund
1 Parameter: Hund
$ ./parameter Hund Katze Maus Baum
4 Parameter: Hund Katze Maus Baum
```

Schleifen Mit dem Kommando `for` können Sie Schleifen konstruieren, die über eine Liste von (durch Freiplatz getrennten) Wörtern laufen:

```
$ for i in 1 2 3
> do
>   echo Und $i!
> done
Und 1!
Und 2!
Und 3!
```

Hierbei nimmt die Variable `i` nacheinander jeden der aufgelisteten Werte an. Jedesmal werden die Kommandos zwischen `do` und `done` ausgeführt.

Das Ganze macht mehr Spaß, wenn die Wörter aus einer Variablen kommen:

```
$ liste='4 5 6'
$ for i in $liste
> do
>   echo Und $i!
> done
Und 4!
Und 5!
Und 6!
```

Schleife über Parameter

Wenn Sie das »`in` …« weglassen, läuft die Schleife über die Parameter von der Kommandozeile:

```
$ cat sort-wc
#!/bin/bash
# Sortiere Dateien nach ihrer Zeilenzahl
for f
do
    echo `wc -l <"$f» Zeilen in $f
```

```
done | sort -n
$ ./sort-wc /etc/passwd /etc/fstab /etc/motd
```

(Das Kommando »`wc -l`« zählt die Zeilen seiner Standardeingabe oder der übergebenen Datei(en).) Beachten Sie, dass Sie die Standardausgabe der *Schleife* mit einer Pipe nach `sort` leiten können!

Fallunterscheidungen Sie können die weiter vorne gezeigten Operatoren `&&` und `||` benutzen, um bestimmte Kommandos nur unter gewissen Umständen auszuführen. Das Skript

```
#!/bin/bash
# grepcp REGEX
rm -rf backup; mkdir backup
for f in *.txt
do
    grep $1 "$f" && cp "$f" backup
done
```

zum Beispiel kopiert nur diejenigen Dateien ins Verzeichnis `backup`, deren Name auf `.txt` endet (dafür sorgt die `for`-Schleife) und die mindestens eine Zeile haben, auf die der reguläre Ausdruck passt, der als Parameter übergeben wurde.

Nützlich für Fallunterscheidungen ist das Kommando `test`, das eine große Auswahl von Bedingungen überprüfen kann. Es liefert den Rückgabewert 0 (Erfolg), wenn die Bedingung zutrifft, sonst einen von Null verschiedenen Rückgabewert (Misserfolg). Betrachten Sie zum Beispiel

`test`

```
#!/bin/bash
# filetest NAME1 NAME2 ...
for name
do
   test -d "$name" && echo $name: Verzeichnis
   test -f "$name" && echo $name: Datei
   test -L "$name" && echo $name: Symbolisches Link
done
```

Dieses Skript betrachtet eine Reihe von übergebenen Dateinamen und gibt für jeden aus, ob er für ein Verzeichnis, eine (normale) Datei oder ein symbolisches Link steht.

Das Kommando `test` existiert sowohl als freistehendes Programm in `/bin/test` als auch als eingebautes Kommando in der Bash und anderen Shells. Die verschiedenen Versionen können (vor allem bei exotischeren Tests) subtil voneinander abweichen. Lesen Sie gegebenenfalls in der Dokumentation nach.

Mit dem `if`-Kommando können Sie (bequem und leserlich) mehr als ein Kommando von einer Fallunterscheidung abhängig machen (Statt »`test …`« können Sie auch »`[ …]`« schreiben):

`if`

```
#!/bin/bash
# filetest2 NAME1 NAME2 ...
for name
do
    if [ -L "$name" ]
    then
        echo $name: Symbolisches Link
    elif [ -d "$name" ]
        echo $name: Verzeichnis
```

```
    elif [ -f "$name" ]
        echo $name: Datei
    else
        echo $name: Keine Ahnung
    fi
done
```

Wenn das Kommando nach dem `if` »Erfolg« meldet (Rückgabewert 0), werden die Kommandos nach `then` ausgeführt, bis zu einem `elif`, `else` oder `fi`. Liefert es hingegen »Misserfolg«, wird als nächstes testhalber das Kommando nach dem nächsten `elif` ausgeführt und dessen Rückgabewert betrachtet. Die Shell macht entsprechend weiter, bis das passende `fi` erreicht ist, wobei die Kommandos hinter dem `else` ausgeführt werden, wenn keines der `if`-Kommandos Erfolg vermelden konnte. Die `elif`- und `else`-Zweige dürfen wegfallen, wenn sie nicht gebraucht werden.

Mehr Schleifen Bei der `for`-Schleife liegt die Anzahl der Schleifendurchläufe von Anfang an fest (die Anzahl der Wörter in der Liste). Oft bekommt man es aber mit Situationen zu tun, wo nicht *a priori* klar ist, wie oft eine Schleife durchlaufen werden soll. Hierfür bietet die Shell die `while`-Schleife an, die (ähnlich wie `if`) ein Kommando ausführt, dessen Erfolg oder Misserfolg darüber entscheidet, wie mit der Schleife verfahren wird: Bei Erfolg werden die »abhängigen« Kommandos ausgeführt, bei Misserfolg wird nach der Schleife im Skript fortgefahren.

`while`

Das folgende Skript liest eine auf der Kommandozeile übergebene Datei der Form

```
Liebe Tante Frieda:frieda@example.net:den tollen Kaffeewärmer
Lieber Onkel Hans:hans@example.com:den schönen Fußball
◁◁◁◁◁
```

und konstruiert aus jeder Zeile eine Dankes-E-Mail (Linux ist halt schon sehr nützlich fürs wirkliche Leben):

```
#!/bin/bash
# birthday FILE
IFS=:
while read anrede adresse geschenk
do
    (echo $anrede!
     echo ""
     echo "Vielen Dank für $geschenk!"
     echo "Ich habe mich sehr darüber gefreut."
     echo ""
     echo "Viele Grüße,"
     echo "Dein Hugo") | mail -s "Vielen Dank!" $adresse
done <$1
```

`read` Das Kommando `read` liest dabei die Eingabedatei Zeile für Zeile und teilt jede Zeile an den Doppelpunkten (Variable `IFS`) in die drei Felder `anrede`, `adresse` und `geschenk` auf, die im Inneren der Schleife dann als Variable zur Verfügung stehen. Die Eingabeumleitung für die Schleife steht etwas konterintuitiv ganz am Ende.

Bitte testen Sie dieses Skript nur mit unverfänglichen E-Mail-Adressen!

Übungen

8.9 [1] Was ist der Unterschied (bei der Schleifenausführung) zwischen

```
for f; do …; done
```

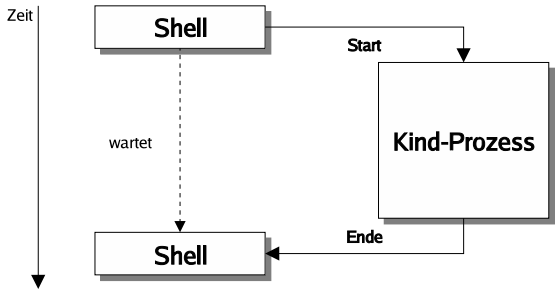


Bild 8.1: Die synchrone Arbeitsweise der Shell

und

```
for f in $*; do …; done
```

? (Probieren Sie es notfalls aus.)

8.10 [2] Warum benutzen wir im Skript `sort-wc` das Kommando

```
wc -l <$f
```

und nicht

```
wc -l $f
```

8.11 [2] Ändern Sie das `grepcp`-Skript so, dass es auch die Liste der zu betrachtenden Dateien von der Kommandozeile übernimmt. (*Tipp:* Das Shell-Kommando `shift` entfernt den ersten Kommandozeilenparameter aus `$` und läßt alle anderen um eine Position aufrücken. Nach einem `shift` ist das vormalige `$2` jetzt `$1`, `$3` ist `$2` und so weiter.)

8.12 [2] Warum liefert das Skript `filetest` für ein symbolisches Link die Ausgabe

```
$ ./filetest foo
foo: Datei
foo: Symbolisches Link
```

(statt nur einfach »`foo: Symbolisches Link`«)?

8.7 Vorder- und Hintergrundprozesse

Wird ein Kommando eingegeben und mit der ↩-Taste bestätigt, führt die Shell die erhaltene Anweisung aus. Interne Kommandos bearbeitet die Shell direkt; bei externen Kommandos erzeugt die Shell einen **Kindprozess**, der zur Ausführung dieses Befehls dient und sich anschließend wieder beendet. Ein Prozess unter Unix ist ein Programm, das läuft; dasselbe Programm kann mehrmals gleichzeitig (zum Beispiel von verschiedenen Benutzern) ausgeführt werden und korrespondiert dann mit mehreren Prozessen. Jeder Prozess kann Kindprozesse anlegen (auch wenn die meisten es – im Gegensatz zu Shells – nicht tun).

Kindprozess

Normalerweise wartet die Shell nun, bis der Kindprozess seine Arbeit erledigt und sich beendet hat. Das merken Sie daran, dass während der Laufzeit

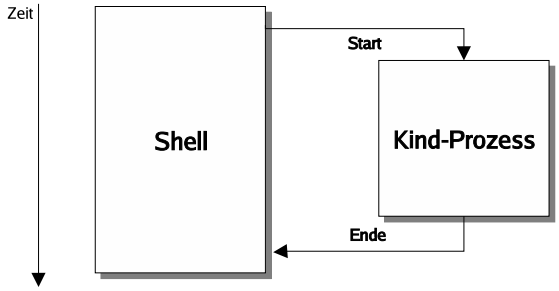


Bild 8.2: Die asynchrone Arbeitsweise der Shell

des Kindprozesses keine neue Eingabeaufforderung erscheint. Nach Beendigung
Rückgabewert des Kindprozesses wird dessen **Rückgabewert** gelesen und ausgewertet, und erst dann erscheint wieder ein Prompt. Die Ausführung der Shell wird quasi mit dem Kind-Prozess abgeglichen. Diese »synchrone« Arbeitsweise ist in Bild 8.1 verdeutlicht; als Benutzer sehen Sie zum Beispiel etwas wie

```
$ sleep 10
                                        Ca. 10 Sekunden lang passiert nichts
$ _
```

(eigentlich kennen Sie das schon).

Soll die Shell nicht warten, bis der Kind-Prozess seine Aufgabe erledigt hat, müssen Sie ihr dies durch das Anhängen eines kaufmännischen Und-Zeichens (`&`) an die Kommandozeile mitteilen. Während der Ableger dann im Hintergrund ab-
Hintergrund-Prozess gearbeitet wird, erscheint nach einer kurzen Meldung zum **Hintergrund-Prozess** sofort wieder die Eingabeaufforderung der Shell:

```
$ sleep 10 &
[2] 6210
                                        Und dann sofort wieder:
$ _
```

Diese Arbeitsweise heißt »asynchron«, da die Shell nicht untätig auf das Ende des Kindprozesses wartet (s. Bild 8.2).

Die Ausgabe »`[2] 6210`« bedeutet, dass das System den Prozess mit der Nummer (oder »Prozess-ID«) 6210 als »Job« Nr. 2 angelegt hat. Auf Ihrem System dürften diese Zahlen vom Beispiel hier abweichen.

Das `&` verhält sich syntaktisch in Wirklichkeit wie der Semikolon, kann also auch als Trennzeichen zwischen Kommandos dienen. Siehe hierzu Übung 8.14.

Hier ein paar Tipps für erfolgreiche Hintergrundverarbeitung:

- Der Hintergrundprozess sollte keine Tastatureingaben erwarten, da die Shell nicht entscheiden kann, welchem Prozess – Vordergrund oder Hintergrund – die Eingaben zuzuordnen sind. Gegebenenfalls kann der Hintergrundprozess seine Eingabe einer Datei entnehmen.
- Der Hintergrundprozess sollte keine Terminalausgaben machen, da sich diese mit den Ausgaben von Vordergrund-Aktivitäten überlagern können. Fehlermeldungen können bei Bedarf in eine Datei umgeleitet oder komplett verworfen werden.

Tabelle 8.3: Optionen für `jobs`

Option	Wirkung
`-l` (long)	zeigt zusätzlich die PID an
`-n` (notify)	zeigt nur Prozesse, die seit dem letzten Aufruf von `jobs` beendet worden sind
`-p` (process)	zeigt nur die PID an

- Wird der Eltern-Prozess abgebrochen, werden meist auch alle zugehörigen Kind-Prozesse (und demzufolge auch deren Kinder usw.) beendet. Davon ausgenommen sind nur Prozesse, die sich komplett von ihren Eltern lossagen, etwa weil sie Systemdienste im Hintergrund erbringen sollen.

Jobkontrolle

Befinden sich mehrere Prozesse im Hintergrund, verliert man leicht die Übersicht. Daher stellt die Bash ein eingebautes Kommando zur Verfügung, mit dem Sie sich über den Zustand von Hintergrundprozessen informieren können – `jobs`. Wird `jobs` ohne Zusätze eingegeben, erhalten Sie eine Liste mit der Jobnummer, dem Prozesszustand und dem Kommandotext. Das sieht dann etwa folgendermaßen aus:

```
$ jobs
[1]   Done                    sleep
$ _
```

In diesem Fall ist der Prozess mit der Jobnummer 1 bereits beendet worden (engl. *done*, erledigt), ansonsten sehen Sie die Meldung »`Running`« (laufend). Das Kommando `jobs` kennt verschiedene Optionen, wobei die wichtigsten in Tabelle 8.3 aufgeführt sind.

Die Bash ermöglicht es, einen Vordergrundprozess mit der Tastenkombination Strg+z anzuhalten. Dieser Prozess wird dann von `jobs` mit dem Status »`Stopped`« gekennzeichnet. Mit dem Befehl `bg` (engl. *background* = Hintergrund) lässt sich ein solcher Prozess in den Hintergrund schicken, wo er dann weiter bearbeitet wird (ansonsten bleibt er angehalten bis zum Sankt-Nimmerleins-Tag oder zum nächsten System-Neustart, was auch immer eher eintritt). So bugsiert die Eingabe »`bg %5`« den Prozess mit der Jobnummer 5 in den Hintergrund.

Umgekehrt können Sie aus mehreren Hintergrundprozessen einen auswählen und mit dem Shell-Kommando `fg` (engl. *foreground*, Vordergrund) in den Vordergrund holen. Die Syntax von `fg` entspricht vollständig der des Kommandos `bg`.

Beenden können Sie einen Vordergrundprozess in der Bash mit der Tastenkombination Strg+c, einen Hintergrundprozess direkt mit dem Kommando `kill`, wobei Sie wie bei `bg` ein Prozentzeichen gefolgt von der Jobnummer angeben.

Übungen

8.13 [2] Verwenden Sie ein geeignet spektakuläres Programm (etwa die OpenGL-Demonstration `glxgears` unter X11, alternativ vielleicht »`xclock -update 1`«), um mit Hintergrundprozessen und Jobkontrolle zu experimentieren. Vergewissern Sie sich, dass Sie Prozesse im Hintergrund starten können, dass Sie Vordergrundprozesse mit Strg+z anhalten und mit `bg` in den Hintergrund schicken können, dass `jobs` die Hintergrundprozesse auflistet und so weiter.

8.14 [3] Beschreiben (und erklären) Sie die Unterschiede zwischen den folgenden drei Kommandozeilen:

```
$  sleep 5 ; sleep 5
$  sleep 5 ; sleep 5 &
```

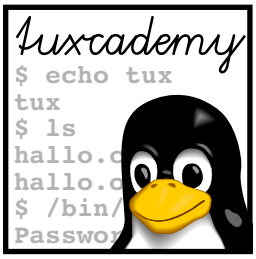
```
$ sleep 5 & sleep 5 &
```

Kommandos in diesem Kapitel

.	Liest eine Datei mit Shell-Kommandos so ein, als ob sie auf der Kommandozeile eingegeben worden wäre	bash(1)	139
bg	Lässt einen (angehaltenen) Prozess im Hintergrund weiterlaufen	bash(1)	145
date	Gibt Datum und Uhrzeit aus	date(1)	130
env	Gibt die Prozessumgebung aus oder startet Programme mit veränderter Umgebung	env(1)	132
export	Definiert und verwaltet Umgebungsvariable	bash(1)	131
fg	Holt einen Hintergrundprozess zurück in den Vordergrund	bash(1)	145
glxgears	Zeigt sich drehende Zahnräder unter X11, mit OpenGL	glxgears(1)	145
hash	Zeigt und verwaltet „gesehene“ Kommandos in der bash	bash(1)	134
history	Zeigt die zuletzt verwendeten bash-Kommandos an	bash(1)	136
jobs	Berichtet über Hintergrundprozesse	bash(1)	145
kill	Hält einen Hintergrundprozess an	bash(1), kill(1)	145
set	Verwaltet Shellvariable	bash(1)	132
source	Liest eine Datei mit Shell-Kommandos so ein, als ob sie auf der Kommandozeile eingegeben worden wäre	bash(1)	139
test	Wertet logische Ausdrücke auf der Kommandozeile aus	test(1), bash(1)	141
unset	Löscht Shell- oder Umgebungsvariable	bash(1)	132
whereis	Sucht ausführbare Programme, Handbuchseiten und Quellcode zu gegebenen Kommandos	whereis(1)	134
which	Sucht Programme in PATH	which(1)	134
xclock	Zeigt eine Uhr an	xclock(1x)	145

Zusammenfassung

- Das Kommando sleep wartet für die als Parameter angegebene Anzahl von Sekunden.
- Das Kommando echo gibt seine Argumente aus.
- Mit date lassen sich Datum und Uhrzeit ermitteln.
- Verschiedene Eigenschaften der Bash unterstützen das interaktive Arbeiten, etwa Kommando- und Dateinamenvervollständigung, Editieren der Kommandozeile, Aliasnamen und Variable.
- Externe Programme können auch asynchron, also im Hintergrund gestartet werden. Die Shell gibt dann sofort eine neue Eingabeaufforderung aus.



9

Das Dateisystem

Inhalt

9.1 Begriffe 148
9.2 Dateitypen 148
9.3 Der Linux-Verzeichnisbaum 150
9.4 Verzeichnisbaum und Dateisysteme 158
9.5 Wechselmedien 159

Lernziele

- Die Begriffe »Datei« und »Dateisystem« verstehen
- Die verschiedenen Dateitypen kennen
- Sich im Verzeichnisbaum eines Linux-Systems zurechtfinden
- Wissen, wie externe Dateisysteme in den Verzeichnisbaum eingebunden werden

Vorkenntnisse

- Linux-Grundkenntnisse (etwa aus den vorherigen Kapiteln)
- Umgang mit Dateien und Verzeichnissen (Kapitel 6)

9.1 Begriffe

Datei Der Begriff **Datei** steht ganz allgemein für eine abgeschlossene Ansammlung von Daten. Für die Art der enthaltenen Daten gibt es keine Einschränkungen; eine Datei kann ein Text sein, der nur wenige Buchstaben lang ist, aber auch ein viele Megabyte großes Archiv, das das gesamte Lebenswerk eines Anwenders umfaßt. Dateien müssen natürlich nicht unbedingt gewöhnlichen Text enthalten. Bilder, Klänge, ausführbare Programme und vieles andere mehr werden ebenfalls in Form von Dateien auf dem Datenträger abgelegt. Um herauszufinden, welche Art
file von Inhalt eine Datei hat, können Sie den Befehl `file` verwenden:

```
$ file /bin/ls /usr/bin/groups /etc/passwd
/bin/ls:         ELF 32-bit LSB executable, Intel 80386,▷
 ◁ version 1 (SYSV), for GNU/Linux 2.4.1,▷
 ◁ dynamically linked (uses shared libs), for GNU/Linux 2.4.1, stripped
/usr/bin/groups: Bourne shell script text executable
/etc/passwd:     ASCII text
```

`file` errät den Typ einer Datei auf der Basis von Regeln, die im Verzeichnis `/usr/share/file` stehen. `/usr/share/file/magic` enthält die Regeln im Klartext. Sie können eigene Regeln definieren, wenn Sie sie in `/etc/magic` ablegen. Näheres verrät `magic`(5).

Zum ordnungsgemäßen Betrieb benötigt ein Linux-System einige tausend verschiedene Dateien. Hinzu kommen noch die von den verschiedenen Benutzern angelegten »eigenen« Dateien.

Dateisystem Ein **Dateisystem** legt fest, nach welcher Methode die Daten auf den Datenträgern angeordnet und verwaltet werden. Auf der Platte liegen ja letzten Endes nur Bytes, die das System irgendwie wiederfinden können muss – und das möglichst effizient, flexibel und auch für sehr große Dateien. Die Details der Dateiverwaltung können unterschiedlich ausgelegt sein (Linux kennt zahlreiche verschiedene Dateisysteme, etwa `ext2`, `ext3`, `ext4`, ReiserFS, XFS, JFS, btrfs, …), aber die logische Sicht auf die Dateien, die die Benutzer zu sehen bekommen, ist im Großen und Ganzen dieselbe: eine baumartige Hierarchie von Datei- und Verzeichnisnamen mit Dateien unterschiedlicher Typen. (Siehe hierzu auch Kapitel 6.)

Der Begriff »Dateisystem« wird in der Linux-Szene in mehreren Bedeutungen verwendet. »Dateisystem« ist außer der in diesem Abschnitt eingeführten Bedeutung »Methode, Bytes auf einem Medium zu arrangieren« für manche Leute auch das, was wir als »Verzeichnisbaum« bezeichnen, außerdem nennt man ein konkretes Medium (Festplatte, USB-Stick, …) mitsamt den darauf befindlichen Daten »Dateisystem« – etwa in dem Sinn, dass man sagt, dass harte Links (Abschnitt 6.4.2) nicht »über Dateisystemgrenzen hinweg«, also nicht zwischen zwei verschiedenen Partitionen auf der Festplatte oder zwischen der Platte und einem USB-Stick, funktionieren.

9.2 Dateitypen

In Linux-Systemen gilt der Grundsatz: »Alles ist eine Datei«. Dies mag zunächst verwirrend scheinen, ist aber sehr nützlich. Prinzipiell können sechs Dateitypen unterschieden werden:

Normale Dateien (engl. *plain files*) Zu dieser Gruppe gehören Texte, Grafiken, Audiodaten etc., aber auch ausführbare Programme. Normale Dateien können mit den üblichen Werkzeugen (Editoren, `cat`, Shell-Ausgabeumlenkung, …) erzeugt werden.

Tabelle 9.1: Linux-Dateitypen

Typ	`ls -l`	`ls -F`	Anlegen mit …
Normale Datei	`-`	`name`	Diverse Programme
Verzeichnis	`d`	`name/`	`mkdir`
Symbolisches Link	`l`	`name@`	`ln -s`
Gerätedatei	`b` oder `c`	`name`	`mknod`
FIFO (*named pipe*)	`p`	`name|`	`mkfifo`
Unix-Domain-Socket	`s`	`name=`	kein Kommando

Verzeichnisse (engl. *directories)* Auch »Ordner« genannt; sie dienen, wie bereits beschrieben, zur Strukturierung des Speicherplatzes. Ein Verzeichnis ist im Prinzip eine Tabelle mit der Zuordnung von Dateinamen zu Inode-Nummern. Verzeichnisse werden mit `mkdir` angelegt.

Symbolische Links Enthalten eine Pfadangabe, die bei Verwendung des Links auf eine andere Datei verweist (ähnlich zu »Verknüpfungen« unter Windows). Siehe auch Abschnitt 6.4.2. Symbolische Links werden mit `ln -s` angelegt.

Gerätedateien (engl. *devices*) Diese Dateien entsprechen Schnittstellen zu beliebigen Geräten wie etwa Laufwerken. So repräsentiert etwa die Datei `/dev/fd0` das erste Diskettenlaufwerk. Jeder Schreib- oder Lesezugriff auf eine solche Datei wird an das zugehörige Gerät weitergeleitet. Gerätedateien werden mit dem Kommando `mknod` angelegt; dies ist normalerweise Territorium des Systemadministrators und wird in dieser Unterlage darum nicht weiter erklärt.

FIFOs Oft auch *named pipes* genannt. Sie erlauben ähnlich wie die Pipes der Shell (Kapitel 7) die direkte Kommunikation zwischen Programmen ohne Verwendung von Zwischendateien: Ein Prozess öffnet den FIFO zum Schreiben und ein anderer zum Lesen. Im Gegensatz zu den Pipes, die die Shell für ihre Pipelines benutzt und die sich zwar aus der Sicht von Programmen wie Dateien benehmen, aber »anonym« sind – sie existieren nicht im Dateisystem, sondern nur zwischen Prozessen, die in einem Verwandtschaftsverhältnis stehen –, haben FIFOs Dateinamen und können darum von beliebigen Programmen wie Dateien geöffnet werden. Außerdem können für FIFOs Zugriffsrechte gesetzt werden (für Pipes nicht). FIFOs werden mit dem Kommando `mkfifo` angelegt.

Unix-Domain-Sockets Ähnlich wie FIFOs sind Unix-Domain-Sockets ein Mittel zur Interprozesskommunikation. Sie verwenden im Wesentlichen dieselbe Programmierschnittstelle wie »echte« Netzwerkkommunikation über TCP/IP, aber funktionieren nur, wenn die Kommunikationspartner auf demselben Rechner laufen. Dafür sind Unix-Domain-Sockets beträchtlich effizienter als TCP/IP. Im Gegensatz zu FIFOs erlauben Unix-Domain-Sockets bidirektionale Kommunikation – beide beteiligten Programme können sowohl Daten lesen als auch schreiben. Unix-Domain-Sockets werden zum Beispiel vom Grafiksystem X11 verwendet, wenn X-Server und -Clients auf demselben Rechner laufen. – Zum Anlegen von Unix-Domain-Sockets gibt es kein spezielles Programm.

Übungen

9.1 [3] Suchen Sie in Ihrem System nach Beispielen für die verschiedenen Dateitypen. (Tabelle 9.1 zeigt Ihnen, woran Sie die betreffenden Dateien erkennen können.)

```
$ cd /
$ ls -l
insgesamt 125
drwxr-xr-x    2 root  root    4096 Dez 20 12:37 bin
drwxr-xr-x    2 root  root    4096 Jan 27 13:19 boot
lrwxrwxrwx    1 root  root      17 Dez 20 12:51 cdrecorder▷
                                                  ◁ -> /media/cdrecorder
lrwxrwxrwx    1 root  root      12 Dez 20 12:51 cdrom -> /media/cdrom
drwxr-xr-x   27 root  root   49152 Mär  4 07:49 dev
drwxr-xr-x   40 root  root    4096 Mär  4 09:16 etc
lrwxrwxrwx    1 root  root      13 Dez 20 12:51 floppy -> /media/floppy
drwxr-xr-x    6 root  root    4096 Dez 20 16:28 home
drwxr-xr-x    6 root  root    4096 Dez 20 12:36 lib
drwxr-xr-x    6 root  root    4096 Feb  2 12:43 media
drwxr-xr-x    2 root  root    4096 Mär 21  2002 mnt
drwxr-xr-x   14 root  root    4096 Mär  3 12:54 opt
dr-xr-xr-x   95 root  root       0 Mär  4 08:49 proc
drwx------   11 root  root    4096 Mär  3 16:09 root
drwxr-xr-x    4 root  root    4096 Dez 20 13:09 sbin
drwxr-xr-x    6 root  root    4096 Dez 20 12:36 srv
drwxrwxrwt   23 root  root    4096 Mär  4 10:45 tmp
drwxr-xr-x   13 root  root    4096 Dez 20 12:55 usr
drwxr-xr-x   17 root  root    4096 Dez 20 13:02 var
```

Bild 9.1: Inhalt des Wurzelverzeichnisses (SUSE)

9.3 Der Linux-Verzeichnisbaum

Ein Linux-System besteht oft aus Hunderttausenden von Dateien. Um einen Überblick zu behalten, gibt es gewisse Konventionen für die Verzeichnisstruktur und die Dateien, die ein Linux-System ausmachen, den *Filesystem Hierarchy Standard*,
FHS kurz »**FHS**«. Die meisten Distributionen halten sich an diesen Standard, allerdings sind kleinere Abweichungen möglich. Der FHS beschreibt alle Verzeichnisse der ersten Hierarchie-Ebene, außerdem definiert er eine zweite Ebene unterhalb von `/usr`.

Wurzelverzeichnis Der Verzeichnisbaum beginnt mit dem **Wurzelverzeichnis** »/«. (Zur Unterscheidung: es gibt auch ein Verzeichnis `/root` – das ist das Heimatverzeichnis des Benutzers `root`.) Das Wurzelverzeichnis enthält entweder nur Unterverzeichnisse oder aber zusätzlich, wenn kein spezielles Verzeichnis `/boot` existiert, den Betriebssystemkern.

Mit dem Befehl »`ls -la /`« können Sie sich im Wurzelverzeichnis `/` die Unterverzeichnisse auflisten lassen. Das Ergebnis sieht beispielsweise so aus wie in Bild 9.1. Die einzelnen Verzeichnisse folgen dem FHS und haben daher in allen Distributionen weitestgehend den gleichen Inhalt. Im folgenden werden die einzelnen Verzeichnisse genauer betrachtet.

Über den FHS besteht weithin Einigkeit, aber er ist genausowenig »verbindlich« wie irgend etwas bei Linux verbindlich ist. Zum einen gibt es sicherlich Linux-Systeme (etwa das auf Ihrer FRITZ!Box oder in Ihrem digitalen Videorecorder), die sowieso im wesentlichen nur der Hersteller anfasst und wo es nichts bringt, den FHS bis ins kleinste Detail einzuhalten. Zum Anderen können Sie auf Ihrem System natürlich machen, was Sie wollen, aber müssen gegebenenfalls die Konsequenzen tragen – Ihr Distributor sichert Ihnen zu, dass er sich an seinen Teil des FHS hält, aber erwartet auf der anderen Seite, dass Sie sich nicht beschweren, wenn Sie nicht 100% nach den Regeln spielen und dann Probleme auftauchen. Wenn Sie zum Beispiel ein Programm in `/usr/bin` installieren und die betreffende Datei beim

nächsten System-Upgrade überschrieben wird, sind Sie selber schuld, weil Sie laut FHS Ihre eigenen Programme nicht nach `/usr/bin` schreiben sollen (`/usr/local/bin` wäre richtig).

Der Betriebssystemkern – `/boot` Im Verzeichnis `/boot` liegt das Betriebssystem im engeren Sinne: `vmlinuz` ist der Kernel von Linux. Im Verzeichnis `/boot` finden sich außerdem Dateien, die für den Bootlader (meist GRUB) von Bedeutung sind.

Bei manchen Systemen befindet sich das Verzeichnis `/boot` auf einer separaten Partition. Das kann nötig sein, wenn das eigentliche Dateisystem verschlüsselt oder anderweitig für den Bootlader schwer zugänglich ist, etwa weil besondere Treiber für den Zugriff auf ein Hardware-RAID-System gebraucht werden.

Allgemeine Systemprogramme – `/bin` Unter `/bin` befinden sich die wichtigsten ausführbaren Programme (meist Systemprogramme), die unbedingt zum Starten des Systems notwendig sind. Dazu gehören z. B. `mount` und `mkdir`. Viele dieser Programme sind so elementar, dass sie nicht nur zum Starten, sondern auch während des Systembetriebs ständig gebraucht werden – etwa `ls` oder `grep`. In `/bin` stehen außerdem Programme, die nötig sind, um ein System wieder flott zu machen, bei dem nur das Dateisystem mit dem Wurzelverzeichnis zur Verfügung steht. Weitere Programme, die beim Start oder zur Reparatur nicht unbedingt gebraucht werden, finden sich auch unter `/usr/bin`.

Spezielle Systemprogramme – `/sbin` Ähnlich wie `/bin` enthält auch `/sbin` Programme, die zum Systemstart oder für Reparaturen nötig sind. Allerdings handelt es sich hierbei zum größten Teil um Systemkonfigurationswerkzeuge, die eigentlich nur `root` ausführen kann. »Normale« Benutzer können mit manchen dieser Programme Informationen abfragen, aber nichts ändern. Analog zu `/bin` gibt es auch ein Verzeichnis `/usr/sbin`, wo weitere systemnahe Programme zu finden sind.

Systembibliotheken – `/lib` Hier finden sich die *shared libraries* der Programme in `/bin` und `/sbin` in Form von Dateien und (symbolischen) Links. Shared Libraries sind Programmstücke, die von verschiedenen Programmen gebraucht werden. Solche gemeinsam verwendeten Bibliotheken sparen eine Menge Systemressourcen, da die meisten Prozesse teilweise gleiche Bestandteile haben und diese Bestandteile dann nur einmal geladen werden müssen; ferner ist es einfacher, Fehler in solchen Bibliotheken zu korrigieren, wenn es sie nur einmal im System gibt und alle Programme den betreffenden Programmcode aus einer zentralen Datei holen. Unter `/lib/modules` liegen übrigens auch die **Kernelmodule**, also Systemkern-Programmcode, der nicht notwendigerweise benötigt wird – Gerätetreiber, Dateisysteme, Netzwerkprotokolle und ähnliches. Diese Module können vom Systemkern bei Bedarf nachgeladen und prinzipiell nach Gebrauch auch wieder entfernt werden.

Kernelmodule

Gerätedateien – `/dev` In diesem Verzeichnis und seinen Unterverzeichnissen findet sich eine Unmenge von Einträgen für die Gerätedateien. **Gerätedateien** bilden die Schnittstelle von der Shell (oder allgemein dem Teil des Systems, den Benutzer auf der Kommandozeile oder als Programmierer zu sehen bekommen) zu den Gerätetreibern im Systemkern. Sie haben keinen »Inhalt« wie andere Dateien, sondern verweisen über »Gerätenummern« auf einen Treiber im Systemkern.

Gerätedateien

Früher war es üblich, dass Linux-Distributoren für jedes nur denkbare Gerät einen Dateieintrag in `/dev` machten. So hatte auch ein auf einem Notebook installiertes Linux-System die erforderlichen Gerätedateien für zehn Festplatten mit je 63 Partitionen, acht ISDN-Adapter, sechzehn serielle und vier parallele Schnittstellen und so weiter. Heutzutage geht der Trend weg von den übervollen `/dev`-Verzeichnissen mit einem Eintrag für jedes vorstellbare Gerät und hin zu enger an den laufenden Kernel gekoppelten Systemen, wo nur Einträge für tatsächlich existierende Geräte erscheinen. Das Stichwort

in diesem Zusammenhang heißt udev (kurz für *userspace* /dev) und wird in *Linux-Administration I* genauer besprochen.

zeichenorientierte Geräte
blockorientierte Geräte

Linux unterscheidet zwischen **zeichenorientierten Geräten** (engl. *character devices*) und **blockorientierten Geräten** (engl. *block devices*). Ein zeichenorientiertes Gerät ist beispielsweise ein Terminal, eine Maus oder ein Modem – ein Gerät, das einzelne Zeichen liefert oder verarbeitet. Ein blockorientiertes Gerät ist ein Gerät, das Daten blockweise behandelt – hierzu gehören zum Beispiel Festplatten oder Disketten, auf denen Sie Bytes nicht einzeln lesen oder schreiben können, sondern nur in Gruppen à 512 (oder so). Je nach ihrer Geschmacksrichtung sind die Gerätedateien in der Ausgabe von »ls -l« mit einen »c« oder einem »b« gekennzeichnet:

```
crw-rw-rw-  1  root  root  10,  4 Oct 16 11:11 amigamouse
brw-rw----  1  root  disk   8,  1 Oct 16 11:11 sda1
brw-rw----  1  root  disk   8,  2 Oct 16 11:11 sda2
crw-rw-rw-  1  root  root   1,  3 Oct 16 11:11 null
```

Treibernummer
Gerätenummer

Anstelle der Speichergröße stehen hier zwei Zahlen: Die erste ist die **Treibernummer** (engl. *major device number*). Sie kennzeichnet den Gerätetyp und legt fest, welcher Treiber im Kernel für die Verwaltung zuständig ist. So haben zum Beispiel alle SCSI-Festplatten traditionell die Treibernummer 8. Die zweite Zahl ist die **Gerätenummer** (engl. *minor device number*). Sie dient dem Treiber zur Unterscheidung verschiedener ähnlicher oder verwandter Geräte oder auch zur Kennzeichnung verschiedener Partitionen einer Platte.

Pseudogeräte

Erwähnenswert sind noch ein paar **Pseudogeräte**. Das *null device*, /dev/null, ist quasi ein Mülleimer für Ausgaben eines Programmes, die nicht gebraucht werden, aber irgendwohin geleitet werden müssen. Bei einem Aufruf wie

```
$ programm >/dev/null
```

wird die Standardausgabe, die sonst auf dem Terminal erscheinen würde, verworfen. Wird /dev/null gelesen, reagiert es wie eine leere Datei und liefert sofort das Dateiende. Für /dev/null müssen alle Benutzer Schreib- und Leserechte haben.

Die »Geräte« /dev/random bzw. /dev/urandom liefern zufällige Bytes in »kryptographischer« Qualität, die erzeugt werden, indem »Rauschen« im System gesammelt wird – etwa die Zeitabstände zwischen dem Eintreffen unvorhersehbarer Ereignisse wie Tastendrücke. Die Daten aus /dev/random eignen sich zur Erzeugung von Schlüsseln für gängige Verschlüsselungsverfahren. Die Datei /dev/zero liefert Nullbytes in beliebiger Menge; Sie können diese unter anderem zum Erzeugen und zum Überschreiben von Dateien mit dem Befehl dd verwenden.

Konfigurationsdateien – /etc Das Verzeichnis /etc ist sehr wichtig, denn hier befinden sich die Konfigurationsdateien für die allermeisten Programme. In /etc/inittab und /etc/init.d beispielsweise stehen die meisten der systemspezifischen Daten, die zum Starten von Systemdiensten erforderlich sind. Die wichtigsten Dateien werden hier etwas detaillierter erklärt. Mit wenigen Ausnahmen hat nur der Benutzer root Schreibrechte, aber jeder Benutzer Leserechte.

/etc/fstab Hier sind alle einhängbaren Dateisysteme mit ihren Eigenschaften (Typ, Zugriffsart, *mount point*) aufgelistet.

/etc/hosts Diese Datei ist eine der Konfigurationsdateien des TCP/IP-Netzwerks. Hier werden die Namen der Netzwerkrechner ihren IP-Adressen zugeordnet. In kleinen Netzwerken und bei Einzelrechnern kann diese Datei einen Name-Server ersetzen.

/etc/inittab Die Datei /etc/inittab ist die Konfigurationsdatei für das init-Programm und damit für den Systemstart.

/etc/init.d In diesem Verzeichnis liegen die »Init-Skripte« für die verschiedenen Systemdienste. Mit ihnen werden beim Systemstart und beim Herunterfahren die Dienste gestartet bzw. gestoppt.

Bei den Red-Hat-Distributionen heißt dieses Verzeichnis `/etc/rc.d/init.d`.

/etc/issue In der Datei `/etc/issue` steht der Begrüßungstext, der vor der Aufforderung zum Anmelden ausgegeben wird. Nach der Installation eines neuen Systems wird in diesem Text meistens der Name des Herstellers präsentiert.

/etc/motd Hier steht die »Nachricht des Tages« (engl. *message of the day*), die nach einer erfolgreichen Anmeldung automatisch auf dem Bildschirm erscheint, noch bevor die Shell die erste Eingabeaufforderung ausgibt. Diese Datei kann der Systemadministrator verwenden, um aktuelle Informationen und Neuigkeiten weiterzugeben[1].

/etc/mtab Dies ist eine Liste aller eingehängten Dateisysteme inklusive ihrer *mount points*. `/etc/mtab` unterscheidet sich von `/etc/fstab` darin, dass in `/etc/mtab` alle aktuell eingehängten Dateisysteme aufgezählt sind, während in `/etc/fstab` nur Voreinstellungen und Optionen dafür stehen, welche Dateisysteme wie eingehängt werden *können* – typischerweise beim Systemstart, aber auch später. Sie können natürlich über die Kommandozeile beliebige Dateisysteme einhängen können, wo Sie wollen, und das wird hier auch protokolliert.

Eigentlich gehört es sich nicht, diese Sorte Information in `/etc` abzulegen, wo die Dateien prinzipiell statisch sein sollen. Hier hat offensichtlich die Tradition die Oberhand gewonnen.

/etc/passwd In `/etc/passwd` findet sich eine Liste aller dem System bekannten Benutzer zusammen mit diversen anderen benutzerspezifischen Informationen. In modernen Systemen befinden sich übrigens trotz des Namens dieser Datei die Kennwörter nicht hier, sondern in der Datei `/etc/shadow`. Jene Datei ist für normale Benutzer nicht lesbar.

Zubehör – /opt Dieses Verzeichnis ist eigentlich dafür gedacht, dass Drittanbieter fertige Softwarepakete anbieten können, die sich installieren lassen sollen, ohne mit den Dateien einer Linux-Distribution oder den lokal angelegten Dateien zu kollidieren. Solche Softwarepakete belegen ein Unterverzeichnis `/opt/`⟨*Paketname*⟩. Von Rechts wegen sollte dieses Verzeichnis unmittelbar nach der Installation einer Distribution auf einer neuen Platte also völlig leer sein.

»Unveränderliche Dateien« – /usr In `/usr` finden sich in diversen Unterverzeichnissen Programme und Dateien, die nicht für den Systemstart oder zur Systemreparatur notwendig oder anderweitig unverzichtbar sind. Die wichtigsten Verzeichnisse sind:

/usr/bin Systemprogramme, die nicht für den Systemstart gebraucht werden und/oder anderweitig nicht so wichtig sind.

/usr/sbin Weitere Systemprogramme für `root`.

/usr/lib Weitere, nicht von Programmen in `/bin` oder `/sbin` benötigte Bibliotheken.

/usr/local Verzeichnis für Dateien, die der lokale Systemadministrator installiert hat. Entspricht von der Idee her dem `/opt`-Verzeichnis – die Distribution darf hier nichts ablegen.

[1] Man sagt, dass die einzige Gemeinsamkeit aller Unix-Systeme der Welt die *message of the day* ist, die darauf hinweist, dass die Platten zu 98% voll sind und die Benutzer überflüssige Dateien entsorgen mögen.

/usr/share Daten, die von der Rechnerarchitektur unabhängig sind. Im Prinzip könnte ein Linux-Netz, das z. B. aus Intel-, SPARC- und PowerPC-Rechnern besteht, sich eine gemeinsame Kopie von `/usr/share` auf einem zentralen Rechner teilen. Heute ist Plattenplatz aber so billig, dass keine Distribution sich die Mühe macht, das tatsächlich zu implementieren.

/usr/share/doc Dokumentation – zum Beispiel HOWTOs

/usr/share/info Info-Seiten

/usr/share/man Handbuchseiten (in Unterverzeichnissen)

/usr/src Quellcode für den Kernel und weitere Programme (sofern vorhanden)

Der Name `/usr` wird häufig als »*Unix system resources*« interpretiert, was aber historisch nicht stimmt: Ursprünglich stammt dieses Verzeichnis aus der Zeit, als in einem Rechner eine kleine schnelle und eine große langsame Festplatte zur Verfügung stand. Auf die kleine Festplatte kamen alle häufig verwendeten Programme und Dateien, auf die große langsame (unter `/usr` eingehängt) große Programme und Dateien, die nicht recht auf die kleine Platte passten, oder solche, die nicht so oft benötigt wurden. Heute können Sie die Trennung anders ausnutzen: Wenn Sie es geschickt anstellen, können Sie `/usr` auf eine eigene Partition legen und diese schreibgeschützt in den Verzeichnisbaum einbinden. Es ist grundsätzlich sogar möglich, `/usr` von einem zentralen Server zu importieren, und so auf Arbeitsplatzrechnern Plattenplatz zu sparen und die Wartung zu vereinfachen (nur der zentrale Server muss gegebenenfalls aktualisiert werden), auch wenn die gesunkenen Plattenpreise das heute nicht mehr nötig machen. Die gängigen Linux-Distributionen unterstützen es sowieso nicht.

`/usr` schreibgeschützt

Das Fenster zum Kernel – `/proc` Das ist mit das interessanteste Verzeichnis und auch eines der wichtigsten. `/proc` ist eigentlich ein Pseudo-Dateisystem. Es belegt keinen Platz auf der Festplatte, sondern die Verzeichnisse und Dateien werden vom Systemkern erzeugt, wenn sich jemand für ihren Inhalt interessiert. Hier finden Sie sämtliche Informationen zu den laufenden Prozessen und außerdem weitere Informationen, die der Kernel über die Hardware des Rechners besitzt. In einigen Dateien finden Sie zum Beispiel eine komplette Hardwareanalyse. Die wichtigsten Dateien sind kurz aufgeführt:

Pseudo-Dateisystem

/proc/cpuinfo Hier sind Informationen über den Typ und die Taktfrequenz der CPU enthalten.

/proc/devices Hier findet sich eine vollständige Liste der Geräte, die vom Kernel unterstützt werden mit deren Gerätenummern. Beim Erstellen der Gerätedateien wird auf diese Datei zugegriffen.

/proc/dma Eine Liste der belegten DMA-Kanäle. Ist auf heutigen PCI-basierten Systemen nicht mehr fürchterlich interessant oder wichtig.

/proc/interrupts Eine Liste aller belegten Hardwareinterrupts. Interruptnummer, Anzahl der bisher ausgelösten Interrupts und die Bezeichnung der möglichen auslösenden Geräte sind angegeben. (Ein Interrupt taucht in dieser Liste nur auf, wenn er wirklich von einem Treiber im Kernel beansprucht wird.)

/proc/ioports Ähnlich wie `/proc/interrupts`, aber für I/O-Ports.

/proc/kcore Diese Datei fällt ins Auge wegen ihrer Größe. Sie ist der Zugang zum gesamten Arbeitsspeicher des Rechners, quasi ein Abbild des RAM, und wird zum Debugging des Systemkerns gebraucht. Diese Datei kann nur mit `root`-Privilegien gelesen werden. Am besten lassen Sie die Finger davon!

/proc/loadavg Diese Datei gibt drei Zahlen aus, die ein Maß für die Auslastung der CPU innerhalb der letzten 1, 5 und 15 Minuten sind. Diese Zahlen werden normalerweise vom Programm `uptime` ausgegeben.

/proc/meminfo Zeigt die Speicherauslastung und die Auslastung des Swap-Bereichs an. Diese Datei wird vom Kommando `free` benutzt.

/proc/mounts Wieder eine Liste aller aktuell gemounteten Dateisysteme, weitestgehend identisch mit `/etc/mtab`.

/proc/scsi In diesem Verzeichnis findet man eine Datei mit dem Namen `scsi`, in der die erkannten Geräte aufgelistet sind. Dann gibt es ein Unterverzeichnis für jeden Typ von SCSI-Hostadapter im System, in dem in einer Datei `0` (bzw. `1`, `2` usw. bei mehreren gleichen Adaptern) Informationen über den Adapter gespeichert sind.

/proc/version Enthält die Versionsnummer und das Übersetzungsdatum des laufenden Kernels.

Früher, bevor es `/proc` gab, mussten Programme wie das Prozessstatus-Anzeigeprogramm `ps`, die Systeminformationen aus dem Kern liefern, einiges über die internen Datenstrukturen des Linux-Kerns wissen und brauchten auch entsprechende Zugriffsrechte, um die betreffenden Daten aus dem laufenden Kern zu lesen. Da die Datenstrukturen sich im Zuge der Linux-Weiterentwicklung öfters änderten, war es oft nötig, zu einer neuen Kern-Version auch neue Versionen dieser Dienstprogramme zu installieren, die an die Änderungen angepasst waren. Das `/proc`-Dateisystem liefert eine Abstraktionsschicht zwischen diesen internen Datenstrukturen und den Dienstprogrammen: Heuer müssen Sie nur noch sicherstellen, dass bei einer Änderung einer internen Datenstruktur das Format der Dateien in `/proc` gleich bleibt – und `ps` & Co. funktionieren weiter wie gehabt.

Hardwaresteuerung – /sys Dieses Verzeichnis finden Sie im Linux-Kernel ab der Version 2.6. Es wird ähnlich wie `/proc` vom Kernel selbst nach Bedarf zur Verfügung gestellt und erlaubt in einer umfangreichen Hierarchie von Unterverzeichnissen eine konsistente Sicht auf die verfügbare Hardware sowie diverse Steuerungseingriffe.

Tendenziell sollen alle Einträge von `/proc`, die nichts mit einzelnen Prozessen zu tun haben, allmählich nach `/sys` wandern. Allerdings wissen die Götter, wann dieses strategische Ziel erreicht sein wird.

Veränderliche Dateien – /var Hier befinden sich veränderliche Dateien, verteilt auf verschiedene Verzeichnisse. Beim Aufruf verschiedener Programme generiert der Benutzer, meist ohne sich dessen im Detail bewusst zu sein, Daten. Das geschieht etwa beim Aufruf von `man`, bei dem komprimierte Hilfedaten entpackt werden (und die entpackten Versionen eine Weile aufgehoben werden, nur falls sie gleich wieder gebraucht werden). Ähnliches erfolgt, wenn gedruckt werden soll: Der Druckauftrag muss zwischengespeichert werden, zum Beispiel unter `/var/spool/cups`. Unter `/var/log` werden An- und Abmeldevorgänge sowie weitere Systemereignisse protokolliert (die Log-Dateien), unter `/var/spool/cron` stehen Informationen für das regelmäßige zeitgesteuerte Starten von Kommandos, und ungelesene elektronische Post der Benutzer steht in `/var/mail`.

Log-Dateien

Nur damit Sie es mal gehört haben (es könnte in der Prüfung vorkommen): Das Systemprotokoll wird auf Linux-Rechnern normalerweise über den »Syslog«-Dienst geregelt. Ein Programm namens `syslogd` nimmt Nachrichten von anderen Programmen entgegen und sortiert diese anhand ihrer Herkunft und Priorität (von »Debugginghilfe« über »Fehler« bis hin zu »Katastrophe, System schmiert gerade ab«) in Dateien in `/var/log` ein, wo

Sie sie dann finden können. Außer in Dateien kann der Syslog-Dienst seine Nachrichten auch anderswohin schicken, etwa auf die Konsole oder über das Netz an einen anderen Rechner, der als zentrale Management-Station arbeitet und alle Protokollnachrichten aus Ihrem Rechenzentrum konsolidiert.

Außer dem `syslogd` gibt es bei manchen Linux-Distributionen auch noch einen `klogd`-Dienst. Seine Aufgabe ist es, Nachrichten vom Betriebssystemkern entgegenzunehmen und an den `syslogd` weiterzureichen. Andere Distributionen kommen ohne einen separaten `klogd` aus, weil deren `syslogd` diese Aufgabe selbst übernehmen kann.

Der Linux-Betriebssystemkern spuckt schon jede Menge Nachrichten aus, bevor das System überhaupt so weit ist, dass `syslogd` (und gegebenenfalls `klogd`) läuft und diese Nachrichten tatsächlich entgegennehmen kann. Da die Nachrichten trotzdem wichtig sein können, speichert der Linux-Kern sie intern ab, und Sie können mit dem Kommando `dmesg` darauf zugreifen.

Vergängliche Daten – `/tmp` Viele Dienstprogramme brauchen temporären Speicherplatz, zum Beispiel manche Editoren oder `sort`. In `/tmp` können beliebige Programme temporäre Dateien ablegen. Viele Distributionen löschen beim Booten zumindest wahlweise alle Dateien unterhalb dieses Verzeichnisses; Sie sollten dort also nichts unterbringen, was Ihnen dauerhaft wichtig ist.

Nach Konvention wird `/tmp` beim Booten geleert und `/var/tmp` nicht. Ob Ihre Distribution das auch so macht, sollten Sie bei Gelegenheit prüfen.

Serverdateien – `/srv` Hier finden Sie Dateien, die von verschiedenen Dienstprogrammen angeboten werden, etwa:

```
drwxr-xr-x    2 root     root         4096 Sep 13 01:14 ftp
drwxr-xr-x    5 root     root         4096 Sep  9 23:00 www
```

Dieses Verzeichnis ist eine relativ neue Erfindung, und es ist durchaus möglich, dass es auf Ihrem System noch nicht existiert. Leider gibt es keinen guten anderen Platz für Web-Seiten, ein FTP-Angebot oder ähnliches, über den man sich einig werden konnte (letzten Endes ja der Grund für die Einführung von `/srv`), so dass auf einem System ohne `/srv` diese Daten irgendwo ganz anders liegen können, etwa in Unterverzeichnissen von `/usr/local` oder `/var`.

Zugriff auf CD-ROMs und Disketten – `/media` Dieses Verzeichnis wird oftmals automatisch angelegt; es enthält weitere, leere Verzeichnisse, etwa `/media/cdrom` und `/media/floppy`, die als Mountpunkt für CD-ROMs und Floppies dienen. Je nach Ihrer Ausstattung mit Wechselmedien sollten Sie sich keinen Zwang antun, neue Verzeichnisse wie `/media/dvd` anzulegen, wenn diese als Mountpoints sinnvoll und nicht von Ihrem Distributionshersteller vorgesehen sind.

Zugriff auf andere Datenträger – `/mnt` Dieses ebenfalls leere Verzeichnis dient zum kurzfristigen Einbinden weiterer Dateisysteme. Bei manchen Distributionen, etwa denen von Red Hat, können sich hier (und nicht in `/media`) Verzeichnisse als Mountpunkte für CD-ROM, Floppy, … befinden.

Heimatverzeichnisse für Benutzer – `/home` Unter diesem Eintrag befinden sich die Heimatverzeichnisse aller Benutzer mit Ausnahme von `root`, der ein eigenes Verzeichnis hat.

Wenn Sie mehr als ein paar hundert Benutzer haben, ist es aus Datenschutz- und Effizienzgründen sinnvoll, die Heimatverzeichnisse nicht alle als unmittelbare Kinder von `/home` zu führen. Sie können in so einem Fall zum

Tabelle 9.2: Zuordnung einiger Verzeichnisse zum FHS-Schema

	statisch	dynamisch
lokal	`/etc`, `/bin`, `/sbin`, `/lib`	`/dev`, `/var/log`
entfernt	`/usr`, `/opt`	`/home`, `/var/mail`

Beispiel die primäre Gruppe der Benutzer als weiteres Unterscheidungskriterium zu Rate ziehen:

```
/home/support/hugo
/home/entwickl/emil
◁◁◁◁◁
```

Heimatverzeichnis des Administrators – `/root` Hierbei handelt es sich um ein ganz normales Heimatverzeichnis ähnlich denen der übrigen Benutzer. Der entscheidende Unterschied ist, dass es nicht im Ordner `/home`, sondern im Wurzelverzeichnis (`/`) liegt.

Der Grund dafür ist, dass das Verzeichnis `/home` oftmals auf einem Dateisystem auf einer separaten Partition oder Festplatte liegt, aber es soll sichergestellt sein, dass `root` auch in seiner gewohnten Umgebung arbeiten kann, wenn dieses separate Dateisystem einmal nicht angesprochen werden kann.

Das virtuelle Fundbüro – `lost+found` (Nur bei `ext`-Dateisystemen; nicht vom FHS vorgeschrieben.) Hier werden Dateien gespeichert, die ein Dateisystemcheck nach einem Systemabsturz auf der Platte findet und die zwar vernünftig aussehen, aber in keinem Verzeichnis zu stehen scheinen. Das Prüfprogramm legt in `lost+found` auf demselben Dateisystem Links auf solche Dateien an, damit der Systemverwalter sich anschauen kann, wo die Datei in Wirklichkeit hingehören könnte; `lost+found` wird schon »auf Vorrat« bereitgestellt, damit das Prüfprogramm es an einer festen Stelle finden kann (es hat nach Konvention auf den `ext`-Dateisystemen immer die Inode-Nummer 11).

Eine andere Motivation für die Verzeichnisanordnung ist wie folgt: Der FHS unterteilt Dateien und Verzeichnisse grob nach zwei Kriterien – Müssen sie lokal verfügbar sein oder können sie auch über das Netz von einem anderen Rechner bezogen werden, und sind ihre Inhalte statisch (Veränderung nur durch Intervention des Administrators) oder ändern sie sich im laufenden Betrieb? (Tabelle 9.2)

Die Idee hinter dieser Einteilung ist es, die Pflege des Systems zu vereinfachen: Verzeichnisse können leicht auf Datei-Server ausgelagert und damit zentral administriert werden. Verzeichnisse, die keine dynamischen Daten enthalten, können nur zum Lesen eingebunden werden und sind damit absturzresistenter.

Übungen

9.2 [1] Wie viele Programme enthält Ihr System an den »gängigen« Plätzen?

9.3 [2] Wird `grep` mit mehr als einem Dateinamen als Parameter aufgerufen, gibt es vor jeder passenden Zeile den Namen der betreffenden Datei aus. Dies ist möglicherweise ein Problem, wenn man `grep` mit einem Shell-Dateisuchmuster (etwa »`*.txt`«) aufruft, da das genaue Format der `grep`-Ausgabe so nicht vorhersehbar ist und Programme weiter hinten in einer Pipeline deswegen durcheinander kommen können. Wie können Sie die Ausgabe des Dateinamens erzwingen, selbst wenn das Suchmuster nur zu einem einzigen Dateinamen expandiert? (*Tipp:* Eine dafür nützliche »Datei« steht im Verzeichnis `/dev`.)

9.4 [3] Das Kommando »`cp bla.txt /dev/null`« tut im Wesentlichen nichts, aber das Kommando »`mv bla.txt /dev/null`« – entsprechende Zugriffsrechte vorausgesetzt – ersetzt `/dev/null` durch `bla.txt`. Warum?

9.5 [2] Welche Softwarepakete stehen auf Ihrem System unter `/opt`? Welche davon stammen aus der Distribution und welche von Drittanbietern? Sollte eine Distribution eine »Schnupperversion« eines Drittanbieter-Programms unter `/opt` installieren oder anderswo? Was halten Sie davon?

9.6 [1] Warum ist es unklug, Sicherheitskopien des Verzeichnisses `/proc` anzulegen?

9.4 Verzeichnisbaum und Dateisysteme

Der Verzeichnisbaum eines Linux-Systems erstreckt sich normalerweise über mehr als eine Partition auf der Festplatte, und auch Wechselmedien wie CD-ROMs, USB-Sticks und ähnliches sowie tragbare MP3-Player, Digitalkameras und ähnliche für einen Computer wie Speichermedien aussehende Geräte müssen berücksichtigt werden. Wenn Sie Microsoft Windows ein bisschen kennen, dann wissen Sie vielleicht, dass dieses Problem dort so gelöst wird, dass die verschiedenen »Laufwerke« über Buchstaben identifiziert werden – bei Linux dagegen werden alle verfügbaren Plattenpartitionen und Medien in den Verzeichnisbaum eingegliedert, der bei »/« anfängt.

Partitionierung

Grundsätzlich spricht nichts Gravierendes dagegen, ein Linux-System komplett auf einer einzigen Plattenpartition zu installieren. Es ist allerdings gängig, zumindest das Verzeichnis `/home` – in dem die Heimatverzeichnisse der Benutzer liegen – auf eine eigene Partition zu tun. Dies hat den Vorteil, dass Sie das eigentliche Betriebssystem, die Linux-Distribution, komplett neu installieren können, ohne um die Sicherheit Ihrer eigenen Daten fürchten zu müssen (Sie müssen nur im richtigen Moment aufpassen, nämlich wenn Sie in der Installationsroutine die Zielpartition(en) für die Installation auswählen). Auch die Erstellung von Sicherheitskopien wird so vereinfacht.

Serversysteme

Auf größeren Serversystemen ist es auch Usus, anderen Verzeichnissen, typischerweise `/tmp`, `/var/tmp` oder `/var/spool`, eigene Partitionen zuzuteilen. Das Ziel ist dabei, dass Benutzer nicht den Systembetrieb dadurch stören können sollen, dass sie wichtige Partitionen komplett mit Daten füllen. Wenn zum Beispiel `/var` voll ist, können keine Protokolldaten mehr geschrieben werden, also möchte man vermeiden, dass Benutzer das Dateisystem mit Unmengen ungelesener Mail, ungedruckten Druckjobs oder riesigen Dateien in `/var/tmp` blockieren. Allerdings wird das System durch so viele Partitionen auch unübersichtlich und unflexibel.

Mehr über die Partitionierung und Strategien dafür finden Sie in der LinupFront-Schulungsunterlage *Linux-Administration I*.

`/etc/fstab`

Die Datei `/etc/fstab` beschreibt die Zusammensetzung des Systems aus verschiedenen Partitionen. Beim Systemstart wird dafür gesorgt, dass die unterschiedlichen Dateisysteme an den richtigen Stellen »eingebunden« – der Linux-Insider sagt »gemountet«, vom englischen *to mount* – werden, worum Sie als einfacher Benutzer sich nicht kümmern müssen. Was Sie aber möglicherweise interessiert, ist, wie Sie an den Inhalt Ihrer CD-ROMs und USB-Sticks kommen, und auch diese müssen Sie einhängen. Wir tun also gut daran, uns kurz mit dem Thema zu beschäftigen, auch wenn es eigentlich Administratoren-Territorium ist.

Zum Einhängen eines Mediums benötigen Sie ausser dem Namen der Gerätedatei des Mediums (in der Regel ein blockorientiertes Gerät wie `/dev/sda1`) ein Verzeichnis irgendwo im Verzeichnisbaum, wo der Inhalt des Mediums erscheinen soll – den sogenannten *mount point*. Dabei kann es sich um jedes beliebige Verzeichnis handeln.


Das Verzeichnis muss dabei nicht einmal leer sein, allerdings können Sie auf den ursprünglichen Verzeichnisinhalt nicht mehr zugreifen, wenn Sie einen Datenträger »darübergemountet« haben. (Der Inhalt erscheint wieder, wenn Sie den Datenträger wieder aushängen.)


Grundsätzlich könnte jemand ein Wechselmedium über ein wichtiges Systemverzeichnis wie `/etc` mounten (idealerweise mit einer Datei namens `passwd`, die einen `root`-Eintrag ohne Kennwort enthält). Aus diesem Grund ist das Einhängen von Dateisystemen an beliebigen Stellen im Dateisystem mit Recht dem Systemverwalter vorbehalten, der keinen Bedarf für solche Sperenzchen haben dürfte; er ist ja schon `root`.


Die »Gerätedatei für das Medium« haben wir weiter oben `/dev/sda1` genannt. Dies ist eigentlich die erste Partition auf der ersten SCSI-Festplatte im System – der Name kann völlig anders lauten, je nachdem was Sie mit welcher Sorte Medium vor haben. Es ist aber ein naheliegender Name für USB-Sticks, die vom System aus technischen Gründen so behandelt werden, als wären sie SCSI-Geräte.

Mit diesen Informationen – Gerätename und *mount point* – kann ein Systemadministrator das Medium etwa wie folgt einbinden:

```
# mount /dev/sda1 /media/usb
```

Eine Datei namens `datei` auf dem Medium würde dann als `/media/usb/datei` im Verzeichnisbaum in Erscheinung treten. Mit einem Kommando wie

```
# umount /media/usb                                   Achtung: kein »n«
```

kann der Administrator diese Einbindung auch wieder lösen.

9.5 Wechselmedien

Das explizite Einbinden von Wechselmedien ist ein mühsames Geschäft, und das explizite Lösen einer Einbindung vor dem Entfernen noch viel mehr – dabei kann gerade letzteres zu Ärger führen, wenn Sie das Medium physikalisch entfernen, bevor Linux komplett damit fertig ist. Linux versucht ja, das System dadurch zu beschleunigen, dass es langsame Vorgänge wie das Schreiben von Daten auf Medien nicht sofort erledigt, sondern irgendwann später, wenn der »richtige Moment« da ist, und wenn Sie natürlich Ihren USB-Stick herausziehen, bevor die Daten wirklich dort gelandet sind, dann haben Sie im besten Fall nichts gewonnen, und im schlimmsten Fall sind die anderen Daten dort im Chaos versunken.

Als Benutzer einer grafischen Arbeitsumgebung auf einem modernen Linux-System haben Sie es leicht: Wenn Sie ein Medium einlegen oder anstöpseln – egal ob Audio-CD, USB-Stick oder Digitalkamera –, erscheint ein Dialog, der Ihnen diverse interessante Aktionen anbietet, die Sie mit dem Medium ausführen wollen könnten. »Einbinden« ist normalerweise eine davon, und das System überlegt sich auch einen schönen *mount point* für Sie. Entfernen können Sie das Medium später ebenso bequem über ein Sinnbild auf dem Bildschirmhintergrund oder in der Kontrollleiste der Arbeitsumgebung. Hierüber müssen wir in dieser Unterlage nicht im Detail reden.

Anders sieht es aus, wenn Sie auf der Kommandozeile arbeiten. Dort müssen Sie Wechselmedien nämlich wie im vorigen Abschnitt skizziert explizit ein- und aushängen. Als »normaler Benutzer« dürfen Sie das aber, wie erwähnt, nicht für beliebige Medien an beliebigen Stellen, sondern nur für solche Medientypen, die Ihr Systemverwalter dafür vorgesehen und auch schon mit »vorgekochten« *mount points* versehen hat. Sie erkennen diese daran, dass sie in der Datei `/etc/fstab` mit der Option `user` oder `users` gekennzeichnet sind:

```
$ grep user /etc/fstab
/dev/hdb        /media/cdrom0   udf,iso9660 ro,user,noauto 0 0
/dev/sda1       /media/usb      auto    user,noauto         0 0
/dev/mmcblk0p1  /media/sd       auto    user,noauto         0 0
```

Für die Details der Einträge in `/etc/fstab` müssen wir Sie auf die Linup-Front-Schulungsunterlage *Linux-Administration I* vertrösten (O. K., `fstab`(5) geht auch, aber unsere Unterlage ist netter); für uns hier und heute soll reichen, dass in diesem Beispiel drei Sorten von Wechselmedien in Frage kommen, nämlich CD-ROMs (der erste Eintrag), USB-basierte Medien wie USB-Sticks, Digitalkameras oder MP3-Player (der zweite Eintrag), und SD-Karten (der dritte Eintrag). Als »normaler Benutzer« müssen Sie sich an die vorgegebenen *mount points* halten und können (nachdem Sie das betreffende Medium platziert haben) Dinge sagen wie

```
$ mount /dev/hdb                    für die CD-ROM
$ mount /media/cdrom0                       dito
$ mount /dev/sda1                   für den USB-Stick
$ mount /media/sd                   für die SD-Karte
```

Das heißt, Linux erwartet *entweder* den Gerätenamen *oder* den *mount point*; das jeweilige Pendant dazu ergibt sich aus der Datei `/etc/fstab`. Aushängen mit `umount` erfolgt analog.

Die Option `user` in `/etc/fstab` ist dafür verantwortlich, dass das funktioniert (sie bewirkt auch noch ein paar andere Dinge, die wir hier nicht im Detail besprechen müssen). Die Option `users` tut grundsätzlich dasselbe; der Unterschied zwischen den beiden – und merken Sie sich das, es könnte in der Prüfung vorkommen – besteht darin, dass bei `user` nur derjenige Benutzer das Dateisystem wieder *aushängen* darf, der es ursprünglich eingehängt hat. Bei `users` darf es jeder Benutzer (!). (Und `root` darf es sowieso immer.)

Übungen

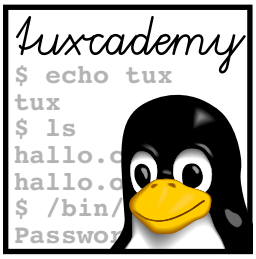
9.7 [1] Legen Sie eine Diskette ins Laufwerk, hängen Sie sie ein, kopieren Sie eine Datei (zum Beispiel `/etc/passwd`) auf die Diskette und hängen Sie die Diskette dann wieder aus. (Sollten Sie ein fortschrittliches System vor sich haben, das keine Disketten mehr annimmt, dann machen Sie dasselbe mit einem USB-Stick oder anderem geeigneten Wechselmedium.)

Kommandos in diesem Kapitel

dmesg	Gibt den Inhalt des Kernel-Nachrichtenpuffers aus	dmesg(8)	156
file	Rät den Typ einer Datei anhand des Inhalts	file(1)	148
free	Zeigt die Speicherauslastung und die Auslastung des Swap-Bereichs an	free(1)	155
klogd	Akzeptiert Protokollnachrichten des Systemkerns	klogd(8)	156
mkfifo	Legt FIFOs (benannte Pipes) an	mkfifo(1)	149
mknod	Legt Gerätedateien an	mknod(1)	149
syslogd	Bearbeitet Systemprotokoll-Meldungen	syslogd(8)	156
uptime	Gibt die Zeit seit dem letzten Systemstart sowie die CPU-Auslastung aus	uptime(1)	154

Zusammenfassung

- Dateien sind abgeschlossene Ansammlungen von Daten, die unter einem Namen gespeichert sind. Linux verwendet die Abstraktion »Datei« auch für Geräte und andere Objekte.
- Die Methode der Anordnung von Daten und Verwaltungsinformationen auf einem Speichermedium nennt man Dateisystem. Derselbe Begriff bezeichnet die gesamte baumartige Hierarchie von Verzeichnissen und Dateien im System oder ein konkretes Medium mit den darauf befindlichen Daten.
- Linux-Dateisysteme enthalten normale Dateien, Verzeichnisse, symbolische Links, Gerätedateien (zwei Sorten), FIFOs und Unix-Domain-Sockets.
- Der *Filesystem Hierarchy Standard* (FHS) beschreibt die Bedeutung der wichtigsten Verzeichnisse in einem Linux-System und wird von den meisten Distributionen eingehalten.
- Wechselmedien müssen vor Gebrauch in den Linux-Verzeichnisbaum eingehängt und nachher wieder ausgehängt werden. Hierzu dienen die Kommandos `mount` und `umount`. Grafische Arbeitsumgebungen machen das mitunter bequemer.



10

Systemadministration

Inhalt

10.1 Administration allgemein 164
10.2 Das privilegierte root-Konto 164
10.3 Administratorprivilegien erlangen 166
10.4 Distributionsabhängige Administrationswerkzeuge 169

Lernziele

- Die Aufgaben eines Systemadministrators einordnen können
- Sich als Administrator anmelden können
- Vor- und Nachteile von (grafischen) Administrationswerkzeugen einschätzen können

Vorkenntnisse

- Grundlegende Linux-Kenntnisse
- Kenntnisse der Administration anderer Betriebssysteme sind hilfreich

10.1 Administration allgemein

Als reiner Benutzer eines Linux-Systems haben Sie es gut: Sie setzen sich an Ihren Rechner, alles ist passend konfiguriert, die ganze Hardware wird unterstützt und funktioniert, Sie müssen sich um nichts kümmern, denn Ihnen steht ein Systemadministrator zu Diensten, der alle Systemarbeiten prompt und umfassend für Sie erledigt (jedenfalls wünschen wir Ihnen das).

Administrator-Aufgabe

Sollten Sie selbst die Position des Systemadministrators innehaben oder anstreben – in der Firma oder auch im privaten Bereich –, dann haben Sie Ihre Arbeit damit schon vorgezeichnet: Sie müssen das System installieren, konfigurieren und allfällige Zusatzgeräte anschließen. Ist das erledigt, müssen Sie das System am Laufen halten, etwa indem Sie das Systemprotokoll auf ungewöhnliche Vorkommnisse überprüfen, regelmäßig alte Protokolldateien entsorgen, Sicherheitskopien anfertigen, neue Software installieren und existierende aktualisieren und so weiter.

Änderungen

Die Systeminstallation ist heute im Zeitalter der Distributionen mit luxuriösen Installationsprogrammen meist kein Hexenwerk mehr. Ein ambitionierter Administrator kann aber sehr viel Zeit damit verbringen, auf seinem System noch die letzten Ressourcen zu mobilisieren. Im Grunde genommen fallen Aufgaben der Systemverwaltung aber nur dann an, wenn eine größere Änderung auftritt, zum Beispiel neue Hard- oder Software integriert werden soll, neue Benutzer hinzukommen oder alte verschwinden, oder Hardwareprobleme auftreten.

Werkzeuge

Viele Linux-Distributionen enthalten spezielle Werkzeuge, die die Systemverwaltung vereinfachen sollen. Diese Werkzeuge erledigen verschiedene Aufgaben von der Benutzerverwaltung über das Anlegen von Dateisystemen bis zum Update. Diese Hilfsprogramme können die Aufgabe sehr erleichtern, manchmal aber auch erschweren. Standardvorgänge werden vereinfacht, aber bei speziellen Einstellungen sollten Sie die genauen Zusammenhänge kennen. Die meisten dieser Werkzeuge stehen außerdem nur für bestimmte Distributionen zur Verfügung.

Verantwortungsgefühl

Die Verwaltung eines Linux-Systems wie jedes anderen Rechnersystems erfordert ein beträchtliches Maß an Verantwortungsgefühl und Sorgfalt. Sie sollten sich nicht als Halbgott (mindestens) verstehen, sondern als Dienstleister. Egal, ob Sie der einzige Systemverwalter sind – etwa auf Ihrem eigenen Rechner – oder in einem Team von Kollegen zusammenarbeiten, um das Netz einer Firma zu unterstützen: Kommunikation ist das A und O. Sie sollten sich angewöhnen, Konfigurationsänderungen und andere Administrationsentscheidungen zu dokumentieren, damit Sie sie später noch nachvollziehen können. Die unter Linux übliche Konfiguration über direkt zu editierende Textdateien macht dies bequem möglich, da Sie die Konfigurationseinstellungen an Ort und Stelle mit Kommentaren versehen können (ein Luxus, den grafisch orientierte Administrationswerkzeuge Ihnen normalerweise nicht gestatten). Machen Sie davon Gebrauch.

Kommunikation

10.2 Das privilegierte `root`-Konto

Der Systemadministrator braucht für viele seiner Aufgaben besondere Rechte. Entsprechend steht ihm ein besonderes Benutzerkonto mit dem Benutzernamen `root` zur Verfügung. Als `root` ist der Administrator der sogenannte **Superuser**. Kurz gesagt: Er darf alles.

Superuser

Uneingeschränkte Privilegien

Die üblichen Dateiberechtigungen und andere Sicherheitsvorkehrungen haben für `root` keine Bedeutung. Er hat Privilegien, die es ihm erlauben, nahezu uneingeschränkt auf alle Daten, Geräte und Komponenten im System zuzugreifen. Damit kann er Veränderungen am System vornehmen, die allen normalen Benutzern von den Sicherheitsmechanismen des Linux-Kernels verboten werden. Das heißt, dass Sie als `root` jede Datei im System verändern können, unabhängig davon, wem sie gehört. Während normale Benutzer keinen Schaden anrichten können (etwa

durch Zerstörung von Dateisystemen oder Manipulationen an den Dateien anderer Benutzer), kennt root keine solchen Einschränkungen.

Diese umfassende Privilegierung des Systemadministrators ist in vielen Anwendungsfällen eher ein Problem. Zwar ist es zum Beispiel notwendig, bei der Anfertigung einer Sicherheitskopie alle Dateien im System zu lesen. Das heißt aber noch lange nicht, dass jemand, der die Sicherheitskopien machen soll (vielleicht ein Werkstudent) auch befugt sein sollte, alle Dateien im System mit einem Texteditor aufzurufen, zu lesen oder zu ändern – oder einen potenziell weltweit zugänglichen Netzwerkdienst zu starten. Es gibt verschiedene Ansätze, die Systemadministratorrechte nur unter kontrollierten Bedingungen einzuräumen (zum Beispiel sudo, ein System, das normalen Benutzern die Ausführung gezielt freigegebener Kommandos mit Administratorrechten erlaubt), gezielt einzelnen Prozessen besondere Privilegien zu vergeben, statt nach dem Prinzip »Alles oder nichts« vorzugehen (Stichwort POSIX Capabilities), oder die Idee eines »allmächtigen« Administrators völlig abzuschaffen (SELinux – *security-enhanced Linux* –, eine frei verfügbare Entwicklung des amerikanischen Geheimdiensts NSA, zum Beispiel enthält ein »rollenbasiertes« Zugriffsrechtesystem, das ohne allmächtigen Administrator auskommen kann).

sudo

POSIX Capabilities

SELinux

Warum enthält Linux überhaupt Vorkehrungen für die Sicherheit des Systems? Der wichtigste Grund ist der, dass Benutzer die Möglichkeit haben sollen zu entscheiden, welcher Zugriff auf ihre eigenen Dateien bestehen soll. Durch das Setzen der Berechtigungsbits (mit dem Befehl chmod) können Benutzer festlegen, dass bestimmte Dateien von bestimmten anderen (oder auch gar keinen) Benutzern gelesen, beschrieben oder ausgeführt werden dürfen. Das sichert die Vertraulichkeit und die Integrität der Daten. Sie wären sicherlich nicht damit einverstanden, dass andere Benutzer Ihr privates Postfach lesen oder hinter Ihrem Rücken den Quellcode eines wichtigen Programms verändern.

Warum Sicherheitsvorkehrungen?

Die Sicherheitsmechanismen sollen auch verhindern, dass Benutzer das System beschädigen. Der Zugriff auf viele der Gerätedateien in /dev, die den Hardwarekomponenten wie etwa den Festplatten entsprechen, wird vom System eingeschränkt. Wenn normale Benutzer direkt auf die Festplatten zugreifen könnten, bestünde die Gefahr, dass alle möglichen Arten von Schäden angerichtet würden (etwa indem der komplette Inhalt einer Festplatte überschrieben wird, oder sich jemand mit Kenntnissen der Dateisystemorganisation Zugriff zu Dateien verschafft, die ihn nichts angehen). Statt dessen zwingt das System normale Benutzer, die Laufwerke über das Dateisystem anzusprechen und sorgt auf diese Weise für den Schutz der Daten.

Zugriffsbeschränkung auf Geräte

Es ist wichtig, festzuhalten, dass solche Schäden in der Regel nicht absichtlich hervorgerufen werden. Die Sicherheitsvorkehrungen des Systems dienen in erster Linie dazu, die Benutzer vor unbeabsichtigten Fehlern und Missverständnissen zu bewahren; erst in zweiter Linie ist es ihr Zweck, die »Privatsphäre« der Benutzer und Daten zu garantieren.

Die Benutzer können auf dem System zu **Gruppen** zusammengefasst werden, für die Sie ebenfalls Zugriffsrechte vergeben können. So könnte etwa ein Team von Programmentwicklern schreibenden und lesenden Zugriff auf eine Reihe von Dateien bekommen, während andere Benutzer diese Dateien nicht verändern können. Jeder Benutzer legt für seine persönlichen Dateien selbst fest, wie öffentlich oder privat der Zugriff geregelt sein soll.

Gruppen

Die Sicherheitsmechanismen verhindern auch, dass normale Benutzer bestimmte Aktionen durchführen können; etwa den Aufruf bestimmter Systemroutinen (engl. *system calls*) aus einem Programm heraus. Es gibt beispielsweise einen Systemaufruf, der das System zum Stillstand bringt, und der von Programmen wie shutdown ausgeführt wird, wenn das System neu gebootet werden soll. Wenn normale Benutzer die Möglichkeit hätten, in ihren Programmen diese Routine aufzurufen, könnten sie versehentlich (oder absichtlich) das System jederzeit anhalten.

Privilegierte Systemroutinen

Der Administrator muss diese Sicherheitsmechanismen meist umgehen, um das System zu pflegen oder aktualisierte Versionen von Programmen einzuspielen zu können. Genau dafür ist das root-Konto gedacht. Ein guter Administrator kann seine Arbeit erledigen, ohne sich um die üblichen Zugriffsrechte und andere Einschränkungen zu kümmern zu müssen, weil diese für root nicht bestehen. Das root-Konto ist nicht etwa besser als ein normales Benutzerkonto, weil es mehr Rechte hat. Die Einschränkung dieser Rechte ist eine Sicherheitsmaßnahme. Weil diese sinnvollen und hilfreichen Schutz- und Sicherheitsmechanismen des Betriebssystems für den Administrator nicht zur Verfügung stehen, ist das Arbeiten mit root-Rechten sehr riskant. Sie sollten deshalb nur diejenigen Befehle als root ausführen, die wirklich die Privilegien benötigen.

Die Sicherheitsprobleme, die andere populäre Betriebssysteme haben, lassen sich in weiten Teilen darauf zurückführen, dass normale Benutzer oft mit Administratorprivilegien agieren und versehentlich aufgerufene Programme wie »Würmer« oder »trojanische Pferde« es leicht haben, sich zum Beispiel als alternative Einwahlprogramme im System zu etablieren. Bei einem korrekt installierten und betriebenen Linux-System ist so etwas kaum möglich, da Benutzer ihre elektronische Post ohne Administratorrechte lesen, für Eingriffe in die Systemkonfiguration wie die Installation eines neuen Einwahlprogramms Administratorrechte aber zwingend erforderlich sind.

Linux ist natürlich nicht prinzipiell gefeit gegen Schädlinge wie »Mail-Würmer«; jemand könnte ein Mailprogramm schreiben und populär machen, das ähnlich wie manche solchen Programme bei anderen Betriebssystemen »aktive Inhalte«, also Skripte oder Binärprogramme in Nachrichten, direkt etwa durch Anklicken ausführt. Ein so eingeschlepptes »bösartiges« Programm könnte unter Linux alle Dateien des Aufrufers löschen oder versuchen, »trojanischen« Code in dessen Arbeitsumgebung unterzubringen, aber es könnte weder andere Benutzer noch das System selbst direkt in Mitleidenschaft ziehen – es sei denn, es nutzt eine Sicherheitslücke in Linux aus, mit der ein lokaler Benutzer »durch die Hintertür« Administratorrechte bekommen kann (solche Lücken werden hin und wieder bekannt, und es werden umgehend Korrekturen veröffentlicht, die Sie natürlich zeitnah installieren sollten).

Übungen

10.1 [2] Was unterscheidet Benutzer und Administrator? Nennen Sie Beispiele für Aufgaben und Tätigkeiten (und entsprechende Befehle), wie sie unter einem Benutzer- bzw. unter dem root-Konto durchgeführt werden!

10.2 [!1] Warum sollten Sie als normaler Benutzer nicht auf dem root-Konto arbeiten?

10.3 [1] Wie sieht es mit der Rechtevergabe bei Ihrem Rechner zu Hause aus? Arbeiten Sie mit Administratorprivilegien?

10.3 Administratorprivilegien erlangen

Um Administratorprivilegien zu bekommen, gibt es zwei Wege:

1. Sie können sich direkt als Benutzer root anmelden und erhalten nach Eingabe des korrekten root-Kennwortes eine Shell mit root-Rechten. Sie sollten es allerdings vermeiden, sich am grafischen Login als root anzumelden, da dann sämtliche grafische Anwendungen inklusive X-Server mit root-Rechten laufen würden, was nicht notwendig ist und zu Sicherheitslücken führen kann. Auch über das Netz sollte ein direktes Anmelden als root nicht erlaubt sein.

Von welchen Terminals aus eine direkte root-Anmeldung zugelassen wird, können Sie über die Datei /etc/securetty bestimmen. Voreinstellung ist meistens »alle virtuellen Konsolen und /dev/ttyS0« (letzteres für Benutzer der »seriellen Konsole«).

2. Sie können mit dem Kommando su aus einer normalen Shell heraus eine neue Shell mit root-Rechten bekommen. su fragt wie das login-Programm nach einem Kennwort und öffnet die root-Shell erst, nachdem das korrekte root-Kennwort eingegeben wurde. In grafischen Umgebungen wie KDE gibt es äquivalente Methoden.

(Siehe hierzu auch *Linux-Grundlagen für Anwender und Administratoren*).

Auch auf Einzelplatzsystemen

Selbst wenn ein Linux-System nur von einer einzigen Person benutzt wird, ist es ratsam, diesen einzigen Benutzer als normalen Benutzer einzutragen. Beim alltäglichen Arbeiten unter root-Rechten lassen sich die meisten Sicherheitsvorkehrungen des Kernels leicht umgehen. Auf diese Weise können schnell Fehler entstehen, die das gesamte System betreffen. Sie können diese Gefahr vermeiden, indem Sie sich unter Ihrem normalen Benutzerkonto anmelden und bei Bedarf vorübergehend mit »/bin/su -« eine Shell mit root-Rechten starten.

Mit su können Sie auch die Identität eines beliebigen anderen Benutzers (hier hugo) annehmen, indem Sie es als

```
$ /bin/su - hugo
```

aufrufen. Dafür müssen Sie das Kennwort des Zielbenutzers kennen, es sei denn, Sie rufen su als root auf.

Auch aus einem weiteren Grund ist die zweite Methode der ersten vorzuziehen. Wenn Sie den Befehl su verwenden, um root zu werden, nachdem Sie sich unter Ihrem eigenen Namen angemeldet haben, dann erzeugt die Eingabe von su einen Eintrag wie:

```
Apr  1 08:18:21 RECHNER su: (to root) user1 on /dev/tty2
```

Protokoll

in den Protokolldateien (etwa in /var/log/messages). Dieser Eintrag besagt, dass der Benutzer user1 erfolgreich einen su-Befehl als root auf Terminal 2 abgesetzt hat. Wenn Sie sich dagegen direkt als root anmelden, wird kein solcher Eintrag in den Logdateien erzeugt; es ließe sich also nicht nachvollziehen, welcher Benutzer mit dem root-Konto herumgespielt hat. Auf einem System mit mehreren Administratoren ist es oft wichtig, herauszufinden, wer wann den Befehl su eingegeben hat.

Ubuntu ist eine der »neumodischen« Distributionen, bei denen ein Anmelden als root verpönt – und in der Standardeinstellung sogar ausgeschlossen – ist. Statt dessen können bestimmte Benutzer über den sudo-Mechanismus Kommandos mit Administratorrechten ausführen. Bei der Installation werden Sie aufgefordert, ein »normales« Benutzerkonto anzulegen, und dieses Benutzerkonto erhält quasi von selbst »indirekte« Administratorprivilegien.

Bei Debian GNU/Linux können Sie wahlweise bei der Installation ein Kennwort für das Konto root vergeben und so eine direkte Anmeldung als Administrator ermöglichen, oder darauf verzichten und wie bei Ubuntu dem ersten, bei der Installation angelegten Benutzer Administratorprivilegien via sudo zukommen lassen.

Eingabeaufforderungen

Auf vielen Systemen unterscheiden sich die Eingabeaufforderungen für root und die übrigen Benutzer. Die klassische Eingabeaufforderung für root enthält ein Doppelkreuz (#), während die Eingabeaufforderungen für die anderen Benutzer ein Dollarzeichen ($) oder ein Größerzeichen (>) enthalten. Die #-Eingabeaufforderung soll Sie daran erinnern, dass Sie gerade root mit allen Rechten sind. Allerdings kann die Eingabeaufforderung leicht geändert werden, und es ist Ihre Entscheidung, ob Sie dieser Konvention folgen wollen oder nicht.


Wenn Sie `sudo` verwenden, dann bekommen Sie natürlich nie eine Eingabeaufforderung für `root` zu sehen.

Missbrauch von `root`

Wie alle mächtigen Werkzeuge kann auch das `root`-Konto missbraucht werden. Deswegen ist es für Sie als Systemadministrator wichtig, das `root`-Kennwort geheimzuhalten. Es sollte nur an solche Benutzer, denen sowohl in fachlicher als auch persönlicher Hinsicht vertraut wird (oder die für ihre Handlungen verantwortlich gemacht werden können), weitergegeben werden. Wenn Sie der einzige Benutzer des Systems sind, betrifft Sie dieses Problem natürlich nicht.

Administration allein oder zu mehreren

Viele Köche verderben den Brei! Auch für die Systemadministration gilt dieses Prinzip. Der größte Vorteil einer alleinigen Nutzung des `root`-Kontos liegt nicht so sehr darin, dass die Möglichkeiten des Missbrauchs minimiert werden (obwohl das sicherlich eine Folge davon ist). Wichtiger ist die Tatsache, dass `root` als einziger Nutzer des `root`-Kontos die gesamte Systemkonfiguration kennt. Wenn außer dem Administrator jemand die Möglichkeit hat, zum Beispiel wichtige Systemdateien zu verändern, dann könnte ohne Wissen des Administrators die Konfiguration des Systems geändert werden, und dieser ist dann nicht mehr auf dem aktuellen Wissensstand, soweit es die Arbeitsweise des Systems betrifft. Im Firmenumfeld ist es aus verschiedenen Gründen – etwa der Sicherung des Systembetriebs zu Urlaubszeiten oder im Falle plötzlicher schwerer Krankheit des Administrators – notwendig, mehrere entsprechend privilegierte Mitarbeiter zu haben; hier ist dann enge Zusammenarbeit und Tuchfühlung angesagt.

Wenn es nur einen Systemverwalter gibt, der für die Konfiguration des Systems verantwortlich ist, haben Sie immer die Gewähr, dass eine Person wirklich weiß, was auf dem System los ist (zumindest theoretisch), und auch die Frage der Verantwortlichkeit ist geklärt. Je mehr Benutzer als `root` arbeiten, um so größer ist die Wahrscheinlichkeit, dass irgendwann jemand unter dem `root`-Konto einen Fehler macht. Auch wenn alle Benutzer, die das `root`-Kennwort kennen, die entsprechende Fachkenntnis besitzen, kann doch jedem einmal ein Fehler unterlaufen. Umsicht und gründliche Ausbildung sind die einzigen Mittel gegen Unfälle.

Verantwortlichkeit


Für die Systemadministration im Team gibt es auch noch ein paar andere nützliche Werkzeuge. Debian GNU/Linux und Ubuntu unterstützen zum Beispiel ein Paket namens `etckeeper`, das es erlaubt, den kompletten Inhalt des `/etc`-Verzeichnisses in einem Revisionskontrollsystem wie Git oder Mercurial abzulegen. Revisionskontrollsysteme (auf die wir hier nicht im Detail eingehen können) ermöglichen es, detailliert Änderungen an einzelnen Dateien in einer Verzeichnishierarchie zu verfolgen, zu kommentieren und gegebenenfalls auch rückgängig zu machen. Es ist mit den Mitteln von Git oder Mercurial sogar möglich, eine Kopie des `/etc`-Verzeichnisses auf einem ganz anderen Rechner abzulegen und automatisch synchron zu halten – ideal als Schutz gegen Unfälle.

Übungen


10.4 [2] Welche Möglichkeiten gibt es, um Administratorrechte zu bekommen? Welche Möglichkeit ist die bessere? Warum?


10.5 [!2] Woran lässt sich bei einem normal konfigurierten System während des Betriebs erkennen, ob Sie unter dem `root`-Konto arbeiten?


10.6 [2] Melden Sie sich als normaler Benutzer (etwa `test`) an. Wechseln Sie zu `root` und wechseln Sie wieder zurück zum Benutzer `test`. Wie arbeiten Sie am zweckmäßigsten, wenn Sie oft zwischen beiden Konten wechseln müssen (etwa um die Auswirkungen gerade gemachter Einstellungen zu kontrollieren)?


10.7 [!2] Melden Sie sich als normaler Benutzer an und wechseln Sie mit `su` auf `root`. Wo finden Sie einen Eintrag, der diesen Wechsel dokumentiert? Schauen Sie sich diese Meldung an!

10.4 Distributionsabhängige Administrationswerkzeuge

Viele Linux-Distributionen versuchen sich von der Menge abzusetzen, indem sie mehr oder weniger aufwendige Werkzeuge mitliefern, die die Administration des Systems vereinfachen sollen. Diese Werkzeuge sind in der Regel gezielt auf die betreffenden Distributionen abgestimmt. Hier ein paar Kommentare zu typischen Vertretern ihrer Zunft:

SUSE-Administratoren wohlvertraut ist der »YaST«, die grafische (aber auch auf einem Textbildschirm lauffähige) Administrationsoberfläche der SUSE-Distributionen. Er erlaubt die mehr oder weniger umfassende Einstellung zahlreicher Aspekte des Systems teils durch direkte Eingriffe in die relevanten Konfigurationsdateien, teils durch die Manipulation von abstrahierten Konfigurationsdateien unter `/etc/sysconfig`, die anschließend durch das Programm `SuSEconfig` in die tatsächlichen Konfigurationsdateien übertragen werden (für gewisse Aufgaben, etwa die Netzwerkkonfiguration, *sind* die Dateien unter `/etc/sysconfig` auch die tatsächlichen Konfigurationsdateien).

Eine alleinseligmachende Lösung aller Probleme der Systemadministration ist YaST leider nicht. Zwar sind viele Aspkete des Systems für die Administration mit YaST aufbereitet, aber mitunter lassen sich wichtige Einstellungen nicht über den YaST vornehmen, oder die betreffenden YaST-Module funktionieren einfach nicht korrekt. Gefährlich wird es, wenn Sie versuchen, den Rechner teils per YaST und teils über manuelle Änderungen an den Konfigurationsdateien zu verwalten: Zwar gibt YaST sich große Mühe, Ihre Änderungen nicht zu überschreiben (was nicht immer so war – bis SuSE 6 oder so waren YaST und SuSEconfig noch rücksichtsloser), aber führt dann auch seine eigenen Änderungen nicht so durch, dass sie sich im System auswirken. An anderen Stellen werden manuelle Änderungen in den Konfigurationsdateien wiederum tatsächlich auch im YaST sichtbar. Sie brauchen also etwas Insider-Wissen und Erfahrung, damit Sie einschätzen können, welche Konfigurationsdateien Sie direkt anfassen dürfen und von welchen Sie besser Ihre schmutzigen Finger lassen.

Novell hat den Quellcode von YaST schon vor einiger Zeit unter die GPL gestellt (zu den Zeiten von SUSE war er zwar verfügbar, aber nicht unter einer »freien« Lizenz). Allerdings hat bisher keine wesentliche andere Distribution YaST für ihre Zwecke adaptiert, geschweige denn zum Standardwerkzeug à la SUSE gemacht.

Das Webmin-Paket von Jamie Cameron (`http://www.webmin.com/`) erlaubt die komfortable Verwaltung diverser Linux-Distributionen (oder Unix-Versionen) über eine webbasierte Oberfläche. Webmin ist sehr umfassend und bietet besondere Fähigkeiten etwa zur Administration »virtueller« Server (für Web-Hoster bzw. deren Kunden). Sie müssen es allerdings möglicherweise selbst installieren, da die meisten Distributionen es nicht mitliefern. Webmin macht seine eigene Benutzerverwaltung, so dass Sie auch solchen Anwendern Administratorrechte geben können, die sich nicht interaktiv am System anmelden können. (Ob das eine gute Idee ist, steht auf einem anderen Blatt.)

Die meisten Administrationswerkzeuge wie YaST oder Webmin teilen sich dieselben Nachteile:

- Sie sind nicht umfassend genug, um alle Aspekte der Administration zu übernehmen, und als Administrator müssen Sie ihre Grenzen genau kennen, um zu wissen, wo Sie selbst Hand anlegen müssen.

- Sie erlauben die Systemadministration auch Leuten, deren Fachkenntnisse nicht wirklich adäquat sind, um die möglichen Konsequenzen ihrer Aktionen zu überschauen oder Fehler zu finden und zu beheben. Das Anlegen eines Benutzers mit einem Administrationswerkzeug ist sicher keine kritische Aufgabe und bestimmt bequemer als das Editieren von vier verschiedenen Systemdateien mit dem `vi`, aber andere Aufgaben wie die Konfiguration eines Firewalls oder Mailservers sind auch mit einem bequemen Administrationswerkzeug nichts für Laien. Die Gefahr besteht darin, dass unerfahrene Administratoren sich mit der Hilfe eines Administrationswerkzeugs an Aufgaben wagen, die nicht viel komplizierter aussehen als andere, die aber ohne umfassendes Hintergrundwissen den sicheren und/oder zuverlässigen Betrieb des Systems gefährden können.

- Sie bieten in der Regel keine Möglichkeit zur Versionierung oder Dokumentation der gemachten Einstellungen und erschweren deswegen die Arbeit im Team oder die Erstellung von Änderungsprotokollen, indem sie erfordern, dass diese extern geführt werden.

- Sie sind oft intransparent, indem sie keine Dokumentation über die Schritte liefern, die zur Ausführung eines Administrationsvorgangs tatsächlich von ihnen auf dem System vorgenommen werden. Damit bleibt das Wissen über die erforderlichen Schritte in den Programmen vergraben; als Administrator haben Sie keine direkte Möglichkeit, von diesen Programmen zu »lernen«, so wie Sie einem erfahreneren Administrator über die Schulter schauen könnten. Die Administrationswerkzeuge halten Sie also künstlich dumm.

- Als Fortsetzung des vorigen Punkts: Wenn Sie mehrere Rechner zu verwalten haben, sind Sie bei den gängigen Administrationswerkzeugen oft gezwungen, dieselben Schritte wiederholt auf jedem einzelnen Rechner durchzuführen. Oft wäre es bequemer, ein Shellskript zu schreiben, das den gewünschten Administrationsvorgang durchführt, und dieses etwa über die »Secure Shell« automatisch auf jedem gewünschten Rechner auszuführen, aber das Administrationswerkzeug sagt Ihnen ja nicht, was Sie in so ein Shellskript hineintun sollten. Die Arbeit damit ist also im größeren Kontext gesehen ineffizient.

Aus verschiedenen praktischen Erwägungen wie diesen heraus raten wir Ihnen also davon ab, sich bei der Systemadministration zu sehr auf die »komfortablen« Werkzeuge der Distributionen zu verlassen. Sie haben große Ähnlichkeit mit Stützrädchen am Fahrrad: Frühes Umkippen wird zwar wirkungsvoll verhindert und es kommt zu schnellen Erfolgserlebnissen, aber je länger die lieben Kleinen damit herumrollern, um so schwieriger wird es, sie an das »richtige« Radfahren (hier: die Administration in den Original-Dateien, mit allen Vorzügen wie Dokumentation, Transparenz, Auditing, Teamfähigkeit, Transportabilität, …) zu gewöhnen.

Übermäßige Abhängigkeit von einem Administrationswerkzeug führt auch zu übermäßiger Abhängigkeit von der Distribution, zu der dieses Administrationswerkzeug gehört. Dies mag man nicht als wirklichen Nachteil empfinden, aber auf der anderen Seite ist einer der wesentlichen *Vorteile* von Linux ja gerade der Umstand, dass es mehrere unabhängige Lieferanten gibt. Sollten Sie also eines Tages beispielsweise genug von den SUSE-Distributionen haben (aus welchen Gründen auch immer) und auf Red Hat oder Debian GNU/Linux umsteigen wollen, dann wäre es höchst hinderlich, wenn Ihre Administratoren nur den YaST kennten und die Linux-Administration von Grund auf neu lernen müßten. (Administrationswerkzeuge aus dritter Quelle wie Webmin haben dieses Problem nicht im selben Maße.)

Übungen

10.8 [!2] Stellt Ihre Distribution Ihnen ein Administrationswerkzeug (etwa YaST) zur Verfügung? Was können Sie damit alles machen?

10.9 [3] (Fortsetzung der vorigen Übung – beim zweiten Durcharbeiten dieser Unterlage.) Finden Sie heraus, wie Ihr Administrationswerkzeug funktioniert. Können Sie die Konfiguration des Systems mit der Hand verändern, so dass das Administrationswerkzeug diese Änderungen mitbekommt? Nur unter bestimmten Umständen?

10.10 [!1] Administrationswerkzeuge wie Webmin sind potentiell für jeden zugänglich, der einen Browser bedienen kann. Welche Vorteile und Nachteile ergeben sich daraus?

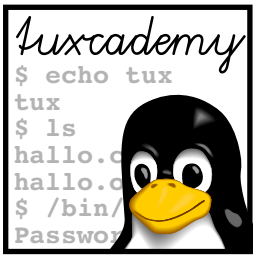
Kommandos in diesem Kapitel

su Startet eine Shell unter der Identität eines anderen Benutzers
su(1) 167

sudo Erlaubt normalen Benutzern das Aufrufen bestimmter Kommandos mit Administratorprivilegien
sudo(8) 165

Zusammenfassung

- Jede Computerinstallation braucht einen gewissen Umfang an Systemadministration. In großen Firmen, Hochschulen und ähnlichen Institutionen werden diese Dienste von (Teams von) hauptamtlichen Administratoren erbracht; in kleinen Firmen oder Privathaushalten fungieren oft (manche) Benutzer als Administratoren.
- Linux-Systeme sind im Großen und Ganzen leicht zu administrieren. Aufwand entsteht vor allem bei der erstmaligen Installation und im laufenden Betrieb bei größeren Konfigurationsänderungen.
- In Linux-Systemen gibt es gewöhnlich ein privilegiertes Benutzerkonto namens `root`, für das die üblichen Zugriffsrechte nicht gelten.
- Man sollte als Administrator nicht ausschließlich als `root` arbeiten, sondern ein gewöhnliches Benutzerkonto verwenden und nur dann `root`-Rechte annehmen, wenn das zwingend erforderlich ist.
- Administrationswerkzeuge wie YaST oder Webmin können einen Teil der Routinearbeiten erleichtern, sind aber kein Ersatz für Administratorwissen und haben oft auch noch andere Nachteile.



11

Benutzerverwaltung

Inhalt

11.1 Grundlagen 174
11.1.1 Wozu Benutzer? 174
11.1.2 Benutzer und Gruppen. 175
11.1.3 »Natürliche Personen« und Pseudobenutzer 177
11.2 Benutzer- und Gruppendaten 178
11.2.1 Die Datei /etc/passwd. 178
11.2.2 Die Datei /etc/shadow. 181
11.2.3 Die Datei /etc/group 184
11.2.4 Die Datei /etc/gshadow 185
11.2.5 Das Kommando getent 185
11.3 Benutzerkonten und Gruppeninformationen verwalten 186
11.3.1 Benutzerkonten einrichten 186
11.3.2 Das Kommando passwd 188
11.3.3 Benutzerkonten löschen 190
11.3.4 Benutzerkonten und Gruppenzuordnung ändern 190
11.3.5 Die Benutzerdatenbank direkt ändern — vipw. 191
11.3.6 Anlegen, Ändern und Löschen von Gruppen 191

Lernziele

- Das Benutzer- und Gruppenkonzept von Linux verstehen
- Die Struktur und Speicherung von Benutzer- und Gruppendaten bei Linux kennen
- Die Kommandos zur Verwaltung von Benutzer- und Gruppendaten anwenden können

Vorkenntnisse

- Kenntnisse über den Umgang mit Konfigurationsdateien

11.1 Grundlagen

11.1.1 Wozu Benutzer?

Früher waren Computer gross und teuer, aber heute sind Büroarbeitsplätze ohne eigenen PC (»persönlichen Computer«) kaum noch vorstellbar, und auch in den meisten häuslichen Arbeitszimmern ist ein Computer anzutreffen. Und während es in der Familie noch genügen mag, wenn Vater, Mutter und die Kinder ihre Dateien nach Verabredung jeweils in verschiedenen Verzeichnissen ablegen, ist das in Unternehmen oder Hochschulen definitiv nicht mehr ausreichend – spätestens wenn Plattenplatz oder andere Dienste auf zentralen Servern zur Verfügung gestellt werden, auf die viele Anwender zugreifen können, muss das Computersystem in der Lage sein, zwischen verschiedenen Benutzern unterscheiden und diesen unterschiedliche Rechte zuordnen zu können. Schließlich geht Frau Schulz aus der Entwicklungsabteilung die Gehaltsliste für alle Angestellten in der Regel genausowenig etwas an wie Herrn Schmidt aus der Personalabteilung die detaillierten Pläne für die neuen Produkte. Und auch am heimischen Herd ist ein bisschen Privatsphäre möglicherweise auch erwünscht – die Weihnachtsgeschenkliste oder Tochters Tagebuch (ehedem mit Schloss versehen) sollen neugierigen Augen vielleicht nicht ganz ohne weiteres zugänglich sein.

Den Umstand, dass Tochters Tagebuch vielleicht sowieso auf Facebook & Co. der ganzen Welt zum Lesen zur Verfügung steht, lassen wir hier mal außer Acht; und selbst wenn das so ist, dann soll die ganze Welt ja wohl trotzdem nicht in Tochters Tagebuch *schreiben* dürfen. (Aus diesem Grund unterstützt auch Facebook die Idee verschiedener Benutzer.)

Der zweite Grund dafür, verschiedene Benutzer zu unterscheiden, folgt aus der Tatsache, dass diverse Aspekte des Systems nicht ohne besondere Privilegien anschau- oder gar änderbar sein sollen. Linux führt aus diesem Grund eine separate Benutzeridentität (`root`) für den Systemadministrator, die es möglich macht, Informationen wie etwa die Benutzerkennwörter vor »gewöhnlichen« Benutzern geheim zu halten. Die Plage älterer Windows-Systeme – Programme, die Sie per E-Mail oder durch ungeschicktes Surfen erhalten und die dann im System alle Arten von Schindluder treiben – kann Sie unter Linux nicht ereilen, da alles, was Sie als gewöhnlicher Benutzer starten können, nicht in der Position ist, systemweit Schindluder zu treiben.

Ganz korrekt ist das leider nicht: Hin und wieder werden Fehler in Linux bekannt, über die ein »normaler Benutzer« theoretisch Dinge tun kann, die sonst dem Administrator vorbehalten sind. Diese Sorte Fehler ist extrem ärgerlich und wird in der Regel sehr zeitnah nach dem Finden behoben, aber es kann natürlich sein, dass so ein Fehler einige Zeit lang unerkannt im System geschlummert hat. Sie sollten daher bei Linux (wie bei allen Betriebssystemen) anstreben, von kritischen Systembestandteilen wie dem Systemkern immer die neueste Version laufen zu lassen, die Ihr Distributor unterstützt.

Auch die Tatsache, dass Linux die Systemkonfiguration vor unbefugtem Zugriff durch normale Benutzer schützt, sollte Sie nicht dazu verleiten, Ihr Gehirn auszuschalten. Wir geben Ihnen hier einige Tipps (etwa dass Sie sich nicht als `root` auf der grafischen Oberfläche anmelden sollen), aber Sie sollten weiter mitdenken. Mails, die Sie auffordern, Adresse *X* anzusurfen und dort Ihre Bank-PIN und drei Transaktionsnummern einzutippen, können Sie auch unter Linux erhalten, und Sie sollten sie genauso ignorieren wie anderswo auch.

Benutzerkonten

Linux unterscheidet verschiedene Benutzer über unterschiedliche **Benutzerkonten** (engl. *accounts*). Typischerweise werden bei den gängigen Distributionen während der Installation zwei Benutzerkonten eingerichtet, nämlich `root` für Administrationsaufgaben und ein weiteres Konto für einen »normalen« Benutzer.

Zusätzliche Konten können Sie als Administrator dann später selbst einrichten, oder sie ergeben sich, wenn der Rechner als Client in einem größeren Netz installiert ist, aus einer anderswo gespeicherten Benutzerkonten-Datenbank.

Linux unterscheidet *Benutzerkonten*, nicht Benutzer. Es hindert Sie zum Beispiel niemand daran, ein separates Benutzerkonto zum E-Mail-Lesen und Surfen im Internet zu verwenden, wenn Sie zu 100% sicher gehen wollen, dass Sachen, die Sie sich aus dem Netz herunterladen, keinen Zugriff auf Ihre wichtigen Daten haben (was ja trotz der Benutzer-Administrator-Trennung sonst passieren könnte). Mit einem bisschen Trickreichtum können Sie sogar einen Browser und ein E-Mail-Programm, die unter Ihrem Surf-Konto laufen, zwischen Ihren »normalen« Programmen anzeigen lassen[1].

Unter Linux ist jedem Benutzerkonto eine eindeutige numerische Kennung zugeordnet, die sogenannte *User ID* oder kurz **UID**. Zu einem Benutzerkonto gehört außerdem noch ein textueller **Benutzername** (etwa `root` oder `hugo`), der für Menschen leichter zu merken ist. An den meisten Stellen, wo es darauf ankommt – etwa beim Anmelden oder bei der Ausgabe einer Liste von Dateien mit ihren Eigentümern – verwendet Linux, wo möglich, den textuellen Namen.

UID

Benutzername

Der Linux-Kern weiß nichts über die textuellen Benutzernamen; in den Prozessdaten und in den Eigentümerangaben im Dateisystem wird immer nur die UID verwendet. Das kann zu Problemen führen, wenn ein Benutzer gelöscht wird, der noch Dateien im System hat, und anschließend die UID einem anderen Benutzer zugewiesen wird. Jener Benutzer »erbt« die Dateien des vorigen UID-Inhabers.

Grundsätzlich spricht nichts Technisches dagegen, dass mehreren Benutzernamen dieselbe (numerische) UID zugeordnet ist. Diese Benutzer haben gleichberechtigten Zugriff auf alle dieser UID gehörenden Dateien, aber jeder kann sein eigenes Kennwort haben. Sie sollten das aber nicht oder nur mit großer Vorsicht ausnutzen.

11.1.2 Benutzer und Gruppen

Um mit einem Linux-Rechner zu arbeiten, müssen Sie sich erst anmelden (neudeutsch »einloggen«), damit das System Sie als Sie erkennt und Ihnen die richtigen Zugriffsrechte zuordnen kann (hierzu später mehr). Alle Aktionen, die Sie während einer Arbeitssitzung (vom Anmelden bis zum Abmelden) ausführen, werden Ihrem Benutzerkonto zugeordnet. Jeder Benutzer hat außerdem ein **Heimatverzeichnis** (engl. *home directory*), in dem er seine »eigenen Dateien« ablegen kann und auf das andere Benutzer oft keinen Lese- und mit sehr großer Sicherheit keinen Schreibzugriff haben. (Nur der Systemadministrator, `root`, darf alle Dateien lesen und schreiben.)

Heimatverzeichnis

Je nachdem, welche Linux-Distribution Sie benutzen (Stichwort: Ubuntu), kann es sein, dass Sie sich nicht explizit beim System anmelden müssen. Dann »weiß« der Rechner allerdings, dass normalerweise Sie kommen, und nimmt einfach an, dass Sie auch wirklich Sie sind. Sie tauschen hier Sicherheit gegen Bequemlichkeit; dieser konkrete Tauschhandel ist vermutlich nur sinnvoll, wenn Sie mit einiger Gewissheit davon ausgehen können, dass niemand außer Ihnen Ihren Rechner einschaltet – und dürfte damit *eigentlich* auf den Computer in Ihrem Single-Haushalt ohne Putzfrau beschränkt sein. Wir haben es Ihnen gesagt.

Mehrere Benutzer, die bestimmte Systemressourcen oder Daten gemeinsam nutzen, können eine **Gruppe** bilden. Linux identifiziert die Gruppenmitglieder

Gruppe

[1] Was dann natürlich wieder etwas gefährlich ist, da Programme, die auf demselben Bildschirm laufen, miteinander kommunizieren können.

entweder durch feste, namentliche Zuordnung oder durch eine vorübergehende Anmeldung ähnlich der Anmeldeprozedur für Benutzer. Gruppen haben keine automatisch vorhandenen »Heimatverzeichnisse«, aber Sie können als Administrator natürlich beliebige Verzeichnisse einrichten, die für bestimmte Gruppen gedacht sind und entsprechende Rechte haben.

Auch Gruppen werden betriebssystemintern durch numerische Kennungen (engl. *group IDs*, kurz GIDs) identifiziert.

Gruppennamen verhalten sich zu GIDs wie Benutzernamen zu UIDs: Der Linux-Kernel kennt nur erstere und legt auch nur erstere in den Prozessdaten und im Dateisystem ab.

Jeder Benutzer gehört zu einer *primären Gruppe* und möglicherweise mehreren *sekundären* oder *zusätzlichen Gruppen*. In einem Unternehmen wäre es beispielsweise möglich, projektspezifische Gruppen einzuführen und jeweils die Projektmitarbeiter in die betreffende Gruppe aufzunehmen, damit sie Zugriff auf gemeinsame Daten in einem Verzeichnis bekommen, das nur für Gruppenmitglieder zugänglich ist.

Für die Zwecke der Rechtevergabe sind alle Gruppen gleichwertig – jeder Benutzer bekommt immer alle Rechte, die sich aus allen Gruppen ergeben, in denen er Mitglied ist. Der einzige Unterschied zwischen der primären Gruppe und den sekundären Gruppen ist, dass Dateien, die ein Benutzer neu anlegt, in der Regel[2] seiner primären Gruppe zugeordnet werden.

Bis einschließlich zum Linux-Kernel 2.4 konnte ein Benutzer maximal 32 zusätzliche Gruppen haben; seit Linux-Kernel 2.6 ist die Anzahl der zusätzlichen Gruppen nicht mehr beschränkt.

Die UID eines Benutzerkontos, die primäre und die sekundären Gruppen und die dazugehörigen GIDs verrät Ihnen das Programm `id`:

```
$ id
uid=1000(hugo) gid=1000(hugo) groups=24(cdrom),29(audio),44(video),▷
 ◁ 1000(hugo)
$ id root
uid=0(root) gid=0(root) groups=0(root)
```

Mit den Optionen `-u`, `-g` und `-G` läßt `id` sich überreden, nur die UID des Kontos, die GID der primären Gruppe oder die GIDs der sekundären Gruppe auszugeben. (Diese Optionen lassen sich nicht kombinieren). Mit der zusätzlichen Option `-n` gibt es Namen statt Zahlen:

```
$ id -G
1000 24 29 44
$ id -Gn
hugo cdrom audio video
```

Das Kommando `groups` liefert dasselbe Resultat wie das Kommando »`id -Gn`«.

`last` Mit dem Kommando `last` können Sie herausfinden, wer sich wann auf Ihrem Rechner angemeldet hat (und im Falle von Anmeldungen über das Netzwerk, von wo aus):

```
$ last
hugo     pts/1        pchugo.example.c Wed Feb 29 10:51   still logged in
oberboss pts/0        pc01.vorstand.ex Wed Feb 29 08:44   still logged in
```

[2] Die Ausnahme besteht darin, dass der Eigentümer eines Verzeichnisses verfügen kann, dass neue Dateien und Verzeichnisse in diesem Verzeichnis der Gruppe zugeordnet werden, der auch das Verzeichnis selbst zugeordnet ist. Aber das nur der Vollständigkeit halber.

```
hugo     pts/2        pchugo.example.c Wed Feb 29 01:17 - 08:44  (07:27)
susi     pts/0        :0               Tue Feb 28 17:28 - 18:11  (00:43)
⋘⋘⋘⋘
reboot   system boot 3.2.0-1-amd64    Fri Feb  3 17:43 - 13:25 (4+19:42)
⋘⋘⋘⋘
```

Die dritte Spalte gibt bei Sitzungen über das Netz den Rechnernamen des `ssh`-Clients an. »`:0`« steht für den grafischen Bildschirm (genaugenommen den ersten X-Server – es könnte mehrere geben).

Beachten Sie auch den `reboot`-Eintrag, der angibt, dass der Rechner neu gestartet wurde. In der dritten Spalte steht hier die Versionsnummer des Linux-Betriebssystemkerns, so wie »`uname -r`« ihn liefert.

Mit einem Benutzernamen als Parameter liefert `last` Informationen über einen bestimmten Benutzer:

```
$ last
hugo     pts/1        pchugo.example.c Wed Feb 29 10:51   still logged in
hugo     pts/2        pchugo.example.c Wed Feb 29 01:17 - 08:44  (07:27)
⋘⋘⋘⋘
```

Es könnte Sie (mit Recht!) stören, dass diese Sorte doch etwas sensitive Information anscheinend beliebigen Systembenutzern ohne Weiteres zugänglich gemacht wird. Wenn Sie als Administrator die Privatsphäre Ihrer Benutzer etwas besser schützen möchten, als Ihre Linux-Distribution das von sich aus macht, können Sie mit dem Kommando

```
# chmod o-r /var/log/wtmp
```

das allgemeine Leserecht für die Datei entfernen, in der `last` die verräterischen Daten findet. Benutzer, die keine Administratorprivilegien haben, sehen dann nur noch etwas wie

```
$ last
last: /var/log/wtmp: Permission denied
```

11.1.3 »Natürliche Personen« und Pseudobenutzer

Außer für »natürliche Personen« – die menschlichen Benutzer des Systems – wird das Benutzer- und Gruppenkonzept auch verwendet, um die Zugriffsrechte auf gewisse Bereiche des Systems zu strukturieren. Das bedeutet, dass es neben den persönlichen Konten der »echten« Benutzer wie Ihnen weitere Konten auf dem System gibt, die nicht tatsächlichen menschlichen Benutzern zugeordnet, sondern systemintern für administrative Funktionen zuständig sind. Hier werden funktionale Rollen definiert, denen eigene Konten und Gruppen zugeordnet werden.

Pseudobenutzer

Nach der Installation von Linux finden Sie in den Dateien `/etc/passwd` und `/etc/group` eine ganze Reihe solcher Pseudobenutzer. Die wichtigste Rolle hat hier der uns schon bekannte Benutzer `root` mit der gleichnamigen Gruppe. UID und GID von `root` sind 0 (Null).

Die Sonderrechte von `root` sind an die UID 0 gekoppelt; die GID 0 hat keine besondere Bedeutung über die normalen Zugriffsrechte hinaus.

Weitere Pseudobenutzer können für bestimmte Programmsysteme (beispielsweise `news` für News mit INN und `postfix` für Mail mit Postfix) oder für bestimmte Komponenten oder Gerätegruppen (beispielsweise Drucker, Band- und Diskettenlaufwerke) existieren. Diese Konten erreichen Sie gegebenenfalls wie andere auch über dem Befehl `su`. Diese Pseudobenutzer sind als Eigentümer von Dateien

Pseudobenutzer für Rechtevergabe

und Verzeichnissen hilfreich, um die mit dem Eigentum an Systemdaten verbundenen Zugriffsrechte flexibel an die speziellen Anforderungen anzupassen, ohne dazu das `root`-Konto benutzen zu müssen. Dasselbe geht auch für Gruppen; beispielsweise haben Mitglieder der Gruppe `disk` Zugriffsrecht auf die Platten auf Blockebene.

Übungen

11.1 [1] Wodurch unterscheidet der Betriebssystemkern verschiedene Benutzer und Gruppen?

11.2 [2] Was passiert, wenn eine UID zweimal mit unterschiedlichem Namen vergeben wird? Ist das erlaubt?

11.3 [1] Was versteht man unter Pseudobenutzern? Nennen Sie Beispiele!

11.4 [2] (Beim zweiten Durcharbeiten.) Ist es akzeptabel, einen Benutzer in die Gruppe `disk` aufzunehmen, dem Sie nicht das `root`-Kennwort anvertrauen würden? Warum (nicht)?

11.2 Benutzer- und Gruppendaten

11.2.1 Die Datei `/etc/passwd`

zentrale Benutzerdatenbank

Die zentrale Benutzerdatenbank ist die Datei `/etc/passwd`. Jeder Benutzer auf dem System hat einen Eintrag in dieser Datei – eine Zeile, in der Attribute wie Linux-Benutzername, »richtiger« Name usw. festgehalten werden. Bereits mit der Erstinstallation eines Linux-Systems sind in der Datei die meisten Pseudobenutzer eingetragen.

Die Einträge in `/etc/passwd` haben folgendes Format:

⟨*Benutzername*⟩:⟨*Kennwort*⟩:⟨*UID*⟩:⟨*GID*⟩:⟨*GECOS*⟩:⟨*Heimatverzeichnis*⟩:⟨*Shell*⟩
```
hugo:x:1000:1000:Hugo Schulz:/home/hugo:/bin/sh
```

⟨*Benutzername*⟩ Dieser Name sollte aus Kleinbuchstaben und Ziffern bestehen; das erste Zeichen sollte ein Buchstabe sein. Unix-Systeme unterscheiden oft nur die ersten 8 Zeichen – Linux hat diese Einschränkung nicht, aber in heterogenen Netzen sollten Sie darauf Rücksicht nehmen.

Widerstehen Sie der Versuchung, Umlaute, Satzzeichen und ähnliches in Benutzernamen aufzunehmen, selbst falls das System sie durchläßt – nicht alle Werkzeuge zum Anlegen neuer Benutzer sind pingelig, und Sie könnten `/etc/passwd` ja auch direkt ändern. Was auf den ersten Blick prächtig zu funktionieren scheint, kann später anderswo zu Problemen führen.

Ebenfalls Abstand nehmen sollten Sie von Benutzernamen, die nur aus Großbuchstaben oder nur aus Ziffern bestehen. Erstere machen möglicherweise Probleme beim Anmelden (siehe Übung 11.6), letztere können zu Verwirrung führen, vor allem wenn der numerische Benutzername nicht mit der numerischen UID des Kontos übereinstimmt. Programme wie »`ls -l`« zeigen nämlich die UID an, wenn es für die betreffende UID keinen Eintrag in `/etc/passwd` gibt, und es ist nicht unbedingt einfach, UIDs in der `ls`-Ausgabe von rein numerischen Benutzernamen zu unterscheiden.

⟨*Kennwort*⟩ Traditionell steht hier das verschlüsselte Kennwort des Benutzers. Unter Linux sind heute »Schattenkennwörter« (***shadow passwords***) üblich; statt das Kennwort in der allgemein lesbaren `/etc/passwd`-Datei abzulegen, steht es in der Datei `/etc/shadow` gespeichert, auf die nur der Administrator und einige privilegierte Prozesse Zugriff haben. In `/etc/passwd` macht ein »x« auf diesen Umstand aufmerksam. Jedem Benutzer steht das Kommando `passwd` zur Verfügung, um sein Kennwort selbst zu verändern. *shadow passwords*

Linux erlaubt verschiedene Verfahren zur Verschlüsselung von Kennwörtern. Klassisch ist das von Unix übernommene und von DES abgeleitete `crypt`-Verfahren; solche Kennwörter erkennen Sie daran, dass sie in verschlüsselter Form genau 13 Zeichen lang sind. Ebenfalls verbreitet sind die sogenannten MD5-Kennwörter; ihre verschlüsselte Form ist länger und beginnt immer mit der Zeichenfolge `$1$`. Manche Linux-Versionen unterstützen weitere Verfahren.

⟨*UID*⟩ Die numerische Benutzerkennung – eine Zahl zwischen 0 und $2^{32}-1$. Nach Konvention sind UIDs zwischen 0 und 99 (einschließlich) für das System reserviert, UIDs zwischen 100 und 499 können an Softwarepakete ausgegeben werden, falls diese Pseudobenutzer benötigen. UIDs für »echte« Benutzer haben bei den meisten Distributionen Werte ab 1000.

Eben weil die Benutzer im System nicht durch die Namen, sondern durch die UID unterschieden werden, behandelt der Kernel intern zwei Konten als völlig identisch, wenn sie unterschiedliche Benutzernamen aber dieselbe UID haben – jedenfalls was die Zugriffsrechte angeht. Bei den Kommandos, die einen Benutzernamen anzeigen (etwa »`ls -l`« oder `id`), wird in solchen Fällen immer der Benutzername verwendet, der beim Anmelden angegeben wurde.

⟨*GID*⟩ Die GID der **primären Gruppe** des Benutzers nach dem Anmelden. primäre Gruppe

Bei den Novell/SUSE- und manchen anderen Distributionen wird eine bestimmte Gruppe, hier beispielsweise `users`, als gemeinsame Standardgruppe für alle Benutzer eingetragen. Diese Methode ist einfach zu verstehen und hat Tradition.

Bei vielen Distributionen, etwa denen von Red Hat oder Debian GNU/Linux, wird für jeden neuen Benutzer automatisch eine eigene Gruppe angelegt, die die gleiche GID hat wie die UID des Benutzerkontos. Die Idee dahinter ist, eine differenziertere Rechtevergabe zu ermöglichen als bei dem Ansatz, alle Benutzer in dieselbe Gruppe `users` zu tun. Denken Sie an die folgende Situation: Emil (Benutzername `emil`) ist der persönliche Assistent der Vorstandsvorsitzenden Susi (Benutzername `susi`). In dieser Funktion muss er hin und wieder auf Dateien zugreifen, die in Susis Heimatverzeichnis gespeichert sind, die aber alle anderen Benutzer nichts angehen. Der von Red Hat, Debian & Co. verfolgte Ansatz »Eine Gruppe pro Benutzer« macht es einfach, den Benutzer `emil` in die *Gruppe* `susi` zu tun und dafür zu sorgen, dass Susis Dateien für alle Gruppenmitglieder lesbar sind (der Standardfall), aber nicht für den »Rest der Welt«. Im Ansatz »Eine Gruppe für alle« wäre es nötig, eine ganz neue Gruppe einzuführen und die Konten `emil` und `susi` entsprechend umzukonfigurieren.

Jeder Benutzer muss durch die Zuordnung in der Datei `/etc/passwd` Mitglied mindestens einer Benutzergruppe sein.

Die sekundären Gruppen (soweit vorhanden) des Benutzers werden durch entsprechende Einträge in der Datei `/etc/group` festgelegt.

⟨*GECOS*⟩ Dies ist das Kommentarfeld, auch *GECOS-Feld* genannt.

GECOS steht für *General Electric Comprehensive Operating System* und hat nichts mit Linux zu tun, abgesehen davon, dass man in diesem Feld in der Frühzeit von Unix Informationen eingefügt hat, die für die Kompatibilität mit einigen Jobversanddiensten für GECOS-Rechner notwendig waren.

Das Feld enthält diverse Informationen über den Benutzer, vor allem seinen »richtigen« Namen und optionale Informationen wie die Zimmer- oder Telefonnummer. Diese Information wird von Programmen wie `mail` und `finger` benutzt. Oft wird der volle Name von News- und Mail-Programmen bei der Zusammenstellung der Absenderadresse verwendet.

Theoretisch gibt es ein Programm namens `chfn`, mit dem Sie als Benutzer den Inhalt Ihres GECOS-Feldes ändern können. Ob das im Einzelfall klappt, ist eine andere Frage, da man zumindest in Firmen Leuten nicht notwendigerweise erlauben will, ihren Namen beliebig zu modifizieren.

⟨*Heimatverzeichnis*⟩ Das hier benannte Verzeichnis ist der persönliche Bereich des Benutzers, in dem er seine eigenen Dateien aufbewahren kann. Ein neu erstelltes Heimatverzeichnis ist selten leer, denn üblicherweise erhält ein neuer Benutzer vom Administrator einige Profildateien als Erstausstattung. Wenn ein Benutzer sich anmeldet, benutzt seine Shell das Heimatverzeichnis als aktuelles Verzeichnis, das heißt, der Benutzer befindet sich unmittelbar nach der Anmeldung zunächst dort.

⟨*Login-Shell*⟩ Der Name des Programms, das von `login` nach erfolgreicher Anmeldung gestartet werden soll – das ist in der Regel eine Shell. Das siebte Feld reicht bis zum Zeilenende.

Der Benutzer kann mit dem Programm `chsh` diesen Eintrag selbst ändern. Die erlaubten Programme (Shells) sind in der Datei `/etc/shells` aufgelistet. Wenn ein Benutzer keine interaktive Shell haben soll, kann auch ein beliebiges anderes Programm mit allen Argumenten in dieses Feld eingetragen werden (ein gängiger Kandidat ist `/bin/true`). Das Feld kann auch leer bleiben. Dann wird automatisch die Standardshell `/bin/sh` gestartet.

Wenn Sie sich unter einer grafischen Oberfläche anmelden, dann werden normalerweise alle möglichen Programme für Sie gestartet, aber nicht notwendigerweise eine interaktive Shell. Der Shell-Eintrag in `/etc/passwd` kommt aber zum Beispiel zum Tragen, wenn Sie ein Terminal-Emulationsprogramm wie `xterm` oder `konsole` aufrufen, denn diese Programme orientieren sich normalerweise an diesem, um Ihre bevorzugte Shell zu identifizieren.

Einige der hier gezeigten Felder können leer bleiben. Absolut notwendig sind nur Benutzername, UID, GID und Heimatverzeichnis. Für die meisten Benutzerkonten werden alle diese Felder ausgefüllt sein, aber Pseudobenutzer benutzen eventuell nur einen Teil der Felder.

Heimatverzeichnisse

Die Heimatverzeichnisse stehen üblicherweise unter `/home` und heißen so wie der Benutzername des Besitzers. In der Regel ist das eine ganz nützliche Übereinkunft, die dazu beiträgt, dass das Heimatverzeichnis eines bestimmten Benutzers leicht zu finden ist. Theoretisch kann ein Heimatverzeichnis aber an beliebiger Stelle im System stehen, und der Name ist auch beliebig.

Bei großen Systemen ist es gängig, zwischen `/home` und dem Benutzernamen-Verzeichnis noch eine oder mehrere Zwischenebenen einzuführen, etwa

`/home/pers/hugo`	*Hugo aus der Personalabteilung*
`/home/entw/susi`	*Susi aus der Entwicklungsabteilung*
`/home/vorst/heinz`	*Heinz aus dem Vorstand*

Dafür gibt es mehrere Gründe. Zum einen ist es so leichter möglich, die Heimatverzeichnisse einer Abteilung auf einem Server der Abteilung zu halten, sie aber gegebenenfalls auf anderen Client-Rechnern zugänglich zu machen. Zum anderen waren Unix-Dateisysteme (und manche Linux-Dateisysteme) langsam im Umgang mit Verzeichnissen, die sehr viele Dateien enthalten, was sich bei einem `/home` mit mehreren tausend Einträgen in unschöner Weise bemerkbar gemacht hätte. Mit heute aktuellen Linux-Dateisystemen (`ext3` mit `dir_index` und ähnlichem) ist letzteres jedoch kein Problem mehr.

Beachten Sie, dass Sie als Administrator die Datei `/etc/passwd` nicht unbedingt direkt von Hand editieren müssen. Es gibt eine Reihe von Programmen, die Ihnen bei der Einrichtung und Pflege der Benutzerkonten helfen. Werkzeuge

Prinzipiell ist es auch möglich, die Benutzerdatenbank anderswo zu lagern als in `/etc/passwd`. Auf Systemen mit sehr vielen Benutzern (Tausenden) ist eine Speicherung etwa in einer relationalen Datenbank vorzuziehen, während sich in heterogenen Netzen eine gemeinsame Benutzerverwaltung für unterschiedliche Plattformen etwa auf der Basis eines LDAP-Verzeichnisses anbietet. Die Details würden allerdings den Rahmen dieses Kurses sprengen.

11.2.2 Die Datei `/etc/shadow`

Aus Sicherheitsgründen werden bei fast allen aktuellen Linux-Distributionen die Benutzerkennwörter in verschlüsselter Form in der Datei `/etc/shadow` gespeichert (engl. *shadow passwords*, »Schattenkennwörter«). Die Datei ist für normale Benutzer nicht lesbar; nur `root` darf sie schreiben, während außer ihm auch die Mitglieder der Gruppe `shadow` die Datei lesen dürfen. Wenn Sie sich die Datei als normaler Benutzer anzeigen lassen wollen, erhalten Sie eine Fehlermeldung.

Die Verwendung von `/etc/shadow` ist nicht Pflicht, aber sehr dringend empfohlen. Allerdings kann es Systemkonfigurationen geben, bei denen die durch Schattenkennwörter erreichte Sicherheit wieder zunichte gemacht wird, etwa wenn Benutzerdaten über NIS an andere Rechner exportiert werden (vor allem in heterogenen Unix-Umgebungen).

Für jeden Benutzer ist in dieser Datei wieder genau eine Zeile eingetragen, das Format ist Format

⟨*Benutzername*⟩:⟨*Kennwort*⟩:⟨*Änderung*⟩:⟨*Min*⟩:⟨*Max*⟩▷
◁:⟨*Warnung*⟩:⟨*Frist*⟩:⟨*Sperre*⟩:⟨*Reserviert*⟩

Zum Beispiel:

```
root:gaY2L19jxzHj5:10816:0:10000::::
daemon:*:8902:0:10000::::
hugo:GodY6c5pZk1xs:10816:0:10000::::
```

Im folgenden die Bedeutung der einzelnen Felder:

⟨*Benutzername*⟩ Entspricht einem Eintrag in der Datei `/etc/passwd`. Dieses Feld »verbindet« die beiden Dateien.

⟨*Kennwort*⟩ Das verschlüsselte Kennwort des Benutzers. Ein leerer Eintrag bedeutet in der Regel, dass der Benutzer sich ohne Kennwort anmelden kann. Steht hier ein Stern oder ein Ausrufungszeichen, kann der betreffende Benutzer sich nicht anmelden. Es ist auch üblich, Benutzerkonten zu sperren, ohne sie komplett zu löschen, indem man einen Stern oder ein Ausrufungszeichen an den Anfang des zugehörigen Kennworts setzt.

⟨*Änderung*⟩ Das Datum der letzten Änderung des Kennworts, in Tagen seit dem 1. Januar 1970.

⟨*Min*⟩ Minimale Anzahl von Tagen, die seit der letzten Kennwortänderung vergangen sein müssen, damit das Kennwort wieder geändert werden kann.

⟨*Max*⟩ Maximale Anzahl von Tagen, die ein Kennwort ohne Änderung gültig bleibt. Nach Ablauf dieser Frist muss der Benutzer sein Kennwort ändern.

⟨*Warnung*⟩ Die Anzahl von Tagen vor dem Ablauf der ⟨*Max*⟩-Frist, an denen der Benutzer eine Warnung erhält, dass er sein Kennwort bald ändern muss, weil die maximale Anzahl abläuft. Die Meldung erscheint in der Regel beim Anmelden.

⟨*Frist*⟩ Die Anzahl von Tagen ausgehend vom Ablauf der ⟨*Max*⟩-Frist, nach der das Konto automatisch gesperrt wird, wenn der Benutzer nicht vorher sein Kennwort ändert. (In der Zeit zwischen dem Ende der ⟨*Max*⟩-Frist und dem Ende dieser Frist kann der Benutzer sich anmelden, muss aber sofort sein Kennwort ändern.)

⟨*Sperre*⟩ Das Datum, an dem das Konto definitiv gesperrt wird, wieder in Tagen seit dem 1. Januar 1970.

Verschlüsselung von Kennwörtern

Kurz noch ein paar Anmerkungen zum Thema »Verschlüsselung von Kennwörtern«. Man könnte auf den Gedanken kommen, dass die Kennwörter, wenn sie verschlüsselt sind, auch wieder *entschlüsselt* werden können. Einem cleveren Cracker, dem die Datei `/etc/shadow` in die Hände fällt, würden so sämtliche Benutzerkonten des Systems offen stehen. Allerdings ist das in Wirklichkeit nicht so, denn die »Verschlüsselung« der Kennwörter ist eine Einbahnstraße: Es ist nicht möglich, aus der »verschlüsselten« Darstellung eines Linux-Kennworts die unverschlüsselte zurückzugewinnen, da das verwendete Verfahren das wirkungsvoll verhindert. Die einzige Möglichkeit, die »Verschlüsselung« zu »knacken«, besteht darin, potenzielle Kennwörter zur Probe zu verschlüsseln und zu schauen, ob dasselbe herauskommt wie das, was in `/etc/shadow` steht.

Nehmen wir mal an, Sie wählen die Zeichen Ihres Kennworts aus den 95 sichtbaren Zeichen des ASCII (es wird zwischen Groß- und Kleinschreibung unterschieden). Das bedeutet, es gibt 95 verschiedene einstellige Kennwörter, $95^2 = 9025$ zweistellige und so weiter. Bei acht Stellen sind Sie schon bei 6,6 Billiarden ($6{,}6 \cdot 10^{15}$) Möglichkeiten. Angenommen, Sie könnten 10 Millionen Kennwörter in der Sekunde zur Probe verschlüsseln (für das traditionelle Verfahren auf heutiger Hardware absolut nicht unrealistisch). Dann müßten Sie knapp 21 Jahre einkalkulieren, um alle Möglichkeiten durchzuprobieren. Wenn Sie in der glücklichen Lage sind, eine moderne Grafikkarte zu besitzen, ist da durchaus noch ein Faktor 50–100 drin, so dass daraus gut zwei Monate werden. Und dann gibt es ja noch nette Dienste wie Amazons EC2, die Ihnen (oder irgendwelchen Crackern) fast beliebige Rechenleistung auf Abruf zur Verfügung stellen, oder das freundliche Russen-Botnet … Fühlen Sie sich also nicht zu sicher.

Es gibt noch ein paar andere Probleme: Das traditionelle Verfahren (meist »crypt« oder »DES« genannt – letzteres, weil es ähnlich zu, aber nicht iden-

tisch mit, dem gleichnamigen Verschlüsselungsverfahren ist[3]) sollten Sie nicht mehr verwenden, wenn Sie es vermeiden können. Es hat nämlich die unangenehme Eigenschaft, nur die ersten acht Zeichen jedes Kennworts zu bearbeiten, und der clevere Cracker kann inzwischen genug Plattenplatz kaufen, um jedenfalls die gängigsten 50 Millionen (oder so) Kennwörter »auf Vorrat« zu verschlüsseln. Zum »Knacken« muss er so nur noch in seinem Vorrat nach der verschlüsselten Form suchen, was sehr schnell geht, und kann dann einfach den Klartext ablesen.

Um das Ganze noch etwas aufwendiger zu machen, wird beim Verschlüsseln eines neu eingegebenen Kennworts traditionell noch ein zufälliges Element addiert (das sogenannte *salt*), das dafür sorgt, dass eine von 4096 Möglichkeiten für das verschlüsselte Kennwort gewählt wird. Der Hauptzweck des *salt* besteht darin, »Zufallstreffer« zu vermeiden, die sich ergeben, wenn Benutzer *X* aus irgendwelchen Gründen einen Blick auf den Inhalt von `/etc/shadow` wirft und feststellt, dass sein verschlüsseltes Kennwort genauso aussieht wie das von Benutzer *Y* (so dass er sich mit seinem *Klartextkennwort* auch als Benutzer *Y* anmelden kann). Als angenehmer Nebeneffekt wird der Plattenplatz für das Cracker-Wörterbuch aus dem vorigen Absatz um den Faktor 4096 in die Höhe getrieben.

Die gängige Methode zur Kennwortverschlüsselung basiert heute auf dem MD5-Algorithmus, erlaubt beliebig lange Kennwörter und verwendet ein 48-Bit-*salt* statt den traditionellen 12 Bit. Netterweise ist das Verfahren auch wesentlich langsamer zu berechnen als »crypt«, was für den üblichen Zweck – die Prüfung beim Anmelden – ohne Bedeutung ist (man kann immer noch ein paar hundert Kennwörter pro Sekunde verschlüsseln), aber die cleveren Cracker schon behindert. (Sie sollten sich übrigens nicht davon irre machen lassen, dass Kryptografen das MD5-Verfahren als solches heutzutage wegen seiner Unsicherheit verpönen. Für die Anwendung zur Kennwortverschlüsselung ist das ziemlich irrelevant.)

Von den verschiedenen Parametern zur Kennwortverwaltung sollten Sie sich nicht zuviel versprechen. Sie werden zwar von der Login-Prozedur auf der Textkonsole befolgt, aber ob sie an anderen Stellen im System (etwa beim grafischen Anmeldebildschirm) genauso beachtet werden, ist systemabhängig. Ebenso bringt es in der Regel nichts, den Anwendern in kurzen Abständen neue Kennwörter aufzuzwingen – meist ergibt sich dann eine Folge der Form `susi1`, `susi2`, `susi3`, … oder sie alternieren zwischen zwei Kennwörtern. Eine *Mindestfrist* gar, die verstreichen muss, bevor ein Benutzer sein Kennwort ändern kann, ist geradezu gefährlich, weil sie einem Cracker möglicherweise ein »Zeitfenster« für unerlaubte Zugriffe einräumt, selbst wenn der Benutzer weiß, dass sein Kennwort kompromittiert wurde.

Das Problem, mit dem Sie als Administrator zu kämpfen haben, ist in der Regel nicht, dass Leute versuchen, die Kennwörter auf Ihrem System mit »roher Gewalt« zu knacken. Viel erfolgversprechender ist in der Regel sogenanntes *social engineering*. Um Ihr Kennwort zu erraten, beginnt der clevere Cracker natürlich nicht mit `a`, `b`, und so weiter, sondern mit den Vornamen Ihres Ehegesponses, Ihrer Kinder, Ihrem Autokennzeichen, dem Geburtsdatum Ihres Hundes et cetera. (Wir wollen Ihnen natürlich in keiner Weise unterstellen, dass *Sie* so ein dummes Kennwort verwenden. Neinnein, *Sie* doch ganz bestimmt nicht! Bei Ihrem Chef sind wir uns da allerdings schon nicht mehr ganz so sicher …) Und dann gibt es

[3] Wenn Sie es genau wissen müssen: Das Klartext-Kennwort fungiert als Schlüssel (!) zur Verschlüsselung einer konstanten Zeichenkette (typischerweise einem Vektor von Nullbytes). Ein DES-Schlüssel hat 56 Bit, das sind gerade 8 Zeichen zu je 7 Bit (denn das höchstwertige Bit im Zeichen wird ignoriert). Dieser Prozess wird insgesamt fünfundzwanzigmal wiederholt, wobei immer die Ausgabe als neue Eingabe dient. Genau genommen ist das verwendete Verfahren auch nicht wirklich DES, sondern an ein paar Stellen abgewandelt, damit es weniger leicht möglich ist, aus kommerziell erhältlichen DES-Verschlüsselungs-Chips einen Spezialcomputer zum Knacken von Kennwörtern zu bauen.

da natürlich noch das altgediente Mittel des Telefonanrufs: »Hallo, hier ist die IT-Abteilung. Wir müssen unsere Systemsicherheitsmechanismen testen und brauchen dazu ganz dringend Ihren Benutzernamen und Ihr Kennwort.«

Es gibt diverse Mittel und Wege, Linux-Kennwörter sicherer zu machen. Neben dem oben genannten verbesserten Verfahren, das die meisten Linux-Systeme heute standardmäßig anwenden, gehören dazu Ansätze wie (zu) simple Kennwörter schon bei der Vergabe anzumeckern oder proaktiv Software laufen zu lassen, die schwache verschlüsselte Kennwörter zu identifizieren versucht, so wie das clevere Cracker auch machen würden (*Vorsicht:* Machen Sie sowas in der Firma nur mit der schriftlichen (!) Rückendeckung Ihres Chefs!). Andere Methoden vermeiden Kennwörter komplett zugunsten von ständig wechselnden magischen Zahlen (Stichwort: SecurID) oder Smartcards. All das würde in dieser Schulungsunterlage zu weit führen, und wir verweisen Sie darum auf die Unterlage *Linux-Sicherheit*.

11.2.3 Die Datei `/etc/group`

Gruppendatenbank

Gruppeninformationen hinterlegt Linux standardmäßig in der Datei `/etc/group`. Diese Datei enthält für jede Gruppe auf dem System einen einzeiligen Eintrag, der ähnlich wie die Einträge in `/etc/passwd` aus Feldern besteht, die durch einen Doppelpunkt »:« voneinander getrennt sind. `/etc/group` enthält genauer gesagt vier Felder pro Zeile:

⟨*Gruppenname*⟩:⟨*Kennwort*⟩:⟨*GID*⟩:⟨*Mitglieder*⟩

Deren Bedeutung ergibt sich wie folgt:

⟨*Gruppenname*⟩ Der textuelle Name der Gruppe, für die Verwendung in Verzeichnislisten usw.

⟨*Kennwort*⟩ Ein optionales Kennwort für diese Gruppe. Damit können auch Benutzer, die nicht per `/etc/shadow` oder `/etc/group` Mitglied der Gruppe sind, mit dem Befehl `newgrp` diese Gruppenzugehörigkeit annehmen. Ein »*« als ungültiges Zeichen verhindert einen Gruppenwechsel von normalen Benutzer in die betreffende Gruppe. Ein »x« verweist auf die separate Kennwortdatei `/etc/gshadow`.

⟨*GID*⟩ Die numerische Gruppenkennung für diese Gruppe

⟨*Mitglieder*⟩ Eine durch Kommas getrennte Liste mit Benutzernamen. Die Liste enthält alle Benutzer, die diese Gruppe als sekundäre Gruppe haben, die also zu dieser Gruppe gehören, aber im GID-Feld der Datei `/etc/passwd` einen anderen Wert stehen haben. (Benutzer mit dieser Gruppe als primärer Gruppe dürfen hier auch stehen, aber das ist unnötig.)

Eine `/etc/group`-Datei könnte zum Beispiel so aussehen:

```
root:x:0:root
bin:x:1:root,daemon
users:x:100:
projekt1:x:101:hugo,susi
projekt2:x:102:emil
```

administrative Gruppen

Die Einträge für die Gruppen `root` und `bin` sind Einträge für administrative Gruppen, ähnlich den imaginären Benutzerkonten auf dem System. Gruppen wie diesen sind viele Dateien auf dem System zugeordnet. Die anderen Gruppen enthalten Benutzerkonten.

GID-Werte

Ähnlich wie bei den UIDs werden auch die GIDs von einer bestimmten Zahl an, typischerweise 100, hochgezählt. Für einen gültigen Eintrag müssen mindestens das erste und dritte Feld (Gruppenname und GID) vorhanden sein. Durch so einen

Eintrag wird einer Gruppennummer, die in der Kennwortdatei einem Benutzer zugeordnet wurde, ein Gruppenname gegeben.

Die Felder für Kennwort und/oder Benutzerliste müssen nur für solche Gruppen ausgefüllt werden, die Benutzern als sekundäre Gruppe zugeordnet sind. Die in der Mitgliederliste eingetragenen Benutzer werden nicht nach einem Kennwort gefragt, wenn sie mit dem Kommando `newgrp` die aktuelle GID wechseln wollen. Wenn im zweiten Feld des Gruppeneintrags ein verschlüsseltes Kennwort eingetragen ist, können Anwender ohne Eintrag in der Mitgliederliste sich mit dem Kennwort legitimieren, um die Gruppenzugehörigkeit anzunehmen.

Mitgliederliste

Gruppenkennwort

In der Praxis werden Gruppenkennwörter so gut wie nie verwendet, da der Verwaltungsaufwand den daraus zu ziehenden Nutzen kaum rechtfertigt. Es ist einerseits bequemer, den jeweiligen Benutzern die betreffende Gruppe direkt als sekundäre Gruppe zuzuordnen (seit dem Linux-Kernel 2.6 gibt es ja keine Beschränkung für die Anzahl der sekundären Gruppen mehr), und andererseits folgt aus einem *einzelnen* Kennwort, das *alle* Gruppenmitglieder kennen müssen, nicht wirklich viel Sicherheit.

Wenn Sie sichergehen wollen, dann sorgen Sie dafür, dass in allen Gruppenkennwort-Feldern ein Stern (»*«) steht.

11.2.4 Die Datei `/etc/gshadow`

Wie bei der Kennwortdatei gibt es auch für die Gruppendatei eine Erweiterung durch das Schattenkennwortsystem. Die Gruppenkennwörter, die sonst in der Datei `/etc/group` analog zu `/etc/passwd` verschlüsselt, aber für alle Benutzer lesbar abgelegt sind, werden dann in der separaten Datei `/etc/gshadow` gespeichert. Dort werden auch zusätzliche Informationen zur Gruppe festgehalten, beispielsweise die Namen der zum Ein- und Austragen von Mitgliedern autorisierten Gruppenverwalter.

11.2.5 Das Kommando `getent`

Natürlich können Sie die Dateien `/etc/passwd`, `/etc/shadow` und `/etc/group` wie alle anderen Textdateien auch mit Programmen wie `cat`, `less` oder `grep` lesen und verarbeiten (OK, OK, für `/etc/shadow` müssen Sie `root` sein). Dabei gibt es aber ein paar praktische Probleme:

- Eventuell bekommen Sie nicht die ganze Wahrheit zu sehen: Es könnte ja sein, dass die Benutzerdatenbank (oder Teile davon) auf einem LDAP-Server, in einer SQL-Datenbank oder einem Windows-Domänencontroller steht und Sie in `/etc/passwd` gar nichts Interessantes finden.
- Wenn Sie gezielt nach einem Benutzereintrag suchen wollen, ist das mit `grep` schon etwas umständlich zu tippen, wenn Sie »falsche Positive« ausschließen wollen.

Das Kommando `getent` macht es möglich, die verschiedenen Datenbanken für Benutzer- und Gruppeninformationen gezielt abzufragen. Mit

```
$ getent passwd
```

bekommen Sie etwas angezeigt, das aussieht wie `/etc/passwd`, aber aus allen Quellen für Benutzerdaten zusammengesetzt ist, die auf dem Rechner aktuell konfiguriert sind. Mit

```
$ getent passwd hugo
```

bekommen Sie den Benutzereintrag für den Benutzer `hugo`, egal wo er tatsächlich gespeichert ist. Statt `passwd` können Sie auch `shadow`, `group` oder `gshadow` angeben, um die betreffende Datenbank zu konsultieren. (Natürlich können Sie auch mit `getent` auf `shadow` und `gshadow` nur als `root` zugreifen.)

Der Begriff »Datenbank« ist hier zu verstehen als »Gesamtheit aller Quellen, aus denen die C-Bibliothek sich Informationen zu diesem Thema (etwa Benutzer) zusammensucht«. Wenn Sie wissen wollen, wo genau diese Informationen herkommen (können), dann lesen Sie `nsswitch.conf`(5) und studieren Sie die Datei `/etc/nsswitch.conf` auf Ihrem System.

Sie dürfen auch mehrere Benutzer- oder Gruppennamen angeben. In diesem Fall werden die Informationen für alle genannten Benutzer oder Gruppen ausgegeben:

```
$ getent passwd hugo susi fritz
```

Übungen

11.5 [1] Welchen Wert finden Sie in der zweiten Spalte der Datei `/etc/passwd`? Warum finden Sie dort diesen Wert?

11.6 [2] Schalten Sie auf eine Textkonsole um (etwa mit Alt+F1) und versuchen Sie sich anzumelden, indem Sie Ihren Benutzernamen in reiner Großschreibung eingeben. Was passiert?

11.7 [2] Wie können Sie prüfen, dass für jeden Eintrag in der `passwd`-Datenbank auch ein Eintrag in der `shadow`-Datenbank vorhanden ist? (`pwconv` betrachtet nur die *Dateien* `/etc/passwd` und `/etc/shadow` und schreibt außerdem `/etc/shadow` neu, was wir hier nicht wollen.)

11.3 Benutzerkonten und Gruppeninformationen verwalten

Nachdem die Installation einer neuen Linux-Distribution abgeschlossen ist, gibt es zunächst nur das `root`-Konto für den Administrator und die Pseudobenutzer. Alle weiteren Benutzerkonten müssen erst eingerichtet werden (wobei die meisten Distributionen heutzutage den Installierer mit sanfter Gewalt dazu nötigen, zumindest *einen* »gewöhnlichen« Benutzer anzulaegen).

Als Administrator ist es Ihre Aufgabe, die Konten für alle Benutzer (real und imaginär) auf Ihrem System anzulegen und zu verwalten. Um dies zu erleichtern, bringt Linux einige Werkzeuge zur Benutzerverwaltung mit. Damit ist das zwar in den meisten Fällen eine problemlose, einfach zu erledigende Angelegenheit, aber es ist wichtig, dass Sie die Zusammenhänge verstehen.

Werkzeuge zur Benutzerverwaltung

11.3.1 Benutzerkonten einrichten

Der Vorgang bei der Einrichtung eines neuen Benutzerkontos ist im Prinzip immer gleich und erfolgt in mehreren Schritten:

1. Sie müssen Einträge in der Kennwortdatei `/etc/passwd` und gegebenenfalls in `/etc/shadow` anlegen.
2. Gegebenenfalls ist ein Eintrag (oder mehrere) in der Gruppendatei `/etc/group` nötig.
3. Sie müssen das Heimatverzeichnis erzeugen, eine Grundausstattung an Dateien hineinkopieren und alles dem neuen Benutzer übereignen.
4. Wenn nötig, müssen Sie den neuen Benutzer noch in weitere Listen eintragen, zum Beispiel für Plattenkontingente, Zugriffsberechtigung auf Datenbanken und spezielle Applikationen.

Alle Dateien, die beim Einrichten eines neuen Kontos bearbeitet werden, sind normale Textdateien. Sie können jeden Schritt ohne weiteres von Hand beziehungsweise mit Hilfe eines Texteditors durchführen. Da dies jedoch eine genauso dröge wie aufwendige Tätigkeit ist, tun Sie besser daran, sich vom System helfen zu lassen. Linux hält hierfür das Programm `useradd` bereit.

`useradd`

Im einfachsten Fall übergeben Sie dem Programm `useradd` lediglich den Namen des neuen Benutzers. Optional können Sie auch noch diverse andere Benutzerparameter setzen; für nicht angegebene Parameter (typischerweise zum Beispiel die UID) werden automatisch »vernünftige« Standardwerte gewählt. Auf Wunsch kann auch das Heimatverzeichnis des Benutzer erzeugt und mit einer Grundausstattung an Dateien versehen werden, die das Programm dem Verzeichnis `/etc/skel` entnimmt. Die Syntax von `useradd` ist:

```
useradd [⟨Optionen⟩] ⟨Benutzername⟩
```

Folgende Optionen stehen dabei unter anderem zur Verfügung:

`-c` ⟨*Kommentar*⟩ Eintrag in in das GECOS-Feld

`-d` ⟨*Heimatverzeichnis*⟩ Fehlt diese Angabe, wird `/home/`⟨*Benutzername*⟩ angenommen

`-e` ⟨*Datum*⟩ Datum, an dem der Zugang automatisch gesperrt wird (Format: »JJJJ-MM-TT«)

`-g` ⟨*Gruppe*⟩ Primäre Gruppe des neuen Benutzers, als Name oder GID. Die Gruppe muss existieren.

`-G` ⟨*Gruppe*⟩**[,**⟨*Gruppe*⟩**]...** Weitere Gruppen, als Namen oder GIDs. Die Gruppen müssen existieren.

`-s` ⟨*Shell*⟩ Login-Shell des Benutzers

`-u` ⟨*UID*⟩ Numerische Benutzerkennung des neuen Benutzers. Die UID darf noch nicht anderweitig vergeben sein, es sei denn, die Option »`-o`« wurde angegeben.

`-m` Legt das Heimatverzeichnis an und kopiert die Datei-Grundausstattung hinein. Diese Grundausstattung kommt aus `/etc/skel`, es sei denn, mit »`-k` ⟨*Verzeichnis*⟩« wurde ein anderes Verzeichnis benannt.

Mit dem Kommando

```
# useradd -c "Hugo Schulz" -m -d /home/hugo -g entw \
>    -k /etc/skel.entw hugo
```

zum Beispiel wird für den Benutzer Hugo Schulz ein Benutzerkonto namens `hugo` angelegt und der Gruppe `entw` zugeordnet. Sein Heimatverzeichnis wird als `/home/hugo` angelegt und die Dateien aus `/etc/skel.entw` als Grundausstattung dort hineinkopiert.

Mit der Option `-D` (bei den SUSE-Distributionen `--show-defaults`) können Sie Vorgabewerte für einige Aspekte neuer Benutzerkonten festlegen. Ohne Zusatzoptionen werden die Werte nur angezeigt:

```
# useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/sh
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
```

Ändern können Sie diese Werte respektive mit den Optionen `-g`, `-b`, `-f`, `-e` und `-s`:

```
# useradd -D -s /usr/bin/zsh                    zsh als Standard-Shell
```

Die letzten beiden Werte in der Liste lassen sich nicht ändern.

`useradd` ist ein relativ niedrig ansetzendes Werkzeug. Im »wirklichen Leben« werden Sie als erfahrener Administrator neue Benutzerkonten wahrscheinlich nicht mit `useradd`, sondern über ein Shellskript anlegen, das Ihre örtlichen Richtlinien mit berücksichtigt (damit Sie nicht immer selber daran denken müssen). Dieses Shellskript müssen Sie leider selbst erstellen – jedenfalls solange Sie nicht Debian GNU/Linux oder eine davon abgeleitete Distribution benutzen (siehe unten).

Vorsicht: Zwar bringt jede ernstzunehmende Linux-Distribution ein Programm namens `useradd` mit, die Implementierungen unterscheiden sich jedoch im Detail.

Die Red-Hat-Distributionen enthalten eine ziemlich »gewöhnliche« Version von `useradd` ohne besondere Extras, die die oben besprochenen Eigenschaften mitbringt.

Das `useradd` der SUSE-Distributionen ist darauf eingerichtet, Benutzer wahlweise in einem LDAP-Verzeichnis statt in `/etc/passwd` anzulegen. (Aus diesem Grund wird die Option `-D` anderweitig verwendet, so dass sie nicht wie bei den anderen Distributionen zum Abfragen oder Setzen von Standardwerten zur Verfügung steht.) Die Details hierzu würden hier etwas zu weit führen.

Bei Debian GNU/Linux und Ubuntu existiert `useradd` zwar, die offizielle Methode zum Anlegen neuer Benutzerkonten ist jedoch ein Programm namens `adduser` (zum Glück gar nicht verwirrend). Der Vorteil von `adduser` besteht darin, dass es die Spielregeln von Debian GNU/Linux einhält und es außerdem ermöglicht, beim Anlegen von Benutzerkonten auch noch beliebige andere Aktionen für das neue Konto auszuführen. Zum Beispiel könnte automatisch ein Verzeichnis im Dokumentbaum eines Web-Servers angelegt werden, in dem der neue Benutzer (und nur dieser) Dateien veröffentlichen kann, oder der Benutzer könnte automatisch zum Zugriff auf einen Datenbankserver autorisiert werden. Die Details stehen in `adduser`(8) und `adduser.conf`(5).

Kennwort

Nach dem Anlegen mit `useradd` ist das neue Konto noch nicht zugänglich; der Systemverwalter muss erst noch ein Kennwort eintragen. Das erklären wir als Nächstes.

11.3.2 Das Kommando `passwd`

Der Befehl `passwd` dient der Vergabe von Benutzerkennwörtern. Wenn Sie als `root` angemeldet sind, dann fragt

```
# passwd hugo
```

nach einem neuen Kennwort für den Benutzer `hugo` (Sie müssen es zweimal angeben, da keine Kontrollausgabe erfolgt).

Das `passwd`-Kommando steht auch normalen Benutzern zur Verfügung, damit sie ihr eigenes Kennwort ändern können (das Ändern der Kennwörter anderer Benutzer ist `root` vorbehalten):

```
$ passwd
Ändern des Passworts für hugo
(aktuelles) UNIX-Passwort: geheim123
Geben Sie ein neues UNIX-Passwort ein: 321mieheg
Geben Sie das neue UNIX-Passwort erneut ein: 321mieheg
passwd: Passwort erfolgreich geändert
```

Normale Benutzer müssen ihr eigenes Kennwort erst einmal richtig angeben, bevor sie ein neues setzen dürfen. Das soll Spaßvögeln das Leben schwer machen, die an Ihrem Computer herumspielen, wenn Sie mal ganz dringend raus mussten und die Bildschirmsperre nicht eingeschaltet haben.

`passwd` dient nebenbei auch noch zur Verwaltung verschiedener Einstellungen in `/etc/shadow`. Sie können sich zum Beispiel den »Kennwortstatus« eines Benutzers anschauen, indem Sie den Befehl `passwd` mit der Option `-S` aufrufen:

```
# passwd -S franz
franz LK 10/15/99 0 99999 7 0
```

Das erste Feld ist dabei (wieder mal) der Benutzername, dann folgt der Kennwortstatus: »`PS`« oder »`P`« heißen hier, dass ein Kennwort gesetzt ist; »`LK`« oder »`L`« stehen für ein gesperrtes Konto, und »`NP`« bezeichnet ein Konto ganz ohne Kennwort. Die übrigen Felder sind respektive das Datum der letzten Kennwortänderung, die Mindest- und Höchstfrist in Tagen für das Ändern des Kennworts, die Warnfrist vor dem Ablauf und die »Gnadenfrist« für die komplette Sperrung des Benutzerkontos nach dem Ablauf des Kennworts. (Siehe hierzu auch Abschnitt 11.2.2.)

Ändern können Sie einige dieser Einstellungen über Optionen von `passwd`. Hier sind ein paar Beispiele:

```
# passwd -l hugo                    Zugang sperren
# passwd -u hugo                    Zugang freigeben
# passwd -n 7 hugo                  Kennwortänderung höchstens alle 7 Tage
# passwd -x 30 hugo                 Kennwortänderung spätestens alle 30 Tage
# passwd -w 3 hugo                  3 Tage Warnfrist vor Kennwortablauf
```

Das Sperren und Freigeben per `-l` und `-u` funktioniert, indem vor das verschlüsselte Kennwort in `/etc/shadow` ein »`!`« gesetzt wird. Da bei der Kennwortverschlüsselung kein »`!`« entsteht, ist es unmöglich, beim Anmelden etwas einzugeben, was auf das »verschlüsselte Kennwort« in der Benutzerdatenbank passt – mithin ist der Zugang über die normale Anmeldeprozedur gesperrt. Sobald das »`!`« entfernt wird, gilt das ursprüngliche Kennwort wieder. (Genial, was?) Allerdings sollten Sie beachten, dass Benutzer sich möglicherweise auch noch auf andere Arten Zugang zum System verschaffen können, die ohne das verschlüsselte Kennwort in der Benutzerdatenbank auskommen – etwa über die Secure Shell mit Public-Key-Authentisierung.

Um die übrigen Einstellungen in `/etc/shadow` zu ändern, müssen Sie das Kommando `chage` heranziehen:

```
# chage -E 2009-12-01 hugo          Sperrung ab 1.12.2009
# chage -E -1 hugo                  Verfallsdatum aufheben
# chage -I 7 hugo                   Gnadenfrist 1 Woche nach Kennwortablauf
# chage -m 7 hugo                   Entspricht passwd -n (Grr.)
# chage -M 7 hugo                   Entspricht passwd -x (Grr, grr.)
# chage -W 3 hugo                   Entspricht passwd -w (Grr, grr, grr.)
```

(`chage` kann alle Parameter ändern, die `passwd` ändern kann, und dann noch ein paar.)

Wenn Sie sich die Optionen nicht merken können, dann rufen Sie `chage` einfach nur mit dem Namen eines Benutzerkontos auf. Das Programm präsentiert Ihnen dann nacheinander die aktuellen Werte zum Bestätigen oder Ändern.

Kennwort im Klartext

Ein bestehendes Kennwort im Klartext auslesen können Sie selbst als Administrator nicht mehr. Auch in der Datei `/etc/shadow` nachzusehen hilft in diesem Fall nichts, denn hier werden alle Kennwörter bereits verschlüsselt abgelegt. Sollte ein Benutzer sein Kennwort vergessen haben, reicht es in der Regel, dessen Kennwort mit `passwd` zu ändern.

Sollten Sie das `root`-Kennwort vergessen haben und gerade nicht als `root` angemeldet sein, bleibt Ihnen noch die Möglichkeit, Linux in eine Shell zu booten oder von einer Rettungsdiskette oder -CD zu booten. (Siehe hierzu Kapitel 16.) Anschließend können Sie mit einem Editor das ⟨*Kennwort*⟩-Feld des `root`-Eintrags in `/etc/passwd` löschen.

Übungen

11.8 [3] Ändern Sie das Kennwort von Benutzer `hugo`. Wie ändert sich die Datei `/etc/shadow`? Fragen Sie den Status zu diesem Kennwort ab.

11.9 [!2] Der Benutzer `trottel` hat sein Kennwort vergessen. Wie können Sie ihm helfen?

11.10 [!3] Stellen Sie die Bedingungen für das Kennwort von Benutzer `hugo` so ein, dass er sein Kennwort frühestens nach einer Woche und spätestens nach zwei Wochen ändern muss. Eine Warnung soll der Benutzer zwei Tage vor Ablauf dieser Zweiwochenfrist erhalten. Kontrollieren Sie anschließend die Einstellungen!

11.3.3 Benutzerkonten löschen

Um ein Benutzerkonto zu löschen, müssen Sie den Eintrag des Benutzers aus `/etc/passwd` und `/etc/shadow` entfernen, alle Verweise auf diesen Benutzer in `/etc/group` löschen und das Heimatverzeichnis sowie alle Dateien entfernen, die der Benutzer erstellt hat oder deren Eigentümer er ist. Wenn der Benutzer z. B. eine Mailbox für eingehende Nachrichten in `/var/mail` hat, sollte auch diese entfernt werden.

`userdel`

Auch für diese Aktionen existiert ein passendes Kommando. Der Befehl `userdel` löscht ein Benutzerkonto komplett. Die Syntax:

```
userdel [-r] ⟨Benutzername⟩
```

Die Option `-r` sorgt dafür, dass das Heimatverzeichnis des Benutzers mit Inhalt sowie seine Mailbox in `/var/mail` entfernt wird, andere Dateien des Benutzers – etwa `crontab`-Dateien usw. – müssen von Hand gelöscht werden. Eine schnelle Methode, die Dateien eines bestimmten Benutzers zu finden und zu löschen, ist der Befehl

```
find / -uid ⟨UID⟩ -delete
```

Ohne die Option `-r` werden nur die Benutzerdaten aus der Benutzerdatenbank gelöscht; das Heimatverzeichnis bleibt stehen.

11.3.4 Benutzerkonten und Gruppenzuordnung ändern

Eine Änderung der Benutzerkonten und -gruppen geschieht traditionell durch das Editieren der Dateien `/etc/passwd` und `/etc/group`. Viele Systeme enthalten auch Befehle wie `usermod` und `groupmod` für denselben Zweck, die Sie im Zweifelsfall vorziehen sollten, weil sie sicherer funktionieren und – meistens – auch bequemer einzusetzen sind.

Das Programm `usermod` akzeptiert im Wesentlichen dieselben Optionen wie `useradd`, aber ändert existierende Benutzerkonten, statt neue anzulegen. Beispielsweise können Sie mit

usermod

```
usermod -g ⟨Gruppe⟩ ⟨Benutzername⟩
```

die primäre Gruppe eines Benutzers ändern.

Achtung! Wenn Sie die UID eines bestehenden Benutzerkontos ändern möchten, können Sie natürlich das ⟨*UID*⟩-Feld in `/etc/passwd` direkt editieren. Allerdings sollten Sie gleichzeitig mit `chown` die Dateiberechtigungen der Dateien dieses Benutzers auf die neue UID übertragen: »`chown -R tux /home/tux`« vergibt die Eignerrechte an allen Dateien in dem Heimatverzeichnis, das von `tux` benutzt wurde, wieder an `tux`, nachdem Sie die UID für dieses Konto geändert haben. Wenn »`ls -l`« eine numerische UID statt eines alphanumerischen Namens ausgibt, weist dies darauf hin, dass mit der UID dieser Dateien kein Benutzername verbunden ist. Mit `chown` können Sie das in Ordnung bringen.

UID ändern

11.3.5 Die Benutzerdatenbank direkt ändern — `vipw`

Das Kommando `vipw` ruft einen Editor (`vi` oder einen anderen) auf, um die Datei `/etc/passwd` zu ändern. Gleichzeitig wird die jeweilige Datei gesperrt, so dass während dieser Änderung niemand z. B. mit `passwd` ebenfalls eine Änderung vornehmen kann (die dann verloren gehen würde). Mit der Option `-s` wird `/etc/shadow` bearbeitet.

Welcher Editor konkret aufgerufen wird, bestimmt der Wert der Umgebungsvariablen `VISUAL`, ersatzweise der Wert der Umgebungsvariablen `EDITOR`; existiert keine der beiden, wird der `vi` gestartet.

Übungen

11.11 [!2] Legen Sie den Benutzer `test` an. Wechseln Sie auf das Benutzerkonto `test` und legen Sie mit `touch` einige Dateien an, einige davon in einem anderen Ordner als dem Heimatverzeichnis (etwa `/tmp`). Wechseln Sie wieder zurück zu `root` und ändern Sie die UID von `test`. Was sehen Sie, wenn Sie mit `ls` die Dateien von Benutzer `test` auflisten?

11.12 [!2] Legen Sie einen Benutzer `test1` über das entsprechende grafische Tool an, einen anderen, `test2`, mit Hilfe des Kommandos `useradd` und einen weiteren, `test3`, von Hand. Betrachten Sie die Konfigurationsdateien. Können Sie problemlos unter allen drei Konten arbeiten? Legen Sie unter jedem Konto eine Datei an.

11.13 [!2] Löschen Sie das Konto von Benutzer `test2` und stellen Sie sicher, dass es auf dem System keine Dateien mehr gibt, die dem Benutzer gehören!

11.14 [2] Ändern Sie die UID des Benutzers `test1`. Was müssen Sie außerdem tun?

11.15 [2] Ändern Sie das Heimatverzeichnis für Benutzer `test1` um von `/home/test1` in `/home/user/test1`.

11.3.6 Anlegen, Ändern und Löschen von Gruppen

Genau wie Benutzerkonten können Sie Gruppen auf mehrere Arten anlegen. Die Methode »von Hand« gestaltet sich hier wesentlich weniger aufwendig als das manuelle Erstellen neuer Benutzerkonten: Da Gruppen kein Heimatverzeichnis

besitzen, genügt es normalerweise, die Datei `/etc/group` mit einem beliebigen Texteditor zu manipulieren und eine entsprechende neue Zeile einzufügen (siehe unten für `vigr`). Bei Verwendung von Gruppenkennwörtern ist auch noch ein Eintrag in `/etc/gshadow` vorzunehmen.

Übrigens spricht nichts dagegen, daß auch Benutzergruppen ein eigenes Verzeichnis erstellt bekommen. Dort können die Gruppenmitglieder dann die Früchte der gemeinsamen Arbeit ablegen. Die Vorgehensweise ist dabei dem Anlegen von Benutzerheimatverzeichnissen ähnlich, allerdings braucht keine Grundausstattung an Konfigurationsdateien kopiert werden.

Zur Gruppenverwaltung gibt es analog zu `useradd`, `usermod` und `userdel` die Programme `groupadd`, `groupmod` und `groupdel`, auf die Sie zurückgreifen sollten, statt `/etc/group` und `/etc/gshadow` direkt zu editieren. Mit `groupadd` können Sie einfach per Kommandozeilenparameter bzw. Voreinstellungen neue Gruppen im System erzeugen:

groupadd

```
groupadd [-g ⟨GID⟩] ⟨Gruppenname⟩
```

Die Option `-g` ermöglicht die Vorgabe einer bestimmten Gruppennummer. Wie erwähnt handelt es sich dabei um eine positive ganze Zahl. Die Werte bis 99 sind meist Systemgruppen vorbehalten. Ohne Angabe wird die nächste freie GID verwendet.

Bereits bestehende Gruppen können Sie mit `groupmod` bearbeiten, ohne direkt in `/etc/group` schreiben zu müssen:

groupmod

```
groupmod [-g ⟨GID⟩] [-n ⟨Name⟩] ⟨Gruppenname⟩
```

Die Option »`-g` ⟨*GID*⟩« ändert die GID der angegebenen Gruppe. Nicht aufgelöste Gruppenzugehörigkeiten von Dateien müssen danach manuell angepasst werden. Die Option »`-n` ⟨*Name*⟩« weist der angegebenen Gruppe einen neuen Gruppennamen zu. Die GID bleibt dabei unverändert, auch manuelle Anpassungen sind nicht erforderlich.

Auch zum Entfernen der Gruppeneinträge existiert ein Hilfsprogramm. Dieses heißt konsequenterweise `groupdel`:

groupdel

```
groupdel ⟨Gruppenname⟩
```

Ebenso wie beim manuellen Entfernen von Gruppeneinträgen empfiehlt es sich auch hier, anschließend alle Dateien des Verzeichnisbaums zu überprüfen und verwaiste Gruppenzugehörigkeiten per `chgrp` anzupassen. Primäre Gruppen von einzelnen Benutzern dürfen nicht entfernt werden. Entweder muss der betroffene Benutzer vor dem Entfernen der Gruppe ebenfalls ausgetragen oder einer anderen primären Gruppe zugeteilt werden.

gpasswd

Das `gpasswd`-Kommando dient in erster Linie zur Manipulation von Gruppenkennwörtern analog zum `passwd`-Kommando. Allerdings kann der Systemadministrator die Verwaltung der Mitgliederliste einer Gruppe an einen oder mehrere Gruppenadministratoren delegieren. Gruppenadministratoren verwenden dafür ebenfalls das `gpasswd`-Kommando:

Gruppenadministrator

```
gpasswd -a ⟨Benutzer⟩ ⟨Gruppe⟩
```

fügt den ⟨*Benutzer*⟩ der ⟨*Gruppe*⟩ hinzu, und

```
gpasswd -d ⟨Benutzer⟩ ⟨Gruppe⟩
```

entfernt ihn wieder daraus. Mit

```
gpasswd -A ⟨Benutzer⟩,… ⟨Gruppe⟩
```

kann der Systemadministrator Benutzer benennen, die als Gruppenadministratoren fungieren sollen.


In den SUSE-Distributionen ist `gpasswd` seit einer Weile nicht mehr enthalten. Statt dessen gibt es modifizierte Versionen der Programme zur Benutzer- und Gruppenverwaltung, die mit einem LDAP-Verzeichnis umgehen können.

Direkt ändern können Sie die Gruppendatenbank als Systemverwalter mit dem Kommando `vigr`. Es funktioniert analog zu `vipw`, ruft also einen Editor auf, mit dem Sie »exklusiven« Zugriff auf `/etc/group` haben. Entsprechend bekommen Sie mit »`vigr -s`« Zugriff auf `/etc/gshadow`. `vigr`

Übungen


11.16 [2] Wozu werden Gruppen gebraucht? Geben Sie mögliche Beispiele an!


11.17 [1] Können Sie ein Verzeichnis anlegen, auf das alle Mitglieder einer Gruppe Zugriff haben?

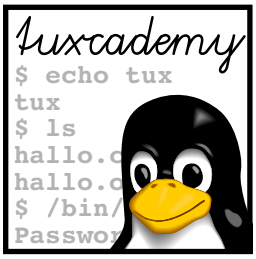

11.18 [!2] Erstellen Sie eine zusätzliche Gruppe `test`. Mitglied dieser Gruppe soll nur `test1` sein. Setzen Sie ein Gruppenkennwort. Melden Sie sich als `test1` und `test2` an und wechseln Sie jeweils in die neue Gruppe!

Kommandos in diesem Kapitel

adduser	Komfortables Kommando zum Anlegen neuer Benutzerkonten (Debian)	adduser(8)	188
chage	Setzt Kennwort-Attribute wie Ablaufdatum und Änderungsfristen	chage(1)	189
chfn	Erlaubt Benutzern das Ändern des GECOS-Felds in der Benutzerdatenbank	chfn(1)	180
getent	Ruft Einträge aus administrativen Datenbanken ab	getent(1)	185
gpasswd	Erlaubt Verwaltung von Gruppenmitgliederlisten durch Gruppenadministratoren und das Setzen des Gruppenkennworts	gpasswd(1)	192
groupadd	Fügt Gruppen in die Gruppendatenbank ein	groupadd(8)	192
groupdel	Löscht Gruppen aus der Gruppendatenbank	groupdel(8)	192
groupmod	Ändert Gruppeneinträge in der Gruppendatenbank	groupmod(8)	192
groups	Gibt die Gruppen aus, in denen ein Benutzer Mitglied ist	groups(1)	176
id	Gibt UID und GIDs eines Benutzers aus	id(1)	176
last	Zeige zuletzt angemeldete Benutzer an	last(1)	176
useradd	Fügt neue Benutzerkonten hinzu	useradd(8)	187
userdel	Entfernt Benutzerkonten	userdel(8)	190
usermod	Modifiziert die Benutzerdatenbank	usermod(8)	190
vigr	Erlaubt das exklusive Ändern von `/etc/group` bzw. `/etc/gshadow`	vipw(8)	193

Zusammenfassung

- Der Zugriff auf das System wird über Benutzerkonten geregelt.
- Jedes Benutzerkonto hat eine numerische UID und (mindestens) einen zugeordneten textuellen Benutzernamen.
- Benutzer können in Gruppen zusammengefasst werden. Gruppen haben Namen und numerische GIDs.
- »Pseudobenutzer« und »-gruppen« dienen zur weiteren Strukturierung der Zugriffsrechte.
- Die zentrale Benutzerdatenbank steht (normalerweise) in der Datei `/etc/passwd`.
- Die verschlüsselten Kennwörter der Benutzer stehen – zusammen mit anderen Kennwortparametern – in der Datei `/etc/shadow`, die nicht mehr für alle Benutzer lesbar ist.
- Gruppeninformationen stehen in den Dateien `/etc/group` und `/etc/gshadow`.
- Kennwörter werden mit dem Programm `passwd` verwaltet.
- Zur Konfiguration der Kennwortparameter in `/etc/shadow` dient das Programm `chage`.
- Benutzerinformationen ändert man mit dem Programm `vipw` oder – besser – mit den dafür vorgesehenen Werkzeugen `useradd`, `usermod` und `userdel`.
- Gruppeninformationen kann man über die Programme `groupadd`, `groupmod`, `groupdel` und `gpasswd` manipulieren.



12

Zugriffsrechte

Inhalt

12.1 Das Linux-Rechtekonzept 196
12.2 Zugriffsrechte auf Dateien und Verzeichnisse 196
12.2.1 Grundlagen 196
12.2.2 Zugriffsrechte anschauen und ändern 197
12.2.3 Dateieigentümer und Gruppe setzen – `chown` und `chgrp` 198
12.2.4 Die *umask* 199
12.3 Zugriffskontrolllisten (ACLs) 201
12.4 Eigentum an Prozessen 202
12.5 Besondere Zugriffsrechte für ausführbare Dateien 202
12.6 Besondere Zugriffsrechte für Verzeichnisse 203
12.7 Dateiattribute 205

Lernziele

- Das Linux-Rechtekonzept beherrschen
- Zugriffsrechte für Dateien und Verzeichnisse vergeben können
- Die Begriffe *umask*, SUID, SGID und *sticky bit* kennen
- Die Dateiattribute der `ext`-Dateisysteme kennen

Vorkenntnisse

- Kenntnisse über das Linux-Benutzer- und Gruppenkonzept (siehe Kapitel 11)
- Kenntnisse über das Linux-Dateikonzept

12.1 Das Linux-Rechtekonzept

Rechtekonzept

Überall, wo mehrere Benutzer auf einem System gleichzeitig arbeiten, muss auch ein Rechtekonzept für Prozesse, Dateien und Verzeichnisse existieren, damit Benutzer *A* nicht ohne weiteres auf Dateien von Benutzer *B* zugreifen kann. Linux implementiert dazu das Standardkonzept aller Unix-Betriebssysteme.

getrennte Rechte

In diesem Konzept sind jede Datei und jedes Verzeichnis genau einem Benutzer (Besitzer) und einer Gruppe zugeordnet. Jede Datei hat getrennte Rechte für den Besitzer, die Mitglieder der Gruppe, der die Datei zugeordnet ist (kurz »die Gruppe« genannt), und alle anderen Benutzer (der »Rest der Welt«). Für diese drei Mengen von Benutzern können jeweils separat Schreib-, Lese- und Ausführungsrechte vergeben werden. Der Eigentümer darf die Zugriffsrechte einer Datei bestimmen. Die Gruppe und alle anderen Benutzer haben nur dann Rechte an einer Datei, wenn der Eigentümer ihnen diese Rechte einräumt. Die gesamte Einstellung der Rechte für eine Datei heißt auch deren **Zugriffsmodus**.

Zugriffsmodus

Zugriffsbeschränkung

In einem Mehrbenutzersystem, in dem private oder gruppeninterne Daten auf einem allgemein zugänglichen Medium gespeichert werden, kann der Eigentümer einer Datei durch entsprechende Zugriffsbeschränkung eine Veränderung oder das Lesen seiner Daten verbieten. Die Rechte einer Datei können für den Eigentümer, die Benutzergruppe und die sonstigen Benutzer separat und unabhängig voneinander festgelegt werden. Dadurch lassen sich bereits mit Hilfe der Zugriffsrechte die Kompetenzen und Zuständigkeiten eines Gruppenarbeitsprozesses auf die Dateien, mit denen die Gruppe arbeitet, abbilden.

12.2 Zugriffsrechte auf Dateien und Verzeichnisse

12.2.1 Grundlagen

Für jede Datei und jedes Verzeichnis im System erlaubt Linux für jede der drei Kategorien von Benutzern – Eigentümer, Mitglieder der Dateigruppe, Rest der Welt – separate Zugriffsrechte. Diese Rechte teilen sich auf in Leserecht, Schreibrecht und Ausführungsrecht.

Zugriffsrechte für Dateien

Für Dateien bedeuten diese Rechte in etwa das, was ihre Namen aussagen: Wer Leserecht hat, darf sich den Inhalt der Datei anschauen, wer Schreibrecht hat, darf ihren Inhalt ändern. Das Ausführungsrecht ist notwendig, um die Datei als Programm starten zu können.

Zum Starten eines binären »Maschinenprogramms« ist nur Ausführungsrecht nötig. Für Dateien mit Shellskripten oder anderen »interpretierten« Programmen brauchen Sie außerdem Leserecht.

Zugriffsrechte für Verzeichnisse

Bei Verzeichnissen sieht es etwas anders aus: Leserecht ist notwendig, um den Inhalt eines Verzeichnisses anschauen zu können – etwa indem Sie das `ls`-Kommando ausführen. Schreibrecht brauchen Sie, um Dateien im Verzeichnis anlegen, löschen oder umbenennen zu können. Das »Ausführungsrecht« steht für die Möglichkeit, das Verzeichnis »benutzen« zu dürfen, in dem Sinne, dass Sie mit `cd` hineinwechseln oder seinen Namen in Pfadnamen von weiter unten im Dateibaum liegenden Dateien benutzen dürfen.

In Verzeichnissen, wo Sie nur das Leserecht haben, können Sie die Dateinamen lesen, aber nichts Anderes über die Dateien herausfinden. Haben Sie für ein Verzeichnis nur »Ausführungsrecht«, dann können Sie Dateien ansprechen, solange Sie wissen, wie sie heißen.

Normalerweise hat es wenig Sinn, Schreib- und Ausführungsrecht für Verzeichnisse getrennt voneinander zu vergeben; in gewissen Spezialfällen kann es allerdings nützlich sein.

Es ist wichtig, hervorzuheben, dass Schreibrecht auf eine *Datei* für das *Löschen* der Datei völlig unnötig ist – Sie brauchen Schreibrecht auf das *Verzeichnis*, in dem die Datei steht, und sonst nichts! Da beim »Löschen« nur ein Verweis auf die tatsächliche Dateiinformation (die Inode) aus dem Verzeichnis entfernt wird, handelt es sich dabei um eine reine Verzeichnisoperation. Das `rm`-Programm warnt Sie zwar, wenn Sie versuchen, eine Datei zu löschen, auf die Sie kein Schreibrecht haben, aber wenn Sie die Operation bestätigen und Schreibrecht auf das Verzeichnis der Datei haben, steht dem erfolgreichen Löschen nichts im Weg. (Linux kennt – wie jedes andere Unix-artige System auch – keine Möglichkeit, eine Datei direkt zu »löschen«; Sie können nur alle Verweise auf die Datei entfernen, woraufhin der Linux-Kernel von sich aus entscheidet, dass niemand mehr auf die Datei zugreifen kann, und ihren Inhalt entsorgt.)

Wenn Sie dagegen Schreibrecht auf die Datei, aber nicht auf ihr Verzeichnis haben, können Sie die Datei nicht komplett löschen. Sie können aber natürlich die Länge der Datei auf 0 setzen und damit den *Inhalt* entfernen, auch wenn die Datei selbst prinzipiell noch existiert.

Für jeden Benutzer gelten diejenigen Zugriffsrechte, die »am besten passen«. Haben zum Beispiel die Mitglieder der Gruppe, der eine Datei zugeordnet ist, kein Leserecht für die Datei, der »Rest der Welt« dagegen schon, dann dürfen die Gruppenmitglieder nicht lesen. Das auf den ersten Blick verlockende Argument, dass, wenn schon der Rest der Welt die Datei lesen darf, die Gruppenmitglieder, die ja in gewissem Sinne auch Teil des Rests der Welt sind, das eigentlich auch dürfen sollten, zählt nicht.

12.2.2 Zugriffsrechte anschauen und ändern

Informationen

Informationen darüber, welche Rechte, Benutzer- und Gruppenzuordnung für eine Datei gelten, bekommen Sie mit »`ls -l`«:

```
$ ls -l
-rw-r--r--  1  hugo  users   4711  Oct 4 11:11 datei.txt
drwxr-x---  2  hugo  group2  4096  Oct 4 11:12 testdir
```

Die Zeichenkette am linken Rand der Tabelle gibt die Zugriffsrechte für den Eigentümer, die Dateigruppe und den »Rest der Welt« an (das allererste Zeichen ist nur der Dateityp und hat mit den Zugriffsrechten nichts zu tun). Die dritte Spalte nennt den Benutzernamen des Eigentümers und die vierte Spalte den Namen der Dateigruppe.

In der Rechteliste stehen »`r`«, »`w`« und »`x`« jeweils für ein vorhandenes Lese-, Schreib- und Ausführungsrecht. Steht nur ein »`-`« in der Liste, dann hat die betreffende Kategorie das betreffende Recht nicht. »`rw-r--r--`« steht also für »Lese- und Schreibrecht für den Eigentümer, aber nur Leserecht für die Gruppenmitglieder und den Rest der Welt«.

Befehl `chmod`

Als Dateieigentümer können Sie mit dem Befehl `chmod` (von *change mode*, »Zugriffsmodus ändern«) die Zugriffsrechte für die Datei setzen. Dabei können Sie die drei Kategorien durch die Abkürzungen »`u`« (*user*) für den Eigentümer (Sie selbst), »`g`« (*group*) für die Mitglieder der Dateigruppe und »`o`« (*others*) für den »Rest der Welt« bestimmen. Die Rechte selbst werden durch die schon erwähnten Kürzel »`r`«, »`w`« und »`x`« festgelegt. Mit »`+`«, »`-`« und »`=`« können Sie verfügen, ob die benannten Rechte respektive zusätzlich zu den existierenden vergeben, von den existierenden »subtrahiert« oder anstelle der bisherigen gesetzt werden sollen. Zum Beispiel:

```
$ chmod u+x datei          Ausführungsrecht für Eigentümer
$ chmod go+w datei         setzt Schreibrecht für Gruppe und RdW
$ chmod g+rw datei         setzt Lese- und Schreibrecht für die Gruppe
```

```
$ chmod g=rw,o=r datei          setzt Lese- und Schreibrecht,
                                löscht Ausführungsrecht für die Gruppe
                                setzt reines Leserecht für den Rest der Welt
$ chmod a+w datei               äquivalent zu ugo+w
```

Tatsächlich sind Rechtespezifikationen um einiges komplexer. Konsultieren Sie die `info`-Dokumentation zu `chmod`, um die Details herauszufinden.

Der Dateieigentümer ist (neben `root`) der einzige Benutzer, der die Zugriffsrechte für eine Datei oder ein Verzeichnis ändern darf. Dieses Privileg ist unabhängig von den tatsächlichen Dateirechten; der Eigentümer darf sich selbst alle Rechte entziehen, aber hindert sich dadurch nicht selber daran, sich später wieder Rechte zu erteilen.

Die allgemeine Syntax des `chmod`-Kommandos ist

```
chmod [⟨Optionen⟩] ⟨Rechte⟩ ⟨Name⟩ …
```

Es können beliebig viele Datei- oder Verzeichnisnamen angegeben werden. Die wichtigsten Optionen sind:

`-R` Wenn ein Verzeichnis angegeben wurde, werden auch die Rechte von Dateien und Verzeichnissen innerhalb dieses Verzeichnisses geändert usw.

`--reference=`⟨*Name*⟩ Verwendet die Zugriffsrechte der Datei ⟨*Name*⟩. In diesem Fall müssen keine ⟨*Rechte*⟩ angegeben werden.

Numerische Rechtedarstellung

Sie können den Zugriffsmodus einer Datei statt wie eben angegeben »symbolisch« auch »numerisch« angeben. In der Praxis ist das sehr verbreitet, wenn Sie alle Rechte für eine Datei oder ein Verzeichnis auf einmal setzen wollen, und funktioniert so: Die drei Rechtetripel werden als dreistellige Oktalzahl dargestellt – die erste Ziffer beschreibt die Rechte des Eigentümers, die zweite die Rechte der Dateigruppe und die dritte die Rechte für den »Rest der Welt«. Jede dieser Ziffern ergibt sich aus der Summe der jeweiligen Rechte, wobei Leserecht 4 zählt, Schreibrecht 2 und Ausführrecht 1. Hier sind ein paar Beispiele für gängige Rechtezuordnungen in »`ls -l`«- und oktaler Darstellung:

```
rw-r--r--  644
r--------  400
rwxr-xr-x  755
```

Mit der numerischen Rechtedarstellung können Sie nur alle Rechte auf einmal setzen – es gibt keine Möglichkeit, wie mit den »`+`«- und »`-`«-Operatoren der symbolischen Darstellung einzelne Rechte zu setzen oder zu entfernen und die anderen dabei unbehelligt zu lassen. Das Kommando

```
$ chmod 644 datei
```

entspricht also der symbolischen Form

```
$ chmod u=rw,go=r datei
```

12.2.3 Dateieigentümer und Gruppe setzen – `chown` und `chgrp`

Das Kommando `chown` erlaubt das Setzen des Datei- oder Verzeichniseigentümers und der Gruppe. Dem Befehl werden die Benutzerkennung des Besitzers und/oder die gewünschte Gruppenkennung und der Dateiname bzw. Verzeichnisname, dessen Eigentümer geändert werden soll, übergeben. Der Aufruf sieht so aus:

```
chown ⟨Benutzername⟩[:][⟨Gruppenname⟩] ⟨Name⟩ …
chown :⟨Gruppenname⟩ ⟨Name⟩ …
```

Werden Benutzername und Gruppenkennung angegeben, werden beide gesetzt; wird nur ein Benutzername angegeben, bleibt die Gruppe so, wie sie war; wird ein Benutzername mit Doppelpunkt angegeben, dann wird die Datei der primären Gruppe des Benutzers zugeordnet; wird nur ein Gruppenname angegeben (mit Doppelpunkt), dann bleibt der Eigentümer so, wie er war. Zum Beispiel:

```
# chown hugo:entw brief.txt
# chown www-data bla.html                    Neuer Benutzer www-data
# chown :entw /home/entwicklung              Neue Gruppe entw
```

`chown` unterstützt auch eine veraltete Syntax, bei der anstatt des Doppelpunkts ein einfacher Punkt verwendet wird.

Um Dateien an andere Benutzer oder beliebige Gruppen zu »verschenken«, müssen Sie Systemverwalter sein. Der Hauptgrund dafür ist, dass normale Benutzer sonst einander ärgern können, wenn das System »Kontingentierung« (Quotas) verwendet, also jeder Benutzer nur über eine bestimmte Menge Plattenplatz verfügen darf.

Mit dem Kommando `chgrp` können Sie die Gruppe einer Datei ändern, und zwar auch als normaler Benutzer – solange Sie Eigentümer der Datei sowie Mitglied der *neuen* Gruppe sind:

```
chgrp ⟨Gruppenname⟩ ⟨Name⟩ …
```

Änderungen des Dateieigentümers oder der Dateigruppe ändern die Zugriffsrechte für die verschiedenen Kategorien nicht.

Auch `chown` und `chgrp` unterstützen die Option `-R` zur rekursiven Anwendung auf eine ganze Verzeichnishierarchie.

Selbstverständlich können Sie auch mit den meisten Dateibrowsern (wie Konqueror oder Nautilus) Rechte, Gruppe und Eigentümer einer Datei ändern.

Übungen

12.1 [!2] Legen Sie eine neue Datei an. Welcher Gruppe ist diese Datei zugeordnet? Verwenden Sie `chgrp`, um die Datei einer Ihrer anderen Gruppen zuzuordnen. Was passiert, wenn Sie die Datei einer Gruppe zuordnen wollen, in der Sie nicht Mitglied sind?

12.2 [4] Vergleichen Sie die Mechanismen, die verschiedene Dateibrowser (zum Beispiel Konqueror, Nautilus, …) zum Setzen von Dateirechten, Eigentümer, Gruppe, … anbieten. Gibt es nennenswerte Unterschiede?

12.2.4 Die *umask*

Neue Dateien werden normalerweise mit dem (oktalen) Zugriffsmodus 666 (Lese- und Schreibrecht für alle) angelegt. Neue Verzeichnisse erhalten den Zugriffsmodus 777. Da das nicht immer so gewollt ist, bietet Linux einen Mechanismus an, mit dem Sie gezielt einzelne Rechte aus diesen Zugriffsmodi entfernen können. Dieser Mechanismus heißt *umask*.

Niemand weiß genau, wo dieser Name herkommt – auch wenn es ein paar Theorien gibt, die aber alle ziemlich unplausibel klingen.

umask-Interpretation

Die *umask* ist eine Oktalzahl, deren Komplement mit dem Standardzugriffsmodus – 666 oder 777 – bitweise UND-verknüpft wird, um den tatsächlichen Zugriffsmodus für die neue Datei oder das Verzeichnis zu erhalten. Mit anderen Worten: Die *umask* können Sie als Zugriffsmodus interpretieren, in dem genau diejenigen Rechte gesetzt sind, die die neue Datei gerade *nicht* haben soll. Ein Beispiel – die *umask* sei 027:

1.	Wert der *umask*:	027	`----w-rwx`
2.	Komplement davon:	750	`rwxr-x---`
3.	Zugriffsmodus einer neuen Datei:	666	`rw-rw-rw-`
4.	Resultat (2 und 3 UND-verknüpft):	640	`rw-r-----`

Die dritte Spalte zeigt die oktalen Werte, die vierte eine symbolische Schreibweise. Die UND-Verknüpfung im Schritt 4 können Sie auch leicht an der vierten Spalte der 2. und 3. Zeile ablesen: In der 4. Zeile ist an jeder Position ein Buchstabe, wo in der 2. *und* der 3. Zeile ein Buchstabe stand – ist auch nur ein Strich (»`-`«) dabei, steht im Resultat auch ein Strich.

Wenn Sie nicht mit Komplement und UND-Verknüpfung operieren möchten, können Sie sich auch einfach vorstellen, dass die *umask* ziffernweise vom oktalen Zugriffsmodus subtrahiert wird und negative Ergebnisse als Null gelten (wir »leihen« uns auch nichts von der nächsten Stelle weiter links). Das heißt für unser Beispiel – Zugriffsmodus ist 666 und *umask* 027 – etwas wie

$$666 \ominus 027 = 640,$$

weil $6 \ominus 0 = 6$, $6 \ominus 4 = 2$ und $6 \ominus 7 = 0$.

Shellkommando `umask`

Die *umask* wird mit dem Shellkommando `umask` gesetzt, das Sie entweder explizit aufrufen oder in einer Startdatei Ihrer Shell – typischerweise `~/.profile`, `~/.bash_profile` oder `~/.bashrc` – setzen können. Die *umask* ist ein Prozessattribut, ähnlich dem aktuellen Verzeichnis oder der Prozessumgebung, das heisst, sie vererbt sich an Kindprozesse, aber Änderungen der *umask* in einem Kindprozess wirken sich nicht auf den Elterprozess aus.

Prozessattribut

Syntax

Das Kommando `umask` bekommt einen Parameter mit auf den Weg, der die gewünschte *umask* angibt:

```
umask [-S|⟨Umask⟩]
```

Symbolische Form

Die *umask* kann als Oktalzahl oder in der beim Kommando `chmod` beschriebenen symbolischen Form angegeben werden – das Gemeine an dieser Stelle ist, dass die symbolische Form die Rechte enthält, die maximal *übrigbleiben* sollen:

```
$ umask 027                                        … ist dasselbe wie …
$ umask u=rwx,g=rx,o=
```

Das heißt, in der symbolischen Form müssen Sie genau das Komplement des Werts ausdrücken, den Sie in der oktalen Form angeben würden – genau die Rechte, die in der oktalen Schreibweise *nicht* vorkommen.

Wenn Sie keinen Wert angeben, zeigt `umask` die aktuelle Maske an. Wenn die Option `-S` gesetzt ist, wird die aktuelle Maske in symbolischer Form ausgegeben (wobei auch hier wieder die Rechte angezeigt werden, die maximal übrigbleiben, nachdem die *umask* angewendet wurde):

```
$ umask
0027
$ umask -S
u=rwx,g=rx,o=
```

Beachten Sie, dass Sie mit der *umask* nur Zugriffsrechte *entfernen* können. Es gibt keine Möglichkeit, einer Datei automatisch Ausführrecht zu geben.

Automatisch Ausführrecht?

Die *umask* hat übrigens Einfluss auf das Kommando `chmod`. Wenn Sie bei `chmod` mit einem »+«-Modus aufrufen (etwa »`chmod +w datei`«), ohne sich dabei auf Eigentümer, Gruppe oder »Rest der Welt« zu beziehen, dann ist das äquivalent zu »`a+`«, aber die in der *umask* gesetzten Rechte werden nicht modifiziert. Betrachten Sie das folgende Beispiel:

umask und `chmod`

```
$ umask 027
$ touch file
$ chmod +x file
$ ls -l file
-rwxr-x---   1 tux   users        0 May 25 14:30 file
```

Das »`chmod +x`« setzt das Ausführrecht für den Dateieigentümer und die Gruppe, aber nicht für den »Rest der Welt«, da die *umask* das Ausführrecht für den »Rest der Welt« enthält. Mit der *umask* können Sie also Vorkehrungen dagegen treffen, Dateien zu viele Rechte zu geben.

Theoretisch funktioniert dasselbe auch für die `chmod`-Operatoren »`-`« und »`=`«, was aber in der Praxis keinen großen Sinn ergibt.

Übungen

12.3 [!1] Wie muss eine numerische *umask* aussehen, die dem Benutzer alle Rechte lässt, aber Gruppenmitgliedern und dem »Rest der Welt« alle Rechte entzieht? Was ist die entsprechende symbolische *umask*?

12.4 [2] Vergewissern Sie sich, dass der Unterschied zwischen den Kommandos »`chmod +x`« und »`chmod a+x`« wirklich so aussieht wie beschrieben.

12.3 Zugriffskontrolllisten (ACLs)

Wie erklärt, erlaubt Linux die getrennte Vergabe von Zugriffsrechten für den Dateieigentümer, die Mitglieder der Dateigruppe und den »Rest der Welt«. Für manche Anwendungen ist dieses dreistufige System aber zu einfach, oder differenzierte Rechtesysteme von anderen Betriebssystemen müssen auf Linux-Systemen abgebildet werden. Hierzu dienen Zugriffskontrolllisten (*access control lists*, *ACLs*).

Linux unterstützt auf den meisten Dateisystemen »POSIX-ACLs« gemäß IEEE 1003.1e (Entwurf 17) mit ein paar Linux-spezifischen Erweiterungen. Hiermit können Sie für Dateien und Verzeichnisse neben der Dateigruppe weitere Gruppen und einzelne Benutzer benennen, denen Sie dann Lese-, Schreib- oder Ausführungsrechte zuordnen, die von denen der Dateigruppe bzw. dem »Rest der Welt« abweichen. Andere Rechte, etwa die der Rechtezuordnung, bleiben dem Dateieigentümer bzw. `root` vorbehalten und können auch mit ACLs nicht weiterdelegiert werden. Die Kommandos `setfacl` und `getfacl` dienen zum Setzen und Abfragen solcher ACLs.

POSIX-ACLs

ACLs sind eine vergleichsweise neue und eher selten verwendete Eigenschaft, und ihre Verwendung unterliegt gewissen Einschränkungen. Der Kernel überwacht zwar ihre Einhaltung, aber zum Beispiel ist nicht jedes Programm in der Lage, ACLs etwa bei Kopieroperationen mitzukopieren – Sie müssen möglicherweise ein angepasstes `tar` (`star`) verwenden, um von einem Dateisystem mit ACLs Sicherheitskopien zu machen. ACLs vertragen sich mit Samba, so dass Windows-Clients die richtigen Rechte zu sehen bekommen, aber wenn Sie Dateisysteme über NFS an andere (proprietäre) Unix-Systeme exportieren, könnte es sein, dass Ihre ACLs von Unix-Clients, die ACLs nicht unterstützen, ignoriert werden.


Nachlesen über ACLs unter Linux können Sie auf `http://acl.bestbits.at/` und in `acl`(5) sowie `getfacl`(1) und `setfacl`(1).

Detaillierte Kenntnisse über ACLs sind für die LPIC-1-Prüfung nicht erforderlich.

12.4 Eigentum an Prozessen

Linux betrachtet nicht nur die mehr oder weniger festen Daten auf einem dauerhaften Speichermedium als Objekte, die einem Eigentümer zugeordnet werden. Auch die Prozesse im System haben einen Eigentümer.

Viele Kommandos, die Sie über die Tastatur eingeben, erzeugen einen Prozess im Arbeitsspeicher des Rechners. Im normalen Betrieb befinden sich immer mehrere Prozesse gleichzeitig im Speicher, die vom Betriebssystem streng voneinander abgegrenzt werden. Die einzelnen Prozesse werden mit allen ihren Daten einem Benutzer als Eigentümer zugeordnet. Dies ist in der Regel der Benutzer, der den Prozess gestartet hat – auch hier haben Prozesse, die mit Administratorrechten gestartet wurden, die Möglichkeit, ihre Identität zu ändern, und der SUID-Mechanismus (Abschnitt 12.5) kann hier ebenfalls eingreifen.

Prozesse haben auch Eigentümer

Die Eigentümer der Prozesse werden vom Programm `ps` angezeigt, wenn es mit der Option `-u` aufgerufen wird.

```
# ps -u
USER   PID %CPU  %MEM SIZE      RSS TTY STAT  START  TIME COMMAND
bin     89  0.0   1.0  788      328   ?    S  13:27  0:00 rpc.portmap
test1  190  0.0   2.0 1100       28   3    S  13:27  0:00 bash
test1  613  0.0   1.3  968       24   3    S  15:05  0:00 vi XF86.tex
nobody 167  0.0   1.4  932       44   ?    S  13:27  0:00 httpd
root     1  0.0   1.0  776       16   ?    S  13:27  0:03 init [3]
root     2  0.0   0.0    0        0   ?   SW  13:27  0:00 (kflushd)
```

12.5 Besondere Zugriffsrechte für ausführbare Dateien

Beim Auflisten der Dateien mit dem Befehl »`ls -l`« bekommen Sie bei manchen Dateien neben den bekannten Zugriffsrechten `rwx` abweichende Anzeigen wie

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

Was soll das? Hierzu müssen wir etwas weiter ausholen:

Angenommen, das Programm `passwd` sei mit folgenden Zugriffsrechten versehen:

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

Ein normaler (unprivilegierter) Benutzer, sagen wir mal `hugo`, möchte nun sein Kennwort ändern und ruft das Kommando `passwd` auf. Als nächstes erhält er die Meldung "*permission denied*". Was ist die Ursache? Der `passwd`-Prozess (der mit den Rechten von `hugo` läuft) versucht die Datei `/etc/shadow` zum Schreiben zu öffnen und scheitert natürlich, da nur `root` die erforderliche Schreibberechtigung besitzt – dies darf auch nicht anders sein, sonst könnte jeder die Kennwörter beliebig manipulieren, etwa um das `root`-Kennwort zu ändern.

SUID-Bit

Mit Hilfe des **Set-UID-Bits** (oft kurz »SUID-Bit« genannt) kann dafür gesorgt werden, dass ein Programm nicht mit den Rechten des Aufrufers, sondern des Dateieigentümers – hier `root` – ausgeführt wird. Im Fall von `passwd` hat der *Prozess*, der `passwd` ausführt, also Schreibrecht auf die Datei `/etc/shadow`, obwohl der aufrufende Benutzer als Nicht-Systemadministrator dieses Schreibrecht sonst nicht hat. Es

liegt in der Verantwortung des Programmierers von `passwd`, dafür zu sorgen, dass mit diesem Recht kein Schindluder getrieben wird, etwa indem Programmierfehler ausgenutzt werden, um beliebige Dateien außer `/etc/shadow` zu manipulieren oder andere Einträge in `/etc/shadow` außer dem Kennwortfeld des aufrufenden Benutzers zu verändern. Unter Linux funktioniert der Set-UID-Mechanismus übrigens nur für Maschinencode-Programme, nicht für Shell- oder andere Interpreter-Skripte.

Die Bell Labs hatten eine Weile lang ein Patent auf den – von Dennis Ritchie erfundenen – SUID-Mechanismus [SUID]. AT&T hatte Unix zuerst nur unter der Voraussetzung verteilt, dass nach der Erteilung des Patents Lizenzgebühren erhoben werden würden; wegen der logistischen Schwierigkeit, Jahre später von Hunderten von Unix-Installationen rückwirkend kleine Geldbeträge einzutreiben, entschloss man sich jedoch, das Patent der Allgemeinheit zur Verfügung zu stellen.

SGID-Bit

Analog zum Set-UID-Bit gibt es auch ein SGID-Bit, mit dem ein Prozess statt mit der Gruppenzugehörigkeit des Aufrufers mit der Gruppenzugehörigkeit der Programmdatei und den damit verbundenen Rechten (typischerweise zum Zugriff auf andere Dateien, die dieser Gruppe zugeordnet sind) ausgeführt wird.

`chmod`-Syntax

Die SUID- und SGID-Modi werden wie alle anderen Modi einer Datei mit dem Systemprogramm `chmod` verändert, indem Sie symbolische Schlüssel wie `u+s` (setzt das SUID-Bit) oder `g-s` (löscht das SGID-Bit) angeben. Auch in oktalen Modi können Sie diese Bits setzen, indem Sie eine vierte Ziffer ganz links hinzufügen: Das SUID-Bit hat den Wert 4, das SGID-Bit den Wert 2 – so können Sie einer Datei den Zugriffsmodus 4755 geben, um sie für alle Benutzer lesbar und ausführbar zu machen (der Eigentümer darf auch schreiben) und das SUID-Bit zu setzen.

`ls`-Ausgabe

Sie erkennen Programme, die mit den Rechten des Eigentümers oder der Gruppe der Programmdatei arbeiten, in der Ausgabe von »`ls -l`« durch die symbolischen Abkürzungen »`s`« anstelle von »`x`« für normal ausführbare Dateien.

12.6 Besondere Zugriffsrechte für Verzeichnisse

SGID für Verzeichnisse

Es gibt eine weitere Ausnahme von der Zuordnung des Eigentums an Dateien nach dem »Verursacherprinzip«: Der Eigentümer eines Verzeichnisses kann bestimmen, dass die in diesem Verzeichnis erzeugten Dateien der gleichen Benutzergruppe gehören wie das Verzeichnis selbst. Das geschieht, indem das SGID-Bit des Verzeichnisses gesetzt wird. (Da Verzeichnisse nicht ausgeführt werden können, ist das SGID-Bit für solche Sachen frei.)

Die Zugriffsrechte auf ein Verzeichnis werden durch das SGID-Bit nicht verändert. Um eine Datei in einem solchen Verzeichnis anzulegen, muss ein Benutzer das Schreibrecht in der für ihn zutreffenden Kategorie (Eigentümer, Gruppe, andere Benutzer) haben. Wenn ein Benutzer zum Beispiel weder der Eigentümer noch Mitglied der Benutzergruppe eines SGID-Verzeichnisses ist, muss das Verzeichnis für alle Benutzer beschreibbar sein, damit er neue Dateien hineinschreiben kann. Die in dem SGID-Verzeichnis erzeugte Datei gehört dann der Gruppe des Verzeichnisses, auch wenn der Benutzer selbst dieser Gruppe nicht angehört.

Der typische Anwendungsfall für das SGID-Bit auf einem Verzeichnis ist ein Verzeichnis, das einer »Projektgruppe« als Datenablage dient. (Nur) Die Mitglieder der Projektgruppe sollen alle Dateien im Verzeichnis lesen und schreiben und auch neue Dateien anlegen können. Das heißt, Sie müssen alle Benutzer, die an dem Projekt mitarbeiten, in die Projektgruppe tun (sekundäre Gruppe reicht):

```
# groupadd projekt                          Neue Gruppe anlegen
# usermod -a -G projekt hugo                hugo in die Gruppe
# usermod -a -G projekt susi                susi auch
```

Jetzt können Sie das Verzeichnis anlegen und der neuen Gruppe zuordnen. Eigentümer und Gruppe bekommen alle Rechte, der Rest der Welt keine; außerdem setzen Sie noch das SGID-Bit:

```
# cd /home/projekt
# chgrp projekt /home/projekt
# chmod u=rwx,g=srwx /home/projekt
```

Wenn `hugo` jetzt eine Datei in `/home/projekt` anlegt, sollte diese der Gruppe `projekt` zugeordnet sein:

```
$ id
uid=1000(hugo) gid=1000(hugo) groups=101(projekt),1000(hugo)
$ touch /tmp/hugo.txt                          Test: gewöhnliches Verzeichnis
$ ls -l /tmp/hugo.txt
-rw-r--r-- 1 hugo hugo 0 Jan  6 17:23 /tmp/hugo.txt
$ touch /home/projekt/hugo.txt                              Projektverzeichnis
$ ls -l /home/projekt/hugo.txt
-rw-r--r-- 1 hugo projekt 0 Jan  6 17:24 /home/projekt/hugo.txt
```

Das ganze hat nur einen Schönheitsfehler, den Sie erkennen, wenn Sie sich die letzte Zeile des Beispiels anschauen: Die Datei hat zwar die richtige Gruppenzugehörigkeit, aber andere Mitglieder der Gruppe `projekt` dürfen sie trotzdem nur lesen. Wenn Sie möchten, dass alle Mitglieder von `projekt` schreiben dürfen, müssen Sie entweder nachträglich `chmod` anwenden (lästig) oder die *umask* so setzen, dass das Gruppenschreibrecht erhalten bleibt (siehe Übung 12.6).

Der SGID-Modus verändert nur das Verhalten des Betriebssystems beim Erzeugen neuer Dateien. Der Umgang mit bereits existierenden Dateien ist in diesen Verzeichnissen völlig normal. Das bedeutet beispielsweise, dass eine Datei, die außerhalb des SGID-Verzeichnisses erzeugt wurde, beim Verschieben dorthin ihre ursprüngliche Gruppe behält (wohingegen sie beim Kopieren die Gruppe des Verzeichnisses bekommen würde).

Auch das Programm `chgrp` arbeitet in SGID-Verzeichnissen völlig normal: der Eigentümer einer Datei kann sie jeder Gruppe zueignen, der er selbst angehört. Gehört der Eigentümer nicht zu der Gruppe des Verzeichnisses, kann er die Datei mit `chgrp` nicht dieser Gruppe übergeben – dazu muss er sie in dem Verzeichnis neu erzeugen.

Es ist möglich, bei einem Verzeichnis das SUID-Bit zu setzen – diese Einstellung hat aber keine Wirkung.

Linux unterstützt noch einen weiteren Spezialmodus für Verzeichnisse, bei dem das Löschen oder Umbenennen von darin enthaltenen Dateien nur dem Besitzer der jeweiligen Datei erlaubt ist:

```
drwxrwxrwt   7 root   root   1024 Apr  7 10:07 /tmp
```

sticky bit

Mit diesem `t`-Modus, dem ***sticky bit***, kann einem Problem begegnet werden, das bei der gemeinsamen Verwendung öffentlicher Verzeichnisse entstehen kann: Das Schreibrecht für das Verzeichnis erlaubt auch das Löschen fremder Dateien, unabhängig von deren Zugriffsmodus und Besitzer! Beispielsweise sind die `/tmp`-Verzeichnisse öffentlicher Raum, in dem von vielen Programmen temporäre Dateien angelegt werden. Um darin Dateien anlegen zu können, haben alle Benutzer für diese Verzeichnisse Schreibrecht. Damit hat jeder Benutzer auch das Recht, Dateien in diesem Verzeichnis zu löschen.

Normalerweise betrachtet das Betriebssystem beim Löschen oder Umbenennen einer Datei die Zugriffsrechte auf die Datei selbst nicht weiter. Wenn auf einem Verzeichnis das *sticky bit* gesetzt wird, kann eine Datei in diesem Verzeichnis anschließend nur von ihrem Eigentümer, dem Eigentümer des Verzeichnisses oder `root` gelöscht werden. Das *sticky bit* kann über die symbolische Angabe `+t` bzw. `-t` gesetzt oder gelöscht werden, oktal hat es in derselben Ziffer wie SUID und SGID den Wert 1.

Das *sticky bit* bekommt seinen Namen von einer weiteren Bedeutung, die es früher in anderen Unix-Systemen hatte: Seinerzeit wurden Programme vor dem Start komplett in den Swap-Speicher kopiert und nach dem Programmlauf wieder daraus entfernt. Programmdateien, auf denen das *sticky bit* gesetzt war, wurden dagegen auch noch nach dem Programmende im Swap-Speicher liegengelassen. Dies beschleunigte weitere Startvorgänge für dasselbe Programm, weil der anfängliche Kopiervorgang entfiel. Linux verwendet wie die meisten heutigen Unix-Systeme *demand paging*, holt also nur die wirklich benötigten Teile des Programmcodes direkt aus der Programmdatei und kopiert gar nichts in den Swap-Speicher; das *sticky bit* hatte unter Linux niemals seine ursprüngliche Sonderbedeutung.

Übungen

12.5 [2] Was bedeutet das spezielle Zugriffsrecht »`s`«? Wo finden Sie es? Können Sie dieses Recht auf eine selbst erstellte Datei setzen?

12.6 [!1] Mit welchem `umask`-Aufruf können Sie eine *umask* einstellen, die im Projektverzeichnis-Beispiel allen Mitgliedern der Gruppe `projekt` das Lesen *und* Schreiben von Dateien im Projektverzeichnis erlaubt?

12.7 [2] Was bedeutet das spezielle Zugriffsrecht »`t`«? Wo finden Sie es?

12.8 [4] (Für Programmierer.) Schreiben Sie ein C-Programm, das ein geeignetes Kommando (etwa `id`) aufruft. Machen Sie dieses Programm SUID-`root` bzw. SGID-`root` und beobachten Sie, was passiert.

12.9 [3] Wenn Sie sie für ein paar Minuten mit einer `root`-Shell alleine lassen, könnten findige Benutzer versuchen, irgendwo im System eine Shell zu hinterlegen, die sie SUID `root` gemacht haben, um auf diese Weise nach Bedarf Administratorrechte zu bekommen. Funktioniert das mit der Bash? Mit anderen Shells?

12.7 Dateiattribute

Außer den Zugriffsrechten unterstützen die `ext2`- und `ext3`-Dateisysteme noch weitere **Dateiattribute**, über die Sie besondere Eigenschaften der Dateisysteme ansprechen können. Die wichtigsten Dateiattribute finden Sie in Tabelle 12.1. Dateiattribute

Am interessantesten sind vielleicht die *append-only*- und *immutable*-Attribute, mit denen Sie Protokoll- und Konfigurationsdateien vor Veränderungen schützen können; nur `root` darf diese Attribute setzen oder löschen, und wenn sie einmal gesetzt sind, sind sie auch für Prozesse verbindlich, die mit den Rechten von `root` laufen. `a`- und `i`-Attribute

Prinzipiell kann natürlich auch ein Angreifer, der `root`-Rechte erlangt hat, die Attribute zurücksetzen. Allerdings rechnen Angreifer nicht notwendigerweise damit.

Auch das `A`-Attribut kann nützlich sein; damit können Sie zum Beispiel auf Notebook-Rechnern dafür sorgen, dass die Platte nicht ständig läuft, und damit `A`-Attribut

Tabelle 12.1: Die wichtigsten Dateiattribute

Attribut	Bedeutung
`A`	*atime* wird nicht aktualisiert (interessant für mobile Rechner)
`a`	(*append-only*) An die Datei kann nur angehängt werden
`c`	Dateiinhalt wird transparent komprimiert (nicht implementiert)
`d`	Datei wird von `dump` nicht gesichert
`i`	(*immutable*) Datei kann überhaupt nicht verändert werden
`j`	Schreibzugriffe auf den Dateiinhalt werden im Journal gepuffert (nur `ext3`)
`s`	Datenblöcke der Datei werden beim Löschen mit Nullen überschrieben (nicht implementiert)
`S`	Schreibzugriffe auf die Datei werden »synchron«, also ohne interne Pufferung, ausgeführt
`u`	Die Datei kann nach dem Löschen wieder »entlöscht« werden (nicht implementiert)

Strom sparen. Normalerweise wird bei jedem Lesezugriff auf eine Datei deren *atime* – der Zeitpunkt des letzten Zugriffs – aktualisiert, was natürlich einen Schreibzugriff auf die *inode* bedeutet. Bestimmte Dateien werden sehr oft im Hintergrund angeschaut, so dass die Platte nie so recht zur Ruhe kommt, und durch geschickte Verwendung des `A`-Attributs können Sie hier mitunter Abhilfe schaffen.

Die `c`-, `s`- und `u`-Attribute hören sich in der Theorie sehr nützlich an, werden von »normalen« Kernels aber zur Zeit (noch) nicht unterstützt. Teils gibt es mehr oder weniger experimentelle Erweiterungen, die diese Attribute ausnutzen, und teils handelt es sich noch um Zukunftsmusik.

`chattr` Attribute setzen und löschen können Sie mit dem `chattr`-Kommando. Das funktioniert ganz ähnlich wie bei `chmod`: Ein vorgesetztes »`+`« setzt ein oder mehrere Attribute, »`-`« löscht ein oder mehrere Attribute, und »`=`« sorgt dafür, dass die betreffenden Attribute die einzigen gesetzten sind. Die Option `-R` sorgt wie bei `chmod` dafür, dass `chattr` auf alle Dateien in Verzeichnissen und den darin enthaltenen Unterverzeichnissen wirkt. Symbolische Links werden dabei ignoriert.

```
# chattr +a /var/log/messages                    Nur anhängen
# chattr -R +j /data/wichtig                     Daten ins Journal …
# chattr -j /data/wichtig/nichtso                … mit Ausnahme
```

`lsattr` Mit `lsattr` können Sie prüfen, welche Attribute für eine Datei gesetzt sind. Das Programm verhält sich ähnlich wie »`ls -l`«:

```
# lsattr /var/log/messages
-----a----------- /var/log/messages
```

Jeder Strich steht hier für ein mögliches Attribut. `lsattr` unterstützt verschiedene Optionen wie `-R`, `-a` und `-d`, die sich so benehmen wie die gleichnamigen Optionen von `ls`.

Übungen

12.10 [!2] Vergewissern Sie sich, dass die `a`- und `i`-Optionen funktionieren wie behauptet.

12.11 [2] Können Sie für eine gegebene Datei *alle* Striche in der `lsattr`-Ausgabe zum Verschwinden bringen?

Kommandos in diesem Kapitel

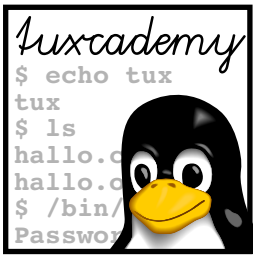
chattr	Setzt Dateiattribute für ext2- und ext3-Dateisysteme	chattr(1)	206
chgrp	Setzt Gruppenzugehörigkeit von Dateien und Verzeichnissen	chgrp(1)	199
chmod	Setzt Rechte für Dateien und Verzeichnisse	chmod(1)	197
chown	Setzt Eigentümer und Gruppenzugehörigkeit für Dateien und Verzeichnisse	chown(1)	198
lsattr	Zeigt Dateiattribute auf ext2- und ext3-Dateisystemen an	lsattr(1)	206
setfacl	Erlaubt die Manipulation von ACLs	setfacl(1)	201
star	POSIX-kompatibles Archivprogramm mit ACL-Unterstützung	star(1)	201
umask	Stellt die *umask* (Rechtekontrolle für neue Dateien) ein	bash(1)	200

Zusammenfassung

- Linux unterscheidet für Dateien das Lese-, Schreib- und Ausführungsrecht, wobei diese Rechte getrennt für den Eigentümer der Datei, die Mitglieder der der Datei zugeordneten Gruppe und den »Rest der Welt« angegeben werden können.
- Die Gesamtheit der Rechte für eine Datei heißt auch Zugriffsmodus.
- Jede Datei (und jedes Verzeichnis) hat einen Eigentümer und eine Gruppe. Zugriffsrechte – Lese-, Schreib- und Ausführungsrecht – werden für diese beiden Kategorien und den »Rest der Welt« getrennt vergeben. Nur der Eigentümer darf die Zugriffsrechte setzen.
- Für den Systemverwalter (root) gelten die Zugriffsrechte nicht – er darf jede Datei lesen oder schreiben.
- Dateirechte werden mit dem Kommando chmod manipuliert.
- Mit chown kann der Systemverwalter die Eigentümer- und Gruppenzuordnung beliebiger Dateien ändern.
- Normale Benutzer können mit chgrp ihre Dateien einer anderen Gruppe zuordnen.
- Mit der *umask* kann man die Standardrechte für Dateien und Verzeichnisse beim Anlegen einschränken.
- SUID- und SGID-Bit erlauben es, Programme mit den Rechten des Dateieigentümers bzw. der der Datei zugeordneten Gruppe anstatt den Rechten des Aufrufers auszuführen.
- Das SGID-Bit auf einem Verzeichnis bewirkt, dass neue Dateien in diesem Verzeichnis der Gruppe des Verzeichnisses zugeordnet werden (und nicht der primären Gruppe des anlegenden Benutzers).
- Das *sticky bit* auf einem Verzeichnis erlaubt das Löschen von Dateien nur dem Eigentümer (und dem Systemadministrator).
- Die ext-Dateisysteme unterstützen besondere zusätzliche Dateiattribute.

Literaturverzeichnis

SUID Dennis M. Ritchie. »Protection of data file contents«. US-Patent 4,135,240. Beantragt am 9.7.1973, erteilt am 16.7.1979.



13

Prozessverwaltung

Inhalt

13.1 Was ist ein Prozess? 210
13.2 Prozesszustände 211
13.3 Prozessinformationen – `ps` 212
13.4 Prozesse im Baum – `pstree` 213
13.5 Prozesse beeinflussen – `kill` und `killall` 214
13.6 `pgrep` und `pkill` 216
13.7 Prozessprioritäten – `nice` und `renice` 217
13.8 Weitere Befehle zur Prozessverwaltung – `nohup`, `top` 219

Lernziele

- Den Prozessbegriff von Linux verstehen
- Die wichtigsten Kommandos zum Abrufen von Prozessinformationen kennen
- Prozesse beeinflussen und beenden können
- Prozessprioritäten beeinflussen können

Vorkenntnisse

- Umgang mit Linux-Kommandos

13.1 Was ist ein Prozess?

Ein Prozess ist im Wesentlichen ein »laufendes Programm«. Prozesse haben Code, der ausgeführt wird, und Daten, auf denen der Code operiert, aber auch diverse Attribute für die interne Verwaltung des Betriebssystems, zum Beispiel die Folgenden:

Prozessnummer

- Die eindeutige Prozessnummer, kurz PID (engl. *process identity*), dient zur unmissverständlichen Identifizierung des Prozesses und kann zu jedem Zeitpunkt nur einem Prozess zugeordnet sein.

Prozessnummer des Elter-Prozesses

- Alle Prozesse kennen die Prozessnummer des Elter-Prozesses, kurz PPID (engl. *parent process ID*). Jeder Prozess kann Ableger (»Kinder«) hervorbringen, die dann einen Verweis auf ihren Erzeuger enthalten. Der einzige Prozess, der keinen solchen Elter-Prozess hat, ist der beim Systemstart erzeugte »Pseudo«-Prozess mit der PID 0. Aus diesem geht der »Init«-Prozess mit der PID 1 hervor, der dann der »Urahn« aller anderen Prozesse im System wird.

Benutzer Gruppen

- Jeder Prozess ist einem Benutzer und einer Reihe von Gruppen zugeordnet. Diese sind wichtig, um die Zugriffsrechte des Prozesses auf Dateien, Geräte usw. zu bestimmen. (Siehe Abschnitt 12.4.) Außerdem darf der Benutzer, dem der Prozess zugeordnet ist, den Prozess anhalten, beenden oder anderweitig beeinflussen. Die Benutzer- und Gruppenzuordnungen werden an Kindprozesse vererbt.

Priorität

- Die CPU-Rechenzeit wird vom System in kleine »Scheiben« (engl. *time slices*) geteilt, die nur Sekundenbruchteile lang sind. Dem aktuellen Prozess steht eine solche Zeitscheibe zur Verfügung, danach wird entschieden, welcher Prozess in der nächsten Zeitscheibe bearbeitet werden soll. Diese Entscheidung wird vom dafür zuständigen *scheduler* (»Planer«) anhand der Priorität des Prozesses getroffen.

  
  In Mehrprozessorsystemen berücksichtigt Linux bei der Zuweisung von Rechenzeit an Prozesse auch die spezielle Topologie des jeweiligen Rechners – es ist einfach, einen Prozess wahlweise auf den verschiedenen Prozessorkernen eines Mehrkern-Prozessors laufen zu lassen, die sich denselben Speicher teilen, während die »Migration« eines Prozesses auf einen anderen Prozessor mit separatem Speicher deutlich mehr Verwaltung bedingt und sich darum seltener lohnt.

weitere Attribute

- Ein Prozess hat weitere Attribute – ein aktuelles Verzeichnis, eine Prozessumgebung, … –, die ebenfalls an Kindprozesse weitergegeben werden.

Prozessdateisystem

Die genannten Informationen können Sie im Verzeichnis `/proc` einsehen. Das Prozessdateisystem von Linux stellt hier zur Laufzeit Daten aus dem Systemkern zur Verfügung, die als Verzeichnisse und Dateien präsentiert werden. Insbesondere gibt es eine Anzahl von Verzeichnissen mit numerischen Namen; jedes Verzeichnis entspricht dabei einem Prozess und sein Name dessen PID. Zum Beispiel:

```
dr-xr-xr-x  3   root  root      0  Oct 16 11:11 1
dr-xr-xr-x  3   root  root      0  Oct 16 11:11 125
dr-xr-xr-x  3   root  root      0  Oct 16 11:11 80
```

Im Verzeichnis für einen Prozess befinden sich diverse »Dateien«, die Informationen über den Prozess enthalten. Näheres hierzu steht in der Handbuchseite `proc`(5).

Jobkontrolle


Bei der Jobkontrolle vieler Shells handelt es sich ebenfalls um Prozessverwaltung; ein »Job« ist ein Prozess, dessen Elterprozess eine Shell ist. Von der entsprechenden Shell aus können deren Jobs mit den Kommandos `jobs`, `bg`

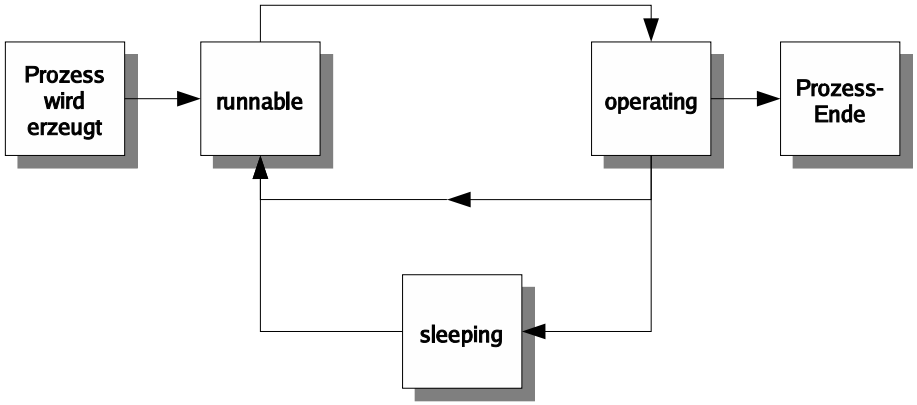


Bild 13.1: Verdeutlichung der Zusammenhänge zwischen den verschiedenen Prozesszuständen

und `fg`, sowie mit den Tastenkombinationen Strg+z und Strg+c (unter anderem) verwaltet werden. Mehr dazu in der Handbuchseite der entsprechenden Shell oder in der Linup-Front-Schulungsunterlage *Linux-Grundlagen für Anwender und Administratoren*.

Übungen

13.1 [3] Wie können Sie sich die Umgebungsvariablen eines beliebigen Ihrer Prozesse anschauen? (*Tipp:* `/proc`-Dateisystem.)

13.2 [2] (Für Programmierer.) Wie groß ist die maximal mögliche PID? Was passiert, wenn diese Grenze erreicht wird? (*Tipp:* Suchen Sie in den Dateien im `/usr/include/linux` nach der Zeichenkette »`PID_MAX`«.)

13.2 Prozesszustände

Prozesszustand

Eine weitere wichtige Eigenschaft eines Prozesses ist sein **Prozesszustand**. Ein Prozess im Arbeitsspeicher wartet darauf, von der CPU ausgeführt zu werden. Diesen Zustand nennt man »lauffähig« (engl. *runnable*). Linux verwendet **präemptives Multitasking**, das heißt, ein Scheduler verteilt die CPU-Zeit in Form von Zeitscheiben an die wartenden Prozesse. Wird ein Prozess in die CPU geladen, nennt man diesen Zustand »rechnend« (engl. *operating*); nach Ablauf der Zeitscheibe wird der Prozess wieder verdrängt und geht zurück in den »lauffähigen« Zustand.

präemptives Multitasking

Im »Außenverhältnis« unterscheidet Linux nicht zwischen diesen beiden Prozesszuständen; entsprechende Prozesse gelten immer als »lauffähig«.

Es kann passieren, dass ein Prozess zwischendurch weiterer Eingaben bedarf oder auf die Beendigung von Operationen mit Peripheriegeräten warten muss; ein solcher Prozess kann keine CPU-Zeit zugeordnet bekommen und hat den Status »schlafend« (engl. *sleeping*). Prozesse, die über die Jobkontrolle der Shell mit Strg+z angehalten wurden, sind im Zustand »gestoppt« (engl. *stopped*). Ist die Ausführung des Prozesses abgeschlossen, beendet sich der Prozess und stellt einen **Rückgabewert** zur Verfügung, mit dem er zum Beispiel signalisieren kann, ob er erfolgreich ausgeführt wurde oder nicht (für eine geeignete Definition von »erfolgreich«).

Rückgabewert

Gelegentlich tauchen Prozesse auf, die mit dem Status »Z« als **Zombie** gekennzeichnet sind. Diese »lebenden Toten« existieren normalerweise nur für einige Augenblicke. Ein Prozess wird zum Zombie, wenn er sich beendet, und verscheidet

Zombie

endgültig, sobald sein Elterprozess den Rückgabewert des Zombies abgefragt hat. Wenn ein Zombie nicht aus der Prozesstabelle verschwindet, bedeutet das, dass der Elternprozess des Zombies eigentlich den Rückgabewert hätte abholen müssen, das aber bisher nicht getan hat. Ein Zombie kann nicht aus der Prozesstabelle entfernt werden. Weil der eigentliche Prozess nicht mehr existiert und weder Arbeitsspeicher noch Rechenzeit verbraucht, hat ein Zombie außer dem unschönen Eintrag im Systemstatus aber keine nachteilige Auswirkung auf das laufende System. Dauerhaft oder massenweise vorhandene Zombies sind normalerweise ein Indiz für einen Programmierfehler im Elterprozess; wenn der Elterprozess sich beendet, dann verschwinden sie auch.

Zombies verschwinden, wenn ihr Elterprozess verschwindet, weil "verwaiste" Prozesse vom Init-Prozess "adoptiert" werden. Da der Init-Prozess die meiste Zeit darauf wartet, dass andere Prozesse sich beenden, damit er deren Rückgabewert abholen kann, werden die Zombies dann ziemlich zügig entsorgt.

Zombies belegen natürlich Einträge in der Prozesstabelle, die möglicherweise für andere Prozesse benötigt werden. Sollte das ein Problem werden, dann sollten Sie sich den Elterprozess genauer anschauen.

Übungen

13.3 [2] Starten Sie einen `xclock`-Prozess im Hintergrund. In der Shellvariablen `$!` finden Sie die PID dieses Prozesses (sie enthält immer die PID des zuletzt gestarteten Hintergrundprozesses). Prüfen Sie den Zustand des Prozesses mit dem Kommando »`grep ^State: /proc/$!/status`«. Stoppen Sie die `xclock` anschließend, indem Sie sie in den Vordergrund holen und mit Strg + z anhalten. Wie ist nun der Prozesszustand? (Alternativ zur `xclock` können Sie auch ein beliebiges anderes langlaufendes Programm verwenden.)

13.4 [4] (Beim zweiten Lesen.) Können Sie absichtlich einen Zombie-Prozess erzeugen?

13.3 Prozessinformationen – `ps`

Die Informationen aus `/proc` rufen Sie in der Regel nicht direkt, sondern mit entsprechenden Kommandos ab.

Auf jedem Unix-artigem System vorhanden ist der Befehl `ps` (*process status*). Ohne Optionen werden alle auf dem aktuellen Terminal laufenden Prozesse berücksichtigt. Dabei werden die Prozessnummer `PID`, das Terminal `TTY`, der Prozesszustand `STAT`, die bisher verbrauchte Rechenzeit `TIME` sowie das jeweilige Kommando tabellarisch aufgelistet:

```
$ ps
  PID TTY STAT TIME COMMAND
  997   1 S    0:00 -bash
 1005   1 R    0:00 ps
$ _
```

Auf dem aktuellen Terminal `tty1` laufen also momentan zwei Prozesse. Neben der `bash` mit der PID 997, die zur Zeit schläft (»`S`« für *sleeping*), wird gerade das Kommando `ps` mit der PID 1005 ausgeführt (»`R`« für *runnable*). Der oben erwähnte Zustand »rechnend« wird in der `ps`-Ausgabe nicht angezeigt.

Syntax

Die Syntax von `ps` ist recht verwirrend. Neben Optionen im Unix98-Stil (wie `-l`) und im GNU-Langformat (etwa `--help`) sind auch Optionen im BSD-Format ohne einleitendes Minuszeichen erlaubt. Von allen möglichen Parametern sind hier einige aufgeführt:

a (engl. *all*) zeigt alle Prozesse mit Terminal

--forest zeigt die Prozesshierarchie

l (engl. *long*) gibt Zusatzinformationen aus, z. B. die Priorität

r (engl. *running*) zeigt nur laufende Prozesse

T (engl. *terminal*) zeigt alle Prozesse im aktuellen Terminal

U ⟨*name*⟩ (engl. *user*) gibt Prozesse von Benutzer ⟨*name*⟩ aus

x zeigt auch Prozesse ohne Terminal

Die ungewöhnliche Syntax von ps erklärt sich dadurch, dass im AT&T-Unix die Optionen traditionell mit Minuszeichen und in BSD traditionell ohne angegeben wurden (dieselbe Option kann in den beiden Geschmacksrichtungen durchaus verschiedene Wirkung haben). Bei der großen Wiedervereinigung in System V Release 4 konnte man so die meisten Optionen mit ihrer gewohnten Bedeutung beibehalten.

Wenn Sie im ps-Aufruf eine PID angeben, bekommen Sie nur Informationen über den betreffenden Prozess gezeigt (falls er existiert):

```
$ ps 1
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init [2]
```

Mit der Option -C erhalten Sie Informationen über den oder die Prozesse, denen ein bestimmtes Kommando zugrundeliegt:

```
$ ps -C konsole
    PID TTY          TIME CMD
   4472 ?        00:00:10 konsole
  13720 ?        00:00:00 konsole
  14045 ?        00:00:14 konsole
```

(Alternativ hilft natürlich auch grep.)

Übungen

13.5 [!2] Was bedeuten die Informationen, die Sie mit dem Kommando ps erhalten? Geben Sie ps ohne Option, dann mit a und schließlich mit ax an. Was bewirkt die Angabe von x?

13.6 [3] Das ps-Kommando erlaubt es Ihnen über die Option -o, die ausgegebenen Felder selbst zu bestimmen. Studieren Sie die Handbuchseite ps(1) und geben Sie eine ps-Kommandozeile an, mit der Sie die PID, PPID, den Prozesszustand und das Kommando ausgeben können.

13.4 Prozesse im Baum – pstree

Wenn Sie nicht unbedingt sämtliche Informationen zu einem Prozess haben möchten, sondern etwas über die Verwandtschaft der Prozesse zueinander herausfinden wollen, bietet sich der Befehl pstree an. pstree zeigt einen Prozessbaum an, in dem die Kindprozesse jeweils in Abhängigkeit von ihrem Elternprozess dargestellt sind. Die Prozesse sind namentlich genannt:

pstree

```
$ pstree
init-+-apache---7*[apache]
     |-apmd
     |-atd
     |-cannaserver
     |-cardmgr
     |-chronyd
     |-cron
     |-cupsd
     |-dbus-daemon-1
     |-events/0-+-aio/0
     |          |-kblockd/0
     |          `-2*[pdflush]
     |-6*[getty]
     |-ifd
     |-inetd
     |-kapmd
     |-kdeinit-+-6*[kdeinit]
     |         |-kdeinit-+-bash---bash
     |         |         |-2*[bash]
     |         |         |-bash---less
     |         |         |-bash-+-pstree
     |         |         |      `-xdvi---xdvi.bin---gs
     |         |         `-bash---emacs---emacsserver
     |         |-kdeinit---3*[bash]
     |         |-kteatime
     |         `-tclsh
     |-10*[kdeinit]
     |-kdeinit---kdeinit
◁◁◁◁◁
```

Identische Prozesse werden in eckigen Klammern mit der entsprechenden Anzahl und »*« angezeigt. Die wichtigsten Optionen von `pstree` sind:

`-p` zeigt zusätzlich zu den Prozessnamen ihre PID an

`-u` zeigt Besitzerwechsel an

`-G` hübscht die Anzeige etwas auf, indem Terminal-Grafikzeichen benutzt werden – ob das wirklich etwas bringt, hängt von Ihrem Terminal ab

Eine angedeutete Baumdarstellung erhalten Sie auch mit »`ps --forest`«. Die Baumstruktur sehen Sie dort in der `COMMAND`-Spalte der Ausgabe.

13.5 Prozesse beeinflussen – `kill` und `killall`

Signale

Das Kommando `kill` schickt **Signale** an ausgewählte Prozesse. Das gewünschte Signal kann entweder als Zahl oder als Text angegeben werden. Ferner müssen Sie noch die jeweilige Prozessnummer übergeben, die Sie mit `ps` herausfinden können:

```
$ kill -15 4711                    Signal SIGTERM an Prozess 4711
$ kill -TERM 4711                  Dasselbe
$ kill -SIGTERM 4711               Nochmal dasselbe
$ kill -s TERM 4711                Nochmal dasselbe
$ kill -s SIGTERM 4711             Nochmal dasselbe
$ kill -s 15 4711                  Raten Sie mal
```

Hier sind die wichtigsten Signale mit den entsprechenden Nummern und ihrer Bedeutung:

SIGHUP (1, *hang up*) veranlasst die Shell, alle ihre Kindprozesse zu beenden, die dasselbe kontrollierende Terminal verwenden wie sie selbst. Bei Hintergrundprozessen ohne kontrollierendes Terminal wird dieses Signal oft dafür verwendet, sie ihre Konfigurationsdatei neu einlesen zu lassen (siehe unten).

SIGINT (2, *interrupt*) Unterbricht den Prozess; entspricht der Tastenkombination Strg+c.

SIGKILL (9, *kill*) Beendet den Prozess und kann von diesem nicht ignoriert werden; quasi »Notbremse«.

SIGTERM (15, *terminate*) Voreinstellung für kill und killall; beendet den Prozess

SIGCONT (18, *continue*) Setzt einen mit SIGSTOP angehaltenen Prozess fort

SIGSTOP (19, *stop*) Hält einen Prozess an.

SIGTSTP (20, *terminal stop*) Entspricht der Tastenkombination Strg+z.

Sie sollten sich nicht zu sehr auf die Signalnummern versteifen, die nicht alle garantiert auf allen Unix-Versionen (oder auch nur Linux-Plattformen) dieselben sind. Was die Signale 1, 9 und 15 angeht, sind Sie auf der sicheren Seite, aber für alles andere benutzen Sie besser die Namen.

Ohne weitere Angabe wird das Signal SIGTERM (engl. *terminate*) gesendet, wodurch der Prozess (meistens) beendet wird. Programme können so geschrieben werden, dass sie Signale »abfangen« (intern bearbeiten) oder ignorieren. Signale, die von einem Prozess weder abgefangen noch ignoriert werden, führen in der Regel zum abrupten Ende des Prozesses. Einige wenige Signale werden standardmäßig ignoriert.

Die Signale SIGKILL und SIGSTOP werden nicht vom Prozess bearbeitet, sondern vom Systemkern, und können nicht abgefangen oder ignoriert werden. SIGKILL beendet einen Prozess, ohne ihm dabei ein Vetorecht einzuräumen (wie SIGTERM das machen würde), und SIGSTOP hält den Prozess an, so dass er keine Rechenzeit mehr zugeordnet bekommt.

Nicht immer wird ein Prozess durch kill zur Einstellung seiner Arbeit veranlasst. Hintergrundprozesse etwa, die losgelöst von Terminals Systemdienste erbringen – die *Daemons* –, lassen sich oft mit dem Signal SIGHUP (engl. *hang up*) zum erneuten Einlesen ihrer Konfigurationsdateien anregen, ohne dass sie dafür neu gestartet werden müssen. *Daemons*

Wie viele andere Linux-Kommandos können Sie kill nur auf Prozesse anwenden, die Ihnen gehören. Nur root ist von dieser Einschränkung ausgenommen.

Manchmal reagiert ein Prozess nicht einmal auf das Signal SIGKILL. Die Ursache hierfür liegt entweder daran, dass es sich um einen Zombie-Prozess handelt (der ja schon tot ist und deswegen nicht nochmal umgebracht werden kann), oder aber in einem blockierten Systemaufruf. Letztere Situation entsteht beispielsweise, wenn ein Prozess auf die Beendigung einer Schreib- oder Leseoperation eines langsamen Gerätes wartet.

Eine Alternative zum Befehl kill ist das Kommando killall. killall agiert genau wie kill – es sendet ein Signal an einen Prozess. Der Unterschied ist, dass hier der Prozess benannt werden muss (statt über seine PID angesprochen) und dass dabei alle Prozesse dieses Namens beeinflusst werden. Wenn kein Signal angegeben ist, wird auch hier standardmäßig SIGTERM gesendet. killall gibt eine Warnung aus, wenn es unter dem angegebenen Namen nichts finden konnte. killall

Die wichtigsten Optionen von killall sind:

-i killall erwartet eine Rückbestätigung, ob es den angegebenen Prozess terminieren darf.

-l gibt eine Liste aller möglichen Signale aus.

-w wartet ab, ob der Prozess, an den das Signal gesendet wurde, beendet ist. `killall` kontrolliert jede Sekunde, ob der Prozess noch existiert oder nicht, und beendet sich erst, wenn letzteres der Fall ist.

Passen Sie mit `killall` auf, wenn Sie es hin und wieder mit Solaris oder BSD zu tun bekommen. Auf diesen Systemen tut das Kommando nämlich genau, was sein Name andeutet – es beendet *alle* Prozesse.

Übungen

13.7 [2] Welche Signale werden standardmäßig ignoriert? (*Tipp:* `signal`(7))

13.6 `pgrep` und `pkill`

So nützlich `ps` und `kill` sind, so umständlich ist es manchmal, genau die Prozesse zu identifizieren, die gerade interessant sind. Natürlich können Sie die Ausgabe von `ps` mit `grep` durchsuchen, aber das »idiotensicher« und ohne übermäßig viele falsche Positive hinzukriegen, ist zumindest mühsam, wenn nicht gar trickreich. Erfreulicherweise hat Kjetil Torgrim Homme uns diese Arbeit abgenommen und das Programm `pgrep` entwickelt, das eine bequeme Suche durch die Prozessliste gestattet: Ein Kommando wie

```
$ pgrep -u root sshd
```

listet zum Beispiel die PIDs aller `sshd`-Prozesse auf, die `root` gehören.

Standardmäßig beschränkt `pgrep` sich auf die Ausgabe der PIDs. Mit der Option `-l` können Sie es aber dazu bringen, außerdem den Kommandonamen auszugeben. Mit `-a` liefert es die komplette Kommandozeile.

Die Option `-d` erlaubt es, ein Trennzeichen anzugeben (normal wäre »`\n`«):

```
$ pgrep -d, -u hugo bash
4261,11043,11601,12289
```

Detailliertere Informationen über die Prozesse können Sie erhalten, indem Sie die PIDs an `ps` verfüttern:

```
$ ps up $(pgrep -d, -u hugo bash)
```

(Mit der Option `p` können Sie `ps` gezielt eine durch Kommas getrennte Liste von PIDs von Interesse übergeben.)

Der Parameter von `pgrep` ist eigentlich ein (erweiterter) regulärer Ausdruck (denken Sie an `egrep`), der dazu verwendet wird, die Prozessnamen zu durchsuchen. Etwas wie

```
$ pgrep '^([bd]a|t?c|k|z|)sh$'
```

sucht also nach den gängigen Shells.

Normalerweise betrachtet `pgrep` nur den Prozessnamen (genauer gesagt die ersten 15 Zeichen des Prozessnamens). Mit der Option `-f` können Sie die komplette Kommandozeile durchsuchen.

Über Optionen können Sie weitere Suchkriterien angeben. Hier eine kleine Auswahl:

-G Nur Prozesse betrachten, deren Gruppe angegeben ist. (Gruppen können über ihre Namen oder GIDs angegeben werden.)

-n Nur den neuesten (zuletzt gestarteten) der gefundenen Prozesse ausgeben.

-o Nur den ältesten (zuerst gestarteten) der gefundenen Prozesse ausgeben.

-P Nur Prozesse betrachten, deren Elterprozesse die angegebenen PIDs haben.

-t Nur Prozesse betrachten, deren kontrollierendes Terminal angegeben ist. (Terminalnamen sollten ohne »/dev/« am Anfang angegeben werden.)

-u Nur Prozesse mit den angegebenen (effektiven) UIDs betrachten.

Wenn Sie Suchkriterien und keinen regulären Ausdruck für den Prozessnamen angeben, dann werden alle Prozesse ausgegeben, die auf die Suchkriterien passen. Wenn Sie beides weglassen, bekommen Sie eine Fehlermeldung.

Das Kommando `pkill` benimmt sich wie `pgrep`, bis darauf, dass es nicht die PIDs der gefundenen Prozesse auflistet, sondern den Prozessen direkt ein Signal schickt (standardmäßig `SIGTERM`). Wie bei `kill` können Sie ein anderes Signal angeben:

```
# pkill -HUP syslogd
```

Die Option `--signal` würde auch funktionieren:

```
# pkill --signal HUP syslogd
```

Der Vorteil von `pkill` gegenüber `killall` ist, dass Sie `pkill` deutlich gezielter einsetzen können.

Übungen

13.8 [!1] Benutzen Sie `pgrep`, um die PIDs aller gerade laufenden Prozesse des Benutzers `hugo` zu bestimmen. (Wenn Sie keinen Benutzer `hugo` haben, dann nehmen Sie einen anderen Benutzer.)

13.9 [2] Starten Sie in zwei verschiedenen Terminalfenstern (ersatzweise Textkonsolen) das Kommando »`sleep 60`«. Verwenden Sie `pkill`, um (a) das zuerst gestartete, (b) das zuletzt gestartete, (c) gezielt das in einem der beiden Terminalfenster gestartete Kommando abzubrechen.

13.7 Prozessprioritäten – `nice` und `renice`

In einem Multitasking-Betriebssystem wie Linux muss die Prozessorzeit auf verschiedene Prozesse verteilt werden. Diese Aufgabe erledigt der Scheduler. Es gibt in der Regel mehr als einen lauffähigen Prozess. Der Scheduler muss nach bestimmten Regeln den lauffähigen Prozessen Rechenzeit zuteilen. Ausschlaggebend dafür ist die **Priorität** der Prozesse. Diese ändert sich dynamisch je nach dem bisherigen Verhalten des Prozesses – »interaktive« Prozesse, also solche, die Ein- und Ausgabe machen, werden gegenüber solchen bevorzugt, die nur Rechenzeit verbrauchen. Priorität

Als Benutzer (oder Administrator) können Sie die Priorität von Prozessen nicht direkt festlegen. Sie können den Kernel lediglich darum bitten, bestimmte Prozesse zu bevorzugen oder zu benachteiligen. Der »*Nice*-Wert«, der den Grad dieser Bevorzugung quantifiziert, wird an Kindprozesse vererbt.

Der *Nice*-Wert für einen neuen Prozess kann mit dem Kommando `nice` festgelegt werden. Die Syntax ist `nice`

```
nice [-⟨Nice-Wert⟩] ⟨Kommando⟩ ⟨Parameter⟩ …
```

(`nice` wird also als »Präfix« eines anderen Kommandos verwendet).

mögliche *Nice*-Werte

Die möglichen *Nice*-Werte sind Zahlen zwischen −20 und +19. Ein negativer *Nice*-Wert erhöht die Priorität, ein positiver Wert erniedrigt sie (je höher der Wert, desto »netter« sind Sie zu den anderen Benutzern des Systems, indem Sie Ihren eigenen Prozessen eine geringere Priorität geben). Ist kein *Nice*-Wert angegeben, wird der Standardwert +10 angenommen. Nur `root` darf Prozesse mit einem negativen *Nice*-Wert aufzurufen (negative *Nice*-Werte sind im allgemeinen nicht nett zu anderen Benutzern).

`renice`

Die Priorität eines bereits laufenden Prozesses können Sie mit dem Befehl `renice` beeinflussen. Hierzu rufen Sie `renice` mit dem gewünschten neuen *Nice*-Wert und der PID (oder den PIDs) der betreffenden Prozesse auf:

```
renice [-⟨Nice-Wert⟩] ⟨PID⟩ …
```

Auch hier gilt, dass nur der Systemverwalter beliebige *Nice*-Werte vergeben darf. Normale Benutzer dürfen mit `renice` den *Nice*-Wert ihrer Prozesse nur erhöhen – es ist zum Beispiel nicht möglich, einem Prozess, der bereits mit dem *Nice*-Wert 5 gestartet wurde, nachträglich wieder den *Nice*-Wert 0 zuzuordnen. Allerdings ist es durchaus erlaubt, ihm den *Nice*-Wert 10 zu geben. (Denken Sie an ein Zahnrad mit Sperrklinke.)

`ulimit`

Eine weitere Möglichkeit, Systemressourcen an Prozesse zu verteilen, ist das in die Shell eingebaute Kommando `ulimit`. Es erlaubt die Kontrolle über die Systemressourcen, die den von der Shell gestarteten Unterprozessen zur Verfügung stehen. Die Syntax ist

```
ulimit [⟨Option⟩ [⟨Limit⟩]]
```

Die wichtigsten ⟨*Optionen*⟩ sind:

`-a` Zeigt alle eingestellten Grenzwerte an

`-d` Schränkt die maximale Größe des Datensegments jedes einzelnen Prozesses ein, der von dieser Shell aus gestartet wird.

`-f` Verbietet dem Anwender, Dateien über einer bestimmten Größe zu erzeugen (nur `ext2`-Dateisystem).

`-n` Schränkt die maximale Anzahl offener Dateien jedes einzelnen von dieser Shell gestarteten Prozesses ein.

`-t` Schränkt die verfügbare CPU-Zeit (Benutzer- und Systemzeit) jedes einzelnen Prozesses auf die angegebene Anzahl Sekunden ein.

`-u` Schränkt die Anzahl der Prozesse je Benutzer ein (dabei werden auch die von anderen Shells gestarteten Prozesse desselben Benutzers mitgezählt).

`-v` Begrenzt den virtuellen Speicher für jeden aus dieser Shell gestarteten Prozess.

Die Grenzen (⟨*Limit*⟩) werden in Kibibytes angegeben, wenn oben keine andere Einheit genannt ist. Wenn beim Aufruf keine Grenze bestimmt wird, gibt `ulimit` die aktuelle Grenze an.

Übungen

13.10 [2] Versuchen Sie, einem Prozess eine höhere Priorität zu geben. Möglicherweise funktioniert das nicht, warum? Überprüfen Sie den Prozesszustand mit `ps`.

13.11 [2] Probieren Sie das `ulimit`-Kommando aus, etwa indem Sie ein Limit für die maximale Dateigröße angeben und dann versuchen, eine größere Datei zu erzeugen (zum Beispiel mit `dd`).

13.12 [2] Was ist der Unterschied zwischen »`ulimit -f`« und Plattenplatzkontingentierung?

13.8 Weitere Befehle zur Prozessverwaltung – `nohup`, `top`

Wenn Sie ein Kommando mittels `nohup` aufrufen, veranlasst das das betreffende Programm dazu, das Signal `SIGHUP` zu ignorieren und damit das Ende des Elternprozesses zu überleben:

`SIGHUP` ignorieren

```
nohup ⟨Kommando⟩ …
```

Der Prozess geht nicht automatisch in den Hintergrund, sondern muss mit einem `&` am Ende der Kommandozeile dorthin geschickt werden. Wenn die Standardausgabe des Programms ein Terminal ist und der Benutzer nichts anderes definiert hat, so wird die Ausgabe automatisch gemeinsam mit der Standardfehlerausgabe in die Datei `nohup.out` umgeleitet. Ist das aktuelle Verzeichnis für den Benutzer nicht schreibbar, wird die Datei im Heimatverzeichnis des Benutzers angelegt.

`top` vereint die Funktionen vieler Befehle zur Prozessverwaltung unter einer Oberfläche. Zudem bietet es nach Aufruf eine sich ständig aktualisierende Prozesstabelle. Interaktiv können verschiedene Operationen ausgeführt werden, eine Übersicht erhalten Sie mit h. Unter anderem ist es möglich, die Liste nach verschiedenen Kriterien zu sortieren, Signale an Prozesse zu versenden (k) und die Priorität zu verändern (r).

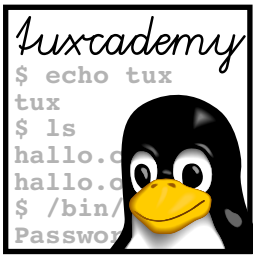
`top`

Kommandos in diesem Kapitel

`kill`	Hält einen Hintergrundprozess an	`bash`(1), `kill`(1)	214
`killall`	Schickt allen Prozessen mit passendem Namen ein Signal	`killall`(1)	215
`nice`	Startet Programme mit einem anderen *nice*-Wert	`nice`(1)	217
`nohup`	Startet ein Programm so, dass es immun gegen `SIGHUP`-Signale ist	`nohup`(1)	219
`pgrep`	Sucht Prozesse anhand ihres Namens oder anderer Kriterien	`pgrep`(1)	216
`pkill`	Signalisiert Prozessen anhand ihres Namens oder anderer Kriterien	`pkill`(1)	217
`ps`	Gibt Prozess-Statusinformationen aus	`ps`(1)	212
`pstree`	Gibt den Prozessbaum aus	`pstree`(1)	213
`renice`	Ändert den *nice*-Wert laufender Prozesse	`renice`(8)	218
`top`	Bildschirmorientiertes Programm zur Beobachtung und Verwaltung von Prozessen	`top`(1)	219
`ulimit`	Legt Ressourcenbegrenzungen für Prozesse fest	`bash`(1)	218

Zusammenfassung

- Ein Prozess ist ein Programm, das ausgeführt wird.
- Ein Prozess hat außer einem Programmtext und den dazugehörigen Daten auch Attribute wie eine Prozessnummer (PID), Elterprozessnummer (PPID), Eigentümer, Gruppen, Priorität, Umgebung, aktuelles Verzeichnis, …
- Alle Prozesse stammen vom `init`-Prozess (PID 1) ab.
- Mit `ps` kann man Informationen über Prozesse abrufen.
- `pstree` stellt die Prozesshierarchie in Baumform dar.
- Prozesse können mit Signalen beeinflusst werden.
- Die Kommandos `kill` und `killall` schicken Signale an Prozesse.
- Mit `nice` und `renice` kann man die Priorität eines Prozesses beeinflussen. `ulimit` erlaubt es, den Ressourcenverbrauch eines Prozesses zu beschränken.
- `top` ist eine komfortable Oberfläche zur Prozessverwaltung.



14

Platten (und andere Massenspeicher)

Inhalt

14.1 Grundlagen 222
14.2 Bussysteme für Massenspeicher 222
14.3 Partitionierung 225
14.3.1 Grundlagen 225
14.3.2 Die traditionelle Methode (MBR) 226
14.3.3 Die moderne Methode (GPT) 227
14.4 Linux und Massenspeicher 229
14.5 Platten partitionieren 231
14.5.1 Prinzipielles. 231
14.5.2 Platten partitionieren mit `fdisk` 233
14.5.3 Platten formatieren mit GNU `parted` 236
14.5.4 `gdisk` 238
14.5.5 Andere Partitionierungsprogramme 238
14.6 Loop-Devices und `kpartx` 239
14.7 Der Logical Volume Manager (LVM) 241

Lernziele

- Verstehen, wie Linux mit Massenspeichergeräten auf ATA-, SATA- und SCSI-Basis umgeht
- MBR- und GPT-Partitionierung verstehen
- Linux-Partitionierungswerkzeuge kennen und anwenden können
- Loop-Devices verwenden können

Vorkenntnisse

- Grundlegende Linux-Kenntnisse
- Kenntnisse über Linux-Hardwareunterstützung

14.1 Grundlagen

RAM ist heutzutage ziemlich billig, aber trotzdem kommen die wenigsten Rechner ohne die dauerhafte Speicherung von Programmen und Daten auf Massenspeichergeräten aus. Dazu zählen unter anderem

- Festplatten mit rotierenden magnetischen Scheiben
- »Solid-State Disks« (SSDs), die aus der Sicht des Computers aussehen wie Festplatten, aber intern Flash-Speicher verwenden
- USB-Sticks, SD-Karten, CF-Karten und andere Wechselmedien mit Flash-Speicher
- RAID-Systeme, die Festplatten bündeln und als ein großes Speichermedium erscheinen lassen
- SAN-Geräte, die »virtuelle« Festplattenlaufwerke im Netz zur Verfügung stellen
- Fileserver, die Dateizugriff auf abstrakter Ebene anbieten (CIFS, NFS, …)

In diesem Kapitel beleuchten wir die Grundlagen der Linux-Unterstützung für die ersten drei Einträge der Liste – Festplatten, SSDs und Flash-Wechselmedien wie USB-Sticks. RAID-Systeme und SAN werden in der Linup-Front-Schulungsunterlage *Linux-Storage und Dateisysteme* besprochen, Fileserver in *Linux-Infrastrukturdienste*.

14.2 Bussysteme für Massenspeicher

IDE, ATA und SATA Bis vor einiger Zeit wurden Festplatten und optische Laufwerke wie CD-ROM- und DVD-Leser und -Brenner über einen »IDE-Controller« angebunden, von dem PCs mit Selbstachtung in der Regel mindestens zwei hatten (mit jeweils zwei »Kanälen«).

IDE

»IDE« ist eigentlich die Abkürzung für »Integrated Drive Electronics«, also locker gesagt »Festplatte mit eingebauter Elektronik«. Die »eingebaute Elektronik«, von der hier die Rede ist, sorgt dafür, dass der Computer die Festplatte als Reihe durchnumerierter Datenblöcke sieht und nichts über Sektoren, Zylinder und Schreib-Lese-Köpfe wissen muss – auch wenn diese elaborate Scharade immer noch zwischen BIOS, Plattencontroller und Platten aufrechterhalten wird. Diese Beschreibung trifft allerdings schon lange auf *alle* Festplatten zu, nicht nur solche mit der wohlbekannten »IDE«-Schnittstelle, die inzwischen offiziell eigentlich »ATA«, kurz für *AT Attachment*«[1] heißt.

Rechner, die Sie heute neu kaufen, haben typischerweise immer noch IDE-Schnittstellen, aber die Methode der Wahl zum Anschluss von Festplatten und optischen Laufwerken ist heute eine serielle Version des ATA-Standards, einfallsreicherweise »Serial ATA« oder kurz »SATA« genannt. Seit es SATA gibt (also seit etwa 2003) sagt man zu herkömmlichem ATA (vulgo »IDE«) auch gerne »P-ATA«, kurz für »Parallel ATA«. Der Unterschied bezieht sich auf das Kabel, das bei herkömmlichem ATA ein unbequem zu verlegendes und elektrisch nicht zu 100% glückliches 40- oder 80-adriges Flachbandkabel ist, mit dem 16 Datenbits parallel übertragen werden können und das mehrere Geräte auf einmal mit dem Controller verbindet. SATA verwendet dagegen (gerne in fröhlichem Orange gehaltene) schmale flache siebenadrige Kabel für serielle Übertragung, eins pro Gerät.

Serial ATA

[1] Erinnert sich hier noch jemand an den IBM PC/AT?

SATA-Kabel und -stecker sind physikalisch deutlich empfindlicher als die entsprechende P-ATA-Hardware, aber machen das durch andere Vorteile wett: Sie behindern die Luftkühlung in einem Rechner weniger und können nicht falsch installiert werden, da sie zwei verschiedene Enden mit unterschiedlichen Steckverbindern haben, die außerdem so ausgelegt sind, dass sie nicht falschherum eingesteckt werden können. Außerdem entfällt die unlogische Trennung zwischen 2,5- und 3,5-Zoll-Plattenlaufwerken, die bei P-ATA verschiedene Steckverbinder verwenden.

Interessanterweise erlaubt das serielle SATA deutlich schnellere Datenübertragung als das herkömmliche ATA, obwohl bei ersterem alle Daten bitweise im Gänsemarsch fließen statt sechzehnfach parallel. Dies hängt mit den elektrischen Eigenschaften der Schnittstelle zusammen, die differentielle Übertragung und eine Signalspannung von nur 0,5 V statt 5 V verwendet. (Deswegen dürfen die Kabel auch länger sein – 1 m statt bisher 45 cm.) Heutige SATA-Schnittstellen können theoretisch bis zu 16 GiBit/s (SATA 3.2) übertragen, was durch Codierung und ähnliche Hemmschuhe auf knapp 2 MiB/s hinausläuft – deutlich mehr, als einzelne Plattenlaufwerke längerfristig verarbeiten können, aber nützlich für RAID-Systeme, bei denen mehrere Platten simultan angesprochen werden können, und für schnelle SSDs. Mit einer weiteren Evolution der Geschwindigkeit von SATA ist eher nicht zu rechnen, da bei SSDs der Trend hin zu einer direkten Anbindung per PCIe geht[2]

Außer der höheren Geschwindigkeit und der bequemeren Verkabelung bietet SATA auch noch den Vorteil des sogenannten *hot-swappings*: Es ist möglich, ein über SATA angeschlossenes Laufwerk auszustöpseln und ein anderes dafür einzustöpseln, ohne dass der Rechner dafür heruntergefahren werden muss. Das setzt natürlich voraus, dass der Rechner ohne die Daten auskommen kann, die auf dem Laufwerk stehen – typischerweise weil es Bestandteil eines RAID-1 oder RAID-5 ist, bei dem die Daten auf dem neuen Laufwerk auf der Basis anderer Laufwerke im System rekonstruiert werden können. Mit herkömmlichem ATA war das nicht oder nur unter Verrenkungen möglich.

Externes SATA (»eSATA«) ist eine Abart von SATA für die Verwendung mit externen Laufwerken. Es hat abweichende Stecker und elektrische Spezifikationen, die mechanisch wesentlich robuster und besser für *hot-swapping* geeignet sind. Inzwischen ist es auf dem Markt fast völlig von USB 3.*x* verdrängt worden, findet sich aber noch in manchen älteren Geräten. eSATA

SCSI und SAS Das *Small Computer System Interface* oder SCSI (übliche Aussprache: »skasi«) dient seit über 25 Jahren zum Anschluss von Platten, Bandlaufwerken und anderen Massenspeichern, aber auch Peripheriegeräten wie Scannern an »kleine« Computer[3]. SCSI-Busse und -Anschlüsse gibt es in einer verwirrenden Vielfalt, beginnend mit dem »traditionellen« 8-Bit-Bus über die schnellen 16 Bit breiten Varianten bis zu neuen, noch schnelleren seriellen Implementierungen (siehe unten). Sie unterscheiden sich außerdem in der Maximalzahl von anschließbaren Geräten pro Bus und in physischen Parametern wie der maximalen Kabellänge und den erlaubten Abständen von Geräten auf dem Kabel. Netterweise sind die meisten dieser Varianten kompatibel oder doch unter Geschwindigkeitsverlust kompatibel machbar. Abarten wie *FireWire* (IEEE-1394) oder *Fibre Channel* werden zumindest von Linux wie SCSI behandelt.

[2] SATA im engeren Sinne erlaubt Geschwindigkeiten bis zu 6 GiBit/s; die höhere Geschwindigkeit von SATA 3.2 wird schon über PCIe erreicht. Die »SATA Express«-Spezifikation definiert eine Schnittstelle, die sowohl SATA- als auch PCIe-Signale transportiert, so dass kompatible Geräte nicht nur an SATA-Express-kompatible Controller angeschlossen werden können, sondern auch an ältere Rechner, die »nur« SATA mit bis zu 6 GiBit/s unterstützen.

[3] Was »klein« in diesem Kontext heißen soll, ist nirgendwo definiert, aber es muss etwas sein wie »kann noch von zwei Personen hochgehoben werden«.

Tabelle 14.1: Verschiedene SCSI-Varianten

Name	Breite	Datenrate	Geräte	Erklärung
SCSI-1	8 Bit	⩽ 5 MiB/s	8	»Urahn«
SCSI-2 »Fast«	8 Bit	10 MiB/s	8	
SCSI-2 »Wide«	16 Bit	20 MiB/s	16	
SCSI-3 »Ultra«	8 Bit	20 MiB/s	8	
SCSI-3 »Ultrawide«	16 Bit	40 MiB/s	16	
Ultra2 SCSI	16 Bit	80 MiB/s	16	LVD-Bus[a]
Ultra-160 SCSI[b]	16 Bit	160 MiB/s	16	LVD-Bus
Ultra-320 SCSI[c]	16 Bit	320 MiB/s	16	LVD-Bus
Ultra-640 SCSI	16 Bit	640 MiB/s	16	LVD-Bus

Heute vorangetrieben werden vor allem die seriellen SCSI-Implementierungen, allen voran »Serial Attached SCSI« (SAS). Ähnlich wie bei SATA ist die Datenübertragung potentiell schneller (im Moment ist SAS etwas langsamer als die schnellste parallele SCSI-Version, Ultra-640 SCSI) und elektrisch weit weniger aufwendig. Insbesondere gibt es bei den schnellen parallelen SCSI-Versionen Probleme mit dem Takt, die auf elektrische Eigenschaften der Kabel und der Terminierung zurückgehen und bei SAS nicht existieren (die lästige Terminierung ist überhaupt nicht mehr nötig).

SAS und SATA sind relativ eng verwandt; die wesentlichen Unterschiede bestehen darin, dass SAS Dinge erlaubt wie die Ansteuerung eines Laufwerks über mehrere Kabelwege für Redundanz (»Multipath-I/O«; SATA macht das nur unter Verrenkungen möglich), umfangreichere Diagnose- und Protokollfunktionen unterstützt und auf einer höheren Signalspannung aufbaut, was längere Kabel (bis zu 8 m) und den Einsatz in größeren Servern ermöglicht.

Die Kompatibilität zwischen SATA und SAS geht so weit, dass Sie SATA-Platten an einem SAS-Controller verwenden können (aber nicht umgekehrt).

Vorkommen

»Reinrassiges« SCSI findet sich bei PC-Systemen vor allem in Servern; Arbeitsplatzrechner und »Konsumenten-PCs« verwenden eher IDE oder SATA zum Anschluss von Massenspeichern und USB für andere Geräte. IDE- und USB-basierte Geräte sind viel billiger als SCSI-basierte Geräte – SATA-Platten kosten zum Beispiel etwa ein Drittel oder ein Viertel so viel wie gleich große SCSI-Platten.

Wobei wir erwähnen müssen, dass die SCSI-Platten in der Regel gezielt für den Servereinsatz gebaut werden und daher auf hohe Geschwindigkeit und lange Lebensdauer im Betrieb getrimmt sind. Die SATA-Platten für Arbeitsplatzrechner kommen nicht mit den gleichen Garantien, sollen nicht lärmen wie ein Düsenjäger beim Start und relativ häufige Start- und Stoppvorgänge aushalten können.

Als Linux-Administrator sollten Sie auch über SCSI-Kenntnisse verfügen, selbst wenn Sie gar kein SCSI im System haben, denn aus der Sicht des Linux-Kerns werden neben SATA- viele USB- oder FireWire-Geräte wie SCSI-Geräte angesprochen und nutzen dieselbe Infrastruktur.

SCSI-ID

Jedes Gerät an einem SCSI-Bus braucht eine eindeutige »SCSI-ID«. Mit dieser Zahl zwischen 0 und 7 (15 bei den breiten Bussen) wird das Gerät adressiert. Die meisten »echten« SCSI-Geräte haben zu ihrer Einstellung »Jumper« oder einen Schalter; bei Fibre-Channel-, USB- oder SATA-Geräten, die über die SCSI-Infrastruktur angesprochen werden, sorgt das System dafür, dass sie eine eindeutige SCSI-ID zugeordnet bekommen.


Um (echtes) SCSI nutzen zu können, muss ein PC mindestens einen Hostadapter (kurz *host*) haben. Hostadapter auf der Hauptplatine und bessere Steckkarten-Hostadapter haben ein SCSI-BIOS, mit dem Booten über SCSI möglich ist. Dort können Sie auch prüfen, welche SCSI-IDs frei und welche belegt sind und von welchem SCSI-Gerät gegebenenfalls gebootet werden soll.

Hostadapter

SCSI-BIOS


Der Hostadapter zählt als Gerät auf dem SCSI-Bus – außer ihm können Sie also noch 7 (oder 15) andere Geräte unterbringen.


Wenn Ihr BIOS von SCSI-Geräten booten kann, können Sie außerdem in der Bootreihenfolge vorgeben, ob die ATA-Platte `C:` den (potenziell) bootfähigen SCSI-Geräten vorgezogen werden soll.

Bootreihenfolge


Wichtig für die korrekte Funktion eines parallelen SCSI-Systems ist die passende **Terminierung** des SCSI-Busses. Diese kann entweder über einen speziellen Stecker (»Terminator«) erfolgen oder bei einzelnen Geräten ein- und ausgeschaltet werden. Fehlerhafte Terminierung kann als Ursache für diverse SCSI-Probleme herangezogen werden, Sie sollten sich bei SCSI-Schwierigkeiten darum immer zuerst vergewissern, dass die Terminierung stimmt. Bei SAS ist keine Terminierung nötig.

Terminierung

USB Mit den neuen schnellen Varianten von USB müssen Sie bei der Anbindung von Massenspeicher kaum noch Kompromisse machen – die Lese- und Schreibgeschwindigkeit wird vom Speichergerät begrenzt und nicht mehr wie bei USB 1.1 und USB 2.0 vom Bus. Linux verwaltet USB-Speichergeräte wie SCSI-Geräte.

Übungen


14.1 [1] Wie viele Platten oder SSDs enthält Ihr Computer? Mit welcher Kapazität? Wie sind sie an den Rechner angebunden (SATA, …)?

14.3 Partitionierung

14.3.1 Grundlagen

Massenspeichergeräte wie Festplatten und SSDs werden üblicherweise »partitioniert«, also in mehrere logische Speichergeräte aufgeteilt, die dann vom Betriebssystem unabhängig angesprochen werden können. Das hat nicht nur den Vorteil, dass es leichter ist, dem jeweiligen Einsatzzweck angepasste Datenstrukturen verwenden zu können – manchmal ist Partitionierung auch die einzige Möglichkeit, ein sehr großes Speichermedium vollständig nutzbar zu machen, wenn Grenzen innerhalb des Betriebssystems eine Verwendung des Mediums »als Ganzes« nicht zulassen (auch wenn das heute eher selten das Problem ist).

Partitionierung hat unter anderem die folgenden Vorteile:

- Logisch getrennte Teile des Systems können getrennt werden. Beispielsweise können Sie die Daten Ihrer Benutzer auf einer anderen Partition ablegen als das eigentliche Betriebssystem. Damit können Sie das Betriebssystem komplett neu installieren, ohne dass die Benutzerdaten gefährdet werden. Bei den oft nur eingeschränkt funktionierenden »Upgrade«-Möglichkeiten selbst aktueller Distributionen ist das wichtig. Auch wenn Inkonsistenzen in einem Dateisystem auftauchen, ist davon zunächst nur eine Partition betroffen.

- Die Struktur des Dateisystems kann den zu speichernden Daten angepasst werden. Die meisten Dateisysteme verwalten den Platz in »Blöcken« fester Größe, wobei jede Datei, egal wie klein sie ist, zumindest einen kompletten

Block belegt. Bei einer Blockgröße von 4 KiB bedeutet das etwa, dass eine 500 Bytes lange Datei nur rund 1/8 ihres Blocks ausnutzt – der Rest liegt als »interner Verschnitt« brach. Wenn Sie wissen, dass in einem Verzeichnis vor allem kleine Dateien liegen werden (Stichwort: Mailserver), dann kann es sinnvoll sein, dieses Verzeichnis auf eine eigene Partition zu legen, deren Dateisystem kleine Blöcke (1 oder 2 KiB) benutzt. Das verringert den Platzverlust durch »internen Verschnitt« erheblich. Einige Datenbankserver dagegen arbeiten am liebsten auf »rohen« Partitionen ganz ohne Dateisystem, die sie komplett selbst verwalten. Auch das muss das Betriebssystem ermöglichen.

- »Wild gewordene« Prozesse oder unvorsichtige Benutzer können den kompletten Plattenplatz auf einem Dateisystem belegen. Es ist zumindest auf wichtigen Serversystemen sinnvoll, Benutzerdaten (inklusive Druckaufträgen, ungelesener E-Mail und Ähnlichem) nur auf solchen Partitionen zu erlauben, die voll laufen können, ohne dass das System selbst dabei Probleme bekommt, etwa indem wichtige Protokolldateien nicht weitergeführt werden können.

Zur Zeit gibt es zwei konkurrierende Methoden dafür, Platten für PCs zu partitionieren. Die traditionelle Methode geht zurück auf die 1980er Jahre, als die ersten Festplatten (mit monumentalen Kapazitäten von 5 oder 10 MB) auf den Markt kamen. In den letzten Jahren wurde eine modernere Methode eingeführt, die mit diversen Einschränkungen der traditionellen Methode aufräumt, aber zum Teil spezielle Werkzeuge benötigt.

Platten werden so gut wie immer partitioniert, auch wenn manchmal nur eine einzige Partition angelegt wird. Bei USB-Sticks verzichtet man mitunter auf eine Partitionierung.

14.3.2 Die traditionelle Methode (MBR)

Die traditionelle Methode legt Partitionsinformationen im *Master Boot Record* (MBR) ab, dem ersten Sektor (Nummer 0) einer Festplatte. (Traditionell sind Sektoren auf PC-Festplatten 512 Bytes lang, aber siehe unten.) Der Platz dort – 64 Bytes ab dem Versatz 446 – reicht für vier **primäre Partitionen**. Wenn Sie mehr als vier Partitionen anlegen wollen, müssen Sie eine dieser primären Partitionen zu einer **erweiterten Partition** erklären. In einer erweiterten Partition können Sie dann **logische Partitionen** anlegen.

primäre Partitionen

erweiterten Partition

logische Partitionen

Die Informationen über logische Partitionen stehen nicht im MBR, sondern am Anfang der jeweiligen (erweiterten bzw. logischen) Partition, sind also über die Platte verstreut.

Die Partitionseinträge speichern heute in der Regel die Anfangssektornummer auf der Platte und die Länge der betreffenden Partition in Sektoren[4]. Da die betreffenden Werte 32 Bit breit sind, ergibt sich daraus bei den gängigen 512-Byte-Sektoren eine maximale Partitionsgröße von 2 TiB.

Inzwischen sind Platten auf dem Markt, die größer sind als 2 TiB. Solche Platten können Sie mit dem MBR-Verfahren nicht komplett ausnutzen. Ein gängiger Trick besteht darin, Platten zu verwenden, die statt 512 Bytes großen Sektoren 4096 Bytes große Sektoren verwenden. Damit kommen Sie auch mit MBR auf 16 TiB pro Partition, allerdings kommt nicht jedes Betriebssystem mit solchen »4Kn«-Platten klar. (Linux ab Kernel 2.6.31, Windows ab 8.1 bzw. Server 2012.)

[4] Früher war es üblich, Partitionen über die Zylinder-, Kopf- und Sektoradresse der betreffenden Sektoren zu beschreiben, das ist aber schon lange verpönt.

Tabelle 14.2: Partitionstypen (hexadezimal) für Linux

Typ	Beschreibung
81	Linux-Daten
82	Linux-Auslagerungsspeicher (*swap space*)
86	RAID-Superblock (altertümlich)
8E	Linux LVM
E8	LUKS (verschlüsselte Partition)
EE	»Schutzpartition« für GPT-partitionierte Platte
FD	RAID-Superblock mit Auto-Erkennung
FE	Linux LVM (altertümlich)

4 KiB große Sektoren sind auf Festplatten auch unabhängig von der Partitionierung vorteilhaft. Die größeren Sektoren sind effizienter für die Speicherung großer Dateien und lassen bessere Fehlerkorrektur zu. Deswegen sind »512e«-Platten auf dem Markt, die intern 4 KiB große Sektoren verwenden, aber nach außen so tun, als hätten sie 512 Byte große Sektoren. Das bedeutet, dass, wenn ein einzelner 512-Byte-Sektor geschrieben werden muss, die angrenzenden 7 Sektoren gelesen und ebenfalls neu geschrieben werden müssen (ein gewisser, aber normalerweise erträglicher Effizienzverlust, da Daten meist in größeren Stücken geschrieben werden). Sie müssen nur bei der Partitionierung darauf achten, dass die 4-KiB-Blöcke, die Linux intern zum Plattenzugriff nutzt, zu den internen 4-KiB-Sektoren der Platte passen – ist das nicht der Fall, kann es sein, dass zum Schreiben eines 4-KiB-Linux-Blocks *zwei* 4-KiB-Sektoren der Platte gelesen *und* geschrieben werden müssen, und das wäre nicht gut. (Zum Glück passen die Partitionierungswerkzeuge mit auf.)

Die Partitionstabelle enthält außer der Startadresse und der Länge der (primären) Partitionen auch einen Partitionstyp, der lose beschreibt, mit welcher Sorte Datenverwaltungsstruktur auf der Partition zu rechnen ist. Eine Auswahl von Linux-Partitionstypen steht in Tabelle 14.2.

Partitionstyp

14.3.3 Die moderne Methode (GPT)

In den späten 1990er Jahren entwickelte Intel eine neue Methode zur Partitionierung, die die Einschränkungen des MBR-Ansatzes aufheben sollte, namentlich *GUID Partition Table* oder GPT.

GPT wurde Hand in Hand mit UEFI entwickelt und ist heute Teil der UEFI-Spezifikation. Allerdings können Sie auch mit einem BIOS-basierten Linux auf GPT-partitionierte Platten zugreifen und umgekehrt.

GPT verwendet 64-Bit-Sektoradressen und erlaubt damit eine maximale Plattengröße von 8 ZiB – Zebibyte, falls Ihnen dieses Präfix nicht geläufig sein sollte. 1 ZiB sind 2^{70} Byte oder grob gesagt gut eine Billion Tebibytes. Damit dürfte sogar die NSA eine Weile auskommen. (Plattenhersteller, die bekanntlich lieber in Zehner- als in Zweierpotenzen rechnen, werden Ihnen eine 8-ZiB-Platte natürlich als 9,4-Zettabyte-Platte verkaufen.)

Bei GPT bleibt der erste Sektor der Platte reserviert für einen »Schutz-MBR«, der die komplette Platte aus MBR-Sicht als partitioniert kennzeichnet. Dies soll Probleme vermeiden, wenn eine GPT-partitionierte Platte an einen Rechner angeschlossen wird, der mit GPT nicht umgehen kann.

Der zweite Sektor (Adresse 1) enthält den »GPT-Kopf« (engl. *GPT header*), der Verwaltungsinformationen für die komplette Platte speichert. Partitionierungsinformationen stehen typischerweise im dritten und den folgenden Sektoren.

GUID	Bedeutung
00000000-0000-0000-0000-000000000000	Unbenutzter Eintrag
C12A7328-F81F-11D2-BA4B-00A0C93EC93B	EFI-Systempartition (ESP)
21686148-6449-6E6F-744E-656564454649	BIOS-Boot-Partition
0FC63DAF-8483-4772-8E79-3D69D8477DE4	Linux-Dateisystem
A19D880F-05FC-4D3B-A006-743F0F84911E	Linux-RAID-Partition
0657FD6D-A4AB-43C4-84E5-0933C84B4F4F	Linux-Auslagerungsspeicher
E6D6D379-F507-44C2-A23C-238F2A3DF928	Linux LVM
933AC7E1-2EB4-4F13-B844-0E14E2AEF915	`/home`-Partition
3B8F8425-20E0-4F3B-907F-1A25A76F98E8	`/srv`-Partition
7FFEC5C9-2D00-49B7-8941-3EA10A5586B7	`dm-crypt`-Partition
CA7D7CCB-63ED-4C53-861C-1742536059CC	LUKS-Partition

Tabelle 14.3: Partitionstyp-GUIDs für GPT (Auswahl)

Der GPT-Kopf verweist auf die Partitionierungsinformationen und deshalb könnten sie eigentlich irgendwo auf der Platte stehen. Allerdings ist es vernünftig, sie direkt hinter den GPT-Kopf zu schreiben. Der UEFI-Standard sieht ein Minimum von 16 KiB für Partitionierungsinformationen vor (egal wie groß die Sektoren auf der Platte sind).

Bei einer Platte mit 512-Byte-Sektoren ist bei 16 KiB Platz für Partitionierungsinformationen der erste anderweitig nutzbare Sektor auf der Platte der mit der Adresse 34. Sie sollten sich aber verkneifen, die erste Partition auf der Platte an diese Adresse zu legen, da Sie sonst Schwierigkeiten mit 512e-Platten bekommen. Der nächste korrekt aufgereihte Sektor ist nämlich der mit der Adresse 40.

Zur Sicherheit werden bei GPT die Partitionierungsinformationen auch noch einmal am Ende der Platte abgelegt.

Traditionell werden Partitionsgrenzen immer an den Anfang einer neuen »Spur« auf der Platte platziert. Spuren sind natürlich ein Relikt aus der Festplatten-Altsteinzeit, da heutige Platten linear adressiert werden (mit anderen Worten, die Sektoren werden vom Anfang bis zum Ende der Platte fortlaufend durchnummeriert) – aber die Idee, eine Platte durch eine Kombination aus einer Anzahl von Schreib-/Leseköpfen, einer Anzahl von »Zylindern« und einer Anzahl von Sektoren pro »Spur« (eine Spur ist ein konzentrischer Kreis, den ein einzelner Kopf auf einem festen Zylinder beschreibt) zu beschreiben, hat sich bemerkenswert lange gehalten. Da die maximale Anzahl von Sektoren pro Spur 63 beträgt, würde das bedeuten, dass die erste Partition beim Block 63 anfängt, und das ist natürlich katastrophal für 512e-Platten.

Seit Windows Vista ist es üblich, die erste Partition 1 MiB nach dem Anfang der Platte (bei 512-Byte-Sektoren mit dem Sektor 2048) zu starten. Das ist auch für Linux keine schlechte Idee, denn in dem üppigen freien Platz zwischen der Partitionstabelle und der ersten Partition läßt sich zum Beispiel der Bootlader GRUB unterbringen. (Der Platz zwischen dem MBR und dem Sektor 63 hat dafür früher aber auch gereicht.)

Die Einträge in der Partitionstabelle sind mindestens 128 Bytes lang und enthalten neben je 16 Bytes für eine GUID für den Partitionstyp und die Partition selbst und je 8 Bytes für Anfangs- und Endblocknummer noch 8 Bytes für »Attribute« und 72 Bytes für einen Partitionsnamen. Man kann debattieren, ob 16-Byte-GUIDs für Partitionstypen nötig sind, aber einerseits heißt das Schema nun mal »GUID Partition Table«, und andererseits ist so immerhin sichergestellt, dass die Partitionstypen so schnell nicht ausgehen. In Tabelle 14.3 ist eine Auswahl zu bestaunen.


Linux kann gemäß GPT partitionierte Medien verwenden. Dazu muss im Kernel die Option »EFI GUID Partition support« (`CONFIG_EFI_PARTITION`) eingeschaltet sein, aber das ist bei aktuellen Distributionen der Fall. Ob die Installationsprozedur es ermöglicht, entsprechend partitionierte Platten anzulegen, ist eine andere Frage, genau wie die Frage, ob der Bootlader damit umgehen kann. Aber das gehört nicht hierher.

14.4 Linux und Massenspeicher

Wird ein Massenspeichergerät an einen Linux-Rechner angeschlossen, versucht der Linux-Kern, allfällige Partitionen zu finden. Für das Gerät selbst und die gefundenen Partitionen werden dann blockorientierte Gerätedateien in `/dev` angelegt. Anschließend können Sie auf die Gerätedateien der Partitionen zugreifen und die dort gespeicherten Verzeichnisstrukturen im Dateisystem des Rechners sichtbar machen.


Ein neues Massenspeichergerät hat zunächst keine Partitionen. In diesem Fall können Sie natürlich mit den entsprechenden Werkzeugen Partitionen anlegen. Das besprechen wir später in diesem Kapitel. Der nächste Schritt nach der Partitionierung besteht darin, auf den Partitionen Dateisysteme zu generieren. Das wird im Detail im Kapitel 15 erklärt.

Die Gerätenamen für Massenspeicher sind üblicherweise `/dev/sda`, `/dev/sdb`, … in der Reihenfolge des Erkennens der jeweiligen Geräte. Partitionen werden durchnummeriert, das Gerät `/dev/sda` hat also die Partitionen `/dev/sda1`, `/dev/sda2`, … Maximal 15 Partitionen pro Gerät sind möglich. Dabei sind, wenn `/dev/sda` eine nach dem MBR-Schema partitionierte Platte ist, `/dev/sda1` bis `/dev/sda4` die primären Partitionen (gegebenenfalls mit einer erweiterten), während allfällige logische Partitionen ab `/dev/sda5` nummeriert werden (egal ob es vier primäre Partitionen auf der Platte gibt oder weniger).


Das »`s`« in `/dev/sda` kommt von »SCSI«. Heutzutage werden fast alle Massenspeicher von Linux wie SCSI-Geräte angesprochen.


Für P-ATA-Platten gibt es auch noch einen anderen, spezifischeren Mechanismus. Dabei werden die IDE-Controller im Rechner direkt angesprochen – die beiden Laufwerke am ersten Controller heißen `/dev/hda` und `/dev/hdb`, die am zweiten `/dev/hdc` und `/dev/hdd`. (Diese Namen werden unabhängig davon verwendet, ob die Laufwerke tatsächlich existieren oder nicht – wenn Sie eine Festplatte und ein CD-ROM-Laufwerk im System haben, tun Sie gut daran, die eine am einen und das andere am anderen Controller anzuschließen, damit die beiden sich nicht gegenseitig behindern. Also haben Sie hinterher `/dev/hda` für die Platte und `/dev/hdc` für das CD-ROM-Laufwerk.) Partitionen auf P-ATA-Platten heißen wie gehabt `/dev/hda1`, `/dev/hda2` und so weiter. In diesem Schema sind 63 (!) Partitionen pro Platte erlaubt.


Wenn Sie heute noch einen Rechner mit P-ATA-Platten haben, werden Sie feststellen, dass in den allermeisten Fällen auch hier die SCSI-Infrastruktur verwendet wird (zu erkennen an den Gerätedateinamen à la `/dev/sda`). Dies ist aus Bequemlichkeits- und Konsistenzgründen sinnvoll. Einige wenige P-ATA-Controller werden von der SCSI-Infrastruktur nicht unterstützt und müssen die alte P-ATA-spezifische Infrastruktur verwenden.


Eine Migration eines existierenden Linux-Systems von »traditionellen« P-ATA-Treibern auf die SCSI-Infrastruktur sollte gut überlegt sein und damit einhergehen, die Konfiguration in `/etc/fstab` so umzustellen, dass Dateisysteme nicht über den Gerätenamen, sondern über Volume Labels oder UUIDs eingehängt werden, die unabhängig vom Gerätenamen der Partition sind. (Siehe hierzu Abschnitt 15.2.3.)

Architektur

Das Massenspeicher-Subsystem des Linux-Kerns hat eine »dreigeschossige« Architektur: Ganz unten befinden sich die Treiber für die einzelnen SCSI-Hostadapter, SATA- und USB-Controller und so weiter, darüber eine generische »Mittelschicht« und ganz oben Treiber für die verschiedenen Geräte (Platten, Bandlaufwerke, …), mit denen Sie auf einem SCSI-Bus rechnen können. Dazu gehört auch ein »generischer« Treiber, der Geräte ohne speziellen Treiber wie Scanner oder CD-ROM-Brenner zugänglich macht. (Falls Sie die noch irgendwo finden können.)

LUNs

Jeder SCSI-Hostadapter unterstützt einen oder mehrere Busse (genannt *channel*), an jeden Bus können bis zu 7 bzw. 15 andere Geräte angeschlossen sein und jedes Gerät selbst kann mehrere *local unit numbers* oder LUNs unterstützen, etwa die einzelnen CDs in einem CD-ROM-Wechsler (selten benutzt). Jedes SCSI-Gerät im System lässt sich also durch das Quadrupel (⟨*host*⟩, ⟨*channel*⟩, ⟨*ID*⟩, ⟨*LUN*⟩) eindeutig beschreiben; meist reichen auch schon (⟨*host*⟩, ⟨*channel*⟩, ⟨*ID*⟩) aus.

Früher konnten Sie im Verzeichnis `/proc/scsi/scsi` Informationen über die angeschlossenen »SCSI«-Geräte abrufen. Auf aktuellen Systemen ist das aber nicht mehr vorhanden, wenn der Linux-Kern nicht mit *legacy* `/proc/scsi` *support* übersetzt wurde.

Heutzutage befinden die Informationen über »SCSI-Controller« sich in `/sys/class/scsi_host` (ein Verzeichnis pro Controller). Das Ganze ist leider nicht ganz so anschaulich wie früher. Sie können trotzdem buddeln gehen:

```
# cd /sys/class/scsi_host/host0/device
# ls
power  scsi_host  subsystem  target0:0:0  uevent
# cd target0:0:0; ls
0:0:0:0  power  subsystem  uevent
# ls 0:0:0:0/block
sda
```

Aufschlussreich ist auch ein Blick in `/sys/bus/scsi/devices`:

```
# ls /sys/bus/scsi/devices
0:0:0:0  10:0:0:0  host1   host2  host4  target0:0:0  target10:0:0
1:0:0:0  host0     host10  host3  host5  target1:0:0
```

Die Gerätenamen `/dev/sda`, `/dev/sdb` usw. haben den Nachteil, nicht besonders aufschlussreich zu sein. Ferner werden sie in der Reihenfolge des Erscheinens der Geräte vergeben. Wenn Sie also heute erst Ihren MP3-Player und dann Ihre Digitalkamera anstöpseln, bekommen sie zum Beispiel die Gerätenamen `/dev/sdb` und `/dev/sdc`; wenn Sie morgen mit der Digitalkamera anfangen und mit dem MP3-Player weitermachen, sind die Namen möglicherweise umgekehrt. Das ist natürlich ärgerlich. Zum Glück vergibt `udev` außer den traditionellen Gerätenamen auch noch einige symbolische Links, die Sie in `/dev/block` finden können:

```
# ls -l /dev/block/8:0
lrwxrwxrwx 1 root root 6 Jul 12 14:02 /dev/block/8:0 -> ../sda
# ls -l /dev/block/8:1
lrwxrwxrwx 1 root root 6 Jul 12 14:02 /dev/block/8:1 -> ../sda1
# ls -l /dev/disk/by-id/ata-ST9320423AS_5VH5TBTC
lrwxrwxrwx 1 root root 6 Jul 12 14:02 /dev/disk/by-id/▷
 ◁ ata-ST9320423AS_5VH5TBTC -> ../../sda
# ls -l /dev/disk/by-id/ata-ST9320423AS_5VH5TBTC-part1
lrwxrwxrwx 1 root root 6 Jul 12 14:02 /dev/disk/by-id/▷
 ◁ ata-ST9320423AS_5VH5TBTC-part1 -> ../../sda1
```

```
# ls -l /dev/disk/by-path/pci-0000:00:1d.0-usb-▷
 ◁ 0:1.4:1.0-scsi-0:0:0:0
lrwxrwxrwx 1 root root 6 Jul 12 14:02 /dev/disk/by-path/▷
 ◁ pci-0000:00:1d.0-usb-0:1.4:1.0-scsi-0:0:0:0  -> ../../sdb
# ls -l /dev/disk/by-uuid/c59fbbac-9838-4f3c-830d-b47498d1cd77
lrwxrwxrwx 1 root root 10 Jul 12 14:02 /dev/disk/by-uuid/▷
 ◁ c59fbbac-9838-4f3c-830d-b47498d1cd77 -> ../../sda1
# ls -l /dev/disk/by-label/root
lrwxrwxrwx 1 root root 10 Jul 12 14:02 /dev/disk/by-label/root ◁
▷ -> ../../sda1
```

Diese Gerätenamen werden von Kenndaten wie der (eindeutigen) Seriennummer des Plattenlaufwerks, der Position auf dem PCIe-Bus, der UUID oder einer Kennung des Dateisystems abgeleitet und sind unabhängig vom Namen der tatsächlichen Gerätedatei.

Übungen

14.2 [!2] In Ihrem System befinden sich zwei Festplatten, die per SATA angeschlossen sind. Die erste Platte hat zwei primäre und zwei logische Partitionen, die zweite Platte ist in eine primäre und drei logische Partitionen. Welche Bezeichnungen haben diese Partitionen unter Linux?

14.3 [!1] Untersuchen Sie das `/dev`-Verzeichnis Ihres Rechners. Welche Speichermedien stehen zur Verfügung und wie heißen die dazugehörigen Gerätedateien? (Schauen Sie auch in `/dev/block` und `/dev/disk`.)

14.4 [1] Stecken Sie einen USB-Stick in Ihren Computer. Prüfen Sie, ob in `/dev` neue Gerätedateien dazugekommen sind. Wenn ja, welche?

14.5 Platten partitionieren

14.5.1 Prinzipielles

Bevor Sie die (möglicherweise einzige) Platte eines Linux-Systems partitionieren, sollten Sie kurz darüber nachdenken, wie ein geeignetes Partitionsschema aussehen könnte und wie groß Sie die Partitionen machen sollten. Nachträgliche Änderungen sind nämlich im besten Fall mühsam und lästig und bedingen im schlimmsten Fall eine komplette Neuinstallation des Systems (was extra mühsam und lästig wäre). (Siehe Abschnitt 14.7 für einen alternativen, weit weniger schmerzbehafteten Ansatz.)

Hier ein paar grundlegende Tipps für die Partitionierung:

- Sie sollten außer der Partition mit dem Wurzelverzeichnis `/` zumindest eine separate Partition für das Dateisystem mit dem Verzeichnis `/home` vorsehen. Das ermöglicht es Ihnen, das Betriebssystem sauber von Ihren eigenen Daten zu trennen, und erleichtert Upgrades der Distribution oder gar einen Wechsel von einer Linux-Distribution zu einer ganz anderen.

  Wenn Sie diesen Ansatz verfolgen, dann sollten Sie wahrscheinlich auch die Verzeichnisse `/usr/local` und `/opt` mit symbolischen Links nach (zum Beispiel) `/home/usr-local` und `/home/opt` verlagern. Auf diese Weise sind auch diese von Ihnen lokal beschickten Verzeichnisse auf »Ihrer« Partition und können zum Beispiel von regelmäßigen Sicherungskopien besser erfasst werden.

- Sie können ein grundlegendes Linux-System ohne Weiteres in eine 2-GiB-Partition quetschen, aber in Anbetracht der heutigen (niedrigen) Kosten pro

Gibibyte für Plattenplatz ist es für die /-Partition empfehlenswert, nicht zu sehr zu knausern. Mit ungefähr 30 GiB sind Sie höchstwahrscheinlich jenseits von Gut und Böse und haben auch noch Platz für Protokolldateien, heruntergeladene Distributionspakete während eines größeren Upgrades und so weiter.

- Auf Server-Systemen kann es sinnvoll sein, auch für `/tmp`, `/var` und ggf. `/srv` eigene Partitionen vorzusehen. Der Hintergrund ist, dass in diesen Verzeichnissen Benutzer Daten ablegen können (außer direkten Dateien etwa ungelesene oder unverschickte E-Mail, unausgedruckte Druckjobs und so weiter). Wenn diese Verzeichnisse auf eigenen Partitionen liegen, können Benutzer so nicht das komplette System vollmüllen und dadurch möglicherweise Probleme verursachen.

- Sie sollten Auslagerungsspeicher (*swap space*) etwa im Wert des RAM-Speichers Ihres Rechners vorsehen, bis zu einem Maximum von 8 GiB oder so. Viel mehr hat wenig Zweck, aber auf Arbeitsplatz- und mobilen Rechnern möchten Sie vielleicht die Möglichkeit ausnutzen, Ihren Rechner in einen »Tiefschlaf« zu versetzen, statt ihn auszuschalten, um einen Neustart zu beschleunigen und gleich wieder dort herauszukommen, wo Sie zuletzt gewesen sind – und die entsprechenden Infrastrukturen verwenden gerne den Auslagerungsspeicher, um den RAM-Inhalt zu sichern.

  
  Früher galt die Faustregel, dass der Auslagerungsspeicher etwa doppelt bis dreimal so groß sein sollte wie die Menge von RAM im System. Diese Faustregel kommt von traditionellen Unix-Systemen, wo der RAM-Speicher als »Cache« für den Auslagerungsspeicher fungiert. In Linux ist das nicht so, sondern RAM und Auslagerungsspeicher werden addiert – auf einem Rechner mit 4 GiB RAM und 2 GiB Auslagerungsspeicher können Sie also Prozesse im Gesamtwert von 6 GiB laufen lassen. Bei einer RAM-Größe von 8 GiB 16 bis 24 GiB Auslagerungsspeicher vorzusehen, wäre absurd.

  
  Sie sollten den RAM-Speicher eines Rechners (vor allem eines Servers) so groß wählen, dass im laufenden Betrieb praktisch kein Auslagerungsspeicher benötigt wird; bei einem Server mit 8 GiB RAM brauchen Sie normalerweise keine 16 GiB Auslagerungsspeicher, aber ein Gigabyte oder zwei, um auf der sicheren Seite zu sein, schaden bestimmt nicht (vor allem bei den heutigen Preisen für Plattenplatz). Dies sorgt dafür, dass bei RAM-Knappheit der Rechner erst mal langsam wird, bevor Prozesse komplett abstürzen, weil sie vom Betriebssystem keinen Speicher bekommen können.

- Wenn Sie mehrere (physikalische) Platten zur Verfügung haben, kann es nützlich sein, das System über die verfügbaren Platten zu verteilen, um die Zugriffsgeschwindigkeit auf einzelne Komponenten zu erhöhen.

  
  Traditionell würde man das Wurzeldateisystem (/ mit den wesentlichen Unterverzeichnissen `/bin`, `/lib`, `/etc` und so weiter) auf eine Platte und das Verzeichnis `/usr` mit seinen Unterverzeichnissen auf einem eigenen Dateisystem auf einer anderen Platte platzieren. Allerdings geht der Trend bei Linux eindeutig weg von einer (zugegeben schon lange künstlichen) Trennung zwischen zum Beispiel `/bin` und `/usr/bin` oder `/lib` und `/usr/lib` und hin zu einem Wurzeldateisystem, das beim Systemstart in einer RAM-Disk angelegt wird. Ob die althergebrachte Trennung von / und `/usr` also in Zukunft noch viel bringt, sei dahingestellt.

  
  Was sich auf jeden Fall lohnen kann, ist, Auslagerungsspeicher über mehrere Platten zu verteilen. Linux verwendet dann immer die am wenigsten anderweitig beschäftigte Platte zum Auslagern.

Immer vorausgesetzt, es ist noch freier Platz auf dem Medium verfügbar, können (auch während des laufenden Betriebs) neue Partitionen angelegt und eingebunden werden. Dieser Vorgang gliedert sich in folgende Schritte:

1. Sichern der aktuellen Bootsektoren und Daten auf der betroffenen Festplatte
2. Aufteilen des Plattenplatzes mit `fdisk` (oder einem vergleichbaren Programm)
3. Ggf. Generieren von Dateisystemen auf den neuen Partitionen (»Formatieren«)
4. Einbinden der neuen Dateisysteme mit `mount` bzw. `/etc/fstab`


Die Punkte 3 und 4 dieser Liste betrachten wir in größerer Ausführlichkeit im Kapitel 15.

Daten und Bootsektor-Inhalte können unter anderem mit dem Kommando `dd` gesichert werden.

```
# dd if=/dev/sda of=/dev/st0
```

sichert zum Beispiel die gesamte Festplatte `sda` auf ein Magnetband.

Sie sollten sich vor Augen halten, dass die Partitionierung eines Speichermediums nichts mit den gespeicherten Daten zu tun hat. Die Partitionstabelle gibt lediglich an, wo auf der Platte der Linux-Kern die Partitionen und damit die Dateistrukturen finden kann. Wenn der Linux-Kern eine Partition erst einmal gefunden hat, ist der Inhalt der Partitionstabelle irrelevant, bis er sich wieder auf die Suche macht, etwa beim nächsten Systemstart. Das gibt Ihnen – wenn Sie mutig (oder tollkühn) sind – weitreichende Möglichkeiten, Ihr System auch im laufenden Betrieb zu modifizieren. Zum Beispiel können Sie Partitionen durchaus vergrößern (wenn unmittelbar hinter dem Ende der Partition unbenutzter Platz steht oder eine Partition, auf deren Inhalt Sie verzichten können) oder verkleinern (und im so freigewordenen Platz gegebenenfalls eine andere Partition unterbringen). Solange Sie mit der gebotenen Vorsicht vorgehen, kann dabei relativ wenig schiefgehen.


Das sollte Sie natürlich in keinster Weise davon abhalten, vor solchen Operationen am offenen Herzen angemessene Sicherheitskopien anzulegen.


Außerdem müssen bei solchen Fisimatenten die Dateisysteme auf den Platten mitspielen (viele Linux-Dateisysteme lassen sich ohne Datenverlust vergrößern und einige auch verkleinern), oder Sie müssen bereit sein, die Daten wegzuschieben, ein neues Dateisystem zu generieren und die Daten dann zurückzuholen.

14.5.2 Platten partitionieren mit `fdisk`

`fdisk` ist ein interaktives Programm zur Manipulation der Partitionstabellen von Festplatten. Dabei können auch »fremde« Partitionstypen, zum Beispiel DOS-Partitionen, erstellt werden. Die Laufwerke werden, über die entsprechenden Gerätedateien (etwa `/dev/sda` für die erste Platte) angesprochen. `fdisk`


`fdisk` beschränkt sich darauf, die Partition in die Partitionstabelle einzutragen und den richtigen Partitionstyp zu setzen. Wenn Sie mit `fdisk` zum Beispiel eine DOS- oder NTFS-Partition anlegen, dann heißt das nur, dass die Partition als solche in der Partitionstabelle steht, nicht dass Sie als nächstes DOS oder Windows NT booten und Dateien in die Partition schreiben können. Davor müssen Sie erst noch ein Dateisystem anlegen, also die nötigen Verwaltungsdatenstrukturen aufbringen. Mit Linux-Bordmitteln können Sie das für viele, aber nicht alle Nicht-Linux-Dateisysteme machen.

Beim Aufruf von »`fdisk` ⟨*Gerät*⟩« meldet sich das Programm mit

Neue (leere) Platte

```
# fdisk /dev/sdb
Welcome to fdisk (util-linux 2.25.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x68d97339.

Command (m for help): _
```

Mit m bekommen Sie eine Liste der verfügbaren Kommandos angezeigt.

Mit `fdisk` können Sie Platten nach dem MBR- wie auch dem GPT-Schema partitionieren. `fdisk` erkennt eine vorhandene Partitionstabelle und richtet sich danach. Auf einer leeren (unpartitionierten) Platte wird zunächst eine MBR-Partitionstabelle angelegt, aber das können Sie dann ändern (wir zeigen Ihnen gleich, wie).

Eine neue Partition können Sie mit dem Kommando »`n`« anlegen:

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048): ↩
Last sector, +sectors or +sizeK,M,G,T,P (2048-2097151,▷
 ◁ default 2097151): +500M

Created a new partition 1 of type 'Linux' and of size 500 MiB.

Command (m for help): _
```

Das Kommando `p` zeigt die aktuelle Partitionstabelle an. Diese könnte etwa so aussehen:

```
Command (m for help): p
Disk /dev/sdb: 1 GiB, 1073741824 bytes, 2097152 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x68d97339

Device     Boot Start     End Sectors  Size Id Type
/dev/sdb1        2048 1026047 1024000  500M 83 Linux
```

Den Partitionstyp können Sie mit dem Kommando `t` ändern. Sie müssen die gewünschte Partition auswählen und können dann den Code (als Hexadezimalzahl) eingeben. Mit `L` bekommen Sie die komplette Liste angezeigt.

Eine Partition, die Sie nicht mehr haben möchten, können Sie mit dem Kommando `d` löschen. Wenn Sie fertig sind, können Sie mit `w` die Partitionstabelle auf die Platte schreiben und das Programm verlassen. Mit `q` verlassen Sie das Programm, ohne die Partitionstabelle neu zu schreiben.

Nach dem Speichern versucht `fdisk`, den Linux-Kern dazu zu bringen, dass er die neue Partitionstabelle einliest; das funktioniert bei neuen oder unbenutzten Platten problemlos, schlägt aber fehl, sobald eine Partition auf der Platte gerade in irgendeiner Form (eingebundenes Dateisystem, aktiver Swapspace, …) benutzt wird. Damit ist eine Neupartitionierung der Platte mit dem /-Dateisystem nur mit einem Neustart des Systems möglich. Einer der seltenen Momente, ein Linux-System neustarten zu müssen …

Optionen

Wie alle Linux-Kommandos kennt `fdisk` auch einige Kommandozeilenoptionen. Die wichtigsten davon lauten:

`-l` zeigt die Partitionstabelle der ausgewählten Platte und beendet danach das Programm.

`-u` (engl. *units*, Einheiten) erlaubt es, die Maßeinheit zu bestimmen, die bei der Ausgabe von Partitionstabellen benutzt wird. Standard ist »Sektoren«; wenn Sie »`-u=cylinders`« angeben, werden statt dessen Zylinder verwendet (aber dafür gibt es heute keinen vernünftigen Grund mehr.

Wenn Sie `fdisk` im MBR-Modus verwenden, dann versucht es, die gängigen Konventionen einzuhalten und dafür zu sorgen, dass die Partitionierung auch mit 4Kn-Platten und 512e-Platten vernünftig funktioniert. Sie sollten, wo möglich, den Vorschlägen des Programms folgen und nicht ohne zwingenden Grund davon abweichen.

Wenn Sie eine Platte gemäß dem GPT-Standard partitionieren wollen und noch keine Partitionstabelle nach GPT auf der Platte steht, können Sie diese mit dem Kommando `g` generieren (*Warnung:* Eine etwa schon existierende MBR-Partitionierungstabelle wird dabei überschrieben):

```
Command (m for help): g
Created a new GPT disklabel (GUID: C2A556FD-7C39-474A-B383-963E09AA7269)
```

(Die angezeigte GUID gilt für die Platte insgesamt.) Anschließend können Sie wie gehabt mit dem Kommando `n` Partitionen anlegen, auch wenn der Dialog geringfügig anders aussieht:

```
Command (m for help): n
Partition number (1-128, default 1): 1
First sector (2048-2097118, default 2048): ↩
Last sector, +sectors or +sizeK,M,G,T,P (2048-2097118, default▷
 ◁ 2097118): +32M

Created a new partition 1 of type 'Linux filesystem' and of size 32 MiB.
```

Auch die Partitionstypauswahl ist anders, weil es statt um zweistellige Hexadezimalzahlen um GUIDs geht:

```
Command (m for help): t
Selected partition 1
Partition type (type L to list all types): L
  1 EFI System                     C12A7328-F81F-11D2-BA4B-00A0C93EC93B
◁◁◁◁◁
 14 Linux swap                     0657FD6D-A4AB-43C4-84E5-0933C84B4F4F
 15 Linux filesystem               0FC63DAF-8483-4772-8E79-3D69D8477DE4
 16 Linux server data              3B8F8425-20E0-4F3B-907F-1A25A76F98E8
 17 Linux root (x86)               44479540-F297-41B2-9AF7-D131D5F0458A
 18 Linux root (x86-64)            4F68BCE3-E8CD-4DB1-96E7-FBCAF984B709
◁◁◁◁◁
Partition type (type L to list all types): _
```

Übungen

14.5 [!2] Legen Sie mit dem Kommando

```
# dd if=/dev/zero of=$HOME/test.img bs=1M count=1024
```

eine leere 1 GiB große Datei an. Verwenden Sie `fdisk`, um die Datei à la MBR zu »partitionieren«: Legen Sie zwei Linux-Partitionen von respektive 256 MiB und 512 MiB an und erzeugen Sie eine Partition für Auslagerungsspeicher mit dem Rest.

14.6 [!2] Wiederholen Sie die vorige Aufgabe, aber legen Sie eine GPT-Partitionstabelle an. Gehen Sie davon aus, dass die 512-MiB-Partition ein `/home`-Verzeichnis enthalten soll.

14.5.3 Platten formatieren mit GNU `parted`

Ein anderes verbreitetes Programm zum Partitionieren von Speichermedien ist `parted` aus dem GNU-Projekt. Vom Leistungsumfang her ist es vergleichbar mit `fdisk`, hat aber einige nützliche Eigenschaften.

Im Gegensatz zu `fdisk` ist `parted` bei den meisten Distributionen nicht standardmäßig installiert, kann aber in der Regel von den Servern der Distribution nachgeladen werden.

Ähnlich wie `fdisk` wird `parted` mit dem Namen des zu partitionierenden Mediums als Parameter aufgerufen:

```
# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) _
```

Eine neue Partition können Sie mit `mkpart` anlegen. Das geschieht entweder interaktiv …

```
(parted) mkpart
Partition name?  []? Test
File system type?  [ext2]? ext4
Start? 211MB
End? 316MB
```

… oder direkt im Kommandoaufruf:

```
(parted) mkpart primary ext4 211MB 316MB
```

Sie können die Kommandos bis auf ein eindeutiges Präfix abkürzen. Statt `mkpart` geht also auch `mkp` (`mk` würde mit dem Kommando `mklabel` kollidieren).

Der Dateisystemtyp wird lediglich dafür verwendet, einen Partitionstyp zu erraten. Sie müssen später immer noch manuell ein Dateisystem auf der Partition generieren.

Die aktuelle Partitionstabelle bekommen Sie mit `print`:

```
(parted) p
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 1074MB
Sector size (logical/physical): 512B/512B
```

```
Partition Table: gpt
Disk Flags:

Number  Start   End    Size   File system  Name     Flags
 1      1049kB  106MB  105MB
 2      106MB   211MB  105MB
 3      211MB   316MB  105MB  ext4         primary

(parted) _
```

(Hier sehen Sie jetzt auch, wo die magischen Zahlen »211MB« und »316MB« weiter oben hergekommen sind.)

`print` hat noch ein paar interessante Unterkommandos: »`print devices`« listet alle verfügbaren Blockgeräte auf, »`print free`« zeigt freien (unpartitionierten) Platz an und »`print all`« gibt die Partitionstabellen aller Blockgeräte aus.

Unerwünschte Partitionen können Sie mit `rm` löschen. Mit `name` können Sie einer Partition einen Namen geben (nur bei GPT). Das Kommando `quit` beendet das Programm.

Wichtig: Während `fdisk` die Partitionstabelle auf der Platte erst aktualisiert, wenn Sie das Programm verlassen, tut `parted` das laufend. Das heißt, Hinzufügungen oder Löschungen von Partitionen wirken sich sofort auf die Platte aus.

Wenn Sie `parted` auf eine neue (unpartitionierte) Platte loslassen, müssen Sie als erstes eine Partitionstabelle generieren. Mit

```
(parted) mklabel gpt
```

wird eine GPT-mäßige Partitionstabelle angelegt, mit

```
(parted) mklabel msdos
```

eine nach dem MBR-Standard. Es gibt keinen Standardwert; ohne Partitionstabelle verweigert `parted` das `mkpart`-Kommando.

Sollten Sie versehentlich eine Partition gelöscht haben, die Sie lieber behalten hätten, kann `parted` Ihnen helfen, sie wiederzufinden. Dazu müssen Sie nur wissen, wo ungefähr auf der Platte die Partition gelegen hat:

```
(parted) rm 3                                                    Upps.
(parted) rescue 200MB 350MB
Information: A ext4 primary partition was found at 211MB -> 316MB.
Do you want to add it to the partition table?

Yes/No/Cancel? yes
```

Damit das funktioniert, muss allerdings ein Dateisystem auf der Partition existieren, da `parted` nach Datenblöcken sucht, die so aussehen, als finge dort ein Dateisystem an.

Neben dem interaktiven Modus erlaubt `parted` es auch, Kommandos direkt auf der Linux-Kommandozeile zu übergeben, etwa so:

```
# parted /dev/sdb mkpart primary ext4 316MB 421MB
Information: You may need to update /etc/fstab.
```

Übungen

14.7 [!2] Wiederholen Sie Übung 14.5 mit `parted` statt `fdisk`, sowohl für das MBR- als auch das GPT-Schema.

14.8 [2] (Wenn Sie Kapitel 15 schon durchgearbeitet haben.) Generieren Sie auf der »Platte« aus den vorigen Aufgaben Dateisysteme auf den Linux-Partitionen. Löschen Sie die Partitionen. Können Sie sie mit dem `rescue`-Kommando von `parted` wiederherstellen?

14.5.4 `gdisk`

Das Programm `gdisk` (»GPT fdisk«) ist spezialisiert auf GPT-partitionierte Platten und kann einige nützliche Dinge tun, die mit den anderen bisher gezeigten Programmen nicht gehen. Allerdings müssen Sie es gegebenenfalls nachinstallieren.

Die elementaren Funktionen von `gdisk` entsprechen im Wesentlichen denen von `fdisk` und `parted`, und wir gehen auf sie nicht nochmal ein (lesen Sie die Dokumentation und machen Sie ein paar Experimente). Ein paar Funktionen von `gdisk` sind allerdings unabhängig erwähnenswert:

- Sie können `gdisk` verwenden, um ein MBR-partitioniertes Medium in ein GPT-partitioniertes Medium umzuwandeln. Das setzt allerdings voraus, dass am Anfang und am Ende des Mediums genug Platz für die GPT-Partitionstabellen vorhanden ist. Bei Medien, wo gemäß heutigen Gepflogenheiten die erste Partition beim Sektor 2048 anfängt, ist ersteres kein Problem, letzteres möglicherweise schon. Sie müssen also gegebenenfalls dafür sorgen, dass die letzten 33 Sektoren des Mediums zu keiner Partition gehören.

  Für die Umwandlung reicht es normalerweise, `gdisk` mit dem Namen der Gerätedatei des zu bearbeitenden Mediums als Parameter zu starten. Sie bekommen dann entweder eine Warnung, dass keine GPT-Partitionstabelle gefunden wurde und `gdisk` die MBR-Partitionstabelle ausgewertet hat (an dieser Stelle können Sie das Programm mit `w` wieder verlassen und sind fertig), oder dass ein intakter MBR, aber eine beschädigte GPT-Partitionstabelle gefunden wurde (dann weisen Sie `gdisk` an, sich nach dem MBR zu richten, und können dann das Programm mit `w` verlassen und sind fertig).

- Der umgekehrte Weg ist auch möglich. Hierzu müssen Sie in `gdisk` mit dem Kommando `r` in das Menü für *recovery/transformation commands* wechseln und können dort das Kommando `g` (*convert GPT into MBR and exit*) wählen. Anschließend können Sie das Programm mit `w` verlassen und das Speichermedium so konvertieren.

Übungen

14.9 [!2] Wiederholen Sie Übung 14.5 mit `gdisk` statt `fdisk` und generieren Sie eine GPT-Partitionstabelle.

14.10 [2] Erzeugen Sie (etwa mit `fdisk`) eine MBR-partitionierte »Platte« und rufen Sie `gdisk` auf, um sie auf GPT-Partitionierung umzustellen. Vergewissern Sie sich, dass ein korrekter »Schutz-MBR« angelegt wurde.

14.5.5 Andere Partitionierungsprogramme

Distributionen

Alternativ haben die meisten Distributionen noch mehr Möglichkeiten, die Partitionierung zu verändern. Die meisten bieten als Alternative zu `fdisk` das Programm `cfdisk`, das bildschirmorientiert und daher etwas komfortabler ist. Noch einfacher zu bedienen sind grafische Programme, etwa der YaST bei den SUSE- oder der »DiskDruid« bei den Red-Hat-Distributionen.

Erwähnenswert ist auch sfdisk, ein komplett uninteraktives Partitionierungsprogramm. sfdisk übernimmt Partitionsinformationen aus einer Eingabedatei und eignet sich daher zum unbeaufsichtigten Partitionieren, etwa im Rahmen einer vollautomatischen Installation. Außerdem können Sie mit sfdisk eine Sicherheitskopie Ihrer Partitionsinformationen machen und entweder als Tabelle ausdrucken oder als Datei auf einer Diskette oder CD ablegen. Diese Kopie lässt sich dann im Fall der Fälle mit sfdisk wieder einspielen.

sfdisk funktioniert nur für MBR-partitionierte Platten. Es gibt ein entsprechendes Programm sgdisk, das äquivalente Aufgaben für GPT-partitionierte Platten erledigt. sfdisk und sgdisk sind allerdings nicht kompatibel – die Optionsstrukturen sind völlig verschieden.

14.6 Loop-Devices und kpartx

Linux besitzt die nützliche Fähigkeit, Dateien wie Speichermedien behandeln zu können. Das heißt, wenn Sie eine Datei haben, können Sie diese partitionieren, Dateisysteme generieren und die »Partitionen« in dieser Datei allgemein so behandeln, als handele es sich um Partitionen einer »echten« Festplatte. Im wirklichen Leben ist das zum Beispiel nützlich, wenn Sie auf CD-ROMs oder DVDs zugreifen möchten, ohne ein entsprechendes Laufwerk im Rechner zu haben (schneller ist es außerdem). Für Unterrichtszwecke bedeutet es, dass Sie diverse Experimente machen können, ohne zusätzliche Festplatten besorgen und an Ihrem Rechner herumschrauben zu müssen.

CD-ROM-Image

Ein CD-ROM-Image erzeugen Sie aus einer existierenden CD-ROM einfach mit dd:

```
# dd if=/dev/cdrom of=cdrom.iso bs=1M
```

Anschließend können Sie das Image direkt zugänglich machen:

```
# mount -o loop,ro cdrom.iso /mnt
```

In diesem Beispiel erscheint der Inhalt der CD-ROM im Verzeichnis /mnt.

Sie können mit dd natürlich auch einfach eine leere Datei anlegen:

```
# dd if=/dev/zero of=disk.img bs=1M count=1024
```

Diese Datei können Sie dann mit einem der üblichen Partitionierungsprogramme »partitionieren«.

```
# fdisk disk.img
```

Bevor Sie mit dem Ergebnis etwas anfangen können, müssen Sie allerdings dafür sorgen, dass die Partitionen Gerätedateien bekommen (im Gegensatz zu »echten« Speichermedien geht das bei simulierten Speichermedien in Dateien nicht automatisch). Vorbedingung dafür ist zunächst eine Gerätedatei für die komplette Datei. Diese – ein sogenanntes *loop device* – können Sie mit dem Kommando losetup anlegen:

```
# losetup -f disk.img
# losetup -a
/dev/loop0: [2050]:93 (/tmp/disk.img)
```

`losetup` verwendet Gerätedateinamen der Form »`/dev/loop`*n*«. Mit der Option »`-f`« sucht das Programm nach dem ersten freien Namen. »`losetup -a`« gibt eine Liste der gerade aktiven *loop devices* aus.

Wenn Sie Ihr Platten-Image einem *loop device* zugeordnet haben, können Sie im nächsten Schritt Gerätedateien für dessen Partitionen erzeugen. Dazu dient das Kommando `kpartx`.

`kpartx` müssen Sie gegebenenfalls auch nachinstallieren. Bei Debian und Ubuntu heißt das Paket `kpartx`.

Das Kommando zum Einrichten der Gerätedateien für die Partitionen auf `/dev/loop0` heißt

```
# kpartx -av /dev/loop0
add map loop0p1 (254:0): 0 20480 linear /dev/loop0 2048
add map loop0p2 (254:1): 0 102400 linear /dev/loop0 22528
```

(ohne das »`-v`« verhält `kpartx` sich ruhig). Die Gerätedateien tauchen anschließend im Verzeichnis `/dev/mapper` auf:

```
# ls /dev/mapper
control  loop0p1  loop0p2
```

Nun hält Sie nichts mehr davon ab, zum Beispiel auf diesen »Partitionen« Dateisysteme zu generieren und sie in die Verzeichnisstruktur Ihres Rechners einzuhängen. Siehe hierzu Kapitel 15.

Wenn Sie die Gerätedateien für die Partitionen nicht mehr benötigen, können Sie sie mit dem Kommando

```
# kpartx -dv /dev/loop0
del devmap : loop0p2
del devmap : loop0p1
```

wieder entfernen. Ein nicht mehr gebrauchtes *loop device* geben Sie mit

```
# losetup -d /dev/loop0
```

frei.

Das Kommando

```
# losetup -D
```

gibt alle *loop devices* frei.

Übungen

14.11 [!2] Verwenden Sie die Test-»Platte« aus Übung 14.5. Ordnen Sie ihr mit `losetup` ein *loop device* zu und machen Sie ihre Partitionen mit `kpartx` zugänglich. Vergewissern Sie sich, dass die richtigen Gerätedateien in `/dev/mapper` erscheinen. Geben Sie anschließend die Partitionen und das *loop device* wieder frei.

14.7 Der Logical Volume Manager (LVM)

Eine Festplatte zu partitionieren und dann Dateisysteme darauf anzulegen ist einfach und naheliegend. Allerdings legen Sie sich damit auch fest: Die Partitionierung einer Festplatte ist später nur unter Verrenkungen zu ändern und erfordert, sofern es sich um die Platte mit dem Wurzeldateisystem handelt, in der Regel den Einsatz eines »Rettungssystems«. Außerdem gibt es keinen zwingenden Grund, warum Sie sich in Ihrer Systemarchitektur von Trivialitäten beeinflussen lassen sollten wie dass Festplatten eine beschränkte Kapazität haben und Dateisysteme nicht größer sein können als die Partitionen, auf denen sie liegen.

Eine Methode, diese Einschränkungen zu transzendieren, ist die Verwendung des *Logical Volume Managers* (LVM). LVM stellt eine Abstraktionsschicht zwischen Platten(partitionen) und Dateisystemen dar – statt Dateisysteme direkt auf Partitionen anzulegen, können Sie Partitionen (oder ganze Platten) in einen »Vorrat« von Plattenspeicherplatz einbringen und sich zum Anlegen von Dateisystemen aus diesem Vorrat bedienen. Dabei können einzelne Dateisysteme durchaus Plattenplatz verwenden, der auf mehr als einer physikalischen Platte liegt.

In der Terminologie von LVM stellen Platten oder Plattenpartitionen sogenannte *physical volumes* (PV) dar, die Sie in eine *volume group* (VG) einbringen können. Auf demselben Rechner können Sie mehr als eine VG verwalten. Der Plattenplatz innerhalb einer VG steht Ihnen zum Anlegen von *logical volumes* (LV) zur Verfügung, die Sie wiederum mit beliebigen Dateisystemen versehen oder als Auslagerungsspeicher nutzen können.

Beim Anlegen von LVs können Sie bestimmen, dass der Speicherplatz geschickt über mehrere physikalische Platten verteilt werden soll (*striping*) oder dass der Inhalt des LV gleichzeitig an mehreren Stellen in der VG abgelegt werden soll (*mirroring*). Ersteres soll die Zugriffsgeschwindigkeit erhöhen (auch wenn die Gefahr besteht, dass bei einem Ausfall *irgendeiner* beteiligten Platte Daten verlorengehen), letzteres die Gefahr eines Datenverlusts verringern (auch wenn Sie dafür mit verlängerten Zugriffszeiten bezahlen). Im wirklichen Leben verläßt man sich hier aber weniger auf die Fähigkeiten von LVM, sondern verwendet für diese Funktionalität lieber (Hardware- oder Software-)RAID.

Eine der netten Eigenschaften von LVM ist, dass Sie LVs im laufenden Betrieb vergrößern oder verkleinern können. Sollte es in einem Dateisystem eng werden, dann können Sie zunächst das unterliegende LV größer machen (jedenfalls solange Ihre VG noch ungenutzten Platz hat – ansonsten müßten Sie erst eine neue Platte einbauen und der VG zuordnen). Anschließend können Sie das Dateisystem auf dem betreffenden LV vergrößern.

Das setzt natürlich voraus, dass das betreffende Dateisystem eine nachträgliche Größenänderung erlaubt. Bei den gängigen Dateisystemen, etwa `ext3` oder `ext4`, ist das jedoch gegeben. Sie unterstützen sogar eine Vergrößerung im laufenden Betrieb. (Zum Verkleinern müssen Sie das Dateisystem aushängen.)

Wenn Sie ein Dateisystem benutzen, das eine Vergrößerung nicht zuläßt, dann bleibt Ihnen nichts anderes übrig als die Daten anderswohin zu kopieren, das Dateisystem neu anzulegen und die Daten wieder zurückzuspielen.

Sollte eine Platte in Ihrer VG anfangen, herumzuzicken, können Sie die darauf abgespeicherten LVs auf eine andere Platte verlagern (wenn Sie in Ihrer VG noch genug Platz haben oder schaffen können). Anschließend können Sie die schadhafte Platte aus der VG hinauskonfigurieren, eine neue Platte einbauen und die LVs zurückmigrieren.


Auch das geht im laufenden Betrieb, ohne dass Ihre Benutzer etwas davon merken – jedenfalls sofern Sie das nötige Kleingeld für »heiß« auswechselbare Platten investiert haben.

Snapshots

Ebenfalls nett sind *Snapshots*, mit denen Sie Sicherheitskopien machen können, ohne Ihr System dafür für Stunden stilllegen zu müssen (wie es nötig wäre, um sicherzustellen, dass sich während des Kopiervorgangs nichts ändert). Dazu können Sie den aktuellen Zustand eines LV auf einem anderen (neuen) LV »einfrieren« – was höchstens ein paar Sekunden dauert – und in aller Ruhe eine Kopie des neuen LV machen, während auf dem alten LV der normale Betrieb weitergeht.


Das LV für den »Schnappschuss« muss nur so groß sein wie der Umfang der Änderungen, die Sie während der Dauer des Kopiervorgangs auf dem Original erwarten (sinnvollerweise mit einer Sicherheitsreserve), da nur die Änderungen im neuen LV gespeichert werden. Es hält Sie also niemand davon ab, einen Snapshot Ihres 10-TB-Dateisystems zu machen, auch wenn Sie keine weiteren 10 TB Plattenplatz frei haben: Wenn Sie nur damit rechnen, dass Daten im Wert von 10 GB geändert werden, während Sie die Kopie auf Band sichern, dann sind Sie mit einem Schnappschuss-LV von 20–30 GB oder so auf jeden Fall auf der sicheren Seite.


Tatsächlich ist es inzwischen auch möglich, beschreibbare Snapshots anzulegen. Das ist zum Beispiel nützlich, wenn Sie mit »virtuellen Maschinen« arbeiten, die zwar über eine Betriebssystem-Grundinstallation verfügen, aber sich in Details unterscheiden. Beschreibbare Snapshots machen es möglich, die Grundinstallation in einem einzigen LV für alle virtuellen Maschinen zu machen und dann die für jede virtuelle Maschine spezifische Konfiguration in je einem LV mit einem beschreibbaren Snapshot abzulegen. (Überstrapazieren sollten Sie das allerdings nicht; wenn Sie am LV mit der Grundinstallation etwas ändern, bekommen die virtuellen Maschinen das nicht mit.)

Der LVM unter Linux ist eine spezielle Anwendung des *device mappers*, einer Systemkomponente, die den flexiblen Umgang mit Blockgeräten ermöglicht. Der *device mapper* erlaubt auch andere Nützlichkeiten wie verschlüsselte Platten oder die platzsparende Plattenplatzvergabe an »virtuelle Server«. Leider haben wir in dieser Schulungsunterlage keinen Platz, um den Einsatz von LVM und *device mapper* im Detail zu erklären, und verweisen Sie dafür auf die Linup-Front-Schulungsunterlage *Linux-Storage und Dateisysteme* (STOR).

Kommandos in diesem Kapitel

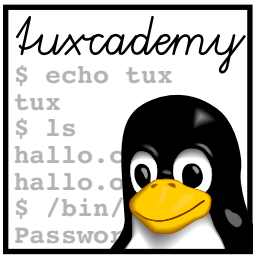
cfdisk	Plattenpartitionierungsprogramm mit textbildschirmorientierter Oberfläche	cfdisk(8)	238
fdisk	Partitionierungswerkzeug für Platten und ähnliche Medien	fdisk(8)	233
gdisk	Partitionierungswerkzeug für GPT-Platten	gdisk(8)	238
kpartx	Erzeugt Blockgeräte aus Partitionstabellen	kpartx(8)	239
losetup	Erzeugt und verwaltet Loop-Devices	losetup(8)	239
parted	Leistungsfähiges Partitionierungswerkzeug aus dem GNU-Projekt	Info: parted	236
sfdisk	Nichtinteraktives Partitionierungsprogramm	sfdisk(8)	238
sgdisk	Nichtinteraktives Partitionierungsprogramm für GPT-Platten	sgdisk(8)	239

Zusammenfassung

- Linux unterstützt alle wesentlichen Arten von Massenspeichern – magnetische Festplatten (SATA, P-ATA, SCSI, SAS, Fibre Channel, USB, …), SSDs, USB-Sticks, SD-Karten, …
- Speichermedien wie Festplatten können partitioniert werden. Partitionen erlauben die unabhängige Verwaltung von Teilen einer Platte, etwa mit verschiedenen Datei- oder Betriebssystemen.
- Linux kann mit Speichermedien umgehen, die gemäß dem MBR- oder GPT-Schema partitioniert sind.
- Die meisten Speichermedien werden von Linux wie SCSI-Geräte verwaltet. Es gibt noch eine ältere Infrastruktur für P-ATA-Platten, die aber nur selten verwendet wird.
- Zur Partitionierung von Platten stehen zahlreiche Linux-Programme wie `fdisk`, `parted`, `gdisk`, `cfdisk` oder `sfdisk` zur Verfügung. Die Distributionen bringen oft eigene Werkzeuge mit.
- *Loop devices* machen aus Dateien blockorientierte Geräte. Partitionen auf *loop devices* können Sie mit `kpartx` zugänglich machen.
- Der Logical Volume Manager (LVM) entkoppelt physikalischen Speicherplatz auf Medien von logischen Speicherstrukturen. Er erlaubt die flexible Verwaltung von Massenspeicher, etwa um Dateisysteme zu erzeugen, die größer sind als ein einzelnes physikalisches Speichermedium. Snapshots helfen beim Anlegen von Sicherheitskopien und bei der Provisionierung von Speicherplatz für virtuelle Maschinen.

Literaturverzeichnis

SCSI-2.4-HOWTO Douglas Gilbert. »The Linux 2.4 SCSI subsystem HOWTO«, Mai 2003. `http://www.tldp.org/HOWTO/SCSI-2.4-HOWTO/`



15

Dateisysteme: Aufzucht und Pflege

Inhalt

15.1 Linux-Dateisysteme 246
15.1.1 Überblick. 246
15.1.2 Die ext-Dateisysteme 249
15.1.3 ReiserFS 257
15.1.4 XFS 259
15.1.5 Btrfs 260
15.1.6 Noch mehr Dateisysteme 262
15.1.7 Auslagerungsspeicher (*swap space*) 263
15.2 Einbinden von Dateisystemen 264
15.2.1 Grundlagen 264
15.2.2 Der mount-Befehl 264
15.2.3 Labels und UUIDs 266
15.3 Das Programm dd 268

Lernziele

- Die wichtigsten Dateisysteme für Linux und ihre Eigenschaften kennen
- Dateisysteme auf Partitionen und Speichermedien generieren können
- Werkzeuge zur Dateisystemprüfung kennen
- Auslagerungsspeicher verwalten können
- Lokale Dateisysteme in die Verzeichnishierarchie einbinden können
- Plattenplatz-Kontingentierung einrichten können

Vorkenntnisse

- Sicherer Umgang mit den Kommandos zum Umgang mit Dateien und Verzeichnissen
- Kenntnisse über Massenspeicher in Linux und Partitionierung (Kapitel 14)
- Vorkenntnisse über den Aufbau von Dateisystemen sind hilfreich

15.1 Linux-Dateisysteme

15.1.1 Überblick

Nachdem Sie eine neue Partition angelegt haben, müssen Sie diese Partition »formatieren«, also die notwendigen Datenstrukturen für die Verwaltung von Dateien und Verzeichnissen auf die Partition schreiben. Wie diese Datenstrukturen im Detail aussehen, hängt vom betreffenden »Dateisystem« ab.

Der Begriff »Dateisystem« ist unter Linux unglücklicherweise mehrfach überladen. Er bedeutet unter anderem:

1. Eine Methode, Daten und Verwaltungsinformationen auf einem Medium zu arrangieren (»das `ext4`-Dateisystem«, »das `btrfs`-Dateisystem«)
2. Einen Teil der Dateihierarchie eines Linux-Systems, der sich auf einem bestimmten Medium oder einer Partition befindet (»das Wurzeldateisystem«, »das `/var`-Dateisystem«)
3. Die Gesamtheit der Dateihierarchie, über Mediengrenzen hinweg (»Named Pipes sind über das Dateisystem zu erreichen«)

Die unter Linux gängigen Dateisysteme (Bedeutung 1 oben) unterscheiden sich mitunter gravierend voneinander. Es gibt einerseits Dateisysteme, die ursprünglich für Linux entwickelt wurden, wie die »`ext`-Dateisysteme« oder Btrfs, und andererseits Dateisysteme, die eigentlich zu anderen Betriebssystemen gehören, die Linux aus Kompatibilitätsgründen aber auch (mehr oder weniger) unterstützt. Dazu gehören die Dateisysteme von DOS, Windows, OS X und einigen Unix-Varianten sowie »Netzdateisysteme« wie NFS oder SMB, die über das lokale Netz den Zugriff auf Dateiserver erlauben.

Viele »native« Dateisysteme von Linux stehen in der Tradition von Dateisystemen, die auf Unix-Systemen üblich waren, etwa dem Berkeley Fast Filesystem (FFS), und sind in ihren Strukturen diesen angelehnt. Allerdings ist die Entwicklung nicht stehengeblieben; modernere Einflüsse werden ständig integriert, um Linux auf dem Stand der Technik zu halten.

Btrfs (ausgesprochen wie (englisch) »butter fs«) von Chris Mason (Fusion-IO) wird weithin als Antwort auf das berühmte ZFS von Solaris gehandelt. (ZFS steht zwar im Quellcode zur Verfügung, kann aber aus lizenzrechtlichen Gründen nicht direkt in den Linux-Kern integriert werden.) Sein Fokus liegt auf »Fehlertoleranz, Reparatur und einfacher Administration«. Inzwischen ist es anscheinend einigermaßen benutzbar, zumindest einige Distributionen verlassen sich darauf.

Superblock

Bei den Linux-Dateisystemen ist üblich, dass am Anfang des Dateisystems ein **Superblock** steht, der Informationen über das Dateisystem als ganzes enthält – also Daten darüber, wann es zuletzt ein- oder ausgehängt wurde, ob das Aushängen »sauber« oder durch einen Systemabsturz erfolgte, und vieles mehr. Der Superblock verweist normalerweise auch auf andere Teile der Verwaltungsdatenstrukturen, etwa wo die Inodes oder die Listen freier und belegter Blöcke zu finden sind und welche Bereiche des Mediums für Daten zur Verfügung stehen.

Es ist gängig, an anderen Stellen des Dateisystems Kopien des Superblocks zu hinterlegen, falls dem Original etwas zustoßen sollte. So machen das zum Beispiel die `ext`-Dateisysteme.

Vor dem Superblock steht auf der Platte meist noch ein »Bootsektor«, in dem Sie theoretisch einen Bootlader unterbringen können (Kapitel 16). Das macht es möglich, Linux zum Beispiel auf einem Rechner zu installieren, auf dem schon Windows installiert ist, und den Bootmanager von Windows zu benutzen, um das System zu starten.

Zum Anlegen eines Dateisystems (Bedeutung 2 oben) existiert unter Linux der Befehl `mkfs`. `mkfs` ist ein vom konkret gewünschten Dateisystemtyp (Bedeutung 1) unabhängiges Programm, das die eigentliche Routine für das jeweilige Dateisystem (Bedeutung 1), `mkfs.`⟨*Dateisystemtyp*⟩, aufruft. Den Dateisystemtyp können Sie mit der Kommandozeilenoption `-t` wählen, mit »`mkfs -t ext2`« würde zum Beispiel das Programm `mkfs.ext2` gestartet (ein anderer Name für `mke2fs`). `mkfs`

Wenn der Rechner unkontrolliert ausgeschaltet wurde oder wenn es zu einem Systemabsturz gekommen ist, müssen Sie damit rechnen, dass sich das Dateisystem in einem inkonsistenten Zustand befindet (auch wenn das im wirklichen Leben selbst bei Abstürzen glücklicherweise sehr selten vorkommt). Fehler im Dateisystem können auftreten, weil die Schreiboperationen durch den Festplattencache im Arbeitsspeicher gepuffert werden und diese Daten beim Abschalten des Rechners verloren gehen, bevor sie auf die Platte geschrieben werden konnten. Andere Fehler entstehen, wenn das System mitten in einer ungepufferten Schreiboperation seinen Dienst aufgibt. inkonsistenter Zustand

Probleme, die vorkommen können, umfassen neben reinen Datenverlusten auch Fehler in der Struktur des Dateisystems, die durch geeignete Programme erkannt und repariert werden können, zum Beispiel Strukturfehler

- fehlerhafte Einträge in Verzeichnissen
- fehlerhafte Einträge in Inodes
- Dateien, die in keinem Verzeichnis eingetragen sind
- Datenblöcke, die zu mehreren verschiedenen Dateien gehören

Viele, aber nicht alle solche Probleme lassen sich automatisch und ohne Datenverlust beheben; das Dateisystem kann aber in der Regel wieder in einen konsistenten Zustand gebracht werden.

Wenn ein Dateisystem nicht korrekt »abgemeldet« wurde, erkennt das System das beim nächsten Neustart an dessen Status. Bei einem ordentlichen Systemabschluss werden die Dateisysteme ausgehängt und dabei das *valid flag* im Superblock gesetzt. Beim Start kann diese Information aus dem Superblock genutzt werden, um die möglicherweise fehlerhaften Dateisysteme automatisch während der Systeminitialisierung testen und, falls erforderlich, reparieren zu lassen – bevor das System versucht, ein Dateisystem einzuhängen, dessen *valid flag* nicht gesetzt ist, schiebt es erst einmal eine Dateisystemprüfung ein. *valid flag*

Bei allen gängigen Linux-Distributionen enthalten die Shellskripte zur Systeminitialisierung, die von `init` beim Start ausgeführt werden, die notwendigen Kommandos zur Durchführung einer Dateisystemprüfung.

Wenn Sie die Konsistenz eines Dateisystems prüfen wollen, müssen Sie damit nicht bis zum nächsten Booten warten. Sie können die Programme zum Dateisystemcheck zu jedem beliebigen Zeitpunkt aufrufen. Sollte ein Dateisystem allerdings Fehler enthalten, darf eine Reparatur nur auf einem Dateisystem durchgeführt werden, das nicht auch gerade eingehängt ist. Diese Einschränkung ist notwendig, damit sich Kernel und Reparaturprogramm bei der Veränderung des Dateisystems nicht in die Quere kommen. Auch aus diesem Grund bietet sich der automatische Test vor dem Zusammenbau des Dateisystems beim Booten an. Konsistenzprüfung

Zur tatsächlichen Konsistenzprüfung dient das Kommando `fsck`. Wie `mkfs` bedient dieses Kommando sich je nach dem Typ des zu prüfenden Dateisystems eines spezifischen Unterkommandos namens `fsck.`⟨*Dateisystemtyp*⟩ – für `ext2` zum Beispiel `fsck.ext2`. `fsck` findet das passende Unterkommando, indem es das zu prüfende Dateisystem untersucht. Mit dem Kommando

```
# fsck /dev/sdb1
```

zum Beispiel können Sie das Dateisystem auf `/dev/sdb1` prüfen.

Das einfache Kommando

```
# fsck
```

prüft nacheinander alle Dateisysteme, die in der Datei `/etc/fstab` aufgelistet sind und deren Eintrag in der sechsten (letzten) Spalte einen von Null verschiedenen Wert haben. (Tauchen dort mehrere Werte auf, werden die Dateisysteme in aufsteigender Reihenfolge dieses Werts geprüft.) Über `/etc/fstab` steht mehr im Abschnitt 15.2.2.

`fsck` hat eine `-t`-Option, die auf den ersten Blick an `mkfs` erinnert, aber eine andere Bedeutung hat: Ein Kommando wie

```
# fsck -t ext3
```

prüft alle Dateisysteme, die in der Datei `/etc/fstab` aufgelistet und dort als vom Typ `ext3` gekennzeichnet sind.

Optionen

Die wichtigsten Optionen von `fsck` sind:

`-A` (All) veranlaßt `fsck`, alle in der Datei `/etc/fstab` aufgeführten Linux-Dateisysteme zu prüfen.

Dabei wird die Prüfreihenfolge in der sechsten Spalte von `/etc/fstab` eingehalten. Gibt es mehrere Dateisysteme mit demselben Wert in dieser Spalte, werden sie parallel geprüft, solange sie auf verschiedenen physikalischen Festplatten liegen.

`-R` Bei `-A` wird das Wurzeldateisystem nicht mit geprüft (sinnvoll, wenn es schon zum Schreiben eingehängt ist).

`-V` Hiermit werden Informationen über den Programmverlauf ausgegeben.

`-N` Zeigt an, was `fsck` täte, ohne es allerdings zu tun.

`-s` Verhindert die parallele Prüfung mehrerer Partitionen. Das Kommando »`fsck`« ohne Parameter ist äquivalent zu »`fsck -A -s`«.

Außer seinen eigenen Optionen können Sie `fsck` nach dem Namen des oder der zu prüfenden Dateisysteme (und gegebenenfalls durch »`--`« abgetrennt) weitere Optionen mitgeben, die es an das spezifische Dateisystemprüfprogramm durchreicht. Die Optionen `-a`, `-f`, `-p` und `-v` werden von den meisten dieser Programme unterstützt. Details dazu finden sich bei den entsprechenden Kommandobeschreibungen. Das Kommando

```
# fsck /dev/sdb1 /dev/sdb2 -pv
```

zum Beispiel würde die Dateisysteme auf den Partitionen `/dev/sdb1` und `/dev/sdb2` automatisch prüfen, allfällige Fehler ohne Rückfrage korrigieren und dabei geschwätzig über den Fortschritt berichten.

Das `fsck`-Programm gibt am Programmende verschiedene Informationen zum Zustand des Dateisystems an die Shell weiter:

- **`0`** Kein Fehler im Dateisystem gefunden.
- **`1`** Fehler im Dateisystem gefunden und korrigiert.
- **`2`** Schwerwiegende Fehler im Dateisystem gefunden und korrigiert. Das System sollte neu gestartet werden.
- **`4`** Fehler im Dateisystem gefunden, aber nicht korrigiert.

8 Es ist ein Fehler bei der Programmausführung aufgetreten.

16 Falsche Benutzung (z. B. Fehler in der Kommandozeile).

128 Fehler in einer Funktion der Shared-Libraries

Es ist beispielsweise denkbar, in einem Init-Skript diese Rückgabewerte zu analysieren und den weiteren Verlauf des Systemstarts entsprechend zu bestimmen. Wenn (mit der Option `-A`) mehrere Dateisysteme geprüft werden, ist der Wert des Rückgabewerts von `fsck` die logische Oder-Verknüpfung aller Rückgabewerte der einzelnen Dateisystemprüfprogramme.

15.1.2 Die ext-Dateisysteme

Geschichte und Eigenschaften Das ursprüngliche *extended file system* für Linux wurde im April 1992 von Rémy Card implementiert. Es war das erste speziell für Linux entworfene Dateisystem (wobei es sich nachdrücklich auf den Grundideen allgemeiner Unix-Dateisysteme abstützte) und räumte mit diversen Einschränkungen des vorher verwendeten Minix-Dateisystems auf.

Das Minix-Dateisystem hatte diverse lästige Einschränkungen, etwa eine auf 64 MiB beschränkte Dateisystemgröße und höchstens 14 Zeichen lange Dateinamen. (Zur Ehrenrettung von Minix ist zu sagen, dass, als Minix neu war, der IBM PC XT als angesagter Computer galt und 64 MiB im PC-Bereich eine nahezu unvorstellbare Plattenkapazität war. Anfang der 1990er knackte es da schon etwas im Gebälk.) Mit `ext` konnte man immerhin 2 GiB große Dateisysteme haben – für die damaligen Verhältnisse brauchbar, heute natürlich eher lachhaft.

Die Ankunft des `ext`-Dateisystems markiert eine weitere wichtige Neuerung im Linux-Kern, nämlich die Einführung des *virtual file system switch* oder VFS. Der VFS abstrahiert Dateisystemoperationen wie das Öffnen und Schließen von Dateien oder das Lesen und Schreiben von Daten und ermöglicht damit die Koexistenz verschiedener Dateisystemimplementierungen in Linux.

Heute wird das ursprüngliche `ext`-Dateisystem nicht mehr benutzt. Wenn wir ab jetzt von »den `ext`-Dateisystemen« reden, dann meinen wir alle ab einschließlich `ext2`.

Die im Januar 1993 ebenfalls von Rémy Card angefangene Nachfolgeversion, `ext2` (das *second extended file system*), stellte eine umfangreiche Überarbeitung des ursprünglichen »erweiterten Dateisystems« dar. Bei der Entwicklung von `ext2` wurden diverse Ideen etwa aus dem *Berkeley Fast Filesystem* von BSD aufgegriffen. `ext2` wird weiterhin gewartet und ist nach wie vor für bestimmte Anwendungen absolut empfehlenswert.

Gegenüber `ext` schiebt `ext2` einige Größenbegrenzungen noch weiter nach draußen – bei der für Intel-basierte Linux-Systeme üblichen Blockgröße von 4 KiB können Dateisysteme 16 TiB und einzelne Dateien 2 TiB groß sein. Eine sehr wichtige Neuerung in `ext2` war die Unterstützung separater Zeitstempel für den letzten Zugriff auf eine Datei, die letzte Inhalts- und die letzte Inode-Veränderung, so dass in diesem Bereich Kompatibilität zum »traditionellen« Unix erreicht war.

`ext2` war von Anfang an auf Zuwachs und Weiterentwicklung ausgelegt: Die meisten Datenstrukturen enthielten zusätzlichen Platz, der teilweise für wichtige Erweiterungen ausgenutzt wurde. Dazu gehören zum Beispiel ACLs und »erweiterte Attribute«.

Seit Ende der 1990er Jahre arbeitete Stephen Tweedie an einem Nachfolger für `ext2`, der ab Ende 2001 unter dem Namen `ext3` in den Linux-Kernel integriert wurde (das war Linux 2.4.15). Die wesentlichen Unterschiede zwischen `ext2` und `ext3` sind:

- ext3 unterstützt Journaling.
- ext3 erlaubt es, Dateisysteme im laufenden Betrieb zu vergrößern.
- ext3 unterstützt effizientere interne Datenstrukturen für Verzeichnisse mit vielen Einträgen.

Dabei ist es weitgehend kompatibel mit ext2. Es ist in der Regel möglich, ext3-Dateisysteme als ext2-Dateisysteme anzusprechen (wobei natürlich die neuen Eigenschaften nicht genutzt werden können) und umgekehrt.

»Journaling« löst ein Problem, das bei zunehmend größeren Dateisystemen sehr hinderlich sein kann, nämlich dass nach einem unvorhergesehenen Systemabsturz eine komplette Konsistenzprüfung des Dateisystems nötig ist. Schreiboperationen auf die Platte führt der Linux-Kern in der Regel nicht sofort aus, sondern speichert die Daten erst einmal im RAM zwischen und schreibt sie erst später auf die Platte, wenn sich das anbietet (etwa weil der Schreib-/Lesekopf des Plattenlaufwerks in der richtigen Gegend ist). Außerdem erfordern es viele Schreiboperationen, an mehreren Stellen auf der Platte Daten zu schreiben, etwa in einen oder mehrere Datenblöcke, in die Inode-Tabelle und die Liste der verfügbaren Blöcke auf der Platte. Wenn im richtigen (bzw. falschen) Moment der Strom wegbleibt, kann es passieren, dass so eine Operation nur halb ausgeführt ist – das Dateisystem ist nicht »konsistent« in dem Sinne, dass zum Beispiel ein Datenblock einer Datei zwar in der Inode zugeordnet ist, aber in der Liste der verfügbaren Blöcke nicht als belegt gekennzeichnet wurde. Das kann natürlich später zu echten Problemen führen.

Bei einem Journaling-Dateisystem wie ext3 betrachtet man jeden Schreibzugriff auf der Platte als »Transaktion«, die entweder komplett ausgeführt werden muss oder gar nicht. Vor der Ausführung der Transaktion ist das Dateisystem *per definitionem* konsistent und hinterher auch. Jede Transaktion wird zuerst in einen speziellen Bereich des Dateisystems, das *Journal*, geschrieben. Wenn sie komplett dort angekommen ist, wird sie als »vollständig« markiert und ist damit aktenkundig. Anschließend kann der Linux-Kern die tatsächlichen Schreiboperationen durchführen. – Stürzt das System ab, muss für ein Journaling-Dateisystem beim Neustart keine komplette Konsistenzprüfung durchgeführt werden, die bei heutigen Dateisystemgrößen Stunden oder gar Tage dauern könnte. Statt dessen wird nur das Journal angeschaut und die als vollständig gekennzeichneten Transaktionen werden ins eigentliche Dateisystem übertragen. Transaktionen, die im Journal nicht als vollständig gekennzeichnet sind, werden verworfen.

Die meisten Journaling-Dateisysteme protokollieren im Journal nur Veränderungen an den »Metadaten« des Dateisystems, also Verzeichnisse, Inodes usw. Die tatsächlichen Nutzdaten werden aus Effizienzgründen normalerweise nicht durch das Journal geschickt. Das bedeutet, dass Sie nach Absturz und Wiederherstellung zwar ein konsistentes Dateisystem haben, ohne Stunden oder Tage mit der kompletten Konsistenzprüfung zu verbringen. Allerdings können durchaus Ihre Nutzdaten durcheinandergekommen sein – zum Beispiel könnte eine Datei noch veraltete Datenblöcke enthalten, weil die aktuellen vor dem Absturz nicht mehr geschrieben werden konnten. Man kann dieses Problem entschärfen, indem man erst die Datenblöcke auf die Platte schreibt und dann die Metadaten ins Journal, aber auch das ist nicht ganz ohne Risiko. ext3 läßt Ihnen die Wahl zwischen drei Betriebsarten – alles ins Journal schreiben (Option beim Einhängen: data=journal), Datenblöcke direkt schreiben und dann Metadatenzugriffe ins Journal (data=ordered), oder keine Einschränkungen (data=writeback). Standard ist data=ordered.

Das doppelte Schreiben von Metadaten oder gar Nutzdaten – einmal ins Journal und dann noch einmal ins eigentliche Dateisystem – geht gegenüber Dateisystemen wie ext2, die das Problem ignorieren, natürlich mit einem gewissen Effizienzverlust einher. Ein Ansatz, das zu beheben, sind *logstrukturierte Dateisysteme*, bei denen das Journal das eigentliche Dateisystem darstellt. In der Linux-Praxis hat sich das aber nicht durchgesetzt. Ein alternativer Ansatz sind sogenannte *copy-on-write filesystems* wie Btrfs, das wir weiter unten noch kurz ansprechen werden.

Die Verwendung eines Dateisystems mit Journal wie ext3 enthebt Sie prinzipiell nicht der Pflicht, hin und wieder komplette Konsistenzprüfungen durchzuführen. Immerhin könnte es sein, dass sich durch Hardwarefehler der Platte, Probleme mit der Verkabelung oder die berühmt-berüchtigten kosmischen Strahlen (nicht lachen, das kann wirklich ein Problem sein!) Fehler in den Datenstrukturen des Dateisystems einschleichen, die Ihnen sonst verborgen bleiben würden, bis sie Unheil anrichten. Aus diesem Grund nötigen die ext-Dateisysteme Sie normalerweise beim Starten dazu, indem sie in gewissen Abständen von sich aus einen Dateisystemcheck anstoßen (typischerweise genau dann, wenn Sie es am wenigsten gebrauchen können). Später in diesem Kapitel werden Sie sehen, wie Sie hier steuernd eingreifen können.

Bei Serversystemen, die Sie selten neu starten und die Sie auch nicht unbedingt einfach so für ein paar Stunden oder Tage offline stellen können, um prophylaktisch das Dateisystem zu prüfen, haben Sie möglicherweise ein größeres Problem. Siehe auch dazu später mehr.

ext4-Dateisystem

Die Krone der ext-Dateisystemevolution repräsentiert derzeit das seit 2006 unter der Federführung von Theodore Ts'o entwickelte ext4-Dateisystem, das seit Ende 2008 (Kernel 2.6.28) als stabil gilt. Ähnlich wie bei ext3 und ext2 wurde Wert auf Rückwärtskompatibilität gelegt: ext2- und ext3-Dateisysteme können als ext4-Dateisysteme angesprochen werden und profitieren dann von einigen internen Verbesserungen in ext4. Der ext4-Code bringt allerdings auch Veränderungen mit, die dazu führen, dass Dateisysteme, die diese ausnutzen, nicht mehr als ext2 oder ext3 zugänglich sind. Hier sind die wichtigsten Verbesserungen in ext4 gegenüber ext3:

- Statt die Datenblöcke von Dateien als Listen von Blocknummern zu verwalten, benutzt ext4 sogenannte *extents*, also Gruppen von physikalisch zusammenhängenden Blöcken auf der Platte. Dies führt zu einer erheblichen Verwaltungsvereinfachung und zu einem Effizienzgewinn, macht Dateisysteme, die Extents benutzen, aber inkompatibel zu ext3. Es vermeidet auch Fragmentierung, also das wilde Verstreuen zusammengehörender Daten über das ganze Dateisystem.
- Beim Schreiben von Inhalten werden erst so spät wie möglich Datenblöcke auf der Platte belegt. Auch das hilft gegen Fragmentierung.
- Anwendungsprogramme können beim Dateisystem voranmelden, wie groß eine Datei werden soll. Das kann wiederum ausgenutzt werden, um zusammenhängenden Plattenplatz zu vergeben und Fragmentierung zu verringern.
- Ext4 verwendet Prüfsummen zur Absicherung des Journals. Dies erhöht die Zuverlässigkeit und vermeidet einige haarige Probleme beim Wiedereinspielen des Journals nach einem Systemabsturz.
- Verschiedene Optimierungen von internen Datenstrukturen erlauben eine erhebliche Beschleunigung von Konsistenzprüfungen.
- Zeitstempel haben jetzt eine Auflösung von Nanosekunden (statt Sekunden) und reichen bis zum Jahr 2242 (statt 2038).

- Einige Größenbeschränkungen wurden verschoben – Verzeichnisse können 64.000 oder mehr Unterverzeichnisse enthalten (vorher 32.000), Dateien können bis zu 16 TiB groß werden und Dateisysteme bis zu 1 EiB.

Trotz dieser nützlichen Weiterentwicklungen ist ext4 laut Ted Ts'o nicht als Innovation zu verstehen, sondern als Lückenfüller, bis noch viel bessere Dateisysteme wie Btrfs (Abschnitt 15.1.5) zur Verfügung stehen.

Alle ext-Dateisysteme verfügen über leistungsfähige Werkzeuge zur Konsistenzprüfung und Reparatur von Dateisystemstrukturen. Für den Einsatz in der Praxis ist das extrem wichtig.

ext-Dateisysteme anlegen Um ein ext-Dateisystem anzulegen, können Sie im einfachsten Fall mkfs mit einer passenden -t-Option benutzen:

```
# mkfs -t ext2 /dev/sdb1                    ext2-Dateisystem
# mkfs -t ext3 /dev/sdb1                    ext3-Dateisystem
# mkfs -t ext4 /dev/sdb1                    ext4-Dateisystem
```

Hinter der -t-Option und ihrem Parameter können noch Parameter kommen, die an das Programm weitergegeben werden, das die eigentliche Arbeit macht – im Falle der ext-Dateisysteme das Programm mke2fs. (Trotz des e2 im Namen kann es auch ext3- und ext4-Dateisysteme erzeugen.)

Ebenfalls funktionieren würden die Aufrufe

```
# mkfs.ext2 /dev/sdb1                       ext2-Dateisystem
# mkfs.ext3 /dev/sdb1                       ext3-Dateisystem
# mkfs.ext4 /dev/sdb1                       ext4-Dateisystem
```

– das sind genau die Kommandos, die mkfs aufrufen würde. Alle drei Kommandos sind eigentlich symbolische Links, die auf mke2fs verweisen; mke2fs schaut nach, unter welchem Namen es aufgerufen wurde, und verhält sich entsprechend.

mke2fs

Sie können das Kommando mke2fs auch direkt aufrufen:

```
# mke2fs /dev/sdb1
```

(Ohne Optionen bekommen Sie ein ext2-Dateisystem.)

Die folgenden Optionen für mke2fs sind nützlich (und möglicherweise für die Prüfung wichtig):

-b ⟨*Größe*⟩ bestimmt die Größe der Blöcke. Typische Werte sind 1024, 2048 oder 4096. Standardwert ist auf Partitionen interessanter Größe 4096.

-c überprüft die Partition auf defekte Blöcke und markiert diese als unbenutzbar.

Heute gängige Festplatten können defekte Blöcke selber erkennen und durch Blöcke aus einer »Geheimreserve« ersetzen, ohne dass das Betriebssystem etwas davon merkt (jedenfalls solange Sie nicht gezielt nachfragen). Solange das gut geht, bringt Ihnen »mke2fs -c« keinen Vorteil. Das Kommando findet erst dann defekte Blöcke, wenn die Geheimreserve erschöpft ist, und in diesem Moment sollten Sie die Festplatte sowieso besser austauschen. (Eine ganz neue Festplatte wäre an dem Punkt wahrscheinlich ein Garantiefall. Alte Möhren sind reif für den Schrott.)

-i ⟨*Anzahl*⟩ legt die »Inodedichte« fest; für jeweils ⟨*Anzahl*⟩ Bytes Plattenplatz wird eine Inode erzeugt. Der Wert muss ein Vielfaches der Blockgröße (Option `-b`) sein; es hat keinen großen Zweck, die ⟨*Anzahl*⟩ kleiner zu wählen als die Blockgröße. Der Mindestwert ist 1024, die Voreinstellung ist der Wert der Blockgröße.

-m ⟨*Anteil*⟩ setzt den Anteil der Datenblöcke, die für `root` reserviert sind (Voreinstellung 5%)

-S veranlasst `mke2fs`, nur die Superblocks und Gruppendeskriptoren neu zu schreiben, die Inodes aber unangetastet zu lassen

-j erzeugt zusätzlich ein Journal und damit ein `ext3`- oder `ext4`-Dateisystem.

Echte `ext4`-Dateisysteme legen Sie am besten mit einem der vorgekochten Aufrufe wie »`mkfs -t ext4`« an, weil `mke2fs` dann weiß, was es machen soll. Wenn Sie es wirklich unbedingt »zu Fuß« erledigen wollen, dann benutzen Sie etwas wie

```
# mke2fs -j -O extents,uninit_bg,dir_index /dev/sdb1
```

Die `ext`-Dateisysteme benötigen (noch) für jede Datei, egal wie klein sie ist, mindestens einen kompletten Datenblock. Wenn Sie also ein `ext`-Dateisystem anlegen, auf dem Sie viele kleine Dateien zu speichern beabsichtigen (Stichwort Mail- oder USENET-Server), dann sollten Sie unter Umständen eine kleinere Blockgröße wählen, um internen Verschnitt zu vermeiden. (Auf der anderen Seite ist Plattenplatz heutzutage wirklich ziemlich billig.)

interner Verschnitt

Die Inode-Dichte (Option `-i`) legt fest, wie viele Dateien Sie in dem Dateisystem haben können – da jede Datei ein Inode benötigt, kann es nicht mehr Dateien geben als Inodes. Die Voreinstellung, für jeden Datenblock auf der Platte ein Inode anzulegen, ist sehr konservativ, aber die Gefahr, aus Inodemangel keine Dateien mehr anlegen zu können, wiegt in den Augen der Entwickler offenbar schwerer als die Platzverschwendung durch unbenutzte Inodes.

Diverse Dinge im Dateisystem benötigen Inodes, aber keine Datenblöcke – zum Beispiel Gerätedateien, FIFOs oder kurze symbolische Links. Selbst wenn Sie so viele Inodes anlegen wie Datenblöcke, können Ihnen die Inodes also immer noch eher ausgehen als die Datenblöcke.

Mit der `mke2fs`-Option `-F` können Sie auch Dateisystemobjekte »formatieren«, die keine Blockgerätedateien sind. Zum Beispiel können Sie CD-ROMs mit einem `ext2`-Dateisystem erzeugen, indem Sie die Kommandosequenz

```
# dd if=/dev/zero of=cdrom.img bs=1M count=650
# mke2fs -F cdrom.img
# mount -o loop cdrom.img /mnt
#                                          … Sachen nach /mnt kopieren…
# umount /mnt
# cdrecord -data cdrom.img
```

benutzen. (`/dev/zero` ist ein »Gerät«, das beliebig viele Nullbytes liefert.) Die resultierenden CD-ROMs enthalten »echte« `ext2`-Dateisysteme mit allen Rechten, Attributen, Zugriffskontrolllisten (ACLs) usw. und können per

```
# mount -t ext2 -o ro /dev/scd0 /media/cdrom
```

(oder ähnlichem) eingehängt werden; statt `/dev/scd0` müssen ggf. Sie den Gerätenamen Ihres optischen Laufwerks einsetzen. (Sie sollten sich verkneifen, hier ein `ext3`-Dateisystem zu verwenden, da das Journal reine Platzverschwendung darstellt. Ein `ext4`-Dateisystem können Sie wiederum ohne Journal anlegen.)

e2fsck **Reparatur von ext-Dateisystemen** e2fsck ist das Prüfprogramm für die ext-Dateisysteme. Damit es von fsck aufgerufen werden kann, gibt es normalerweise auch symbolische Links wie fsck.ext2.


Ähnlich wie bei mke2fs funktioniert auch e2fsck für ext3- und ext4-Dateisysteme.


Direkt aufrufen können Sie das Programm natürlich auch, was Ihnen möglicherweise etwas Tipparbeit spart, wenn Sie Optionen übergeben wollen. Allerdings können Sie dann nur den Namen einer einzigen Partition (naja, genaugenommen eines einzigen Blockgeräts) übergeben.

Optionen Die wichtigsten Optionen von e2fsck sind:

-b ⟨*Nummer*⟩ liest den Superblock aus dem mit der ⟨*Nummer*⟩ bezeichneten Partitionsblock (anstelle des ersten Superblocks).

-B ⟨*Grösse*⟩ gibt die Größe einer Blockgruppe zwischen zwei Kopien des Superblocks an; bei den ext-Dateisystemen werden Sicherungskopien des Superblocks normalerweise alle 8192 Blöcke angelegt, auf größeren Platten alle 32768 Blöcke. (Abfragen können Sie das mit dem weiter unten erklärten Kommando tune2fs; schauen Sie in der Ausgabe von »tune2fs -l« nach »*Blocks per group*«.)

-f (engl. *force*) erzwingt die Überprüfung des Dateisystems unbedingt, auch wenn im Superblock eingetragen ist, dass das Dateisystem in Ordnung ist.

-l ⟨*Datei*⟩ liest die Liste kaputter Blöcke (*bad blocks*) aus der angegebenen Datei, diese Blöcke werden als »benutzt« markiert.

-c (engl. *check*) durchsucht das Dateisystem nach kaputten Blöcken.


Die Optionen -l und -c sollten Sie, wie gesagt, nicht überbewerten; eine Festplatte, bei der e2fsck kaputte Blöcke finden kann, gehört eigentlich aussortiert.

-p (engl. *preen*) veranlasst die automatische Reparatur aller gefunden Fehler ohne interaktive Aktion.

-v (engl. *verbose*) gibt während des Ablaufs Informationen über den Status des Programms und über das Dateisystem aus.

Die Gerätedatei gibt das Blockgerät an, dessen Dateisystem geprüft werden soll. Wenn dieses Gerät kein ext-Dateisystem enthält, bricht das Kommando au-
Schritte tomatisch ab. Die einzelnen Schritte, die beim Aufruf von e2fsck durchgeführt werden, sind:

1. Die angegebenen Befehlsargumente werden geprüft.
2. Es wird kontrolliert, ob das gewählte Dateisystem eingehängt ist.
3. Das Dateisystem wird geöffnet.
4. Es wird geprüft, ob der Superblock lesbar ist.
5. Es wird geprüft, ob die Datenblöcke lesbar oder fehlerhaft sind.
6. Die Informationen aus dem Superblock über Inodes, Blöcke und Größen werden mit dem aktuellen Zustand des Systems verglichen.
7. Es wird überprüft, ob die Verzeichniseinträge mit den Inodes übereinstimmen.
8. Es wird geprüft, ob jeder als belegt gekennzeichnte Datenblock existiert und genau einmal von einem Inode referenziert wird.

9. Die Anzahl der Links in den Verzeichnissen wird mit den Link-Zählern der Inodes verglichen (muss übereinstimmen).
10. Die Gesamtzahl der Blöcke muss gleich sein der Anzahl der freien Blöcke plus der Anzahl der belegten Blöcke.

`e2fsck` gibt genau wie `fsck` einen Rückgabewert zurück, dessen numerische Werte dieselbe Bedeutung haben. Rückgabewert

Alle möglichen Dateisystemfehler aufzuzählen, die von `e2fsck` behandelt werden können, ist an dieser Stelle nicht möglich. Hier sind nur einige wichtige Beispiele dargestellt:

- Ansonsten gültig aussehende Dateien, deren Inode in keinem Verzeichnis eingetragen ist, bekommen ein Link im Verzeichnis `lost+found` des Dateisystems mit der Inode-Nummer als Name. Als Administrator können Sie diese Dateien anschauen und versuchen, sie an ihren eigentlichen Platz im Dateisystem zurückzuverschieben. Zu einem solchen Fehler kann es kommen, wenn z. B. das System zusammenbricht, nachdem eine Datei erstellt wurde, aber bevor der dazugehörige Verzeichniseintrag geschrieben werden konnte.
- Der Referenzzähler eines Inodes ist größer als die Anzahl der Namenseinträge in Verzeichnissen, die auf dieses Inode verweisen. `e2fsck` passt den Referenzzähler im Inode an.
- `e2fsck` findet freie Blöcke, die als belegt eingetragen sind (das passiert z. B., wenn das System zusammenbricht, nachdem eine Datei gelöscht wurde, aber bevor der Eintrag im Superblock und den Bitmaps erfolgen konnte).
- Die Gesamtzahl der Blöcke ist inkorrekt (belegte und freie Blöcke zusammen sind ungleich der Gesamtzahl der Blöcke).

Nicht alle Fehler lassen sich einfach reparieren. Was tun, wenn der Superblock nicht lesbar ist? Dann lässt sich das Dateisystem nicht mehr einhängen, und in der Regel scheitert auch `e2fsck`. Hier können Sie eine Kopie des Superblocks verwenden, von denen jede Blockgruppe auf der Partition eine enthält. Mit der Option `-b` lässt `e2fsck` sich zwingen, einen bestimmten Block als Superblock anzusehen. Der entsprechende Befehl lautet dann z. B.: Komplizierte Fehler

```
# e2fsck -f -b 8193 /dev/sda2
```

Wenn sich das Dateisystem mit `fsck` nicht automatisch reparieren lässt, dann gibt es noch die Möglichkeit, direkt in das Dateisystem einzugreifen. Dazu ist allerdings eine sehr detaillierte Kenntnis der inneren Struktur notwendig, die über den Lehrinhalt dieses Kurses hinausgeht. – Zu diesem Zweck existieren zwei nützliche Werkzeuge. Das erste ist das Programm `dumpe2fs`. Mit diesem Kommando werden die internen Datenstrukturen eines `ext`-Dateisystems sichtbar gemacht. Die Interpretation der Ausgabe erfordert die o. g. detaillierten Kenntnisse. Eine Reparatur eines `ext`-Dateisystems lässt sich mit den Dateisystem-Debugger `debugfs` durchführen.

Von Programmen wie `debugfs` sollten Sie tunlichst die Finger lassen, wenn Sie nicht *genau* wissen, was Sie tun. Zwar gibt Ihnen `debugfs` die Möglichkeit, die Datenstrukturen des Dateisystems auf niedriger Ebene sehr detailliert zu beeinflussen, aber Sie können damit leicht noch mehr kaputt machen, als ohnehin schon kaputt ist. Nachdem wir Sie also angemessen ermahnt haben, können wir Ihnen sagen, dass Sie mit

```
# debugfs /dev/sda1
```

das ext-Dateisystem auf /dev/sda1 zur Inspektion öffnen können (debugfs erlaubt Ihnen das Schreiben auf das Dateisystem vernünftigerweise nur, wenn Sie es mit der Option -w aufgerufen haben). debugfs zeigt eine Eingabeaufforderung an; mit »help« erhalten Sie eine Liste der möglichen Kommandos. Diese steht auch in der Anleitung, die Sie unter debugfs(8) finden können.

ext-Dateisystemparameter abfragen und ändern Wenn Sie eine Partition angelegt und bereits mit einem ext-Dateisystem versehen haben, können Sie nachträglich die Formatparameter ändern. Dazu gibt es das Kommando tune2fs. Dieses Kommando ist mit äußerster Vorsicht zu verwenden und sollte auf keinen Fall auf ein zum Schreiben eingehängtes Dateisystem angewendet werden:

Formatparameter ändern

```
tune2fs [⟨Optionen⟩] ⟨Gerät⟩
```

Die folgenden Optionen sind wichtig:

-c ⟨*Anzahl*⟩ setzt die maximale Anzahl der Einhängevorgänge zwischen zwei routinemäßigen Dateisystemprüfungen. Der von mke2fs vergebene Standardwert ist eine Zufallszahl irgendwo in der Gegend von 30 (damit nicht alle Dateisysteme auf einen Sitz prophylaktisch geprüft werden). Der Wert 0 bedeutet »unendlich viele«.

-C ⟨*Anzahl*⟩ setzt die aktuelle Zahl von Einhängevorgängen. Damit können Sie fsck betrügen oder (indem Sie es auf einen Wert setzen, der größer ist als der aktuelle mit -c eingestellte Wert) eine Dateisystemprüfung beim nächsten Systemstart erzwingen.

-e ⟨*Verhalten*⟩ legt das Verhalten beim Auftreten von Fehlern fest. Es gibt folgende Möglichkeiten:

continue Normal weitermachen

remount-ro Schreiben auf das Dateisystem verbieten

panic Kernel-Panik erzwingen

In jedem Fall wird beim nächsten Systemstart eine Konsistenzprüfung gemacht.

-i ⟨*Intervall*⟩⟨*Einheit*⟩ setzt die maximale Zeit zwischen zwei routinemäßigen Dateisystemprüfungen. ⟨*Intervall*⟩ ist eine ganze Zahl; die ⟨*Einheit*⟩ ist d für Tage, w für Wochen und m für Monate. Der Wert 0 steht für »unendlich lange«.

-l zeigt die Informationen im Superblock an.

-m ⟨*Prozent*⟩ setzt den Anteil der Datenblöcke, die für root oder den mit -u festgelegten Benutzer reserviert sind (Voreinstellung 5%)

-L ⟨*Name*⟩ setzt einen Partitionsnamen (bis zu 16 Buchstaben). Kommandos wie mount und fsck erlauben es, Partitionen über ihren Namen statt den Namen der Gerätedatei zu identifizieren.

Um mit tune2fs ein existierendes ext3-Dateisystem zu einem ext4-Dateisystem zu machen, müssen Sie die Kommandos

```
# tune2fs -O extents,uninit_bg,dir_index /dev/sdb1
# e2fsck -fDp /dev/sdb1
```

ausführen (vorausgesetzt, das betreffende Dateisystem steht auf /dev/sdb1). Achten Sie darauf, in der Datei /etc/fstab sicherzustellen, dass das Dateisystem anschließend auch als ext4 eingehängt wird (siehe Abschnitt 15.2).


Beachten Sie allerdings, dass alle existierenden Dateien dann noch ext3-Verwaltungsstrukturen verwenden – die Neuerungen wie Extents werden nur für Dateien benutzt, die Sie danach anlegen. Das Defragmentierungsprogramm e4defrag soll alte Dateien umwandeln, ist aber noch nicht ganz fertig.


Wenn Sie die Möglichkeit dazu haben, sollten Sie ein Dateisystem nicht »am Platz« umwandeln, sondern lieber den Inhalt sichern, das Dateisystem als ext4 neu anlegen und den Inhalt anschließend zurückspielen. Die Leistung von ext4 ist auf ursprünglich als ext4 angelegten Dateisystemen nämlich ungleich höher als auf umgewandelten ext3-Dateisystemen – ein Faktor 2 ist da mitunter durchaus drin.


Sollten Sie noch ext2-Dateisysteme herumliegen haben, aus denen Sie ext3-Dateisysteme machen wollen: Das geht einfach, indem Sie ein Journal anlegen. Auch dabei hilft tune2fs:

```
# tune2fs -j /dev/sdb1
```

Natürlich müssen Sie auch hier gegebenenfalls die Datei /etc/fstab anpassen.

Übungen


15.1 [!2] Generieren Sie auf einem geeigneten Medium (Plattenpartition, USB-Stick, mit dd angelegte Datei) ein ext4-Dateisystem.


15.2 [2] Ändern Sie die maximale Einhängezahl des in Übung 15.1 angelegten Dateisystems auf 30. Außerdem sollen 30% des Speicherplatzes für den Benutzer test reserviert sein.

15.1.3 ReiserFS

Allgemeines ReiserFS ist ein für den allgemeinen Einsatz gedachtes Linux-Dateisystem. Es wurde von einem Team unter Leitung von Hans Reiser entwickelt und debütierte in Linux 2.4.1 (das war 2001). Damit war es das erste für Linux verfügbare Dateisystem mit Journal und verfügte auch über einige andere Neuerungen, die das damals populärste Linux-Dateisystem, ext2, nicht bieten konnte:

- ReiserFS-Dateisysteme konnten mit einem speziellen Werkzeug vergrößert und verkleinert werden. Vergrößern konnte man Dateisysteme sogar im laufenden Betrieb.
- Kleine Dateien und die Endstücke größerer Dateien konnten zusammengepackt werden, um »internen Verschnitt« zu vermeiden, der bei Dateisystemen wie ext2 dadurch entsteht, dass Plattenplatz in Einheiten der Blockgröße (meist 4 KiB) vergeben werden. Selbst eine nur 1 Byte lange Datei benötigt bei ext2 & Co. einen Datenblock von 4 KiB, und das könnte man als Verschwendung ansehen (eine 4097 Byte lange Datei braucht zwei Datenblöcke, und das ist fast genauso schlimm). Bei ReiserFS können mehrere solche Dateien sich einen Datenblock teilen.

  
  Grundsätzlich wäre es natürlich möglich und wurde auch diskutiert, dieses *tail packing* auch für die ext-Dateisysteme zur Verfügung zu stellen. Plattenplatz ist aber inzwischen so billig, dass der Leidensdruck nicht mehr so groß ist und man den Zuwachs an Komplexität scheut, der damit verbunden wäre.

- Inodes werden nicht komplett bei der Generierung des Dateisystems angelegt, sondern nach Bedarf. Dies vermeidet eine pathologische Fehlersituation, die bei den ext-Dateisystemen auftreten kann, wo zwar noch Datenplatz im Dateisystem zur Verfügung steht, aber alle Inodes belegt sind, so dass keine weiteren Dateien mehr angelegt werden können.

Bei den ext-Dateisystemen wird das Problem gemildert, indem typischerweise für jeden Datenblock auf der Platte eine Inode reserviert wird (Inode-Dichte entspricht der Blockgröße). Damit ist es schwierig, diesen Fehler zu produzieren.

- ReiserFS verwaltet Verzeichniseinträge nicht in Listen wie ext2, sondern in Bäumen. Damit ist es effizienter für Verzeichnisse mit sehr vielen Einträgen.

  Ext3 und vor allem ext4 können das inzwischen auch.

  Tatsächlich verwendet ReiserFS dieselbe B+-Baumstruktur nicht nur für Verzeichniseinträge, sondern auch für Inodes, Dateimetadaten und Blocklisten für Dateien, was an einigen Stellen zu Leistungszuwachs führt, an anderen aber eine Verlangsamung bewirkt.

ReiserFS war lange Zeit das voreingestellte Dateisystem für die SUSE-Distributionen, die das Projekt auch finanziell unterstützten. Seit 2006 hat Novell/SUSE sich allerdings von ReiserFS abgewendet und ist auf ext3 umgeschwenkt; ganz neue Versionen des SLES verwenden standardmäßig Btrfs für das Wurzeldateisystem.

Im wirklichen Leben sollten Sie um das Reiser-Dateisystem (und seinen designierten Nachfolger, Reiser4) einen weiten Bogen machen, solange Sie keine älteren Systeme warten müssen, die es benutzen. Das hat weniger damit zu tun, dass Hans Reiser als Mörder seiner Ehefrau verurteilt wurde (das spricht natürlich nicht für ihn als Mensch, aber sowas kommt nicht nur unter Linux-Kernel-Entwicklern vor), sondern eher damit, dass das Reiser-Dateisystem technisch zwar seine guten Seiten hat, aber auf ziemlich tönernen Füßen steht. Beispielsweise verletzen gewisse Verzeichnisoperationen in ReiserFS die Unix-artigen Dateisystemen ansonsten zugrundeliegenden Annahmen, so dass zum Beispiel Mailserver, die Postfächer auf einem ReiserFS lagern, weniger resistent gegenüber Systemabstürzen sind als solche, die andere Dateisysteme verwenden. Ein weiteres gravierendes Problem, das wir weiter unten noch genauer besprechen werden, ist die Existenz technischer Macken im Programm für die Dateisystemreparatur. Außerdem – und das ist wahrscheinlich das gravierendste Problem – kümmert sich niemand mehr richtig darum.

mkreiserfs dumpreiserfs resize_reiserfs

Reiserfs anlegen mkreiserfs dient dem Anlegen eines Reiser-Dateisystems. Die mögliche Angabe einer logischen Blockgröße wird derzeit ignoriert, es werden immer 4-KiB-Blöcke angelegt. Mit dumpreiserfs können Sie Informationen über Reiser-Dateisysteme auslesen. resize_reiserfs erlaubt es, die Größe von momentan nicht verwendeten ReiserFS-Partitionen zu ändern. Im Betrieb gelingt dies durch ein Kommando in der Art »`mount -o remount,resize=`⟨*Blockanzahl*⟩ ⟨*Mountpunkt*⟩«.

Konsistenzprüfung für ReiserFS Auch für das Reiser-Dateisystem gibt es ein Programm zur Prüfung und Reparatur, namentlich reiserfsck. reiserfsck führt einen Konsistenztest durch und versucht eigenständig, gefundene Schäden zu reparieren, so ähnlich e2fsck. Dieses Programm ist nur dann nötig, wenn das Dateisystem wirklich beschädigt sein sollte. Wurde das ReiserFS lediglich nicht ordnungsgemäß aus dem Dateibaum abgehängt, übernimmt der Kernel die Wiederherstellung anhand des Journals automatisch.

reiserfsck hat einige kapitale Probleme. Eines davon ist, dass es beim in manchen Fehlersituationen nötigen Neuaufbau der Baumstruktur völlig durcheinander kommt, wenn Datendateien (!) Blöcke enthalten, deren Inhalt als Superblock eines anderen Reiser-Dateisystems missverstanden werden könnte – wie das zum Beispiel vorkommt, wenn Sie ein Abbild (engl. *image*) eines Reiser-Dateisystems in einer Datei haben, etwa weil Sie eine

Virtualisierungsumgebung wie VirtualBox oder VMware mit Reiser-formatierten »virtuellen« Festplatten benutzen. Damit ist das Reiser-Dateisystem für ernsthaftes Arbeiten eigentlich disqualifiziert. Wir haben Sie gewarnt.

Übungen

15.3 [!1] Wie lautet das Kommando, um die erste logische Partition der zweiten Platte im Reiser-Dateisystem zu formatieren?

15.1.4 XFS

Das Dateisystem XFS wurde von SGI (ehedem Silicon Graphics, Inc.) für Linux zur Verfügung gestellt; es handelt sich dabei um das Dateisystem der SGI-Unix-Variante IRIX, das in der Lage ist, effizient mit sehr großen Dateien umzugehen. Alle wesentlichen Linux-Distributionen bieten XFS-Unterstützung an, auch wenn wenige es als Standard verwenden; gegebenenfalls müssen Sie die XFS-Werkzeuge zusätzlich installieren.

In manchen Kreisen ist »XFS« die Abkürzung für den »X11 Font Server«. Das Kürzel kann zum Beispiel in Paketnamen von Distributionen auftauchen. Lassen Sie sich nicht verwirren.

Ein XFS-Dateisystem können Sie auf einer leeren Partition (oder in einer Datei) mit dem Kommando

```
# mkfs -t xfs /dev/sda2
```

(setzen Sie den gewünschten Gerätenamen ein) generieren. Die eigentliche Arbeit macht (natürlich) ein Programm namens `mkfs.xfs`. Sie können es über verschiedene Optionen beeinflussen; schlagen Sie in der Dokumentation (`xfs`(5) und `mkfs.xfs`(8)) nach.

Wenn es Ihnen auf Leistung ankommt, können Sie zum Beispiel das Journal auf ein anderes (physikalisches) Speichermedium verlegen, indem Sie eine Option wie »`-l logdev=/dev/sdb1,size=10000b`« angeben. (Dabei sollte das eigentliche Dateisystem natürlich nicht auf `/dev/sdb` liegen, und die Partition für das Journal sollte nicht anderweitig benutzt werden.)

Die XFS-Hilfsprogramme enthalten ein `fsck.xfs` (das Sie über »`fsck -t xfs`« aufrufen können), aber dieses Programm tut nicht wirklich etwas – es ist nur da, damit das System etwas aufrufen kann, wenn »alle« Dateisysteme geprüft werden sollen (was einfacher ist, als eine spezielle Ausnahme für XFS in `fsck` einzubauen). Tatsächlich werden XFS-Dateisysteme automatisch beim Einhängen geprüft, wenn sie nicht sauber ausgehängt wurden. Sollten Sie außer der Reihe die Konsistenz eines XFS prüfen oder eines reparieren müssen, steht Ihnen das Programm `xfs_repair`(8) zur Verfügung – »`xfs_repair -n`« prüft, ob Reparaturen nötig sind; ohne die Option werden die Reparaturen vorgenommen.

In Extremfällen kann es vorkommen, dass `xfs_repair` ein Dateisystem nicht reparieren kann. In dieser Situation können Sie mit `xfs_metadump` einen Abzug der Dateisystem-Metadaten machen und an die Entwickler schicken:

```
# xfs_metadump /dev/sdb1 sdb1.dump
```

(Das Dateisystem darf dazu nicht eingehängt sein.) Der Abzug ist eine Binärdatei, die keine Dateiinhalte enthält und in der alle Dateinamen unkenntlich gemacht sind. Es besteht also keine Gefahr, dass vertrauliche Daten weitergegeben werden.


Einen mit `xfs_metadump` angelegten Abzug können Sie mit `xfs_mdrestore` wieder in ein Dateisystem eintragen (ein »echtes« Speichermedium oder ein Image in einer Datei). Dateiinhalte sind natürlich keine dabei, da sie schon im Abzug nicht enthalten sind. Wenn Sie kein XFS-Entwickler sind, dann ist dieses Kommando für Sie wahrscheinlich nicht übermäßig interessant.

Das Kommando `xfs_info` gibt Informationen über ein (eingehängtes) XFS-Dateisystem aus:

```
# xfs_info /dev/sdb1
meta-data=/dev/sdb1              isize=256    agcount=4, agsize=16384 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=65536, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =Intern                 bsize=4096   blocks=853, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =keine                  extsz=4096   blocks=0, rtextents=0
```

Sie können hier zum Beispiel sehen, dass das Dateisystem aus 65536 Blöcken à 4 KiB besteht (`bsize` und `blocks` im `data`-Abschnitt), während das Journal 853 4-KiB-Blöcke im selben Dateisystem belegt (`Intern`, `bsize` und `blocks` im `log`-Abschnitt).


Dieselben Daten werden übrigens auch von `mkfs.xfs` ausgegeben, nachdem es ein XFS-Dateisystem angelegt hat.

Sie sollten es sich verkneifen, XFS-Dateisysteme mit `dd` zu kopieren (oder zumindest sehr vorsichtig vorgehen). Das liegt daran, dass jedes XFS-Dateisystem eine eindeutige UUID enthält und Programme wie `xfsdump` (das Sicherheitskopien anlegt) durcheinanderkommen können, wenn sie zwei unabhängige Dateisysteme mit derselben UUID antreffen. Zum Kopieren von XFS-Dateisystemen verwenden Sie besser `xfsdump` und `xfsrestore` oder aber `xfs_copy`.

15.1.5 Btrfs

Btrfs gilt als das angesagte Linux-Dateisystem für die Zukunft. Es kombiniert die traditionell mit einem unixartigen Dateisystem assoziierten Eigenschaften mit einigen innovativen Ideen, die teils an das ZFS von Solaris angelehnt sind. Dazu gehören nebst Eigenschaften, die sonst vom Logical Volume Manager (LVM; Abschnitt 14.7) übernommen werden – wie dem Anlegen von Dateisystemen, die mehrere physikalische Speichermedien überspannen –, oder die die RAID-Unterstützung des Linux-Kerns leistet – etwa das redundante Speichern derselben Daten auf mehreren physikalischen Speichermedien – auch transparente Komprimierung von Daten, Konsistenzprüfung von Datenblöcken mit Prüfsummen und ähnliches mehr. Das »Killerfeature« sind wahrscheinlich »Schnappschüsse« (engl. *snapshots*), mit denen verschiedene Versionszustände von Dateien oder ganzen Dateihierarchien gleichzeitig zur Verfügung gestellt werden können.


Btrfs ist ein paar Jahre jünger als ZFS, und sein Design enthält darum einige Nützlichkeiten, die noch nicht erfunden waren, als ZFS neu war. ZFS gilt heutzutage als der »Stand der Kunst« bei Dateisystemen, aber es ist zu erwarten, dass Btrfs ZFS irgendwann in der nicht allzufernen Zukunft überrunden wird.


Btrfs beruht prinzipiell auf der Idee des *copy on write*. Das heißt, wenn Sie einen Schnappschuss eines Btrfs-Dateisystems anlegen, wird erst mal überhaupt nichts kopiert, sondern nur vermerkt, dass eine Kopie existiert. Auf die Daten kann aus dem ursprünglichen Dateisystem und dem Schnappschuss heraus zugegriffen werden, und solange nur gelesen wird, können

die Dateisysteme sich den kompletten Datenbestand teilen. Erst wenn entweder im ursprünglichen Dateisystem oder im Schnappschuss Schreibvorgänge passieren, werden nur die betroffenen Datenblöcke kopiert. Die Datenbestände selbst werden in effizienten Datenstrukturen, sogenannten B-Bäumen, abgelegt.

Btrfs-Dateisysteme können Sie wie üblich mit `mkfs` anlegen:

```
# mkfs -t btrfs /dev/sdb1
```

Sie können auch mehrere Speichermedien angeben, die dann alle zum neuen Dateisystem gehören. Dabei legt Btrfs Metadaten wie Verzeichnisinformationen redundant auf mehreren Medien ab; Daten werden standardmäßig über verschiedene Platten verteilt (*striping*), um den Zugriff zu beschleunigen[1]. Sie können sich aber andere Speicherformen wünschen:

```
# mkfs -t btrfs -L MeinBtrfs -d raid1 /dev/sdb1 /dev/sdc1
```

In diesem Beispiel wird ein Btrfs-Dateisystem generiert, das die Partitionen `/dev/sdb1` und `/dev/sdc1` umfasst und mit »MeinBtrfs« gekennzeichnet wird. Daten werden redundant auf beiden Platten gespeichert (»`-d raid1`«).

Innerhalb von Btrfs-Dateisystemen können Sie »Subvolumes« anlegen, die einer Art von Partition auf Dateisystemebene entsprechen. Subvolumes sind die Einheiten, von denen Sie später Schnappschüsse anlegen können. Wenn Ihr System Btrfs für das Wurzeldateisystem verwendet, würde sich zum Beispiel das Kommando

```
# btrfs subvolume create /home
```

anbieten, damit Sie Ihre eigenen Daten in einem separaten Subvolume zusammenhalten können. Subvolumes nehmen nicht viel Platz weg, so dass Sie durchaus eher mehr davon anlegen sollten als weniger – insbesondere eins für jedes Verzeichnis, von dem Sie später unabhängige Snapshots anlegen wollen, da es nicht möglich ist, Verzeichnisse nachträglich zu Subvolumes zu erklären.

Einen Schnappschuss von einem Subvolume können Sie mit

```
# btrfs subvolume snapshot /mnt/sub /mnt/sub-snap
```

anlegen. Der Schnappschuss (hier `/mnt/sub-snap`) ist zunächst nicht von dem Original-Subvolume (hier `/mnt/sub`) zu unterscheiden; beide enthalten dieselben Dateien und sind beschreibbar. Zunächst wird auch kein zusätzlicher Speicherplatz verbraucht – erst wenn Sie im Original oder dem Schnappschuss Dateien ändern oder neue schreiben, wird das kopiert, was nötig ist.

Btrfs macht Konsistenzprüfungen im laufenden Betrieb und versucht Probleme zu beheben, sobald sie auftreten. Mit dem Kommando »`btrfs scrub start`« können Sie einen »Hausputz« starten, der die Prüfsummen aller Daten und Metadaten auf einem Btrfs-Dateisystem nachrechnet und fehlerhafte Blöcke gegebenenfalls anhand einer anderen Kopie (falls vorhanden) repariert. Das kann natürlich lange dauern; mit »`btrfs scrub status`« können Sie abfragen, wie der Hausputz vorangeht, mit »`btrfs scrub cancel`« können Sie ihn unterbrechen und mit »`btrfs scrub resume`« wieder aufnehmen.

Es gibt ein `fsck.btrfs`-Programm, das aber nichts macht außer einer Meldung auszugeben, dass es nichts macht. Das Programm ist nötig, damit während des

[1] Mit anderen Worten, für Metadaten wird RAID-1 verwendet und für Daten RAID-0.

normalen Systemstarts, wo alle Dateisysteme auf Konsistenz geprüft werden, irgendetwas ausgeführt werden kann. Um Btrfs-Dateisysteme tatsächlich zu prüfen oder zu reparieren, gibt es das Kommando »`btrfs check`«. Normalerweise macht es nur eine Konsistenzprüfung, und wenn es mit der Option »`--repair`« aufgerufen wird, versucht es gefundene Probleme tatsächlich zu reparieren.

Btrfs ist sehr vielseitig und komplex und wir können hier nur einen kleinen Einblick geben. Konsultieren Sie die Dokumentation (beginnend bei `btrfs`(8)).

Übungen

15.4 [!1] Generieren Sie mit »`mkfs -t btrfs`« ein Btrfs-Dateisystem auf einer freien Partition.

15.5 [2] Legen Sie in Ihrem Btrfs-Dateisystem ein Subvolume `sub0` an. Erzeugen Sie in `sub0` einige Dateien. Machen Sie dann einen Schnappschuss unter dem Name `snap0`. Überzeugen Sie sich, dass `sub0` und `snap0` denselben Inhalt haben. Löschen oder ändern Sie einige Dateien in `sub0` und `snap0` und vergewissern Sie sich, dass die beiden Subvolumes voneinander unabhängig sind.

15.1.6 Noch mehr Dateisysteme

`tmpfs` `tmpfs` ist eine flexible Implementierung eines »RAM-Disk-Dateisystems«, bei dem Dateien nicht auf einer Platte, sondern im virtuellen Speicher des Rechners abgelegt werden. Sie sind dadurch schneller zugänglich, aber selten gebrauchte Daten können trotzdem im Auslagerungsspeicher zwischengelagert werden. Die Größe eines `tmpfs` ist variabel bis zu einer Obergrenze. Für `tmpfs` gibt es kein spezielles Generierungsprogramm, sondern Sie können es einfach anlegen, indem Sie es einbinden: Das Kommando

```
# mount -t tmpfs -o size=1G,mode=0700 tmpfs /scratch
```

zum Beispiel erzeugt ein `tmpfs` von maximal 1 GiB Größe unter dem Namen `/scratch`, auf das nur der Eigentümer des Verzeichnisses `/scratch` Zugriff hat. (Auf das Einbinden von Dateisystemen kommen wir in Abschnitt 15.2 zurück.)

Populär auf älteren Windows-Rechnern, USB-Sticks, Digitalkameras, MP3-Playern und anderen »Speichermedien« ohne große Ansprüche an Effizienz und
VFAT Flexibilität ist das ehrwürdige Dateisystem VFAT von Microsoft. Linux kann entsprechend formatierte Medien natürlich einhängen, lesen und schreiben und solche Dateisysteme selbstverständlich auch anlegen, zum Beispiel mit

```
# mkfs -t vfat /dev/mmcblk0p1
```

(setzen Sie auch hier den gewünschten Gerätenamen ein). An diesem Punkt wird es Sie nicht mehr überraschen, dass `mkfs.vfat` nur ein anderer Name für das Programm `mkdosfs` ist, das alle möglichen Sorten von MS-DOS- und Windows-Dateisystemen anlegen kann – bis hin zum Dateisystem des Atari ST seligen Angedenkens. (Da es Linux-Varianten gibt, die auf Atari-Rechnern laufen, ist das vielleicht nicht komplett aus der Luft gegriffen.)

`mkdosfs` unterstützt diverse Optionen, mit denen Sie die Art des anzulegenden Dateisystems bestimmen können. Die meisten davon haben heute keine praktische Bedeutung mehr, und `mkdosfs` tut in den meisten Fällen auch automatisch das Richtige. Wir wollen jetzt nicht in eine Taxonomie von FAT-Dateisystem-Varianten abschweifen und belassen es bei einem Hinweis darauf, dass der wesentliche Unterschied zwischen FAT und VFAT darin besteht, dass letztere Dateisysteme Dateinamen erlauben, die nicht dem älteren strikten 8 + 3-Schema folgen. Die *File Allocation Table*, die Datenstruktur, die sich merkt, welche Datenblöcke zu welcher Datei gehören und die dem

Dateisystem seinen Namen gegeben hat, existiert auch in verschiedenen Geschmacksrichtungen, von denen `mkdosfs` sich diejenige aussucht, die dem Medium am ehesten entgegen kommt – Disketten werden mit 12-Bit-FAT versehen und Festplatten(partitionen) bzw. (heutzutage) USB-Sticks beachtlicher Kapazität mit 32-Bit-FATs; in letzterem Fall heißt das resultierende Dateisystem dann »VFAT32«.

NTFS

NTFS, das Dateisystem von Windows NT und seinen Nachfolgern bis hin zu Windows Vista, ist ein etwas leidiges Thema. Naheliegenderweise besteht beträchtliches Interesse daran, Linux den Umgang mit NTFS-Partitionen zu ermöglichen – außer natürlich seitens Microsoft, wo man bisher nicht geruht hat, der Allgemeinheit zu erklären, wie NTFS tatsächlich funktioniert. (Es ist bekannt, dass NTFS auf dem »Berkeley Fast Filesystem« von BSD beruht, das ganz gut verstanden ist, aber Microsoft hat es in der Zwischenzeit so vermetzgert, dass es kaum wiederzuerkennen ist.) In der Linux-Szene gab es mehrere Anläufe, NTFS-Unterstützung zu programmieren, indem man versucht hat, NTFS unter Windows zu verstehen, aber ein ganz durchschlagender Erfolg ist noch nicht gelungen. Im Moment gibt es einen Kernel-Treiber mit guter Lese-, aber fragwürdiger Schreibunterstützung und einen Treiber, der im Userspace läuft und dem Vernehmen nach beim Lesen *und* Schreiben gut funktioniert. Schließlich gibt es noch die »ntfsprogs«, ein Paket mit Hilfsprogrammen zur Verwaltung von NTFS-Dateisystemen und zum rudimentären Zugriff auf dort gespeicherte Daten. Nähere Informationen können Sie unter `http://www.linux-ntfs.org/` finden.

15.1.7 Auslagerungsspeicher (*swap space*)

Swap-Partition

Unabhängig von Partitionen für Dateisysteme sollten Sie stets noch eine **Swap-Partition** erstellen. Dort kann Linux bei Bedarf RAM-Inhalte auslagern; der effektive Arbeitsspeicher, der Ihnen zur Verfügung steht, ist dann also größer als das RAM in Ihrem Rechner.

Bevor Sie eine Swap-Partition in Betrieb nehmen, müssen Sie sie mit dem Kommando `mkswap` »formatieren«:

```
# mkswap /dev/sda4
```

Dadurch werden einige Verwaltungsinformationen in die Partition geschrieben.

Beim Systemstart ist es nötig, eine Swap-Partition zu »aktivieren«. Dies entspricht dem Einhängen von Partitionen mit einem Dateisystem und erfolgt mit dem Kommando `swapon`:

```
# swapon /dev/sda4
```

Anschließend sollte die Partition in der Datei `/proc/swaps` auftauchen:

```
# cat /proc/swaps
Filename                Type       Size    Used   Priority
/dev/sda4               partition  2144636 380    -1
```

Nach Gebrauch wird die Swap-Partition mit `swapoff` wieder deaktiviert:

```
# swapoff /dev/sda4
```

Um das Aktivieren und Deaktivieren von Swap-Partitionen kümmert sich das System selbst, wenn Sie sie in `/etc/fstab` eingetragen haben. Siehe dazu Abschnitt 15.2.2.

Sie können bis zu 32 Swap-Partitionen (bis Kernel 2.4.10: 8) parallel betreiben; die maximale Größe hängt von Ihrer Rechnerarchitektur ab und ist nirgendwo wirklich dokumentiert, aber »monsterriesengroß« ist eine gute Approximation. Früher war sie bei den meisten Linux-Plattformen knapp 2 GiB.


Wenn Sie mehrere Platten zur Verfügung haben, sollten Sie den Auslagerungsspeicher über diese verteilen, was der Geschwindigkeit merklich aufhelfen sollte.


Linux kann Swap-Partitionen mit Prioritäten versehen. Dies lohnt sich, wenn Sie verschieden schnelle Platten im System haben, auf denen Auslagerungsspeicher konfiguriert ist, weil Linux dann die schnellen Platten bevorzugen kann. Lesen Sie in `swapon`(8) nach.


Außer Partitionen können Sie auch Dateien als Swap-Bereiche verwenden. Seit Linux 2.6 ist das sogar nicht mal mehr langsamer! Damit ist es möglich, für seltene riesengroße Arbeitslasten temporär Platz zu schaffen. Eine Swap-Datei müssen Sie zuerst als Datei voller Nullen anlegen, etwa mit

```
# dd if=/dev/zero of=swapfile bs=1M count=256
```

bevor Sie sie mit dem Kommando `mkswap` präparieren und mit `swapon` aktivieren. (Verkneifen Sie sich Tricks mit `dd` oder `cp`; eine Swap-Datei darf keine »Löcher« enthalten.)


Informationen über die aktuellen Swap-Bereiche finden Sie in der Datei `/proc/swaps`.

15.2 Einbinden von Dateisystemen

15.2.1 Grundlagen

Um auf die auf einem Medium (Festplatte, USB-Stick, Diskette, …) gespeicherten Daten zuzugreifen, könnten Sie prinzipiell direkt die Gerätedateien ansprechen. Das wird auch zum Teil gemacht, zum Beispiel bei Bandlaufwerken. Die gängigen Befehle für Dateioperationen (`cp`, `mv` und so weiter) können aber nur über den Verzeichnisbaum auf Dateien zugreifen. Um solche Befehle verwenden zu können, müssen Sie die Datenträger über ihre Gerätedateien in den Verzeichnisbaum einhängen. Dieses Einhängen wird mit dem Befehl `mount` vorgenommen.

Einhängen

Die Stelle im Verzeichnisbaum, an der ein Datenträger eingehängt wird, heißt *mount point*. Dabei kann es sich um jedes beliebige Verzeichnis handeln. Das Verzeichnis muss dabei nicht leer sein, allerdings können Sie auf den ursprünglichen Verzeichnisinhalt nicht mehr zugreifen, wenn Sie einen Datenträger »drübergehängt« haben.


Der Inhalt erscheint wieder, wenn Sie den Datenträger wieder mit `umount` aushängen. Trotzdem sollten Sie nichts über `/etc` und ähnliche Systemverzeichnisse drübermounten …

15.2.2 Der `mount`-Befehl

Das Kommando `mount` dient prinzipiell zum Einhängen von Dateisystemen in den Verzeichnisbaum. Es kann auch verwendet werden, um die aktuell eingehängten Dateisysteme anzuzeigen, einfach indem Sie es ohne Parameter aufrufen:

```
$ mount
/dev/sda2 on / type ext3 (rw,relatime,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
◁◁◁◁◁
```

Zum Einhängen eines Mediums, etwa einer Festplattenpartition, müssen Sie dessen Gerätedatei und den gewünschten *mount point* angeben:

```
proc       /proc         proc  defaults                  0 0
/dev/sda2  /             ext3  defaults,errors=remount-ro 0 1
/dev/sda1  none          swap  sw                        0 0
/dev/sda3  /home         ext3  defaults,relatime         0 1
/dev/sr0   /media/cdrom0 udf,iso9660 ro,user,exec,noauto  0 0
/dev/sdb1  /media/usb    auto  user,noauto               0 0
/dev/fd0   /media/floppy auto  user,noauto,sync          0 0
```

Bild 15.1: Die Datei `/etc/fstab` (Beispiel)

```
# mount -t ext2 /dev/sda5 /home
```

Dabei ist es nicht unbedingt notwendig, mit der Option `-t` den Dateisystemtyp anzugeben, da der meistens automatisch vom Kernel erkannt wird. Wenn die Partition in `/etc/fstab` eingetragen ist, reicht es, entweder den *mount point oder* die Gerätedatei anzugeben:

```
# mount /dev/sda5                    Eine Möglichkeit …
# mount /home                        … und eine andere
```

Die Datei `/etc/fstab` beschreibt allgemein gesagt den Aufbau der gesamten Dateisystemstruktur aus verschiedenen Dateisystemen, die auf unterschiedlichen Partitionen, Platten usw. liegen können. Neben Gerätenamen und den dazugehörigen *mount points* können Sie diverse Optionen angeben, die beim Einbinden des Dateisystems verwendet werden. Welche Optionen jeweils möglich sind, ist dateisystemabhängig; viele Optionen sind aber `mount`(8) zu entnehmen. `/etc/fstab`

Eine typische `/etc/fstab`-Datei könnte aussehen wie in Bild 15.1. Die Wurzelpartition steht in der Regel an erster Stelle. Außer den »normalen« Dateisystemen werden hier auch Pseudodateisysteme wie `devpts` oder `proc` eingebunden sowie die Swap-Bereiche aktiviert.

Das dritte Feld beschreibt den Typ des jeweiligen Dateisystems. Einträge wie `ext3` und `iso9660` sprechen für sich selbst (wenn `mount` mit der Typangabe selber nichts anfangen kann, versucht es die Arbeit an ein Programm namens `/sbin/mount.`⟨*Typ*⟩ zu delegieren), `swap` steht für Auslagerungsplatz (engl. *swapspace*), der nicht eingehängt werden muss, und `auto` heißt, dass `mount` versuchen soll, den Dateisystemtyp zu erraten. Typ

Zum Raten bedient `mount` sich des Inhalts der Datei `/etc/filesystems`, oder, falls diese nicht existiert, der Datei `/proc/filesystems`. (`/proc/filesystems` wird auch gelesen, wenn `/etc/filesystems` mit einer Zeile endet, die nur einen Stern (»*«) enthält.) In jedem Fall werden nur diejenigen Zeilen beachtet, die nicht mit `nodev` gekennzeichnet sind. Zu Ihrer Erbauung hier ein Ausschnitt aus einer typischen `/proc/filesystems`-Datei:

```
nodev   sysfs
nodev   rootfs
◁◁◁◁◁
nodev   mqueue
        ext3
nodev   nfs
        vfat
        xfs
        btrfs
◁◁◁◁◁
```

`/proc/filesystems` wird vom Kernel dynamisch auf der Basis derjenigen Dateisysteme erzeugt, für die tatsächlich Treiber im System vorliegen. `/etc/filesystems` ist nützlich, wenn Sie (etwa für die Benutzung mit `auto`) eine Ratereihenfolge vorgeben wollen, die von der bei `/proc/filesystems` resultierenden – die Sie nicht beeinflussen können – abweicht.

Bevor `mount` sich auf `/etc/filesystems` beruft, versucht es sein Glück mit den Bibliotheken `libblkid` oder `libvolume_id`, die beide unter anderem in der Lage sind, zu bestimmen, welche Sorte Dateisystem auf einem Medium vorliegt. Sie können diese Bibliotheken mit den dazugehörigen Kommandozeilenprogrammen `blkid` bzw. `vol_id` ausprobieren:

```
# blkid /dev/sdb1
/dev/sdb1: LABEL="TESTBTRFS" UUID="d38d6bd1-66c3-49c6-b272-eabdae▷
 ◁ 877368" UUID_SUB="3c093524-2a83-4af0-8290-c22f2ab44ef3" ▷
 ◁ TYPE="btrfs" PARTLABEL="Linux filesystem" ▷
 ◁ PARTUUID="ade1d2db-7412-4bc1-8eab-e42fdee9882b"
```

Optionen

Das vierte Feld enthält die Optionen. Hier sind folgende Einträge zu sehen:

`defaults` Ist nicht wirklich eine Option, sondern nur ein Platzhalter für die Standardoptionen (siehe `mount`(8)).

`noauto` Gegenteil von `auto`, bewirkt, dass das Dateisystem beim Systemstart *nicht* automatisch eingebunden wird.

`user` Grundsätzlich kann nur der Benutzer `root` Datenträger einhängen (der normale Benutzer kann lediglich den einfachen `mount`-Befehl zum Anzeigen von Informationen benutzen), es sei denn, `user` ist gesetzt. In diesem Fall dürfen normale Benutzer »`mount` ⟨*Gerät*⟩« oder »`mount` ⟨*Mountpunkt*⟩« sagen; dabei wird immer das benannte Gerät unter dem angegebenen Mountpunkt eingehängt. Mit der Option `user` darf nur der Benutzer das Gerät wieder aushängen, der es auch eingehängt hat (`root` natürlich auch); es gibt eine ähnliche Option `users`, bei der jeder Benutzer das Gerät wieder aushängen darf.

`sync` Schreibvorgänge werden nicht im RAM gepuffert, sondern direkt das Medium geschrieben. Damit wird einem Anwendungsprogramm eine Schreiboperation erst dann als abgeschlossen gemeldet, wenn die Daten tatsächlich auf dem Medium stehen. Dies ist für Disketten oder USB-Sticks sinnvoll, die Sie sonst aus dem Laufwerk holen (oder ausstöpseln) könnten, während noch ungeschriebene Daten im RAM stehen.

`ro` Das Dateisystem wird nur zum Lesen eingebunden (Gegenteil von `rw`).

`exec` Vom Dateisystem aus können ausführbare Dateien aufgerufen werden. Das Gegenteil ist `noexec`; `exec` ist hier angegeben, weil die Option `user` unter anderem `noexec` impliziert.

Wie Sie im Eintrag für `/dev/hdb` sehen, können spätere Optionen frühere überschreiben: `user` enthält implizit die Option `noexec`, das `exec` weiter rechts in der Liste überschreibt jedoch diese Voreinstellung.

15.2.3 Labels und UUIDs

Wir haben Ihnen gezeigt, wie Sie Dateisysteme über Gerätenamen wie `/dev/sda1` einhängen können. Das hat aber den Nachteil, dass die Zuordnung von Gerät zu Gerätedatei nicht zwangsläufig stabil ist: Sobald Sie eine Platte austauschen oder umpartitionieren oder wenn Sie eine neue Platte einbauen, kann sich die Zuordnung ändern und Sie müssen die Konfiguration in `/etc/fstab` anpassen. Bei einigen Gerätetypen, wie etwa USB-Medien, können Sie sich konstruktionsbedingt auf gar nichts verlassen. Abhilfe schaffen hier Labels und UUIDs.

Label

Bei einem **Label** (engl. für Etikett, Schild) handelt es sich um einen frei vergebbaren Text von maximal 16 Zeichen Länge, der im Superblock des Dateisystems abgelegt wird. Wenn Sie beim Anlegen des Dateisystems das Label vergessen haben, so können Sie es jederzeit mit `e2label` nachtragen oder ändern. Das Kommando

```
# e2label /dev/sda3 HOME
```

zum Beispiel erlaubt Ihnen, auf `/dev/sda3` nun als `LABEL=HOME` zuzugreifen, z. B. mit

```
# mount LABEL=HOME /home
```

Das System sucht dann alle verfügbaren Partitionen nach einem Dateisystem ab, das dieses Label enthält.

Dasselbe funktioniert mit der `-L`-Option von `tune2fs`:

```
# tune2fs -L HOME /dev/sda3
```

Die anderen Dateisysteme haben auch Mittel und Wege, um ein Label zu setzen. Bei Btrfs zum Beispiel können Sie direkt beim Generieren des Dateisystems eins vergeben (Option »`-L`«), oder Sie verwenden

```
# btrfs filesystem label /dev/sdb1 MEINLABEL
```

UUID

Haben Sie sehr viele Platten oder Rechner, so dass Ihnen Labels nicht die erforderliche Eindeutigkeit liefern, so können Sie auf einen *universally unique identifier* (**UUID**) zurückgreifen. Ein UUID sieht typischerweise so aus:

```
$ uuidgen
bea6383f-22a7-453f-8ef5-a5b895c8ccb0
```

und wird beim Anlegen eines Dateisystems automatisch und zufällig erzeugt. Dadurch wird sichergestellt, dass keine zwei Dateisysteme den gleichen UUID haben. Ansonsten ist die Handhabung wie bei Labels, nur dass Sie jetzt `UUID=bea6383f-22a7-453f-8ef5-a5b895c8ccb0` verwenden müssen (Schluck.). Auch UUIDs können Sie mit `tune2fs` setzen oder durch

```
# tune2fs -U random /dev/sda3
```

neu erzeugen lassen. Das sollte aber nur selten notwendig sein, beispielsweise wenn Sie eine Platte ersetzen oder Dateisysteme geklont haben.

Herausfinden können Sie den UUID eines `ext`-Dateisystems übrigens mit

```
# tune2fs -l /dev/sda2 | grep UUID
Filesystem UUID:              4886d1a2-a40d-4b0e-ae3c-731dd4692a77
```

Bei anderen Dateisystemen (XFS, Btrfs) können Sie den UUID eines Dateisystems zwar abfragen (`blkid` ist Ihr Freund), aber nicht ohne Weiteres neu setzen.

Das Kommando

```
# lsblk -o +UUID
```

gibt Ihnen einen Überblick über alle Blockgeräte und ihre UUIDs.

Auch Partitionen mit Auslagerungsspeicher können Sie über Labels oder UUIDs ansprechen:

```
# swapon -L swap1
# swapon -U 88e5f06d-66d9-4747-bb32-e159c4b3b247
```

Die UUID einer Swap-Partition können Sie mit `blkid` oder `lsblk` herausfinden oder in `/dev/disk/by-uuid` nachschauen. Sollte Ihre Swap-Partition keine UUID und/oder kein Label haben, können Sie `mkswap` verwenden, um welche zu vergeben.

Sie können auch in der Datei `/etc/fstab` Labels und UUIDs verwenden (man könnte das auch für den Zweck der Übung halten). Schreiben Sie ins erste Feld statt des Gerätenamens

```
LABEL=home
```

oder

```
UUID=bea6383f-22a7-453f-8ef5-a5b895c8ccb0
```

Das funktioniert natürlich auch für Auslagerungsspeicher.

Übungen

15.6 [!2] Betrachten Sie die Einträge in den Dateien `/etc/fstab` und `/etc/mtab`. Wie unterscheiden sich diese?

15.3 Das Programm `dd`

`dd` ist ein Programm, das Dateien »blockweise« kopiert. Besonders gerne wird es verwendet, um »Images«, also Komplettkopien von Dateisystemen, anzulegen – etwa zur Systemrestaurierung im Falle eines katastrophalen Plattendefekts.

`dd` (kurz für *copy and convert*[2]) liest Daten blockweise aus einer Eingabedatei und schreibt sie unverändert in eine Ausgabedatei. Dabei spielt die Art der Daten keine Rolle. Ob es sich bei den Dateien um reguläre Dateien oder um Gerätedateien handelt, spielt für `dd` ebenfalls keine Rolle.

Eine schnell zurückspielbare Sicherheitskopie Ihrer Systempartition können Sie mit `dd` wie folgt erzeugen:

```
# dd if=/dev/sda2 of=/data/sda2.dump
```

Hierdurch wird die zweite Partition der ersten SCSI-Festplatte in eine Datei `/data/sda2.dump` gesichert – diese Datei sollte sich sinnvollerweise auf einer anderen Platte befinden. Sollte Ihre erste Platte beschädigt werden, können Sie nach Austausch der Festplatte gegen ein baugleiches (!) Modell in kürzester Zeit den ursprünglichen Zustand wieder herstellen:

```
# dd if=/data/sda2.dump of=/dev/sda2
```

(Wenn `/dev/sda` Ihre Systemplatte ist, müssen Sie natürlich von einem Rettungs- oder Live-System gebootet haben.)

Damit das gut funktioniert, muss, wie gesagt, die Geometrie der neuen Festplatte mit der Geometrie der alten übereinstimmen. Außerdem braucht die neue Partitionstabelle Festplatte eine Partitionstabelle, die mit der der alten übereinstimmt. Auch die

[2] Ehrlich! Das `dd`-Kommando ist inspiriert von einem entsprechenden Kommando, das es auf IBM-Großrechnern gab (deswegen auch die für Unix-Begriffe ziemlich krause Parametersyntax). Dort hieß das Kommando `CC` wie *copy and convert*, aber der Name `cc` war unter Unix schon vom C-Übersetzer belegt.

Partitionstabelle können Sie (jedenfalls bei MBR-partitionierten Platten) mit dd sichern:

```
# dd if=/dev/sda of=/media/floppy/mbr_sda.dump bs=512 count=1
```

So verwendet sichert dd nicht etwa die ganze Festplatte auf Diskette, sondern schreibt das Ganze in »Häppchen« von 512 Byte (bs=512) und zwar genau ein Häppchen (count=1). Im Endeffekt landet der gesamte MBR auf der Floppy. Zwei Fliegen mit einem Streich: die Stufe 1 des Bootladers befindet sich nach der Restaurierung ebenfalls wieder auf der Festplatte:

```
# dd if=/media/floppy/mbr_sda.dump of=/dev/sda
```

Eine Häppchengröße muss hier nicht angegeben werden, die Datei wird nur einmal geschrieben und ist (hoffentlich) nur 512 Byte groß.

Achtung: Die Partitionsinformationen für logische Partitionen stehen nicht im MBR! Wenn Sie logische Partitionen verwenden, sollten Sie ein Kommando wie sfdisk benutzen, um das komplette Partitionsschema zu sichern – siehe unten.

Zum Sichern der Partitionierungsinformationen von GPT-partitionierten Platten verwenden Sie zum Beispiel gdisk (das Kommando b).

dd kann auch verwendet werden, um auf den Inhalt von CDROMs oder DVDs dauerhaft über Festplatte zugreifen zu können. Das Kommando »dd if=/dev/cdrom of=/data/cdrom1.iso« packt den Inhalt der CDROM auf Festplatte. Da die Datei ein ISO-Image ist, also ein Dateisystem enthält, das der Linux-Kern interpretieren kann, lässt sie sich auch mounten. Nach »mount -o loop,ro /data/cdrom1.iso /mnt« können Sie auf den Inhalt des Images zugreifen. Selbstverständlich können Sie diese Zugriffsmöglichkeit auch dauerhaft mittels /etc/fstab realisieren.

Kommandos in diesem Kapitel

blkid Findet Attribute von blockorientierten Geräten und gibt sie aus blkid(8) 266

dd „*Copy and convert*“, kopiert Dateien und Dateisysteme blockweise und macht einfache Konvertierungen dd(1) 268

debugfs Dateisystem-Debugger zur Reparatur schlimm verkorkster Dateisysteme. Nur für Gurus! debugfs(8) 255

dumpe2fs Gibt interne Strukturen des ext2-Dateisystems aus. Nur für Gurus! dumpe2fs(8) 255

dumpreiserfs Gibt interne Strukturen des Reiser-Dateisystems aus. Nur für Gurus! dumpreiserfs(8) 258

e2fsck Prüft ext2- und ext3-Dateisysteme auf Konsistenz e2fsck(8) 253

e2label Ändert das Label eines ext2/3-Dateisystem e2label(8) 266

fsck Organisiert Dateisystemkonsistenzprüfungen fsck(8) 247

lsblk Listet verfügbare Blockgeräte auf lsblk(8) 267

mkdosfs Legt FAT-formatierte Dateisysteme an mkfs.vfat(8) 262

mke2fs Legt ein ext2- oder ext3-Dateisystem an mke2fs(8) 252

mkfs Organisiert das Anlegen von Dateisystemen mkfs(8) 246

mkfs.vfat Legt FAT-formatierte Dateisysteme an mkfs.vfat(8) 262

mkfs.xfs Legt XFS-formatierte Dateisysteme an mkfs.xfs(8) 259

mkreiserfs Legt Reiser-Dateisysteme an mkreiserfs(8) 258

mkswap Initialisiert eine Auslagerungs-Partition oder -Datei mkswap(8) 263

mount Hängt ein Dateisystem in den Dateibaum ein mount(8), mount(2) 264

reiserfsck Prüft ein Reiser-Dateisystem auf Konsistenz reiserfsck(8) 258

resize_reiserfs Ändert die Größe eines Reiser-Dateisystems resize_reiserfs(8) 258

swapoff Deaktiviert eine Auslagerungs-Partition oder -Datei swapoff(8) 263

swapon Aktiviert eine Auslagerungs-Partition oder -Datei swapon(8) 263

tune2fs Stellt Parameter von ext2- und ext3-Dateisystemen ein tunefs(8) 256, 267

vol_id Bestimmt Dateisystemtypen und gibt Label und UUID aus vol_id(8) 266

xfs_info Entspricht xfs_growfs, aber ändert das Dateisystem nicht xfs_growfs(8) 260

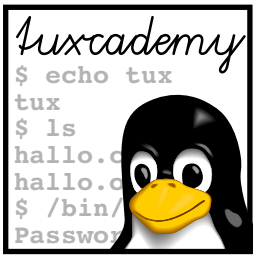
xfs_mdrestore Überträgt einen Metadaten-Abzug in ein XFS-Dateisystem xfs_mdrestore(8) 259

xfs_metadump Erzeugt Metadaten-Abzüge von XFS-Dateisystemen xfs_metadump(8) 259

xfs_repair „Repariert“ XFS-Dateisysteme xfs_repair(8) 259

Zusammenfassung

- Nach dem Partitionieren muss auf einer neuen Partition ein Dateisystem angelegt werden, bevor sie verwendet werden kann. Linux hält hierzu das Programm mkfs bereit (mit dateisystemspezifischen Hilfsprogrammen, die die eigentliche Arbeit machen).
- Unsauber ausgehängte Dateisysteme können Inkonsistenzen aufweisen. Bemerkt Linux beim Neustart solche Dateisysteme, werden diese automatisch geprüft und, wenn möglich, repariert. Über Programme wie fsck und e2fsck können solche Prüfvorgänge auch manuell angestoßen werden.
- Das Kommando mount dient zum Einhängen von Dateisystemen in den Dateibaum.
- Mit dd können Partitionen auf Blockebene gesichert werden.



16

Linux booten

Inhalt

16.1 Grundlagen 272
16.2 GRUB Legacy 275
16.2.1 Grundlagen von GRUB 275
16.2.2 Die Konfiguration von GRUB Legacy 276
16.2.3 Installation von GRUB Legacy 277
16.3 GRUB 2 278
16.3.1 Sicherheitsaspekte 279
16.4 Kernelparameter 280
16.5 Probleme beim Systemstart 282
16.5.1 Fehlersuche 282
16.5.2 Typische Probleme 282
16.5.3 Rettungssysteme und Live-Distributionen 284

Lernziele

- Die Bootlader GRUB Legacy und GRUB 2 kennen und konfigurieren können
- Probleme beim Systemstart erkennen und beheben können

Vorkenntnisse

- Grundlegendes Verständnis der Vorgänge beim Starten eines PCs
- Umgang mit Konfigurationsdateien

16.1 Grundlagen

Wenn Sie einen Linux-Rechner einschalten, läuft ein interessanter und aufwendiger Prozess ab, mit dem der Rechner sich initialisiert, testet und schließlich das eigentliche Betriebssystem (Linux) startet. In diesem Kapitel betrachten wir diesen Vorgang etwas detaillierter und erklären Ihnen, wie Sie ihn an Ihre Bedürfnisse anpassen und gegebenenfalls Probleme finden und beheben können.

Der neudeutsche Begriff »booten« kommt vom englischen *to boot*, kurz für *to bootstrap*, kurz für *to pull oneself up by one's bootstraps*, also etwa »sich an den eigenen Stiefelschlaufen hochziehen«. Hierzulande ist das am ehesten zu vergleichen mit der Leistung des »Lügenbarons« Karl Friedrich Hieronymus von Münchhausen, der sich bekanntlich an den eigenen Haaren aus dem Sumpf zog – die Begriffe »sumpfen« und »Sumpflader« haben sich leider (oder zum Glück?) aber noch nicht wirklich eingebürgert.

Unmittelbar nach dem Einschalten des Computers übernimmt die Firmware – je nach Alter des Rechners das *Basic Input/Output System* (BIOS) oder das *Unified Extensible Firmware Interface* (UEFI) – die Kontrolle. Was dann passiert, hängt von der Firmware ab.

BIOS-Startvorgang Bei BIOS-basierten System sucht das BIOS abhängig von der in seiner Konfiguration eingestellten Bootreihenfolge auf Medien wie CD-ROM oder Festplatte nach einem Betriebssystem. Bei Disketten oder Festplatten werden die ersten 512 Bytes des Bootmediums gelesen. Hier stehen spezielle Informationen zum System-Start. Generell heißt dieser Bereich **Bootsektor**, der Bootsektor einer Festplatte außerdem **Master Boot Record** (MBR).

Bootsektor
Master Boot Record

Der MBR ist uns schon bei der Diskussion des gleichnamigen Partitionierungsschemas in Kapitel 14 begegnet. Hier geht es jetzt um den Anfang des MBR vor der Partitionstabelle.

Die ersten 446 Bytes des MBR enthalten ein Ministartprogramm, das seinerseits für den Start des Betriebssystems zuständig ist, den **Bootlader**. Der Rest wird von der Partitionstabelle eingenommen. 446 Bytes reichen nicht für den kompletten Bootlader, aber es ist genug Platz für ein kleines Programm, das mit Hilfe des BIOS den Rest des Bootladers von der Platte holen kann. Im Bereich zwischen dem MBR und dem Beginn der ersten Partition – mindestens Sektor 63, heute eher Sektor 2048 – ist genug Platz für den restlichen Bootlader. (Wir kommen auf das Thema gleich noch zurück.)

Bootlader

Heutige Bootlader für Linux (insbesondere der »Grand Unified Bootloader« oder **GRUB**) können die gängigen Linux-Dateisysteme lesen und sind darum in der Lage, den Betriebssystemkern auf einer Linux-Partition zu finden, ins RAM zu laden und zu starten.

GRUB

Bootmanager

GRUB funktioniert nicht nur als Bootlader, sondern auch als **Bootmanager**. Als solche kann er nach Wahl des Benutzers verschiedene Linux-Kerne bzw. sogar andere Betriebssysteme starten.

Bootfähige CD-ROMs oder DVDs spielen eine wichtige Rolle bei der Installation oder Aktualisierung von Linux-Systemen oder als Grundlage von Live-Systemen, die direkt von einem nur lesbaren Medium ablaufen, ohne auf der Platte installiert werden zu müssen. Um einen Linux-Rechner von CD zu booten, müssen Sie im einfachsten Fall nur dafür sorgen, dass das CD-Laufwerk weiter vorne in der Bootreihenfolge der Firmware steht als die Festplatte, und den Rechner starten, während die gewünschte CD im Laufwerk liegt.

Das Booten von CD-ROMs folgt in der BIOS-Tradition anderen Regeln als das Booten von Diskette oder Festplatte. Im wesentlichen definiert der

»El-Torito«-Standard (in dem diese Regeln niedergelegt sind) zwei Ansätze: Beim einen wird auf der CD-ROM ein bis zu 2,88 MiB großes Abbild einer bootfähigen Diskette abgelegt, die die Firmware findet und bootet; beim anderen wird direkt von der CD-ROM gebootet, was einen speziellen Bootlader (für Linux etwa `ISOLINUX`) voraussetzt.

Mit der entsprechenden Hard- und Software (letztere heute meist in der Firmware enthalten) kann ein PC auch über das Netz booten. Kernel, Root-Dateisystem und alles weitere können dabei auf einem entfernten Server liegen, so dass der Rechner plattenfrei und damit ohrenfreundlich sein kann. Die Details würden ein bisschen weit führen und sind auch für LPIC-1 irrelevant; suchen Sie gegebenenfalls nach Stichwörtern wie »PXE« oder »Linux Terminal Server Project«.

UEFI-Startvorgang UEFI-basierte Systeme verwenden keine Bootsektoren, sondern die UEFI-Firmware enthält bereits einen Bootmanager, der Informationen über das zu startende Betriebssystem in nichtflüchtigem Speicher (NVRAM) ausnutzt. Bootlader für verschiedene Betriebssysteme auf dem Rechner stehen als reguläre Dateien in einer »EFI-Systempartition« (*EFI system partition*, ESP), von wo die Firmware sie lesen und starten kann. Im NVRAM steht entweder der Name des zu startenden Bootladers, oder das System verwendet den Standardnamen `/EFI/BOOT/BOOTX64.EFI`. (Das `X64` steht hier für »64-Bit-Intel-artiger PC«. Theoretisch funktioniert UEFI auch für 32-Bit-Systeme; eine grandiose Idee ist das aber nicht unbedingt.) Der betriebsspezifische Bootlader (etwa GRUB) kümmert sich dann um den Rest wie beim BIOS-Startvorgang.

Die ESP muss offiziell ein FAT32-Dateisystem enthalten (es gibt Linux-Distributionen, die FAT16 verwenden, aber das macht Probleme mit Windows 7, das FAT32 verlangt). Eine Größe von 100 MiB reicht für gewöhnlich aus, aber manche UEFI-Implementierungen haben Probleme mit FAT32-ESPs, die kleiner als 512 MiB sind, und das `mkfs`-Kommando von Linux verwendet standardmäßig FAT16 für Partitionen mit bis zu 520 MiB. Bei den heutigen Plattenpreisen gibt es keinen Grund, nicht zur Sicherheit eine ESP von um die 550 MiB anzulegen.

Es ist grundsätzlich möglich, einfach einen kompletten Linux-Kernel als `BOOTX64.EFI` in die ESP zu schreiben und so ganz ohne einen Bootlader im engeren Sinne auszukommen. PC-Linux-Distributionen machen das normalerweise nicht so, aber dieser Ansatz ist für Embedded-Systeme interessant.

Viele UEFI-basierte Systeme erlauben als Alternative auch das Booten à la BIOS von MBR-partitionierten Platten, also mit einem Bootsektor. Die Bezeichnung hierfür ist *Compatibility Support Module* (CSM). Mitunter wird dieser Modus automatisch verwendet, wenn auf der ersten erkannten Festplatte ein traditioneller MBR gefunden wird. Das verhindert einen UEFI-Startvorgang von einer ESP auf einer MBR-partitionierten Platte und ist nicht 100% hasenrein.

CSM

UEFI-basierte Systeme booten von einer CD-ROM, indem sie (ähnlich wie bei Platten) nach einer Datei namens `/EFI/BOOT/BOOTX64.EFI` suchen. (Wobei es möglich ist, CD-ROMs zu machen, die auf UEFI-Systemen UEFI-mäßig und auf BIOS-Systemen per El Torito booten.)

UEFI Secure Boot

»UEFI Secure Boot« soll verhindern, dass Rechner durch »Rootkits« kompromittiert werden, die sich in den Startvorgang einklinken und das System übernehmen, bevor das eigentliche Betriebssystem gestartet wird. Hierbei weigert die Firmware sich, Bootlader zu starten, die nicht mit einem passenden Schlüssel digital signiert sind. Zulässige Bootlader sind wiederum dafür verantwortlich, nur Betriebssystemkerne zu starten, die mit einem passenden Schlüssel digital signiert

sind, und entsprechende Betriebssystemkerne sind gehalten, bei dynamisch ladbaren Treibern auf korrekte digitale Signaturen zu bestehen. Ziel ist, dass auf dem System, zumindest was das Betriebssystem angeht, nur »vertrauenswürdige« Software läuft.

Ein Nebeneffekt ist, dass man auf diese Weise potentiell unerwünschte Betriebssysteme behindern oder ausschließen kann. Prinzipiell könnte eine Firma wie Microsoft auf die PC-Industrie Druck ausüben, nur von Microsoft signierte Bootlader und Betriebsssysteme zu akzeptieren; da dies aber auf das Missfallen diverser Kartellbehörden stoßen dürfte, ist mit so einem Schritt als Teil offizieller Firmenpolitik nicht wirklich zu rechnen. Eher besteht die Gefahr, dass die Hersteller von PC-Hauptplatinen und UEFI-Implementierungen nur die Anwendung »Windows booten« testen und debuggen und die Linux-Bootlader aufgrund von versehentlichen Fehlern in der Firmware nicht oder nur mit Verrenkungen zum Laufen zu bringen sind.

Shim

Linux unterstützt UEFI Secure Boot auf verschiedene Arten. Es gibt einen Bootlader namens »Shim« (entwickelt von Matthew Garrett), den ein Distributor von Microsoft signieren lassen kann. UEFI startet Shim und Shim startet dann einen anderen Bootlader oder Betriebssystemkern. Diese können signiert sein oder auch unsigniert; die von UEFI Secure Boot in Aussicht gestellte Sicherheit bekommen Sie natürlich nur mit den Signaturen. Sie können Ihre eigenen Schlüssel installieren und dann auch Ihre eigenen (selbstübersetzten) Kernels signieren.

Die Details dafür würden hier zu weit führen. Konsultieren Sie ggf. die Linup-Front-Schulungsunterlage *Linux-Systemanpassungen*.

PreLoader

Eine Alternative zu Shim ist »PreLoader« (von James Bottomley, vertrieben von der Linux Foundation). PreLoader ist einfacher als Shim und erlaubt es, einen (möglicherweise unsignierten) nachgeordneten Bootlader beim System zu akkreditieren und später ohne Rückfrage zu booten.

Platten: MBR vs. GPT Die Frage, welches Partitionierungsschema eine Platte verwendet, und die Frage, ob der Rechner mit BIOS (oder CSM) oder UEFI startet, haben eigentlich wenig miteinander zu tun. Es ist zumindest mit Linux ohne Weiteres möglich, ein BIOS-basiertes System von einer GPT-partitionierten Platte oder ein UEFI-basiertes System von einer MBR-partitionierten Platte zu starten (letzteres möglicherweise über CSM).

Um ein BIOS-basiertes System von einer GPT-partitionierten Platte zu starten, ist es sinnvoll, eine »BIOS-Boot-Partition« einzurichten und dort den Teil des Bootladers abzulegen, der nicht in den MBR passt. Die Alternative – den freien Platz zwischen MBR und Anfang der ersten Partition auszunutzen – ist bei GPT-partitionierten Platten nicht verlässlich, da die GPT-Partitionstabelle diesen Platz zumindest teilweise einnimmt und/oder die erste Partition direkt hinter der GPT-Partitionstabelle anfangen kann. Die BIOS-Boot-Partition muss nicht groß sein; 1 MiB ist höchstwahrscheinlich dicke genug.

Nach dem Bootlader Der Bootlader lädt den Linux-Betriebssystemkern und übergibt diesem die Kontrolle. Damit ist er selbst überflüssig und kann aus dem System entfernt werden; auch die Firmware wird ab jetzt links liegen gelassen – der Kernel ist ganz auf sich gestellt. Er muss insbesondere auf alle Treiber zugreifen können, die zur Initialisierung des Speichermediums mit dem Wurzeldateisystem und dieses Dateisystems selbst nötig sind (der Bootlader hat für seine Plattenzugriffe die Firmware verwendet), typischerweise also zumindest einen Treiber für einen IDE-, SATA- oder SCSI-Controller sowie für das passende Dateisystem. Diese Treiber müssen entweder fest im Kernel eingebaut sein

oder werden – heute die bevorzugte Methode – dem »frühen Userspace« (*early userspace*) entnommen, der konfiguriert werden kann, ohne dass dafür eine Neuübersetzung des Kernels nötig ist. (Sobald das Wurzeldateisystem eingehängt ist, geht es bequem weiter, da alle Treiber von dort gelesen werden können.) Zu den Aufgaben des Bootladers gehört auch das Laden der initialen Userspace-Daten.

Früher nannte man den »frühen Userspace« eine »initiale RAM-Disk«, weil die Daten als (in der Regel nur lesbares) Medium *en bloc* in den Speicher geladen und vom Betriebssystemkern wie eine blockorientierte Platte behandelt wurden. Es gab spezielle komprimierte Dateisysteme für diese Anwendung. Heute verwendet man meist ein Verfahren, bei dem die Daten für den frühen Userspace in Form eines `cpio`-Archivs vorliegen, das der Kernel direkt in den Plattencache extrahiert, so als hätten Sie jede Datei aus dem Archiv direkt von einem (hypothetischen) Speichermedium gelesen. Das macht es einfacher, den frühen Userspace nach Gebrauch zu entsorgen.

Der Kernel verwendet `cpio` statt `tar`, weil `cpio`-Archive im vom Kernel benutzten Format besser standardisiert und leichter auszupacken sind als `tar`-Archive.

Sobald der »frühe Userspace« zur Verfügung steht, wird ein Programm namens `/init` aufgerufen, das sich um die weitere Initialisierung kümmert. Dazu gehören Aufgaben wie das Identifizieren des Speichermediums, das als Wurzeldateisystem verfügbar gemacht werden soll, das Laden allfälliger benötigter Treiber, um auf das Medium und das Dateisystem zuzugreifen (natürlich kommen die Treiber ebenfalls aus dem frühen Userspace), unter Umständen die (rudimentäre) Konfiguration des Netzwerks, falls das Wurzeldateisystem auf einem entfernten Dateiserver liegt, und so weiter. Anschließend bringt der frühe Userspace das gewünschte Wurzeldateisystem unter »/« in Position und übergibt die Kontrolle an das eigentliche Init-Programm – heute zumeist System-V-Init (Kapitel 17) oder systemd (Kapitel 18), jeweils unter dem Namen `/sbin/init`. (Sie können sich mit der Kernel-Kommandooption `init=` ein anderes Programm wünschen.)

Wenn kein früher Userspace existiert, stellt der Betriebssystemkern das in seiner Kommandozeile mit der Option `root=` angegebene Medium als Wurzeldateisystem zur Verfügung und startet das mit der Option `init=` angegebene Programm, ersatzweise `/sbin/init`.

Übungen

16.1 [2] Wo auf einer MBR-partitionierten Festplatte darf sich der Bootlader befinden? Warum?

16.2 GRUB Legacy

16.2.1 Grundlagen von GRUB

GRUB ist inzwischen bei den allermeisten PC-basierten Linux-Distributionen der Standard-Bootlader. Gegenüber früheren Bootladern wie LILO hat er einige Vorteile, vor allem die Tatsache, dass er mit den gängigen Linux-Dateisystemen umgehen kann. Das heißt, er kann den Kernel direkt aus einer Datei wie `/boot/vmlinuz` lesen und ist damit immun gegen Probleme, die sich ergeben können, wenn Sie einen neuen Kernel installieren oder Ihr System anderweitig ändern. Ferner ist GRUB alles in allem komfortabler – er bietet zum Beispiel eine interaktive GRUB-Shell mit diversen Befehlen an und erlaubt so das Ändern der Boot-Konfiguration für spezielle Anforderungen oder im Fall von Problemen.

GRUB-Shell

Die GRUB-Shell gibt Zugriff auf das Dateisystem, ohne dass die üblichen Zugriffsrechte beachtet werden. Sie sollte darum niemals Unbefugten zur Verfügung gestellt, sondern auf wichtigen Rechnern durch ein Kennwort geschützt werden. Siehe auch Abschnitt 16.3.1.

Im Moment gibt es zwei verbreitete Versionen von GRUB: Die alte Version (»GRUB Legacy«) findet sich in älteren Linux-Distributionen – vor allem auch solchen mit »Enterprise«-Anspruch –, während die neuen Distributionen auf die modernere Version GRUB 2 (Abschnitt 16.3) setzen.

Die grundlegende Funktionsweise von GRUB Legacy entspricht der in Abschnitt 16.1 skizzierten Prozedur. Bei einem BIOS-basierten Bootvorgang lokalisiert das BIOS im MBR der Boot-Platte die erste Stufe (*stage 1*) des Bootladers im Werte von 446 Bytes. Die erste Stufe ist in der Lage, aufgrund von im Programm (als Teil der 446 Bytes) abgelegten Sektorlisten und den Plattenfunktionen des BIOS die nächste Stufe auf der Platte zu finden und zu lesen[1].

Die »nächste Stufe« ist normalerweise die »Stufe 1,5« (*stage 1.5*), die im anderweitig unbenutzten Platz unmittelbar hinter dem MBR und vor dem Anfang der ersten Partition liegt. Die Stufe 1,5 hat rudimentäre Unterstützung für Linux-Dateisysteme und kann die »Stufe 2« (*stage 2*) von GRUB im Dateisystem finden (in der Regel unter `/boot/grub`). Die Stufe 2 darf dann irgendwo auf der Platte liegen, kann ebenfalls Dateisysteme lesen, holt sich die Konfigurationsdatei, zeigt das Menü an und lädt und startet schließlich das gewünschte Betriebssystem (im Falle von Linux gegebenenfalls inklusive dem *early userspace*).

Die Stufe 1 könnte die Stufe 2 auch direkt lesen, nur würde das denselben Restriktionen unterliegen wie das Lesen der Stufe 1.5 (kein Dateisystemzugriff und nur innerhalb der ersten 1024 Zylinder). Deswegen wird das normalerweise nicht so gemacht.

GRUB kann die meisten Unix-artigen Betriebssysteme für x86-Rechner direkt laden und starten, darunter Linux, Minix, NetBSD, GNU Hurd, Solaris, ReactOS, Xen und VMware ESXi[2]. Der relevante Standard heißt »Multiboot«. Nicht Multiboot-kompatible Systeme (beispielsweise Windows) lädt GRUB, indem er den Bootlader des betreffenden Betriebssystems aufruft – sogenanntes *chain loading*.

Damit GRUB Legacy mit GPT-partitionierten Platten funktioniert, brauchen Sie eine BIOS-Boot-Partition, um dort die Stufe 1,5 unterzubringen. Es gibt eine Version von GRUB Legacy, die mit UEFI-Systemen zurechtkommt, aber Sie verwenden für UEFI-Startvorgänge besser einen anderen Bootlader.

16.2.2 Die Konfiguration von GRUB Legacy

`/boot/grub/menu.lst`

Die zentrale Konfigurationsdatei von GRUB Legacy befindet sich in der Regel unter `/boot/grub/menu.lst`. Hier werden die Grundeinstellungen vorgenommen und die einzelnen zu bootenden Betriebssysteme festgelegt und konfiguriert. Die Datei könnte z. B. aussehen wie folgt:

```
default 1
timeout 10

title linux
   kernel (hd0,1)/boot/vmlinuz root=/dev/sda2
   initrd (hd0,1)/boot/initrd
title failsafe
   kernel (hd0,1)/boot/vmlinuz.bak root=/dev/sda2 apm=off acpi=off
```

[1] Jedenfalls solange diese innerhalb der ersten 1024 »Zylinder« der Platte zu finden ist. Das hat historische Gründe und ist notfalls durch geeignete Partitionierung zu erreichen.

[2] Von irgendwoher muss das »U« in GRUB ja kommen.

```
   initrd (hd0,1)/initrd.bak
title einanderessystem
   root (hd0,2)
   makeactive
   chainloader +1
title floppy
   root (fd0)
   chainloader +1
```

Die einzelnen Parameter bedeuten dabei folgendes:

default Gibt das standardmäßig zu bootende System an. Achtung: GRUB fängt bei 0 an zu zählen! Der obige Eintrag startet also, wenn beim Booten nichts anderes angegeben wird, den Eintrag `failsafe`.

timeout Soviel Sekunden wird das GRUB-Menü angezeigt, bevor der `default`-Eintrag gebootet wird.

title Eröffnet einen Betriebssystemeintrag und vergibt dessen Namen, der im GRUB-Menü angezeigt wird.

kernel Gibt den zu bootenden Linux-Kernel an. `(hd0,1)/boot/vmlinuz` bedeutet z. B., dass der Kernel in `/boot/vmlinuz` auf der 1. Partition auf der 0. Festplatte zu finden ist, im Beispiel für `linux` also auf `/dev/hda2`. Achtung: Die 0. Festplatte ist die 1. Festplatte in der BIOS-Bootreihenfolge! Es gibt keine Unterscheidung zwischen IDE und SCSI! Und: Grub fängt bei 0 an zu zählen … Die genaue Zuordnung der einzelnen Laufwerke findet GRUB übrigens in der Datei `device.map`.

Nach der Angabe des Kernelortes können noch beliebige Kernelparameter übergeben werden. Dazu gehört auch der `boot=`-Eintrag.

initrd Gibt den Ort des `cpio`-Archivs für den "frühen Userspace" an.

root Legt für Fremd-Systeme die Systempartition fest. Hier können Sie auch Medien angeben, auf denen nur manchmal etwas Bootbares ist, etwa das Floppy-Laufwerk – dadurch können Sie von Floppy booten, ohne dass im BIOS etwas dergleichen eingestellt ist.

chainloader +1 Bezeichnet den von der Fremd-System-Systempartition zu ladenden Bootlader (in der Regel den Inhalt des Bootsektors der Systempartition).

makeactive Macht die angesprochene Partition temporär bootfähig. Bestimmte Systeme (nicht Linux) brauchen das, um von der betreffenden Partition booten zu können. Übrigens: GRUB kennt noch eine Reihe weiterer solcher Einträge, beispielsweise den Eintrag `map`, welcher es ermöglicht, einem System vorzuspiegeln, dass es auf einer anderen Festplatte (als z. B. der oft ungeliebten zweiten) installiert ist als in Wirklichkeit.

16.2.3 Installation von GRUB Legacy

Mit Installation ist nicht die Installation eines Pakets der Distribution gemeint, sondern die Installation des GRUB-Bootsektors respektive der Stufe 1 (und höchstwahrscheinlich der Stufe 1,5). Das müssen Sie aber nur selten machen, etwa bei der ursprünglichen Systeminstallation (wo die Installationsprozedur Ihrer Distribution das für Sie übernehmen sollte). Stufe 1

Für die Installation verwenden Sie das Kommando `grub`, das die GRUB-Shell aufruft. Am komfortabelsten ist es, dabei eine »Batch«-Datei zu verwenden, da Sie sonst nach einer falschen Eingabe noch einmal alles neu tippen müssen. Einige Distributionen (z. B. die von SUSE/Novell) liefern auch schon eine solche Datei mit. Der Installationsvorgang könnte dann so aussehen: GRUB-Shell

```
# grub --batch --device-map=/boot/grub/device.map < /etc/grub.inst
```

Die Option `--device-map` legt eine `device.map`-Datei unter dem angegebenen Namen an, falls noch keine existiert.

`/etc/grub.inst` Die Datei `/etc/grub.inst` kann beispielsweise den folgenden Inhalt haben:

```
root (hd0,1)
setup (hd0)
quit
```

`root` kennzeichnet dabei die Partition, auf der sich das »Heimatverzeichnis« von GRUB befindet (meistens `/boot/grub` – in diesem Verzeichnis werden die anderen Bestandteile von GRUB gesucht).

Die Partition, die Sie hier mit `root` angeben, hat nichts mit der Partition zu tun, auf der das Wurzelverzeichnis Ihrer Linux-Distribution liegt und die Sie mit `root=` in den Menüeinträgen für Ihre Linux-Kernels angeben. Jedenfalls nicht notwendigerweise. Siehe hierzu auch Abschnitt 16.4.

`setup` installiert GRUB auf dem angegebenen Gerät, hier im MBR von `hd0`. Das `setup`-Kommando von GRUB ist eine vereinfachte Fassung eines allgemeineren Kommandos namens `install`, das in den meisten Fällen funktionieren sollte.

`grub-install`

Alternativ zur beschriebenen Methode können Sie auch das Skript `grub-install` zur Installation der GRUB-Komponenten benutzen, das manche Distributionen mitbringen.

Plattenbezeichnung In der GRUB-Shell können Sie übrigens auch leicht herausfinden, was Sie als Plattenbezeichnung in `root-` oder `kernel`-Klauseln angeben müssen. Hier hilft das Kommando `find` der GRUB-Shell:

```
# grub
◁◁◁◁◁
grub> find /boot/vmlinuz
 (hd0,1)
```

16.3 GRUB 2

Neuimplementierung GRUB 2 ist eine komplette Neuimplementierung des Bootladers, bei der keine besondere Rücksicht auf Kompatibilität zu GRUB Legacy genommen wurde. GRUB 2 wurde im Juni 2012 offiziell freigegeben, auch wenn bereits diverse Distributionen frühere Versionen standardmäßig verwenden.

Beim LPIC-1-Zertifikat gehört GRUB 2 ab der Version 3.5 (vom 2. Juli 2012) der Lernziele zum Prüfungsstoff.

GRUB 2 besteht wie bisher aus verschiedenen aufeinander aufbauenden Stufen:

- Die Stufe 1 (`boot.img`) wird bei BIOS-Systemen im MBR (oder dem Bootsektor einer Partition) abgelegt. Sie kann über das BIOS den ersten Sektor der Stufe 1,5 laden, der dann wiederum den Rest der Stufe 1,5 lädt.
- Die Stufe 1,5 (`core.img`) steht entweder zwischen dem MBR und der ersten Partition (bei MBR-partitionierten Platten) oder in der BIOS-Boot-Partition (bei GPT-partitionierten Platten). Die Stufe 1.5 besteht aus einem ersten Sektor, der an das Bootmedium angepasst ist (Platte, CD-ROM, Netz, …) sowie aus einem »Kernel«, der rudimentäre Funktionen wie Geräte- und Dateizugriff, die Verarbeitung einer Kommandozeile usw. enthält, und einer beliebigen Liste von Modulen.


Über diese modulare Struktur läßt die Stufe 1.5 sich gut an Größenrestriktionen anpassen.

- Bei GRUB 2 gibt es keine explizite Stufe 2 mehr; fortgeschrittene Funktionalität wird in Modulen zur Verfügung gestellt und von der Stufe 1,5 nach Bedarf geladen. Die Module stehen in `/boot/grub`, und die Konfigurationsdatei in `/boot/grub/grub.cfg`.


Bei UEFI-basierten Systemen steht der Bootloader auf der ESP in einer Datei namens `EFI/`⟨*Betriebssystem*⟩`/grubx64.efi`. Dabei ist ⟨*Betriebssystem*⟩ etwas wie `debian` oder `fedora`. Schauen Sie auf Ihrem UEFI-basierten Linux-System mal ins Verzeichnis `/boot/efi/EFI`.


Das »`x64`« in »`grubx64.efi`« steht wieder für »64-Bit-PC«.

Konfigurationsdatei

Die Konfigurationsdatei für GRUB 2 sieht deutlich anders aus als die für GRUB Legacy und ist auch ein gutes Stück komplizierter (sie ähnelt eher einem Bash-Skript als einer GRUB-Legacy-Konfigurationsdatei). Die Autoren von GRUB 2 gehen auch davon aus, dass Sie als Systemverwalter diese Datei nicht von Hand anlegen und warten. Statt dessen gibt es ein Kommando namens `grub-mkconfig`, das eine `grub.cfg`-Datei erzeugen kann. Dazu bedient es sich einer Reihe von Hilfsprogrammen (Shellskripten) in `/etc/grub.d`, die zum Beispiel in `/boot` nach Linux-Kernels suchen, um sie ins GRUB-Bootmenü aufzunehmen. (`grub-mkconfig` schreibt die neue Konfigurationsdatei auf seine Standardausgabe; das Kommando `update-grub` ruft `grub-mkconfig` auf und leitet dessen Ausgabe nach `/boot/grub/grub.cfg` um.)

grub-mkconfig

update-grub

Sie sollten die Datei `/boot/grub/grub.cfg` also nicht direkt ändern, da Ihre Distribution zum Beispiel nach der Installation eines Kernel-Updates `update-grub` aufrufen dürfte und Ihre Änderungen an `grub.cfg` überschrieben werden würden.

Normalerweise können Sie zum Beispiel zusätzliche Einträge ins GRUB-2-Bootmenü aufnehmen, indem Sie sie in die Datei `/etc/grub.d/40_custom` schreiben. Der Inhalt dieser Datei wird von `grub-mkconfig` 1 : 1 in die Datei `grub.cfg` kopiert. Alternativ dazu können Sie Konfigurationseinstellungen in der Datei `/boot/grub/custom.cfg` machen, die – falls vorhanden – von `grub.cfg` eingelesen wird.

Hier der Vollständigkeit halber noch ein Auszug aus einer typischen `grub.cfg`-Datei. In Analogie zum Beispiel in Abschnitt 16.2.2 könnte ein Menüeintrag zum Starten von Linux bei GRUB 2 ungefähr so aussehen:

```
menuentry 'Linux' --class gnu-linux --class os {
  insmod gzio
  insmod part_msdos
  insmod ext2
  set root='(hd0,msdos2)'
  linux  /boot/vmlinuz root=/dev/hda2
  initrd /boot/initrd.img
}
```

(`grub-mkconfig` generiert normalerweise etwas Komplizierteres.) Beachten Sie, dass die GRUB-Module zum Entkomprimieren (`gzio`), für die Unterstützung von MS-DOS-artiger Partitionierung (`part_msdos`) und des `ext2`-Dateisystems explizit geladen werden. Die Nummerierung von Partitionen fängt bei GRUB 2 bei 1 an (bei GRUB Legacy war es noch 0), so dass `(hd0,msdos2)` auf die zweite MS-DOS-Partition auf der ersten Platte verweist. Statt `kernel` wird `linux` zum Starten eines Linux-Kerns verwendet.

16.3.1 Sicherheitsaspekte

Bootparameter?

Die GRUB-Shell bietet viele Möglichkeiten, unter anderem, die Möglichkeit, ohne `root`-Kennwort auf die Dateisysteme zuzugreifen! Auch die Eingabe von Bootparametern kann eine Gefahr darstellen, da Sie so auch in eine Shell booten können.

Kennwort

GRUB bietet die Möglichkeit, die beschriebenen Sicherheitslücken durch Setzen eines Kennworts zu schließen.

Das Setzen des Kennwortes geschieht für GRUB Legacy in der zentralen Konfigurationsdatei `menu.lst`. Hier muss im globalen Abschnitt der Eintrag »`password --md5` ⟨*verschlüsseltes Kennwort*⟩« eingefügt werden. Das verschlüsselte Kennwort erhalten Sie durch den Befehl `grub-md5-crypt` (oder das Kommando »`md5crypt`« in der GRUB-Shell) und können es dann z. B. auf der grafischen Oberfläche per *copy & paste* in die Datei übernehmen. Dann wird bei jeder interaktiven Einflußnahme im GRUB-Menü das entsprechende Kennwort abgefragt.

Booten einzelner Systeme verhindern

Sie können übrigens auch das Booten einzelner Systeme verhindern, indem Sie in der Datei `menu.lst` im entsprechenden spezifischen Teil die Option `lock` hinzufügen. Dann wird das oben gesetzte Kennwort auch dann abgefragt, wenn das betreffende System gestartet werden soll. Alle anderen Systeme können nach wie vor ohne Kennwort gestartet werden.

Übungen

16.2 [2] Welches ist die Konfigurationsdatei für Ihren Bootlader? Erstellen Sie einen neuen Eintrag, mit dem Sie ein weiteres Betriebssystem starten könnten. Machen Sie vorher eine Sicherungskopie der Datei.

16.3 [!3] Verhindern Sie, dass ein normaler Benutzer unter Umgehung von `init` direkt in eine Shell booten kann. Wie generieren Sie eine Kennwortabfrage beim Booten für ein bestimmtes Betriebssystem?

16.4 Kernelparameter

Laufzeitkonfigurierung des Linux-Kernels

Linux kann vom Bootlader eine Kommandozeile übernehmen und diese während des Systemstarts auswerten. Mit den Argumenten auf dieser Kommandozeile können Gerätetreiber eingestellt und verschiedene Kerneloptionen geändert werden. Dieser Mechanismus zur Laufzeitkonfigurierung des Linux-Kernels ist vor allem bei den generischen Kernels auf den Installationsmedien einer Linux-Distributionen sehr hilfreich, um einen Rechner mit einer problematischen Hardwarekonfiguration zum Laufen zu bringen. Bei GRUB können Sie die Parameter in einer Boot-Konfiguration einfach dem `kernel`-Eintrag folgen lassen.

Alternativ können Sie die Parameter interaktiv beim Booten eingeben. Hierzu müssen Sie möglicherweise schnell genug die Aufmerksamkeit von GRUB erregen (etwa indem Sie eine Pfeiltaste oder Umschalttaste drücken, während das Auswahlmenü oder der Begrüßungsbildschirm erscheint). Anschließend können Sie im Menü den gewünschten Eintrag ansteuern und e tippen. GRUB präsentiert Ihnen dann den gewünschten Eintrag, den Sie nach Ihrem Gusto anpassen können, bevor Sie den Bootvorgang fortsetzen können.

Konfiguration bestimmter Gerätetreiber

Es gibt verschiedene Arten von Argumenten. Eine erste Gruppe überlagert die voreingestellten Parameter. Hierzu gehören z. B. `root` oder `rw`. Eine andere Gruppe von Argumenten dient der Konfiguration bestimmter Gerätetreiber. Wenn eines dieser Argumente auf der Kommandozeile auftaucht, wird die Initialisierungsfunktion für den entsprechenden Gerätetreiber mit den hier angegebenen Parametern anstelle der in den Kernelsourcen festgelegten Vorgaben aufgerufen.

Die meisten Linux-Distributionen verwenden heute modulare Kernel, die nur sehr wenige Gerätetreiber tatsächlich fest integriert enthalten. Modulare Treiber können Sie nicht über die Kernel-Kommandozeile konfigurieren.

Sollte es beim Booten zu Problemen mit einem fest in den Kernel eingebauten Treiber kommen, können Sie diesen Treiber meistens abschalten, indem Sie als Argument für den entsprechenden Bootparameter einfach nur die Zahl 0 angeben.

allgemeine Einstellungen

Schließlich gibt es noch Argumente für allgemeine Einstellungen. Dazu gehören z. B. `init` oder `reserve`. Im folgenden sind einige typische Argumente aufgeführt. Es handelt sich dabei nur um einige Beispiele aus einer Vielzahl von möglichen Argumenten. Weitere Parameter finden Sie in der Dokumentation der Kernel-Quellen. Genaue Details für spezielle Hardware müssen entweder im Handbuch oder im Internet recherchiert werden.

`ro` Dies veranlasst den Kernel, das Wurzeldateisystem ohne Schreibberechtigung einzuhängen.

`rw` Dies veranlasst den Kernel, das Wurzeldateisystem mit Schreibberechtigung einzuhängen, auch wenn in der Kerneldatei etwas anderes festgelegt ist.

`init=`⟨*Programm*⟩ Startet ⟨*Programm*⟩ (z. B. `/bin/bash`) anstelle des sonst üblichen `/sbin/init`)

⟨*Runlevel*⟩ Startet in den Runlevel ⟨*Runlevel*⟩, wobei ⟨*Runlevel*⟩ in der Regel eine Ziffer zwischen 1 und 5 ist. Ansonsten ergibt der Runlevel sich aus der Datei `/etc/inittab`. (Irrelevant für Rechner, die systemd benutzen.)

`single` Startet in den Einbenutzermodus.

`maxcpus=`⟨*Zahl*⟩ Verwendet auf einem System mit mehreren Prozessoren (oder, heutzutage, Prozessorkernen) nur so viele wie angegeben. Dies ist nützlich zur Fehlersuche oder für Leistungsmessungen.

`mem=`⟨*Größe*⟩ Gibt die zu verwendende Speichergröße an. Dies ist einerseits nützlich, wenn der Kernel nicht von selbst die korrekte Speichergröße erkennt (heutzutage eher unwahrscheinlich) oder Sie testen wollen, wie sich das System mit wenig Speicher benimmt. Die ⟨*Größe*⟩ ist eine Zahl, optional gefolgt von einer Einheit (»`G`« für Gibibyte, »`M`« für Mebibyte oder »`K`« für Kibibyte).

Ein typischer Fehler ist etwas wie »`mem=512`«. Linux geht sparsam mit den Systemressourcen um, aber in 512 Byte RAM (!) kann es dann doch nicht laufen …

`panic=`⟨*Sekunden*⟩ Löst im Falle eines katastrophalen Kernel-Absturzes (im Jargon *kernel panic* genannt, Edsger Dijkstras Ausspruch »Anthropomorphe Begriffsbildung im Umgang mit Computern ist ein Zeichen professioneller Unreife« zum Trotz) nach ⟨*Sekunden*⟩ einen automatischen Neustart aus.

`hd`*x*`=noprobe` Veranlasst den Kernel, das festplattenartige Gerät `/dev/hd`*x* (IDE-Platte, CD-ROM, …) komplett zu ignorieren. Es genügt nicht, das Gerät im BIOS auszuschalten, da Linux es trotzdem findet und anspricht.

`noapic` und ähnliche Parameter wie `nousb`, `apm=off`, `acpi=off` sagen Linux, dass es bestimmte Bereiche der Kernel-Funktionalität nicht verwenden soll. Diese Optionen können dazu dienen, Linux auf ungewöhnlichen Rechnern überhaupt zum Laufen zu bringen, damit Sie Probleme in den betreffenden Bereichen genauer analysieren und beheben können.

Eine komplette Liste aller Parameter, die auf der Kernel-Kommandozeile angegeben werden können, steht in der Datei `Documentation/kernel-parameters.txt`, die Bestandteil des Linux-Quellcodes ist. (Bevor Sie allerdings nur wegen dieser Datei Kernel-Quellen installieren, sollten Sie im Internet nach ihr suchen.)

Optionen auf der Kernel-Kommandozeile, die keine Kernelparameter sind, gibt der Kernel übrigens als Umgebungsvariable an den `init`-Prozess weiter.

Umgebungsvariable für `init`

16.5 Probleme beim Systemstart

16.5.1 Fehlersuche

Normalerweise ist die Lage einfach: Sie schalten den Rechner ein, gehen gemütlich zur Kaffeemaschine (oder auch nicht – siehe Abschnitt 17.1), und wenn Sie wiederkommen, grüßt Sie der grafische Anmelde-Bildschirm. Aber was tun, wenn das mal nicht so klappt?

Die Diagnose von Problemen beim Systemstart ist mitunter nicht ganz einfach – jede Menge Meldungen rauschen über den Bildschirm oder werden (bei manchen Distributionen) überhaupt nicht mehr angezeigt, sondern hinter einem netten Bildchen versteckt. Der Systemprotokolldienst (`syslogd`) wird auch erst nach einer Weile hochgefahren. Zum Glück läßt Linux Sie aber nicht im Regen stehen, wenn Sie die Meldungen des Kernels in Ruhe nachlesen wollen.

Für die Zwecke der Protokollierung läßt sich der Systemstartvorgang in zwei Phasen einteilen: Die »frühe« Phase umfasst alles vom ersten Lebenszeichen des Kernels bis zu dem Moment, wo der Systemprotokolldienst eingeschaltet wird. Die »späte« Phase beginnt genau dann und endet im Prinzip erst beim Herunterfahren des Rechners.

Die Meldungen der frühen Phase schreibt der Kernel in einen internen Puffer, der mit dem Kommando `dmesg` ausgelesen werden kann. Manche Distributionen arrangieren, dass diese Meldungen baldmöglichst an den Systemprotokolldienst übergeben werden, damit sie auch im »offiziellen« Protokoll auftauchen.

In der späten Phase läuft der Systemprotokolldienst, den wir hier nicht detailliert besprechen – er ist Thema in der Linup-Front-Schulungsunterlage *Linux-Administration II* (und der Prüfung LPI-102). Uns soll für jetzt genügen, dass bei den meisten Distributionen fast alle Meldungen, die über den Protokolldienst laufen, in der Datei `/var/log/messages` abgelegt werden. Dort finden sich auch die Meldungen, die vom Zeitpunkt des Starts des Protokolldiensts an beim Startvorgang anfallen.

Bei Debian GNU/Linux enthält `/var/log/messages` nur einen Teil der Systemmeldungen, nämlich alles, was keine gravierende Fehlermeldung ist. Wenn Sie *alles* sehen wollen, müssen Sie sich `/var/log/syslog` anschauen – darin stehen alle Meldungen ausser (aus Gründen der Privatsphäre) denen, die sich mit Authentisierung beschäftigen. Auch die Kernel-Nachrichten der »frühen Phase« übrigens.

Theoretisch können nach dem Start von `init` und vor dem Start des Systemprotokolldienstes Meldungen, die nicht vom Kernel kommen, verloren gehen. Aus diesem Grund ist der Systemprotokolldienst in der Regel einer der ersten Dienste, die nach `init` gestartet werden.

16.5.2 Typische Probleme

Hier sind einige Schwierigkeiten, die beim Booten auftreten können:

Der Rechner startet überhaupt nicht Wenn der Computer gar nichts macht, liegt in der Regel ein Hardwaredefekt vor. (Wenn Sie so einen Fall am Telefon diagnostizieren müssen, dann fragen Sie unbedingt nach den offensichtlichen Sachen wie »Steckt der Stromstecker in der Steckdose?« – vielleicht hat die Putzkolonne nach Feierabend dringend ihren Staubsauger einstöpseln müssen – und »Ist der Kippschalter hinten am Netzteil auf *An*?«. Manchmal sind es die einfachen Probleme.) Dasselbe ist wahrscheinlich der Fall, wenn der Computer nur Pieptöne von sich gibt oder rhythmisch mit seinen Leuchtdioden blinkt, aber nicht wirklich zu booten anfängt.

Die Pieptöne oder Blinkzeichen können Eingeweihten eine grobe Fehlerdiagnose ermöglichen. Details der Hardware-Fehlersuche würden hier aber endgültig zu weit führen.

Sachen gehen schief, bevor der Bootlader startet Die Firmware nimmt verschiedene Selbsttests vor und gibt Fehlermeldungen auf dem Bildschirm aus, wenn Sachen nicht stimmen (etwa fehlerhafter RAM-Speicher festgestellt wird). Auch die Behebung solcher Probleme diskutieren wir hier nicht weiter. Wenn alles glatt geht, sollte Ihr Rechner das Bootlaufwerk identifizieren und den Bootlader starten.

Der Bootlader läuft nicht durch Das kann daran liegen, dass das Betriebssystem ihn nicht finden kann (etwa weil das Laufwerk, auf dem er installiert ist, nicht in der Bootreihenfolge in der Firmware vorkommt) oder er beschädigt ist. Im ersteren Fall sollten Sie sicherstellen, dass Ihre Firmware das Richtige tut (kein Thema für uns). Im letzteren Fall sollten Sie zumindest eine rudimentäre Fehlermeldung bekommen, die Ihnen zusammen mit der Dokumentation des Bootladers eine Diagnose erlauben müßte.

GRUB als zivilisiertes Programm liefert Fehlermeldungen im Klartext, die in der Info-Dokumentation von GRUB genauer erklärt sind.

Die Abhilfe für die meisten prinzipiellen (im Gegensatz zu Konfigurations-) Probleme mit dem Bootlader, sofern sie nicht offensichtlich auf Platten- oder Firmware-Fehler zurückzuführen sind, besteht darin, das System von CD-ROM zu booten – das »Rettungssystem« der Distribution oder eine »Live-Distribution« wie Knoppix bieten sich an – und den Bootlader neu zu installieren.

Dasselbe gilt für Probleme wie zum Beispiel eine ruinierte Partitionstabelle im MBR. Sollten Sie versehentlich Ihren MBR überschrieben haben, können Sie vom Rettungssystem aus mit `dd` ein Backup einspielen (Sie haben doch eins, nicht wahr?), die Partitionierung mit `sfdisk` erneuern (Sie haben doch sicher einen Ausdruck der Partitionstabelle irgendwo gut findbar verwahrt, nicht wahr?) oder Rettungsversuche mit `gdisk` unternehmen, und dann den Bootlader neu schreiben.

Für den Fall eines ultimaten Partitionstabellen-Super-GAUs gibt es auch Programme, die die ganze Platte nach Stellen absuchen, die wie der Superblock eines Dateisystems aussehen, und die Partitionierung auf diese Weise rekonstruieren (helfen) können. Wir drücken Ihnen die Daumen, dass Sie sowas nie brauchen werden.

Der Kernel startet nicht Wenn der Bootlader seine Arbeit getan hat, sollte der Kernel zumindest starten (was sich durch eine gewisse Aktivität auf dem Bildschirm bemerkbar macht). Die Kernels der Distributionen sind generisch genug, dass sie auf den meisten PCs laufen sollten, aber es kann trotzdem Probleme geben, etwa wenn Sie einen Rechner mit extrem moderner Hardware haben, die der Kernel noch nicht kennt (was fatal ist, wenn zum Beispiel ein Treiber für den Plattencontroller fehlt) oder Sie am frühen Userspace herumgebastelt haben (Pfui, wenn Sie nicht wissen, was Sie tun!). Unter Umständen kann es hier möglich sein, durch BIOS-Einstellungen (etwa Zurückschalten eines SATA-Plattencontrollers in einen »traditionellen«, IDE-kompatiblen Modus) oder das Deaktivieren gewisser Funktionsbereiche des Kernels (siehe Abschnitt 16.4) den Rechner zum Booten zu bringen. Hier lohnt es sich, einen anderen Computer für Internet-Recherchen per Google & Co. parat zu haben.

Wenn Sie am Kernel herumbasteln oder eine neue Version Ihres Distributions-Kernels einspielen wollen, dann sorgen Sie auf jeden Fall dafür, noch einen Kernel in der Hinterhand zu haben, von dem Sie wissen, dass er funktioniert. Wenn Sie immer einen brauchbaren Kernel im Menü Ihres Bootladers stehen haben, sparen Sie sich das lästige Hantieren mit CDs.

Andere Probleme Wenn der Kernel seine Initialisierungen vollzogen hat, wird die Kontrolle an den »Init«-Prozess übergeben. Darüber erfahren Sie mehr in Kapitel 17. Sie müssten dann aber aus dem Gröbsten heraus sein.

16.5.3 Rettungssysteme und Live-Distributionen

Als Systemadministrator sollten Sie immer ein »Rettungssystem« für Ihre Distribution zur Hand haben, denn typischerweise brauchen Sie es gerade dann, wenn Sie am wenigsten in einer Position sind, es sich kurzfristig zu besorgen. (Das gilt vor allem, wenn Ihr Linux-Rechner Ihr einziger Computer ist.) Ein Rettungssystem ist eine abgespeckte Version Ihrer Distribution, die Sie von einer CD oder DVD (früher Diskette(n)) starten können und die in einer RAM-Disk läuft.

Sollte Ihre Linux-Distribution nicht über ein eigenes Rettungssystem auf Diskette oder CD verfügen, dann besorgen Sie sich eine »Live-Distribution« wie zum Beispiel Knoppix. Live-Distributionen werden von CD (oder DVD) gestartet und funktionieren ohne Installation auf Ihrem Rechner. Knoppix finden Sie als ISO-Image im Internet (`http://www.knoppix.de/`) oder hin und wieder als Beigabe zu Computerzeitschriften.

Der Vorteil von Rettungssystemen und Live-Distributionen besteht darin, dass sie funktionieren, ohne dass Ihre Festplatte beteiligt ist. Sie können also Dinge tun wie das Wurzeldateisystem einem `fsck` zu unterziehen, die nicht erlaubt sind, während Ihr System von Platte läuft. Hier sind ein paar Probleme und ihre Lösungen:

Kernel zerschossen? Booten Sie das Rettungssystem und installieren Sie das betreffende Paket neu. Im einfachsten Fall gehen Sie nach dem Start des Rettungssystem auf das Wurzeldateisystem Ihrer installierten Distribution:

```
# mount -o rw /dev/sda1 /mnt                    Gerätename kann abweichen
# chroot /mnt
# _                                  Jetzt sehen wir die installierte Distribution
```

Danach können Sie die Netzschnittstelle aktivieren oder ein Kernel-Paket von einem USB-Stick oder CD-ROM kopieren und mit dem Paketwerkzeug Ihrer Distribution installieren.

`root`-Kennwort vergessen? Booten Sie das Rettungssystem und wechseln Sie wie im vorigen Punkt in die installierte Distribution. Anschließend hilft

```
# passwd
```

(Dieses Problem können Sie natürlich auch ohne Rettungssystem über einen Neustart mit »`init=/bin/bash rw`« als Kernel-Parameter in den Griff bekommen.)

Live-Distributionen wie Knoppix sind auch nützlich, um im Computergeschäft zu prüfen, ob Linux die Hardware des Rechners unterstützt, um den Sie schon eine Weile lang lüstern herumschleichen. Wenn Knoppix eine Hardware erkennt, dann können Sie das anderen Linux-Distributionen in aller Regel auch beibringen. Wenn Knoppix eine Hardware nicht erkennt, dann ist das mitunter kein prinzipielles Problem (es könnte irgendwo im Netz einen Treiber geben, von dem Knoppix nichts weiß), aber Sie sind zumindest gewarnt.

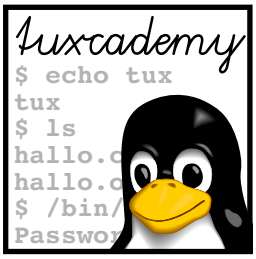
Wenn es zu Ihrer Distribution eine passende Live-Version gibt – bei Ubuntu zum Beispiel sind die Live- und die Installations-CD identisch –, dann sind Sie natürlich besonders fein raus, weil die Live-Distribution dann normalerweise dieselbe Hardware erkennt wie die zu installierende Distribution.

Kommandos in diesem Kapitel

dmesg Gibt den Inhalt des Kernel-Nachrichtenpuffers aus dmesg(8) 282
grub-md5-crypt Bestimmt MD5-verschlüsselte Kennwörter für GRUB Legacy grub-md5-crypt(8) 279

Zusammenfassung

- Ein Bootlader ist ein Programm, das ein Betriebssystem laden und starten kann.
- Ein Bootmanager ist ein Bootlader, der die Auswahl zwischen verschiedenen Betriebssystemen oder Betriebssysteminstallationen erlaubt.
- GRUB ist ein leistungsfähiger Bootmanager mit besonderen Eigenschaften – etwa der Möglichkeit, auf Dateien zuzugreifen, und einer eingebauten Kommandoshell.
- Die GRUB-Shell hilft bei der Installation von GRUB sowie bei der Konfiguration einzelner Bootvorgänge.



17

System-V-Init und der Init-Prozess

Inhalt

17.1 Der Init-Prozess 288
17.2 System-V-Init 288
17.3 Upstart 295
17.4 Herunterfahren des Systems 297

Lernziele

- Die System-V-Init-Infrastruktur verstehen
- Den Aufbau der Datei `/etc/inittab` kennen
- Runlevels und Init-Skripte verstehen
- Das System geordnet herunterfahren oder neu starten können

Vorkenntnisse

- Kenntnisse der Linux-Systemadministration
- Wissen über Vorgänge beim Systemstart (Kapitel 16)

17.1 Der Init-Prozess

Nachdem die Firmware, der Bootlader, der Betriebssystemkern und (gegebenenfalls) der frühe Userspace ihre Arbeit erledigt haben, übernimmt der »Init-Prozess« das Ruder. Seine Aufgabe besteht darin, den Systemstart zu Ende zu führen und den weiteren Systembetrieb zu koordinieren. Linux sucht und startet dafür ein Programm namens `/sbin/init`.

`/sbin/init`

Der Init-Prozess hat die Prozess-ID 1. Wenn es einen frühen Userspace gibt, dann »erbt« er diese von dem Prozess, der erzeugt wurde, um `/init` auszuführen, und später nur seinen Programmtext durch den des Init-Prozesses ersetzt.

Der Init-Prozess nimmt übrigens eine Sonderstellung ein: er ist der einzige Prozess, der mit »`kill -9`« nicht abgebrochen werden kann. (Er kann sich allerdings aus freien Stücken entleiben.)

Wenn der Init-Prozess sich tatsächlich beendet, läuft der Kernel trotzdem weiter. Es gibt Puristen, die ein Programm als Init-Prozess starten, das Paketfilterregeln in Kraft setzt und sich dann beendet. So ein Rechner als Firewall ist für alle praktischen Zwecke unknackbar, läßt sich aber auch nicht ohne Rettungssystem neu konfigurieren …

Sie können dem Linux-Kernel sagen, dass er ein anderes Programm als Init-Prozess ausführen soll. Dazu müssen Sie beim Booten eine Option wie »`init=/sbin/meininit`« angeben. An dieses Programm werden keine besonderen Anforderungen gestellt, aber Sie sollten daran denken, dass, wenn der Init-Prozess sich beendet, Sie keinen neuen mehr bekommen.

17.2 System-V-Init

Grundlagen Die traditionelle Infrastruktur, die bisher von den meisten Linux-Distributionen verwendet wurde, heißt »System-V-Init«. Das »V« ist eine römische 5, man sagt »System Five«, weil die Infrastruktur an das ehrwürdige Unix System V von AT&T angelehnt ist, wo etwas sehr Ähnliches das erste Mal auftauchte. Das war in den 1980er Jahren.

System-V-Init

Es gab schon länger den Verdacht, dass eine vor rund 30 Jahren entworfene Infrastruktur den heutigen Anforderungen an das Init-System eines Linux-Rechners nicht mehr vollständig genügt. (Zur Erinnerung: Als System-V-Init neu war, war das angesagte Unix-System eine VAX mit 30 seriellen Terminals.) Heutige Rechner müssen zum Beispiel mit häufigen Hardware-Änderungen umgehen können (Stichwort USB), und das ist etwas, womit System-V-Init sich relativ schwer tut. Deswegen wurden in den letzten Jahren einige Alternativen zu System-V-Init vorgeschlagen. Eine davon – systemd von Lennart Poettering und Kay Sievers – hat das Rennen gemacht und ist der aktuelle oder künftige Standard praktisch aller wesentlichen Linux-Distributionen (wir besprechen sie genauer in Kapitel 18). Eine andere ist Upstart von Scott James Remnant (siehe Abschnitt 17.3).

Runlevel

Eine der wesentlichen Eigenschaften von System-V-Init sind die **Runlevel**, die beschreiben, in welchem Zustand sich das System befindet und welche Dienste es anbietet. Außerdem kümmert der Init-Prozess sich darum, dass auf virtuellen Konsolen, direkt seriell angeschlossenen Terminals u. ä. ein Anmelden möglich ist, und verwaltet den Zugriff auf das System über etwaige Modems. All dies wird in der Datei `/etc/inittab` konfiguriert.

`/etc/inittab`

Die Syntax von `/etc/inittab` (Bild 17.1) ist genau wie die vieler anderer Linux-

```
# Standard-Runlevel
id:5:initdefault

# Erstes auszuführendes Skript
si::bootwait:/etc/init.d/boot

# Runlevels
l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
#l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

ls:S:wait:/etc/init.d/rc S
~~:S:respawn:/sbin/sulogin

# Ctrl-Alt-Del
ca::ctrlaltdel:/sbin/shutdown -r -t 4 now

# Terminals
1:2345:respawn:/sbin/mingetty --noclear tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Serielles Terminal
# S0:12345:respawn:/sbin/agetty -L 9600 ttyS0 vt102

# Modem
# mo:235:respawn:/usr/sbin/mgetty -s 38400 modem
```

Bild 17.1: Eine typische `/etc/inittab`-Datei (Auszug)

Konfigurationsdateien etwas eigentümlich (auch wenn daran letzten Endes natürlich AT&T schuld ist). Alle Zeilen, die nicht entweder leer oder eine Kommentarzeile – wie üblich durch »#« eingeleitet – sind, bestehen aus vier durch Doppelpunkte getrennten Feldern:

»Etikett« Die Aufgabe des ersten Felds ist, die Zeile eindeutig zu identifizieren. Sie dürfen sich hier eine beliebige Kombination aus bis zu vier Zeichen ausdenken. (Treiben Sie es nicht auf die Spitze und beschränken Sie sich auf Buchstaben und Ziffern.) Das Etikett wird nicht anderweitig verwendet.

Das eben Gesagte stimmt nicht zu 100% für die Zeilen, die sich um Terminals kümmern (siehe unten). Dort entspricht nach Konvention das Etikett dem Namen des betreffenden Geräts, aber ohne »`tty`« am Anfang, also `1` für `tty1` oder `S0` für `ttyS0`. Niemand weiß genau, warum.

Runlevels Die Runlevels, für die diese Zeile gilt. Wir haben noch nicht genau erklärt, wie Runlevels funktionieren, also sehen Sie es uns für den Moment nach, dass wir uns darauf beschränken, zu erwähnen, dass Runlevels mit Ziffern benannt werden und die betreffende Zeile in allen Runlevels beachtet wird, deren Ziffer in diesem Feld steht.

Es gibt außer denen mit Ziffern als Namen auch noch einen Runlevel namens »`S`«. Näheres dazu siehe unten.

Aktion Das dritte Feld gibt an, wie mit dieser Zeile umgegangen wird. Die wesentlichen Möglichkeiten sind

`respawn` Der in dieser Zeile beschriebene Prozess wird sofort wieder gestartet, falls er sich beendet hat. Typischerweise wird das für Terminals verwendet, die, nachdem der aktuelle Benutzer mit seiner Sitzung fertig ist, dem nächsten Benutzer taufrisch zur Verfügung gestellt werden sollen.

`wait` Der in dieser Zeile beschriebene Prozess wird einmal ausgeführt, wenn das System in den betreffenden Runlevel wechselt, und `init` wartet darauf, dass er fertig wird.

`bootwait` Der in dieser Zeile beschriebene Prozess wird beim Systemstart ausgeführt. `init` wartet darauf, dass er fertig wird. Das Runlevel-Feld in dieser Zeile wird ignoriert.

`initdefault` Das Runlevel-Feld in dieser Zeile gibt an, welchen Runlevel das System beim Start anstreben soll.

Hier steht bei LSB-konformen Distributionen normalerweise »`5`«, wenn das System eine Anmeldung auf dem Grafikbildschirm akzeptieren soll, sonst »`3`«. Näheres siehe unten.

Wenn dieser Eintrag (oder die ganze Datei `/etc/inittab` fehlt), müssen Sie auf der Konsole einen Runlevel angeben.

`ctrlaltdel` Gibt an, was das System tun soll, wenn der Init-Prozess ein `SIGINT` geschickt bekommt – was normalerweise passiert, wenn jemand auf der Konsole die Tastenkombination Strg+Alt+Entf drückt. Normalerweise läuft das auf irgendeine Form von `shutdown` hinaus (siehe Abschnitt 17.4).

Es gibt noch ein paar andere Aktionen. `powerwait`, `powerfail`, `powerokwait` und `powerfailnow` zum Beispiel dienen dazu, System-V-Init mit USVs zusammenarbeiten zu lassen. Die Details stehen in der Dokumentation (`init`(8) und `inittab`(5)).

Kommando Das vierte Feld beschreibt ein Kommando, das ausgeführt werden soll. Es reicht bis zum Ende der Zeile und Sie können hineinschreiben, was Sie mögen.

Wenn Sie Änderungen an `/etc/inittab` vorgenommen haben, wirken diese sich nicht sofort aus. Sie müssen zuerst das Kommando »`telinit q`« ausführen, um `init` dazu zu bringen, dass es die Datei neu einliest.

Das Bootskript Beim System-V-Init startet der Init-Prozess ein Shellskript, das Bootskript, typischerweise `/etc/init.d/boot` (Novell/SUSE), `/etc/rc.d/init.d/boot` (Red Hat) oder `/etc/init.d/rcS` (Debian). (Der genaue Name steht in `/etc/inittab`, suchen Sie nach einem Eintrag mit `bootwait` als Aktion.)

Dieses Bootskript übernimmt Aufgaben wie etwa die Fehlerprüfung und etwaige Korrektur der in `/etc/fstab` eingetragenen Dateisysteme, die Initialisierung des Rechnernamens und der Linux-Uhr und andere wichtige Vorarbeiten für einen stabilen Systembetrieb. Dann werden, falls notwendig, Kernelmodule nachgeladen, die Dateisysteme eingehängt und ähnliches. Die spezifischen Aktionen und ihre genaue Reihenfolge hängen von der verwendeten Linux-Distribution ab.

Heutzutage beschränkt `boot` sich in der Regel darauf, die Dateien in einem Verzeichnis wie `/etc/init.d/boot.d` (SUSE) aufzurufen. Diese Dateien werden in der Reihenfolge ihrer Namen abgearbeitet. In diesem Verzeichnis können Sie auch weitere Shellskripte unterbringen, um während der Systeminitialisierung eigenen Code auszuführen.

Übungen

17.1 [2] Können Sie herausfinden, wo Ihre Distribution die Skripte ablegt, die das Bootskript ausführt?

17.2 [!2] Nennen Sie einige typische Aufgaben des Bootskripts. In welcher Reihenfolge sollten diese ausgeführt werden.

Runlevel Nach der Ausführung des Bootskripts versucht der Init-Prozess, das System in einen der verschiedenen Runlevels zu bringen. Welcher das ist, geht ebenfalls aus `/etc/inittab` hervor oder wird beim Booten des Systemkerns auf dessen Kommandozeile festgelegt.

Runlevels

Die verschiedenen Runlevels und ihre Bedeutung sind inzwischen über die meisten Distributionen hinweg standardisiert, etwa wie folgt:

standardisierte Runlevels

1 Einbenutzermodus ohne Netzwerk

2 Mehrbenutzermodus ohne Netzwerkserver

3 Mehrbenutzermodus mit Netzwerkservern

4 Unbenutzt, kann bei Bedarf individuell konfiguriert werden

5 Wie Runlevel 3, aber mit grafischer Anmeldung

6 Reboot

0 System-Halt

Der Runlevel `S` (oder `s`) wird beim Start durchlaufen, bevor das System in einen der Runlevels 2 bis 5 übergeht. Auch wenn Sie das System in den Runlevel 1 versetzen, kommen Sie letzten Endes im Runlevel `S` heraus.

Beim Systemstart wird in der Regel einer der Runlevels 3 oder 5 angestrebt – Runlevel 5 ist typisch für Arbeitsplatzsysteme, die eine grafische Umgebung anbieten, während Runlevel 3 für Serversysteme sinnvoll ist, die möglicherweise gar nicht über eine Grafikkarte verfügen. Im Runlevel 3 können Sie immer eine Grafikumgebung nachträglich starten oder die grafische Anzeige auf einen anderen Rechner umleiten, indem Sie sich von diesem aus über das Netz anmelden.

Diese Runlevel-Vorgaben entstammen dem LSB-Standard. Nicht alle Distributionen setzen sie tatsächlich um; Debian GNU/Linux zum Beispiel überlässt die Runlevel-Zuordnung weitgehend dem lokalen Administrator.

Sie können auch die Runlevels 7 bis 9 verwenden, müssen die dann aber selber konfigurieren.

Befehl `telinit`

Während des Betriebs wird der Runlevel mit dem Befehl `telinit` gewechselt. Dieser Befehl kann nur mit `root`-Rechten ausgeführt werden: »`telinit 5`« zum Beispiel wechselt sofort in den Runlevel 5. Alle laufenden im neuen Runlevel nicht mehr benötigten Dienste werden sofort beendet, während die noch nicht laufenden, aber im neuen Runlevel erforderlichen gestartet werden.

Statt `telinit` können Sie auch `init` verwenden (ersteres ist sowieso nur ein symbolisches Link auf letzteres). Das Programm prüft beim Start seine PID, und wenn die nicht 1 ist, benimmt es sich wie `telinit`, sonst wie `init`.

`runlevel`

Mit dem Kommando `runlevel` können der vorige und der aktuelle Runlevel abgefragt werden:

```
# runlevel
N 5
```

Hier befindet das System sich im Runlevel 5, in den es, wie der Wert »`N`« für den »vorigen Runlevel« andeutet, direkt nach dem Systemstart gekommen ist. Eine Ausgabe wie »`5 3`« würde bedeuten, dass die letzte Runlevel-Änderung darin bestand, das System aus dem Runlevel 5 in den Runlevel 3 zu bringen.

Ein paar Runlevels haben wir Ihnen noch verschwiegen, nämlich die »Bedarfs-Runlevels« (engl. *on-demand runlevels*) `A`, `B` und `C`. Sie können in `/etc/inittab` Einträge haben, die für einen dieser drei Runlevels vorgesehen sind und die Aktion `ondemand` verwenden, also etwa

```
xy:AB:ondemand:...
```

Wenn Sie etwas sagen wie

```
# telinit A
```

dann werden diese Einträge ausgeführt, aber der eigentliche Runlevel nicht gewechselt: Wenn Sie vor dem `telinit` in Runlevel 3 waren, dann sind Sie hinterher immer noch dort. – `a`, `b` und `c` sind Synonyme für `A`, `B` und `C`.

Übungen

17.3 [!2] Lassen Sie sich den momentanen Runlevel anzeigen. Was genau wird angezeigt? Wechseln Sie in Runlevel 2. Sehen Sie sich jetzt den aktuellen Runlevel an.

17.4 [2] Testen Sie die Bedarfs-Runlevels: Fügen Sie eine Zeile zu `/etc/inittab` hinzu, die zum Beispiel den Runlevel `A` betrifft. Lassen Sie `init` die `inittab`-Datei neu einlesen. Geben Sie dann das Kommando »`telinit A`«.

Init-Skripte Die Dienste, die in den verschiedenen Runlevels laufen, werden über die Skripte im Verzeichnis `/etc/init.d` (Debian, Ubuntu, SUSE) oder `/etc/rc.d/init.d` (Red Hat) gestartet und gestoppt. Die Skripte werden jeweils beim Wechsel von einem Runlevel in den anderen abgearbeitet, können aber auch von Hand aufgerufen werden. Hier lassen sich eigene Skripte für jedes beliebige Zusatzprogramm einfügen. Alle diese Skripte werden unter dem Begriff »Init-Skripte« zusammengefasst.

Die Init-Skripte verstehen in der Regel die Parameter `start`, `stop`, `status`, `restart` und `reload`, mit denen Sie die entsprechenden Dienste starten, stoppen, usw. können. Mit dem Aufruf »`/etc/init.d/network restart`« werden zum Beispiel die Netzwerkkarten deaktiviert und mit erneuerter Konfiguration wieder hochgefahren.

Selbstverständlich müssen Sie beim Runlevelwechsel oder beim Systemstart nicht alle Dienste von Hand starten: Zu jedem Runlevel *r* gibt es in `/etc` (Debian und Ubuntu), `/etc/rc.d` (Red Hat) bzw. `/etc/init.d` (SUSE) ein Unterverzeichnis `rc`*r*`.d`. Über diese Verzeichnisse werden die Dienste für die einzelnen Runlevel definiert und die Übergänge zwischen den Runlevels gesteuert. In den Verzeichnissen befinden sich symbolische Links zu den Skripten im `init.d`-Verzeichnis. Über diese Links werden die betreffenden Skripte von einem Skript `/etc/init.d/rc` (typisch) beim Betreten und Verlassen der Runlevel gestartet oder gestoppt.

Runlevel-Unterverzeichnisse

Dabei werden die Namen der symbolischen Links herangezogen, um die Reihenfolge der Start- und Stoppoperationen zu bestimmen. Zwischen den verschiedenen Diensten bestehen ja Abhängigkeiten – es hätte keinen großen Sinn, zum Beispiel Netzwerkdienste wie die Samba- oder Web-Server zu starten, bevor nicht die grundlegende Netzwerkunterstützung aktiviert wurde. Bei den meisten Linux-Distributionen werden die Dienste für einen Runlevel aktiviert, indem das `rc`-Programm alle symbolischen Links in dessen Verzeichnis, die mit dem Buchstaben »`S`« anfangen, in lexikographischer Reihenfolge mit dem Parameter `start` aufruft. Da die Namen der Links nach dem »`S`« eine zweistellige Zahl enthalten, können Sie durch geschickte Wahl dieser Zahl bestimmen, wann im Laufe dieses Vorgangs welcher Dienst gestartet wird. Entsprechend werden zum Deaktivieren der Dienste für einen Runlevel beim *Verlassen* des Runlevels alle symbolischen Links in dessen Verzeichnis, die mit dem Buchstaben »`K`« anfangen, in lexikographischer Reihenfolge mit dem Parameter `stop` aufgerufen.

Dienste aktivieren

Soll ein Dienst, der im alten Runlevel gelaufen ist, auch im neuen Runlevel laufen, so kann ein unnötiges Stoppen und Starten vermieden werden. Dazu wird vor der Ausführung eines `K`-Links geprüft, ob im Verzeichnis für den neuen Runlevel für denselben Dienst ein `S`-Link existiert. In diesem Fall wird auf das Stoppen und sofortige Neustarten verzichtet.

Debian GNU/Linux geht hier anders vor: Immer wenn ein neuer Runlevel *r* betreten wird, werden *alle* symbolischen Links im *neuen* Verzeichnis (`/etc/rc`*r*`.d`) ausgeführt. Dabei bekommen Links mit »`K`« am Anfang den Parameter `stop` und alle mit »`S`« den Parameter `start` übergeben.

Dienste konfigurieren

Um die Dienste in einem Runlevel zu verändern bzw. einen neuen Runlevel anzulegen, können Sie prinzipiell die symbolischen Links direkt manipulieren. Bei den meisten Distributionen ist das aber inzwischen verpönt.

Die Red-Hat-Distributionen setzen für die Runlevel-Konfiguration ein Programm namens `chkconfig` ein. »`chkconfig quota 35`« fügt z. B. den Quota-Dienst, nein nicht in Runlevel 35, sondern in die Runlevel 3 und 5 ein. »`chkconfig -l`« bietet einen schönen Überblick über die konfigurierten Runlevel.

Die SUSE-Distributionen verwenden ein Programm namens `insserv`, um die Dienste in den verschiedenen Runlevels zu ordnen. Es benutzt Informationen in den einzelnen Init-Skripten, um eine Reihenfolge für das Starten und Stoppen der Dienste zu berechnen, die den jeweiligen Abhängigkeiten Sorge trägt. Außerdem bietet YaST2 einen grafischen »Runlevel-Editor«, und es existiert ein `chkconfig`-Programm, das allerdings nur ein Frontend zu `insserv` darstellt.

Auch bei Debian GNU/Linux müssen Sie die Links nicht mit der Hand anlegen, sondern können sich des Programms `update-rc.d` bedienen. Allerdings sind dort manuelle Eingriffe durchaus noch statthaft – `update-rc.d` dient eher dazu, dass Pakete bei der Installation ihre Init-Skripte in die Startsequenz integrieren können. Mit einem

```
# update-rc.d meinpaket defaults
```

wird das Skript `/etc/init.d/meinpaket` in den Runlevels 2, 3, 4 und 5 gestartet und in den Runlevels 0, 1 und 6 gestoppt. Sie können dieses Benehmen über Optionen ändern. Wenn Sie nichts anderes angeben, verwendet `update-rc.d` die Folgenummer 20 zur Einordnung des Diensts in die Startreihenfolge – im Gegensatz zu SUSE und Red Hat ist das nicht automatisiert. – Das `insserv`-Kommando steht bei Debian GNU/Linux als optionales Paket zur Verfügung; ist es installiert, dann kann es zumindest diejenigen Init-Skripte, die die nötigen Metadaten enthalten, wie bei den SUSE-Distributionen automatisch verwalten. Allerdings ist das noch nicht konsequent durchgezogen.

Übungen

17.5 [!2] Was müssen Sie tun, damit der `syslog`-Dienst seine Konfiguration neu einliest?

17.6 [1] Wie können Sie bequem die momentane Runlevel-Konfiguration überprüfen?

17.7 [!2] Entfernen Sie den Dienst `cron` aus Runlevel 2.

Einbenutzermodus

Der Einbenutzermodus Im **Einbenutzermodus** (Runlevel `S`) kann nur der Systemadministrator an der Konsole arbeiten. Die Möglichkeit, mit Alt+Funktionstaste auf andere Konsolen zu wechseln, existiert nicht. Der Einbenutzermodus wird meist für Administrationsaufgaben verwendet, speziell wenn das Filesystem repariert werden muss oder zum Einrichten von Kontingenten (`quota`).

Hierzu kann das Wurzeldateisystem beim Booten nur zum Lesen eingehängt werden. Dazu müssen Sie im Bootlader auf der Kommandozeile des Kernels die Option `S` übergeben. Wenn Sie das System im Einbenutzermodus starten, können Sie die Schreibberechtigung für das Wurzeldateisystem auch »von Hand« ändern. Dazu existieren die Optionen `remount` und `ro`: »`mount -o remount,ro /`« setzt das Dateisystem auf »nur lesbar«. »`mount -o remount,rw /`« macht das Ganze wieder rückgängig.

Um ein Dateisystem im Betrieb auf »nur lesbar« setzen zu können, darf kein Prozess eine Datei im Dateisystem zum Schreiben geöffnet haben. Gegebenenfalls müssen Sie also mit `kill` alle Programme beenden, die noch offene Dateien haben. Das sind in der Regel Daemons wie `syslogd` oder `cron`.

Es hängt von Ihrer Distribution ab, ob und wie Sie aus dem Einbenutzermodus wieder herauskommen.

Debian GNU/Linux empfiehlt zum Verlassen des Einbenutzermodus nicht »`telinit 2`«, sondern einen Systemneustart, da beim Eintritt in den Einbenutzermodus über »`telinit 1`« alle Prozesse entfernt werden, die nicht in den Einbenutzermodus gehören. Dabei verschwinden auch einige Hintergrundprogramme, die im Runlevel `S` gestartet werden und für einen geordneten (Mehrbenutzer-)Systembetrieb notwendig sind, so dass es nicht klug ist, aus dem Runlevel `S` wieder in einen Mehrbenutzer-Runlevel zu wechseln.

Übungen

17.8 [!1] Versetzen Sie das System in den Einbenutzermodus (*Tipp:* `telinit`). Was müssen Sie machen, um tatsächlich in den Einbenutzermodus zu gelangen?

17.9 [1] Überzeugen Sie sich davon, dass Sie im Einbenutzermodus wirklich der einzige Benutzer auf dem Rechner sind und dass keine Hintergrundprozesse laufen.

17.3 Upstart

Während System-V-Init traditionell von einer »synchronen« Philosophie ausgeht – das Init-System ändert seinen Zustand höchstens auf explizite Benutzeranforderung hin, und die bei Zustandsänderungen ausgeführten Schritte, etwa Init-Skripte, werden nacheinander ausgeführt –, verwendet Upstart einen »ereignisgesteuerten« Ansatz. Das heißt, das System soll auf externe Ereignisse (zum Beispiel das Einstecken eines USB-Geräts) reagieren können. Das geschieht »asynchron«. Das Starten und Stoppen von Diensten erzeugt neue Ereignisse, so dass – und das ist einer der wichtigsten Unterschiede zwischen System-V-Init und Upstart – zum Beispiel beim unerwarteten Absturz eines Systemdiensts Upstart den Dienst automatisch neu starten kann. Den System-V-Init würde das dagegen völlig kalt lassen.

Upstart ist mit Absicht so geschrieben, dass es zu System-V-Init kompatibel ist, jedenfalls in dem Sinne, dass Init-Skripte für Dienste identisch weiterverwendet werden können.

Upstart wurde von Scott James Remnant, seinerzeit Angestellter von Canonical (der Firma hinter Ubuntu), entwickelt und debütierte dementsprechend in dieser Distribution. Seit Ubuntu 6.10 (»Edgy Eft«) ist Upstart das Standard-Init-System von Ubuntu, wobei es zunächst in einem System-V-Init-kompatiblen Modus betrieben wurde; seit Ubuntu 9.10 (»Karmic Koala«) wird der »native« Modus benutzt.

Allerdings ist Ubuntu gerade dabei, auf systemd (siehe Kapitel 18) umzuschwenken.

Für die LPIC-1-Zertifizierung müssen Sie ab Version 3.5 (vom 2.7.2012) wissen, dass Upstart existiert und was seine wesentlichen Eigenschaften sind. Details von Konfiguration und Betrieb sind nicht gefragt.

Angeblich soll Upstart auch den Bootvorgang beschleunigen, indem Dienste parallel initialisiert werden können. In der Praxis ist das nicht der Fall, da der limitierende Faktor beim Booten vor allem die Geschwindigkeit ist, mit der Blöcke von Platte ins RAM geschaufelt werden können. Auf der Linux Plumbers Conference 2008 zeigten Arjan van de Ven und Auke Kok, dass es möglich ist, einen Asus EeePC in 5 Sekunden in ein benutzbares System zu booten (also nicht à la Windows in einen Desktop mit rödelnder Festplatte im Hintergrund). Diese Arbeit beruhte auf System-V-Init und nicht Upstart.

Jobs

Die Konfiguration von Upstart basiert auf der Idee sogenannter »Jobs«, die die Rolle von Init-Skripten einnehmen (wobei Init-Skripte, wie gesagt, auch unterstützt werden). Upstart unterscheidet zwischen »Aufgaben« (*tasks*) – Jobs, die nur für begrenzte Zeit laufen und sich dann selbst beenden – und »Diensten« (*services*) – Jobs, die dauerhaft »im Hintergrund« laufen.

Auch Aufgaben können sehr lange laufen. Das wesentliche Unterscheidungskriterium ist, dass Dienste – denken Sie an einen Mail-, Datenbank- oder Web-Server – sich nicht aus eigenem Antrieb beenden, Aufgaben dagegen schon.

Jobs werden über Dateien konfiguriert, die im Verzeichnis `/etc/init` stehen. Die Namen dieser Dateien ergeben sich aus einem Jobnamen und der Endung »`.conf`«. Ein Beispiel für eine solche Datei sehen Sie in Bild 17.2.

```
# rsyslog - system logging daemon
#
# rsyslog is an enhanced multi-threaded replacement for the traditional
# syslog daemon, logging messages from applications

description     "system logging daemon"

start on filesystem
stop on runlevel [06]

expect fork
respawn

exec rsyslogd -c4
```

Bild 17.2: Upstart-Konfigurationsdatei für Job `rsyslog`

Eines der Hauptziele von Upstart ist, die großen Mengen an schablonenhaftem Code zu vermeiden, der die meisten Init-Skripte von System-V-Init kennzeichnet. Die Konfigurationsdatei für Upstart beschränkt sich deshalb darauf, anzugeben, wie der Dienst aufzurufen ist (»`exec rsyslogd -c4`«). Ferner legt sie fest, ob der Dienst nach einem Absturz neu gestartet werden soll (»`respawn`«) und wie Upstart herausfinden kann, welcher Prozess beobachtet werden muss (»`expect fork`« gibt an, dass der `rsyslog`-Prozess sich selbst in den Hintergrund schickt, indem er einen Kindprozess erzeugt und sich dann beendet – auf diesen Kindprozess muss Upstart dann aufpassen). – Vergleichen Sie das mal mit `/etc/init.d/syslogd` (oder so ähnlich) bei einem typischen System-V-Init-basierten Linux.

Während bei »klassischem« System-V-Init der Systemverwalter explizit »global« die Reihenfolge vorgeben muss, in der die Init-Skripte für einen bestimmten Runlevel ausgeführt werden, bestimmen bei Upstart die Jobs »lokal«, wo sie sich im Gefüge allfälliger Abhängigkeiten einordnen. Die »`start on` …«- und »`stop on` …«-Zeilen geben an, welche Ereignisse dazu führen, dass der Job gestartet oder angehalten wird. In unserem Beispiel wird `rsyslog` gestartet, sobald das Dateisystem zur Verfügung steht, und gestoppt, wenn das System in die »Runlevels« 0 (Halt) oder 6 (Neustart) übergeht. Die Runlevel-Verzeichnisse mit symbolischen Links von System-V-Init sind damit überflüssig.

Upstart unterstützt Runlevels vor allem aus Kompatibilität zur Tradition und um die Migration von System-V-Init-basierten Systemen zu Upstart zu erleichtern. Nötig wären sie im Prinzip nicht, aber im Moment werden sie noch gebraucht, um das System herunterzufahren (!).

Neuere System-V-Init-Implementierungen versuchen ebenfalls, Abhängigkeiten zwischen Diensten zu realisieren, in dem Sinne, dass Init-Skript X immer erst nach Init-Skript Y ausgeführt werden kann und so weiter. (Dies läuft auf eine automatische Vergabe der Prioritätsnummern in den Runlevel-Verzeichnissen hinaus.) Dafür werden Metadaten herangezogen, die in standardisierten Kommentaren am Anfang der Init-Skripte stehen. Die Möglichkeiten dabei bleiben aber weit hinter denen von Upstart zurück.

Beim Systemstart generiert Upstart das Ereignis `startup`, sobald es seine eigene Initialisierung abgeschlossen hat. Damit können andere Jobs ausgeführt werden. Die komplette Startsequenz ergibt sich aus dem `startup`-Ereignis und aus Ereignissen, die bei der Ausführung weiterer Jobs erzeugt und von anderen Jobs abgewartet werden.

Zum Beispiel stößt bei Ubuntu 10.04 das `startup`-Ereignis die Aufgabe `mountall` an, die die Dateisysteme verfügbar macht. Wenn sie abgeschlos-

sen ist, wird unter anderem das Ereignis `filesystem` erzeugt, das wiederum den Start des Diensts `rsyslog` aus Bild 17.2 auslöst.

Bei Upstart dient das Kommando `initctl` zur Interaktion mit dem `init`-Prozess:

```
# initctl list                                        Welche Jobs laufen gerade?
alsa-mixer-save stop/waiting
avahi-daemon start/running, process 578
mountall-net stop/waiting
rc stop/waiting
rsyslog start/running, process 549
<<<<<
# initctl stop rsyslog                                          Job anhalten
rsyslog stop/waiting
# initctl status rsyslog                                  Wie ist der Status?
rsyslog stop/waiting
# initctl start rsyslog                                    Job wieder starten
rsyslog start/running, process 2418
# initctl restart rsyslog                                     Stop und Start
rsyslog start/running, process 2432
```

Die Kommandos »`initctl stop`«, »`initctl start`«, »`initctl status`« und »`initctl stop`« können Sie auch zu »`stop`«, »`start`«, … abkürzen.

17.4 Herunterfahren des Systems

Sie sollten einen Linux-Rechner nicht einfach ausschalten, da das zu Datenverlusten führen kann – möglicherweise stehen noch Daten im RAM, die eigentlich auf Platte geschrieben gehören, aber noch auf den richtigen Zeitpunkt dafür warten. Außerdem kann es sein, dass Benutzer über das Netz auf dem Rechner angemeldet sind, und es wäre nicht die feine englische Art, sie mit einem unangekündigten Systemhalt oder -neustart zu überraschen. Dasselbe gilt für Benutzer, die über das Netz Dienste verwenden, die der Rechner anbietet.

Es ist selten nötig, einen Linux-Rechner anzuhalten, der eigentlich ununterbrochen laufen sollte. Sie können ungeniert Software installieren oder entfernen und das System auch ziemlich radikal umkonfigurieren, ohne dafür das Betriebssystem neu starten zu müssen. Die einzigen Fälle, wo das wirklich nötig ist, betreffen Änderungen am Kernel (etwa das Einspielen von Sicherheits-Updates) oder den Einbau neuer oder Austausch defekter Hardware innerhalb des Rechnergehäuses.

Am ersten Fall (Kerneländerungen) wird übrigens gearbeitet. Die `kexec`-Infrastruktur erlaubt es, einen zweiten Kernel in den Speicher zu laden und direkt (also ohne den Umweg über einen Systemneustart) in diesen hineinzuspringen. Es ist also gut möglich, dass Sie in Zukunft immer den neuesten Kernel laufen lassen können und dafür nicht mal Ihren Rechner herunterfahren müssen.

Mit der richtigen (teuren) Hardware können Sie auch den zweiten Fall in weiten Teilen ausklammern: Entsprechende Serversysteme gestatten Ihnen den Austausch von CPUs, RAM-Modulen und Platten im laufenden Betrieb.

Es gibt verschiedene Möglichkeiten, das System anzuhalten oder neu zu starten:

- Durch einen beherzten Druck auf den Ein-Aus-Schalter des Systems. Wenn Sie diesen so lange festhalten, bis Ihr Rechner hörbar den Betrieb einstellt, Ein-Aus-Schalter

ist das System ausgeschaltet. Allerdings sollten Sie das nur in Fällen akuter Panik tun (es brennt im Maschinensaal oder es gibt einen plötzlichen Wassereinbruch).

shutdown
- Mit dem Befehl `shutdown`. Dieses Kommando ist die sauberste Form, das System herunterzufahren oder neu zu starten.
- Für System-V-Init: Mit dem Befehl »`telinit 0`« wird der Runlevel 0 angefordert. Dieser ist äquivalent zum Systemhalt.

halt
- Mit dem Kommando `halt`. Dies ist eigentlich eine direkte Aufforderung an den Kernel, das System anzuhalten, bei den meisten Distributionen wird mit `halt` aber in Wirklichkeit »`shutdown -h`« aufgerufen, wenn sich das System nicht schon im Runlevel 0 oder 6 befindet.

reboot

Für Neustarts gibt es das Kommando `reboot`, das sich aber wie `halt` normalerweise auf `shutdown` abstützt. (Eigentlich sind `halt` und `reboot` sogar dasselbe Programm.)

Die betreffenden Kommandos sind alle dem Systemadministrator vorbehalten.

Vielleicht funktioniert auch die Tastenkombination Alt+Strg+Entf, wenn sie in `/etc/inittab` entsprechend konfiguriert ist (siehe Abschnitt 17.1).

Auch grafische Displaymanager bieten oft eine Option zum Herunterfahren oder Neustarten des Systems. Hier ist dann zu konfigurieren, ob dazu das `root`-Kennwort eingegeben werden muss oder nicht.

Schließlich interpretieren moderne PCs einen (kurzen) Druck auf den Ein-Aus-Schalter mitunter als »Fahr mich ordentlich herunter«, nicht als »Lass mich abstürzen«.

Normalerweise werden Sie die zweite Variante verwenden, nämlich das Programm `shutdown`. Es sorgt dafür, dass alle angemeldeten Benutzer von dem bevorstehenden Systemhalt informiert werden, verhindert neue Anmeldungen und führt je nach Option schließlich die erforderlichen Aktionen aus, um das System anzuhalten.

```
# shutdown -h +10
```

fährt zum Beispiel das System nach zehn Minuten herunter. Mit der `-r` wird das System neu gestartet. Ohne Angabe einer Option finden Sie sich nach Ablauf der angegebenen Zeit im Einbenutzermodus wieder.

Sie können den Zeitpunkt des Systemhalts/Neustarts auch als absolute Zeit angeben:

```
# shutdown -h 12:00                                    High Noon
```

Das Schlüsselwort `now` ist bei `shutdown` ein anderer Ausdruck für »`+0`« – sofortige Aktion. Tun Sie das nur, wenn Sie sicher sind, dass niemand das System anderweitig verwendet.

Folgendes passiert beim Auslösen des Kommandos `shutdown` genau:

1. Alle Benutzer bekommen die Mitteilung, dass und wann das System heruntergefahren wird.
2. Das `shutdown`-Kommando legt automatisch die Datei `/etc/nologin` an, die von `login` (genauer gesagt der PAM-Infrastruktur) gefunden wird; ihre Existenz verhindert, dass Anwender (außer `root`) sich anmelden können.


Benutzer, die das System abblitzen läßt, bekommen zum Trost den Inhalt der Datei `/etc/nologin` angezeigt.

Die Datei wird in der Regel beim Systemstart automatisch wieder gelöscht.

3. Es findet ein Wechsel in den Runlevel 0 bzw. 6 statt. Dadurch werden alle Dienste über ihre Init-Skripte beendet (genauer gesagt, alle Dienste, die in Runlevel 0 bzw. 6 nicht vorkommen, und das sind in der Regel alle …).
4. Alle noch laufenden Prozesse erhalten zuerst das Signal `SIGTERM`. Die Programme fangen ggf. dieses Signal ab und schließen ihre Aktivität ordentlich ab, bevor sie sich beenden.
5. Eine kurze Zeitspanne später werden alle bis dahin nicht beendeten Prozesse »mit Gewalt« durch das Signal `SIGKILL` beendet.
6. Die Dateisysteme werden ausgehängt und die Swapbereiche deaktiviert.
7. Zum Schluß werden alle Aktivitäten des Systems beendet und gegebenenfalls der Warmstart ausgelöst oder der Rechner über APM oder ACPI ausgeschaltet. Funktioniert das nicht, erscheint auf der Konsole die Meldung »`System halted`«. Ab diesem Moment können Sie selbst zum Schalter greifen.


Sie können `shutdown` nach der Anhaltezeit weiteren Text übergeben, den die Benutzer dann mit angezeigt bekommen:

```
# shutdown -h 12:00 '
Systemhalt wegen Hardwareaufrüstung.
Sorry für die Störung!
'
```


Wenn Sie `shutdown` ausgeführt und es sich dann doch noch anders überlegt haben, können Sie einen ausstehenden Systemhalt oder -neustart mit

```
# shutdown -c "Doch kein Systemhalt"
```

wieder stornieren (den Erklärungstext dürfen Sie natürlich frei wählen.)

Übrigens: Den Mechanismus, den `shutdown` benutzt, um Benutzer über einen bevorstehenden Systemhalt (oder Ähnlichem) zu benachrichtigen, können Sie auch anderweitig verwenden. Das Kommando heißt `wall` (kurz für »*write to all*«): `wall`

```
$ wall "Um 15 Uhr Kuchen im Pausenraum!"
```

liefert auf den Terminals aller angemeldeten Benutzer eine Meldung der Form

```
Broadcast message from hugo@red (pts/1) (Sat Jul 18 00:35:03 2015):

Um 15 Uhr Kuchen im Pausenraum!
```


Naja, wenn Sie die Meldung als normaler Benutzer abschicken, bekommen sie alle Benutzer, die nicht mit »`mesg n`« ihr Terminal für solche Nachrichten gesperrt haben. Wenn Sie auch solche Benutzer erreichen wollen, müssen Sie die Nachricht als `root` abschicken.


Selbst wenn Sie nicht auf einem Textterminal angemeldet sind, sondern in einer grafischen Arbeitsumgebung: Die heutigen Arbeitsumgebungen schnappen solche Nachrichten auf und zeigen sie Ihnen in einem Extra-Fensterchen (oder so; das kommt auch auf die Arbeitsumgebung an).

Wenn Sie `root` sind und der Parameter von `wall` aussieht wie der Name einer existierenden Datei, dann wird die Datei gelesen und deren Inhalt als Nachricht verschickt:

```
# echo "Um 15 Uhr Kuchen im Pausenraum!" >kuchen.txt
# wall kuchen.txt
```

Als normaler Benutzer dürfen Sie das nicht, aber Sie können `wall` die Nachricht auf der Standardeingabe übergeben. (Als `root` können Sie das natürlich auch.) Verwenden Sie das nicht für *Krieg und Frieden*.

Wenn Sie `root` sind, können Sie mit der Option `-n` (kurz für `--nobanner`) die Überschriftenzeile »`Broadcast message` …« unterdrücken.

Übungen

17.10 [!2] Fahren Sie Ihr System von jetzt an in fünfzehn Minuten herunter und teilen Sie den Benutzern mit, dass das Ganze nur ein Test ist. Wie verhindern Sie dann den tatsächlichen Shutdown (damit das ganze wirklich nur ein Test ist)?

17.11 [1] Was passiert, wenn Sie (als `root`) `wall` den Namen einer nicht existierenden Datei als Parameter übergeben?

17.12 [2] `wall` ist eigentlich ein Sonderfall des Kommandos `write`, mit dem Sie auf unsäglich primitive Weise mit anderen Benutzern auf demselben Rechner »chatten« können. Probieren Sie `write` aus, im einfachsten Fall zwischen zwei unterschiedlichen Benutzern in verschiedenen Fenstern oder Konsolen. (`write` war interessanter, als man noch eine VAX mit 30 Terminals hatte.)

Kommandos in diesem Kapitel

`chkconfig` Schaltet Systemdienste ein oder aus (SUSE, Red Hat) `chkconfig`(8) 293

`halt` Fährt das System herunter `halt`(8) 298

`initctl` Zentrales Steuerungsprogramm für Upstart `initctl`(8) 297

`insserv` Aktiviert oder deaktiviert Init-Skripte (SUSE) `insserv`(8) 293

`mesg` Schaltet `wall`-Nachrichten für ein Terminal ein oder aus `mesg`(1) 299

`reboot` Startet den Rechner neu `reboot`(8) 298

`runlevel` Zeigt den vorigen und den aktuellen Runlevel an `runlevel`(8) 292

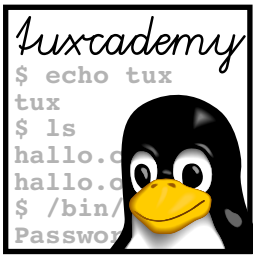
`shutdown` Fährt das System herunter oder startet es neu, mit Verzögerung und Warnung an angemeldete Benutzer `shutdown`(8) 297

`update-rc.d` Installiert und entfernt System-V-Initskript-Links (Debian) `update-rc.d`(8) 293

`wall` Schreibt eine Nachricht auf die Terminals aller angemeldeten Benutzer `wall`(1) 299

Zusammenfassung

- Nach dem Start initialisiert der Systemkern das System und übergibt dann die Kontrolle an das Programm `/sbin/init` als ersten Benutzerprozess.
- Der `init`-Prozess steuert das System und kümmert sich insbesondere um die Aktivierung der im Hintergrund laufenden Dienste und die Verwaltung von Terminals, virtuellen Konsolen und Modems.
- Die Datei `/etc/inittab` enthält die Konfiguration von `init`.
- Das System unterscheidet verschiedene »Runlevels« (Betriebszustände), die sich durch unterschiedliche Mengen von laufenden Diensten definieren.
- Für größere Systemarbeiten steht der Einbenutzermodus zur Verfügung.
- Das Kommando `shutdown` dient zum komfortablen (und für andere Benutzer freundlichen) Herunterfahren oder Neustarten des Systems.
- Mit dem Kommando `wall` können Sie eine Nachricht an alle (angemeldeten) Benutzer schicken.
- Linux-Systeme müssen selten neu gestartet werden – eigentlich nur, wenn ein neuer Systemkern oder neue Hardware installiert worden ist.



18

Systemd

Inhalt

18.1 Überblick 304
18.2 Unit-Dateien 306
18.3 Typen von Units 310
18.4 Abhängigkeiten 311
18.5 Ziele 312
18.6 Das Kommando `systemctl` 315
18.7 Installation von Units 318

Lernziele

- Die systemd-Infrastruktur verstehen
- Den Aufbau von Unit-Dateien kennen
- Ziele (*targets*) verstehen und konfigurieren können

Vorkenntnisse

- Kenntnisse der Linux-Systemadministration
- Wissen über Vorgänge beim Systemstart (Kapitel 16)
- Wissen über System-V-Init (Kapitel 17)

18.1 Überblick

Systemd von Lennart Poettering und Kay Sievers ist eine weitere Alternative zum althergebrachten System-V-Init-System. Ähnlich wie Upstart löst systemd sich von den starren Vorgaben von System-V-Init, aber setzt verschiedene Konzepte für die Aktivierung und Kontrolle von Diensten mit wesentlich größerer Konsequenz um als Upstart.

Systemd wird von allen wesentlichen Linux-Distributionen als zukünftiges Standard-Init-System angesehen. Bei vielen – etwa Debian, Fedora, RHEL, CentOS, openSUSE und SLES – ist es mittlerweile vorinstalliert. Sogar Ubuntu, eigentlich Haupturheber von Upstart, hat sich inzwischen für systemd erklärt.

Systemd und Abhängigkeiten

Während System-V-Init und Upstart explizite Abhängigkeiten von Diensten untereinander ausnutzen – Dienste zum Beispiel, die den Systemprotokolldienst benutzen, können erst gestartet werden, wenn der Systemprotokolldienst läuft –, dreht systemd die Abhängigkeiten um: Ein Dienst, der den Systemprotokolldienst voraussetzt, tut das ja nicht, weil der Protokolldienst laufen muss, sondern weil er selbst Protokollnachrichten loswerden möchte. Dazu muss er auf den Kommunikationskanal zugreifen, über den der Systemprotokolldienst erreicht werden kann. Es reicht also völlig aus, wenn *systemd selbst* diesen Kommunikationskanal anlegt und ihn an den Systemprotokolldienst weiterreicht, sobald dieser tatsächlich zur Verfügung steht – der Dienst, der protokollieren möchte, wartet, bis seine Nachrichten tatsächlich angenommen werden können. Systemd kann also im Wesentlichen erst alle Kommunikationskanäle erzeugen und dann alle Dienste ohne Rücksicht auf Abhängigkeiten gleichzeitig starten. Die Abhängigkeiten regeln sich – ganz ohne explizite Regeln in der Konfiguration – von selbst.

Dieser Ansatz funktioniert auch später im Betrieb: Wenn ein Dienst angesprochen wird, der gerade nicht läuft, kann systemd ihn bei Bedarf starten.

Derselbe Ansatz kann grundsätzlich auch für Dateisysteme benutzt werden: Wenn ein Dienst eine Datei auf einem Dateisystem öffnen möchte, das aktuell nicht zur Verfügung steht, wird der Zugriff aufgehalten, bis das Dateisystem tatsächlich angesprochen werden kann.

Units
Ziele

Systemd verwendet »Units« als Abstraktion für zu verwaltende Systemaspekte wie Dienste, Kommunikationskanäle oder Geräte. Sogenannte Ziele (*targets*) treten an die Stelle der Runlevel von SysV-Init und dienen zur Zusammenfassung verschiedener Units. Zum Beispiel gibt es ein Ziel `multiuser.target`, das dem Runlevel 3 im »klassischen« Schema entspricht. Ziele können von der Verfügbarkeit von Geräten abhängen – zum Beispiel könnte ein Ziel namens `bluetooth.target` angefordert werden, wenn ein USB-Bluetooth-Adapter eingesteckt wird, und die nötige Software dafür starten. (System-V-Init startet die Bluetooth-Software, sofern sie konfiguriert ist, unabhängig davon, ob tatsächlich Bluetooth-Hardware zur Verfügung steht.)

Außerdem hat systemd noch weitere interessante Eigenschaften, die System-V-Init und Upstart nicht zu bieten haben. Unter anderem:

- Systemd unterstützt die Aktivierung von Diensten »bei Bedarf«, nicht nur in Abhängigkeit von erkannter Hardware (wie beim Bluetooth-Beispiel oben), sondern zum Beispiel auch über Netzwerkverbindungen, D-Bus-Anfragen oder die Verfügbarkeit bestimmter Pfade im Dateisystem.
- Systemd erlaubt eine sehr umfangreiche Kontrolle der gestarteten Dienste, etwa was die Prozessumgebung, Ressourcenlimits und Ähnliches angeht. Dazu gehören auch Sicherheitsverbesserungen, etwa indem Dienste nur eingeschränkten Zugriff auf das Dateisystem erhalten oder ein privates `/tmp`-Verzeichnis oder eine private Netzwerkumgebung zugeordnet bekommen.


Bei System-V-Init läßt sich das fallweise über die Init-Skripte abhandeln, aber das ist im Vergleich sehr primitiv, mühselig einzurichten und unbequem zu warten.

- Systemd verwendet den `cgroups`-Mechanismus des Linux-Kerns, um sicherzustellen, dass zum Beispiel beim Stoppen von Diensten alle dazugehörigen Prozesse beendet werden.
- Systemd kümmert sich auf Wunsch um die Protokollausgabe von Diensten; es reicht, wenn diese auf ihre Standardausgabe schreiben.
- Systemd erleichtert die Wartung von Konfigurationen, indem Voreinstellungen der Distribution und lokale Anpassungen sauber getrennt werden.
- Systemd enthält eine Menge in C geschriebene Werkzeuge, die sich um die Systeminitialisierung kümmern und im Wesentlichen das tun, was sonst mit distributionsspezifischen Shellskripten im Runlevel S erledigt wird. Ihre Verwendung kann den Systemstart deutlich beschleunigen und trägt in diesem Bereich zu einer distributionsübergreifenden Standardisierung bei.

Kompatibilität

Systemd ist auf maximale Kompatibilität zu System-V-Init und anderen »Traditionen« getrimmt. Zum Beispiel unterstützt er die Init-Skripte von System-V-Init, wenn für einen Dienst keine systemd-eigene Konfigurationsdatei zur Verfügung steht, oder entnimmt die beim Systemstart einzuhängenden Dateisysteme der Datei `/etc/fstab`.

`systemctl`

Sie können das Kommando `systemctl` verwenden, um mit einem laufenden systemd zu interagieren, etwa um Dienste gezielt zu starten oder anzuhalten:

```
# systemctl status rsyslog.service
● rsyslog.service - System Logging Service
   Loaded: loaded (/lib/systemd/system/rsyslog.service; enabled)
   Active: active (running) since Do 2015-07-16 15:20:38 CEST;▷
    ◁ 3h 12min ago
     Docs: man:rsyslogd(8)
           http://www.rsyslog.com/doc/
 Main PID: 497 (rsyslogd)
   CGroup: /system.slice/rsyslog.service
           └─497 /usr/sbin/rsyslogd -n
# systemctl stop rsyslog.service
Warning: Stopping rsyslog.service, but it can still be activated by:
  syslog.socket
# systemctl start rsyslog.service
```

Solche Änderungswünsche für den Systemzustand bezeichnet systemd als »Jobs« und ordnet sie in eine Warteschlange ein.


Transaktionen

Systemd betrachtet Anfragen nach Zustandsänderungen als »Transaktionen«. Wenn eine Unit gestartet oder angehalten werden soll, wird sie (und allfällige von ihr abhängige Units) einer temporären Transaktion hinzugefügt. Anschließend prüft systemd, ob die Transaktion konsistent ist – insbesondere, dass keine kreisförmigen Abhängigkeiten existieren. Ist dies nicht der Fall, versucht er, die Transaktion zu reparieren, indem nicht zwingend nötige Jobs entfernt werden, um die Schleife(n) aufzubrechen. Auch nicht zwingend nötige Jobs, die dazu führen würden, dass laufende Dienste angehalten werden, werden aussortiert. Schließlich prüft systemd, ob die Jobs in der Transaktion bereits in der Warteschlange stehenden Jobs widersprechen, und lehnt die Transaktion möglicherweise ab. Nur falls die Transaktion konsistent ist und die Minimierung ihrer Auswirkungen auf das System erfolgreich war, werden ihre Jobs in die Warteschlange eingegliedert.

Übungen

18.1 [!1] Benutzen Sie »`systemctl status`«, um sich ein Bild von den auf Ihrem Rechner aktiven Units zu machen. Rufen Sie den detaillierten Zustand einiger interessant aussehender Units auf.

18.2 Unit-Dateien

Einer der gravierenden Vorteile von systemd ist, dass er ein einheitliches Dateiformat für alle Konfigurationsdateien verwendet – egal, ob es um zu startende Dienste, Geräte, Kommunikationskanäle, Dateisysteme oder andere von systemd verwaltete Artefakte geht.

Dies im Gegensatz zur traditionellen System-V-Init-basierten Infrastruktur, wo fast jede Funktionalität anders konfiguriert wird: Permanent laufende Hintergrunddienste in `/etc/inittab`, Runlevels und die Dienste darin über Init-Skripte, Dateisysteme in `/etc/fstab`, bei Bedarf zu aktivierende Dienste in `/etc/inetd.conf`, … Jede dieser Dateien ist syntaktisch verschieden von allen anderen, während bei systemd nur die Details der möglichen (und sinnvollen) Konfigurationseinstellungen voneinander abweichen – das Dateiformat ist jeweils dasselbe.

Eine sehr wichtige Beobachtung ist: Unit-Dateien sind »deklarativ«. Das heißt, sie erklären nur, wie die gewünschte Konfiguration *aussieht* – im Gegensatz zu den Init-Skripten von System-V-Init, die ausführbaren Code enthalten, der versucht, die gewünschte Konfiguration *herzustellen*.

Init-Skripte bestehen typischerweise aus jeder Menge schablonenhaft aussehendem Code, der von der jeweiligen Distribution abhängt und den Sie trotzdem Zeile für Zeile lesen und verstehen müssen, wenn Sie ein Problem haben oder etwas Besonderes erreichen möchten. Für etwas komplexere Hintergrunddienste sind Init-Skripte von hundert Zeilen oder mehr nicht ungewöhnlich. Unit-Dateien für systemd dagegen kommen normalerweise ohne Weiteres mit einem Dutzend Zeilen oder zwei aus, und diese Zeilen sind in der Regel ziemlich leicht zu verstehen.

Natürlich enthalten auch Unit-Dateien hin und wieder Shell-Kommandos, etwa um zu erklären, wie ein bestimmter Dienst gestartet oder gestoppt werden soll. Allerdings sind das allermeistens relativ offensichtliche Einzeiler.

Syntax Die grundlegende Syntax von Unit-Dateien ist in `systemd.unit`(5) erklärt. Ein Beispiel für eine Unit-Datei finden Sie in Bild 18.1. Typisch ist die Einteilung in Abschnitte, die mit einem Titel in eckigen Klammern anfangen[1]. Alle Unit-Dateien (egal wofür sie gedacht sind) können `[Unit]`- und `[Install]`-Abschnitte haben (hierzu gleich mehr). Dazu kommen Abschnitte, die vom Einsatzzweck der Unit-Datei abhängen.

Wie üblich werden Leerzeilen und Kommentarzeilen ignoriert. Kommentarzeilen können mit `#` oder `;` anfangen. Überlange Zeilen können mit einem `\` am Zeilenende umbrochen werden, der beim Einlesen durch ein Leerzeichen ersetzt wird. Groß- und Kleinschreibung ist wichtig!

Optionen

Zeilen, die keine Abschnittstitel, Leerzeilen oder Kommentarzeilen sind, enthalten Einstellungen von »Optionen« in der Form »⟨*Name*⟩ = ⟨*Wert*⟩«. Manche Optionen dürfen mehrmals auftreten, und dann kommt es auf die Option an, wie systemd damit umgeht: Mehrfachoptionen bilden oft eine Liste; wenn Sie einen leeren Wert angeben, werden alle vorigen Einstellungen ignoriert. (Wenn das so ist, steht das in der Dokumentation.)

[1] Die Syntax ist inspiriert von den `.desktop`-Dateien der »XDG Desktop Entry Specification« [XDG-DS14], die wiederum von den INI-Dateien von Microsoft Windows inspiriert sind.

```
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify
#  it under the terms of the GNU Lesser General Public License as
#  published by the Free Software Foundation; either version 2.1
#  of the License, or (at your option) any later version.

[Unit]
Description=Console Getty
Documentation=man:agetty(8)
After=systemd-user-sessions.service plymouth-quit-wait.service
After=rc-local.service
Before=getty.target

[Service]
ExecStart=-/sbin/agetty --noclear --keep-baud console ▷
 ◁ 115200,38400,9600 $TERM
Type=idle
Restart=always
RestartSec=0
UtmpIdentifier=cons
TTYPath=/dev/console
TTYReset=yes
TTYVHangup=yes
KillMode=process
IgnoreSIGPIPE=no
SendSIGHUP=yes

[Install]
WantedBy=getty.target
```

Bild 18.1: Eine systemd-Unit-Datei: `console-getty.service`


Optionen, die nicht in der Dokumentation stehen, werden von systemd mit einer Warnung angemeckert und ansonsten überlesen. Wenn ein Abschnitts- oder Optionsname mit »X-« anfängt, wird er komplett ignoriert (Optionen in einem »X-«-Abschnitt brauchen selber kein »X-«-Präfix).


Ja/Nein-Einstellungen in Unit-Dateien können auf verschiedene Weise vorgenommen werden. `1`, `true`, `yes` und `on` stehen für »Ja«, `0`, `false`, `no` und `off` für »Nein«.


Zeitangaben können Sie ebenfalls auf verschiedene Arten machen. Einfache ganze Zahlen werden als Sekunden interpretiert[2]. Wenn Sie eine Einheit anhängen, gilt die Einheit (erlaubt sind unter anderem `us`, `ms`, `s`, `min`, `h`, `d`, `w` in ansteigender Folge von Mikrosekunden bis Wochen – siehe `systemd.time`(7)). Sie können mehrere Zeitangaben mit Einheiten aneinanderhängen (à la »`10min 30s`«), und dann werden die Zeiten addiert (hier 630 Sekunden).

Einstellungen suchen und finden Systemd sucht Unit-Dateien entlang einer Liste von Verzeichnissen, die standardmäßig hart ins Programm codiert ist. Verzeichnisse weiter vorne in der Liste haben Vorrang gegenüber Verzeichnissen weiter hinten.


Die Details sind bis zu einem gewissen Grad systemabhängig. Die gängige Liste ist normalerweise etwas wie

`/etc/systemd/system`	*Lokale Konfiguration*
`/run/systemd/system`	*Dynamisch erzeugte Unit"=Dateien*
`/lib/systemd/system`	*Unit"=Dateien für Distributionspakete*

Lokale Anpassungen

Systemd hat diverse clevere Möglichkeiten, mit denen Sie Einstellungen anpassen können, ohne dass Sie die (in der Regel von Ihrer Distribution zur Verfügung gestellten) Unit-Dateien ändern müssen – was unbequem wäre, wenn die Distribution die Unit-Dateien aktualisiert. Stellen Sie sich zum Beispiel vor, Sie möchten einige Einstellungen in der Datei `example.service` modifizieren:

- Sie können die Datei `example.service` der Distribution von `/lib/systemd/system` nach `/etc/systemd/system` kopieren und die gewünschten Änderungen anpassen. Die Unit-Datei der Distribution wird dann überhaupt nicht mehr angeschaut.
- Sie können ein Verzeichnis `/etc/systemd/system/example.service.d` anlegen und dort eine Datei – zum Beispiel `local.conf` – platzieren. Die Einstellungen in dieser Datei übersteuern gezielt gleichnamige Einstellungen in `/lib/systemd/system/example.service`, wobei alle Einstellungen, die nicht in `local.conf` stehen, erhalten bleiben.

  
  Achten Sie darauf, dass `local.conf` alle nötigen Abschnittstitel enthält, damit die Einstellungen korrekt zugeordnet werden können.

  
  Es hindert Sie niemand daran, in `/etc/systemd/system/example.service.d` mehrere Dateien abzulegen. Die einzige Vorbedingung ist, dass die Dateinamen auf `.conf` enden müssen. Systemd macht keine Aussagen darüber, in welcher Reihenfolge die Dateien angeschaut werden – am besten sorgen Sie also dafür, dass jede Einstellung nur in einer einzigen Datei vorkommt.

[2] Allermeistens jedenfalls – es gibt (dokumentierte) Ausnahmen.

Schablonen-Unit-Dateien Mitunter können mehrere Dienste dieselbe oder eine sehr ähnliche Unit-Datei benutzen. In diesem Fall ist es bequem, nicht mehrere Kopien derselben Unit-Datei warten zu müssen. Denken Sie zum Beispiel an die Terminal-Definitionszeilen in `/etc/inittab` – es wäre gut, hier nicht eine Unit-Datei pro Terminal haben zu müssen.

Systemd unterstützt dies durch Unit-Dateien mit Namen wie `example@.service`. Sie könnten zum Beispiel eine Datei `getty@.service` haben und anschließend über ein symbolisches Link von `getty@tty2.service` auf `getty@.service` eine virtuelle Konsole auf `/dev/tty2` konfigurieren. Wenn diese Konsole aktiviert werden soll, liest systemd die Datei `getty@.service` und ersetzt überall die Zeichenkette `%I` durch das, was im Dienstnamen zwischen `@` und `.` steht, also `tty2`. Das Resultat der Ersetzung wird dann als Konfiguration in Kraft gesetzt.

Instantiierung

Tatsächlich ersetzt systemd nicht nur `%I`, sondern auch noch einige andere Sequenzen (und das nicht nur in Schablonen-Unit-Dateien). Die Details stehen in `systemd.unit`(5), im Abschnitt »Specifiers«.

Grundlegende Einstellungen Alle Unit-Dateien können die Abschnitte `[Unit]` und `[Install]` haben. Der erstere enthält allgemeine Informationen über die Unit, der letztere liefert Details für die Installation der Unit (etwa um explizite Abhängigkeiten einzuführen – darüber reden wir später).

Hier sind einige der wichtigsten Optionen aus dem `[Unit]`-Abschnitt (die komplette Liste steht in `systemd.unit`(5)):

`Description` Eine Beschreibung der Unit (als freier Text). Wird verwendet, um Benutzungsoberflächen freundlicher zu machen.

`Documentation` Eine durch Leerzeichen getrennte Liste von URLs mit Dokumentation für die Unit. Erlaubt sind die Protokollschemata `http:`, `https:`, `file:`, `info:` und `man:` (die letzteren drei verweisen auf lokale Dokumentation). Ein leerer Wert leert die Liste komplett.

`OnFailure` Eine durch Leerzeichen getrennte Liste von anderen Units, die aktiviert werden, wenn diese Unit in den `failed`-Zustand übergeht.

`SourcePath` Der Pfadname einer Konfigurationsdatei, aus der diese Unit-Datei generiert wurde. Dies ist interessant für Werkzeuge, die aus externen Konfigurationsdateien Unit-Dateien für systemd erzeugen.

`ConditionPathExists` Prüft, ob unter dem angegebenen absoluten Pfadnamen eine Datei (oder ein Verzeichnis) existiert. Wenn nein, wird die Unit als `failed` geführt. Wenn vor dem Pfadnamen ein »`!`« steht, dann darf unter dem Namen *keine* Datei (bzw. kein Verzeichnis) zu finden sein. (Es gibt noch eine ganze Menge andere »`Condition…`«-Tests – zum Beispiel können Sie die Ausführung von Units davon abhängig machen, ob das System eine bestimmte Rechnerarchitektur hat oder in einer virtuellen Umgebung läuft, ob das System mit Netz- oder Batteriestrom oder auf einem Rechner mit einem bestimmten Namen läuft und so weiter. Lesen Sie in `systemd.unit`(5) nach.)

Übungen

18.2 [!2] Stöbern Sie in den Unit-Dateien Ihres Systems unter `/lib/systemd/system` (oder `/usr/lib/systemd/system`, je nach Distribution). Wie viele verschiedene `Condition…`-Optionen können Sie finden?

18.3 Typen von Units

Systemd unterstützt eine breite Auswahl von »Units«, also Systemkomponenten, die er verwalten kann. Am leichtesten können Sie sie anhand der Dateiendungen auseinanderhalten, die die dazugehörigen Unit-Dateien haben. Wie in Abschnitt 18.2 erwähnt, teilen alle Arten von Units sich das gleiche grundlegende Dateiformat. Hier ist eine Aufzählung der wichtigsten Unit-Typen:

`.service` Ein Prozess auf dem Rechner, der von systemd ausgeführt und kontrolliert wird. Dies umfasst sowohl Hintergrunddienste, die über längere Zeit (möglicherweise die komplette Laufzeit des Rechners) aktiv bleiben, als auch einmalig ausgeführte Prozesse (etwa während des Systemstarts).

Wenn ein Dienst unter einem bestimmten Namen (etwa `example`) angefordert wird, aber keine passende Unit-Datei (hier `example.service`) gefunden wird, sucht systemd nach einem gleichnamigen Init-Skript von System-V-Init und generiert daraus dynamisch eine Service-Unit. (Die Kompatibilität ist sehr weitreichend, aber nicht zu 100% vollständig.)

`.socket` Ein TCP/IP- oder lokales Socket, also ein Kommunikationsendpunkt, mit dem Clientprogramme Kontakt aufnehmen können, um einen Dienst anzusprechen. Systemd verwendet Socket-Units für die Aktivierung von Hintergrunddiensten bei Bedarf.

Socket-Units haben immer eine zugeordnete Service-Unit, die gestartet wird, wenn systemd Aktivität auf dem betreffenden Socket bemerkt.

`.mount` Ein »Mount-Punkt« auf dem System, also ein Verzeichnis, wo ein Dateisystem eingehängt werden soll.

Die Namen dieser Units ergeben sich aus dem Pfadnamen, indem alle Schrägstriche (»/«) in Bindestriche (»-«) und alle anderen nicht (gemäß ASCII) alphanumerischen Zeichen außer »_« in eine hexadezimale Ersatzdarstellung à la `\x2d` umgewandelt werden (».« wird nur umgewandelt, wenn es das erste Zeichen des Pfadnamen ist). Der Name des Wurzelverzeichnisses (»/«) wird zu »-«, aber Schrägstriche am Anfang oder Ende aller anderen Namen werden entfernt. Der Verzeichnisname `/home/lost+found` zum Beispiel wird also zu `home-lost\textbackslash x2bfound`.

Sie können diese Ersetzung mit dem Kommando »`systemd-escape -p`« nachvollziehen:

```
$ systemd-escape -p /home/lost+found
-home-lost\x2bfound
$ systemd-escape -pu home-lost\\x2bfound
/home/lost+found
```

Die Option »`-p`« kennzeichnet den Parameter als Pfadnamen. Die Option »`-u`« macht die Umwandlung rückgängig.

`.automount` Gibt an, dass ein Mount-Punkt bei Bedarf automatisch eingehängt werden soll (statt prophylaktisch beim Systemstart). Die Namen dieser Units ergeben sich auch durch die eben diskutierte Pfadnamen-Transformation. Die Details des Einhängens müssen durch eine korrespondierende Mount-Unit beschrieben werden.

`.swap` Beschreibt Auslagerungsspeicher auf dem System. Die Namen dieser Units ergeben sich durch die Pfadnamen-Transformation angewendet auf die betreffenden Gerätedatei- oder Dateinamen.

.target Ein »Ziel«, also ein Synchronisierungspunkt für andere Units beim Systemstart oder beim Übergang in andere Systemzustände. Vage analog zu den Runlevels von System-V-Init. Siehe Abschnitt 18.5.

.path Beobachtet eine Datei oder ein Verzeichnis und startet eine andere Unit (standardmäßig eine gleichnamige Service-Unit), wenn zum Beispiel Änderungen an der Datei festgestellt werden oder eine Datei in ein ansonsten leeres Verzeichnis geschrieben wird.

.timer Startet eine andere Unit (standardmäßig eine gleichnamige Service-Unit) zu einem bestimmten Zeitpunkt oder wiederholt in bestimmten Intervallen. Damit wird systemd zu einem Ersatz für `cron` und `at`.

(Es gibt noch ein paar andere Unit-Typen, die zu erklären hier zu weit führen würde.)

Übungen

18.3 [!2] Suchen Sie auf Ihrem System nach Beispielen für alle diese Unit-Typen. Schauen Sie sich die Unit-Dateien an. Konsultieren Sie ggf. die Man-Pages für die einzelnen Typen.

18.4 Abhängigkeiten

Wie wir schon erwähnt haben, kommt systemd weitgehend ohne explizite Abhängigkeiten aus, weil er in der Lage ist, implizite Abhängigkeiten (etwa von Kommunikationskanälen) auszunutzen. Trotzdem ist es hin und wieder nötig, explizite Abhängigkeiten anzugeben. Dafür stehen einige Optionen zur Verfügung, die im `[Unit]`-Abschnitt einer Unit-Datei (zum Beispiel `example.service`) angegeben werden können. Zum Beispiel:

Requires Gibt eine Liste anderer Units an. Wenn die aktuelle Unit aktiviert wird, werden die aufgelisteten Units auch aktiviert. Wenn eine der aufgelisteten Units deaktiviert wird oder ihre Aktivierung fehlschlägt, dann wird auch die aktuelle Unit deaktiviert. (Mit anderen Worten, die aktuelle Unit »hängt von den angegebenen Units ab«.)

Die `Requires`-Abhängigkeiten haben nichts damit zu tun, in welcher *Reihenfolge* die Units gestartet oder angehalten werden – das müssen Sie gegebenenfalls separat mit `After` oder `Before` konfigurieren. Wenn keine explizite Reihenfolge angegeben wurde, startet systemd alle Units gleichzeitig.

Sie können solche Abhängigkeiten auch angeben, ohne die Unit-Datei zu ändern, indem Sie ein Verzeichnis `/etc/systemd/system/example.service.requires` kreieren und darin symbolische Links auf die gewünschten Unit-Dateien anlegen. Ein Verzeichnis wie

```
# ls -l /etc/systemd/system/example.service.requires
lrwxrwxrwx 1 root root 34 Jul 17 15:56 network.target -> ▷
 ◁ /lib/systemd/system/network.target
lrwxrwxrwx 1 root root 34 Jul 17 15:57 syslog.service -> ▷
 ◁ /lib/systemd/system/syslog.service
```

entspricht der Einstellung

```
[Unit]
Requires = network.target syslog.service
```

in `example.service`.

Wants Eine abgeschwächte Form von `Requires`. Hier werden die genannten Units zwar zusammen mit der aktuellen Unit gestartet, aber wenn ihre Aktivierung fehlschlägt, hat das keinen Einfluss auf den Gesamtvorgang. Dies ist die empfohlene Methode, den Start einer Einheit vom Start einer anderen Einheit abhängig zu machen.

Auch hier können Sie die Abhängigkeiten »extern« angeben, indem Sie ein Verzeichnis `example.service.wants` anlegen.

Conflicts Die Umkehrung von `Requires` – die hier angegebenen Units werden angehalten, wenn die aktuelle Unit gestartet wird, und umgekehrt.

Wie `Requires` macht `Conflicts` keine Aussage über die Reihenfolge, in der Units gestartet oder angehalten werden.

Wenn eine Unit U mit einer anderen Unit V im Konflikt steht und beide gleichzeitig gestartet werden sollen, dann schlägt der Vorgang fehl, wenn beide Units verbindlicher Teil des Vorgangs sind. Wenn eine (oder auch beide) Units nicht verbindlicher Teil des Vorgangs sind, wird der Vorgang modifiziert: Wenn nur eine Unit nicht verbindlich ist, wird diese nicht gestartet, wenn beide nicht verbindlich sind, wird die im `Conflicts` genannte Unit gestartet und die, deren Unit-Datei die `Conflicts`-Einstellung enthält, gestoppt.

Before (und `After`) Diese Listen von Units bestimmen die Startreihenfolge. Wenn `example.service` die Einstellung »`Before=example2.service`« enthält und beide Units gestartet werden, dann wird der Start von `example2.service` verschoben, bis `example.service` gestartet ist. `After` ist das Gegenstück zu `Before`, das heißt, wenn `example2.service` die Einstellung »`After=example.service`« enthält und beide Units gestartet werden, dann ergibt sich derselbe Effekt – `example2.service` wird zurückgestellt.

Das hat wohlgemerkt nichts mit den Abhängigkeiten von `Requires` oder `Conflicts` zu tun. Es ist üblich, Units zum Beispiel sowohl in `Requires` als auch `After` aufzulisten. Dann wird die aufgelistete Unit vor der Unit gestartet, deren Unit-Datei die Einstellungen enthält.

Beim Deaktivieren der Units wird die umgekehrte Reihenfolge eingehalten. Wird eine Unit mit einer `Before`- oder `After`-Abhängigkeit zu einer anderen Unit deaktiviert, während die andere gestartet wird, findet das Deaktivieren vor dem Aktivieren statt, egal in welche Richtung die Abhängigkeit zeigt. Wenn es keine `Before`- oder `After`-Abhängigkeit zwischen zwei Units gibt, dann werden sie gleichzeitig gestartet oder angehalten.

Übungen

18.4 [!1] Welchen Vorteil versprechen wir uns davon, Abhängigkeiten über symbolische Links in Verzeichnissen wie `example.service.requires` zu konfigurieren anstatt in der Unit-Datei `example.service`?

18.5 [2] Finden Sie in Ihrer Systemkonfiguration Beispiele für `Requires`-, `Wants`- und `Conflicts`-Abhängigkeiten, mit und ohne korrespondierende `Before`- oder `After`-Abhängigkeiten.

18.5 Ziele

Ziele (*targets*) sind für systemd im Großen und Ganzen das, was für System-V-Init die Runlevels sind: Die Möglichkeit, eine Menge von Diensten bequem zu

Tabelle 18.1: Gängige Ziele für systemd (Auswahl)

Ziel	Bedeutung
`basic.target`	Grundlegender Systemstart ist abgeschlossen (Dateisysteme, Auslagerungsspeicher, Sockets, Timer und anderes).
`ctrl-alt-del.target`	Wird ausgeführt, wenn Strg + Alt + Entf gedrückt wurde. Meist dasselbe wie `reboot.target`.
`default.target`	Ziel, das systemd beim Systemstart anstrebt. In der Regel entweder `multi-user.target` oder `graphical.target`.
`emergency.target`	Startet eine Shell auf der Konsole des Rechners. Für Notfälle. Wird normalerweise über die Option »`systemd.unit=emergency.target`« auf der Kernel-Kommandozeile angesteuert.
`getty.target`	Aktiviert die statisch definierten `getty`-Instanzen (für Terminals). Entspricht den `getty`-Zeilen in `/etc/inittab` bei System-V-Init.
`graphical.target`	Etabliert eine grafische Anmeldeaufforderung. Hängt von `multiuser.target` ab.
`halt.target`	Hält das System an (ohne es auszuschalten).
`multi-user.target`	Etabliert ein Mehrbenutzersystem ohne grafische Anmeldeaufforderung. Wird von `graphical.target` benutzt.
`network-online.target`	Dient als Abhängigkeit für Units, die Netzwerkdienste benötigen (*nicht* Netzwerkdienste anbieten), etwa Mount-Units für entfernte Dateisysteme. Wie genau festgestellt wird, ob das Netzwerk zur Verfügung steht, ist von der Methode der Netzwerkkonfiguration abhängig.
`poweroff.target`	Hält das System an und schaltet es aus.
`reboot.target`	Startet das System neu.
`rescue.target`	Führt grundlegende Systeminitialisierung durch und startet eine Shell.

Tabelle 18.2: Kompatibilitäts-Ziele für System-V-Init

Ziele	Äquivalent
`runlevel0.target`	`poweroff.target`
`runlevel1.target`	`rescue.target`
`runlevel2.target`	`multi-user.target` (empfohlen)
`runlevel3.target`	`graphical.target` (empfohlen)
`runlevel4.target`	`graphical.target` (empfohlen)
`runlevel5.target`	`graphical.target` (empfohlen)
`runlevel6.target`	`reboot.target`

beschreiben. Während System-V-Init nur eine relativ kleine Anzahl von Runlevels erlaubt und ihre Konfiguration ziemlich kompliziert ist, ermöglicht systemd die problemlose Definition diverser Ziele.

Unit-Dateien für Ziele haben keine speziellen Optionen (der Standardvorrat aus `[Unit]` und `[Install]` sollte reichen). Ziele dienen nur dazu, andere Units über Abhängigkeiten zusammenzufassen oder standardisierte Namen für Synchronisierungspunkte in Abhängigkeiten festzulegen (`local-fs.target` zum Beispiel kann verwendet werden, um Units, die die lokalen Dateisysteme benötigen, erst dann zu starten, wenn diese zur Verfügung stehen). Eine Übersicht der wichtigsten Ziele steht in Tabelle 18.1

Im Interesse der Rückwärtskompatibilität zu System-V-Init definiert systemd einige Ziele, die mit den klassischen Runlevels korrespondieren. Betrachten Sie hierzu Tabelle 18.2.

Standard-Ziel

Sie können das Standard-Ziel festlegen, das systemd beim Start des Systems anstrebt, indem Sie ein symbolisches Link von `/etc/systemd/system/default.target` auf die Unit-Datei des gewünschten Ziels anlegen:

```
# cd /etc/systemd/system
# ln -sf /lib/systemd/system/multi-user.target default.target
```

(Dies ist das moralische Äquivalent zur `initdefault`-Zeile in der Datei `/etc/inittab` bei System-V-Init.) Eine bequemere Methode ist das Kommando »`systemctl set-default`«:

```
# systemctl get-default
multi-user.target
# systemctl set-default graphical
Removed symlink /etc/systemd/system/default.target.
Created symlink from /etc/systemd/system/default.target to ▷
 ◁ /lib/systemd/system/graphical.target.
# systemctl get-default
graphical.target
```

(Wie Sie sehen können, macht das aber auch nichts Anderes als das symbolische Link umzubiegen.)

Ziel gezielt aktivieren

Um ein bestimmtes Ziel gezielt zu aktivieren (so wie Sie bei System-V-Init gezielt in einen bestimmten Runlevel wechseln können), verwenden Sie das Kommando »`systemctl isolate`«:

```
# systemctl isolate multi-user
```

(bei Bedarf wird »`.target`« an den Parameter angehängt). Dieses Kommando startet alle Units, von denen das Ziel abhängt, und hält alle anderen Units an.

»`systemctl isolate`« funktioniert nur für Units, in deren `[Unit]`-Abschnitt »`AllowIsolate`« eingeschaltet ist.

Zum Anhalten des Systems und zum Wechseln in den Rescue-Modus (Anhänger des System-V-Init würden »Einbenutzermodus« dazu sagen) gibt es die Abkürzungen

```
# systemctl rescue
# systemctl halt
# systemctl poweroff                    Wie halt, aber mit Ausschalten
# systemctl reboot
```

Diese Kommandos entsprechen im Wesentlichen ihren Äquivalenten mit »systemctl isolate«, aber geben außerdem eine Warnung an allfällige Benutzer aus. Sie können (und sollten!) natürlich trotzdem das Kommando shutdown benutzen.

Zurück in die Standardbetriebsart kommen Sie mit

```
# systemctl default
```

Übungen

18.6 [!2] Von welchen anderen Units hängt das multi-user.target ab? Hängen diese Units wiederum von anderen Units ab?

18.7 [2] Verwenden Sie »systemctl isolate«, um Ihr System in den Rescue-Modus (Einbenutzermodus) zu versetzen, und »systemctl default«, um zurück in den Standardmodus zu kommen. (*Tipp:* Machen Sie das von einer Textkonsole aus.)

18.8 [2] Starten Sie Ihr System mit »systemctl reboot« neu und dann anschließend nochmal mit shutdown. Würdigen Sie den Unterschied.

18.6 Das Kommando systemctl

Zur Steuerung von systemd dient das Kommando systemctl. Ein paar Anwendungen haben wir schon gesehen, hier kommt jetzt eine etwas systematischere Liste. Das ist allerdings immer noch nur ein kleiner Ausschnitt der Möglichkeiten.

Die allgemeine Struktur von systemctl-Aufrufen ist

```
# systemctl ⟨Unterkommando⟩ ⟨Parameter⟩ …
```

systemctl unterstützt einen ziemlich großen Zoo von Unterkommandos. Welche Parameter (und Optionen) erlaubt sind, hängt vom jeweiligen Unterkommando ab.

Oft werden Unit-Namen als Parameter erwartet. Diese können Sie entweder mit der Endung angeben (also zum Beispiel example.service) oder ohne (example). Im letzteren Fall hängt systemd eine Endung an, die ihm passend erscheint – beim Kommando start zum Beispiel ».service«, beim Kommando isolate dagegen ».target«. Unit-Namen als Parameter

Kommandos für Units Die folgenden Unterkommandos beschäftigen sich mit Units und ihrer Verwaltung:

list-units Zeigt die Units an, die systemd kennt. Sie können mit der Option -t einen Unit-Typ (service, socket, …) oder eine durch Kommas getrennte Liste von Unit-Typen angeben und die Ausgabe damit auf Units der betreffenden Typen einschränken. Sie können auch ein Shell-Suchmuster übergeben und damit gezielt nach bestimmten Units suchen:

```
# systemctl list-units "ssh*"
UNIT        LOAD   ACTIVE SUB     DESCRIPTION
ssh.service loaded active running OpenBSD Secure Shell server

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization
         of SUB.
SUB    = The low-level unit activation state, values depend on
         unit type.

1 loaded units listed. Pass --all to see loaded but inactive units,
too. To show all installed unit files use 'systemctl
list-unit-files'.
```

Wie üblich sind Anführungszeichen hier eine gute Idee, damit die Shell nicht über die Suchmuster herfällt, die für systemd gemeint sind.

start Startet eine oder mehrere als Parameter benannte Units.

Hier können Sie auch Shell-Suchmuster verwenden. Die Suchmuster wirken allerdings nur auf Units, die systemd kennt; inaktive Units, die nicht in einem Fehlerstatus sind, werden also nicht erfasst, ebensowenig wie aus Schablonen instantiierte Units, deren genauer Name vor der Instantiierung nicht bekannt ist. Sie sollten die Suchmuster also nicht überstrapazieren.

stop Stoppt eine oder mehrere als Parameter benannte Units (auch wieder mit Suchmustern).

reload Lädt die Konfiguration für die aufgezählten Units neu (falls die den Units zugrundeliegenden Programme mitspielen). Suchmuster sind erlaubt.

Hier geht es um die Konfiguration der Hintergrunddienste selbst, nicht die Konfiguration der Dienste aus der Sicht von systemd. Wenn Sie möchten, dass systemd seine eigene Konfiguration in Bezug auf die Hintergrunddienste neu lädt, müssen Sie das Kommando »`systemctl daemon-reload`« verwenden.

Was bei »`systemctl reload`« konkret passiert, hängt vom betreffenden Hintergrunddienst ab (meistens wird `SIGHUP` verschickt). Sie können das in der Unit-Datei für den Dienst konfigurieren.

restart Startet die aufgezählten Units neu (Suchmuster sind erlaubt). Wenn eine Unit noch nicht läuft, wird sie einfach nur gestartet.

try-restart Wie `restart`, aber Units, die nicht laufen, werden nicht neu gestartet.

reload-or-restart (und `reload-or-try-restart`) Lädt die Konfiguration der benannten Units neu (wie bei `reload`), falls die Units das erlauben, oder startet sie neu (wie bei `restart` oder `try-restart`), falls nicht.

Statt `reload-or-try-restart` können Sie als Tippvereinfachung auch `force-reload` sagen (das ist zumindest ein bisschen kürzer).

isolate Die benannte Unit wird gestartet (mitsamt ihren Abhängigkeiten) und alle anderen werden angehalten. Entspricht einem Wechsel des Runlevels bei System-V-Init.

kill Schickt ein Signal an einen oder mehrere Prozesse der Unit. Mit der Option `--kill-who` können Sie angeben, welcher Prozess gemeint ist. (Die Möglichkeiten sind `main`, `control` und `all` – letzteres ist die Voreinstellung –, und `main` und `control` sind in `systemctl`(1) genauer erklärt.) Mit der Option `--signal` (kurz `-s`) können Sie das zu verschickende Signal bestimmen.

status Gibt den aktuellen Status der benannten Unit(s) aus, gefolgt von den neuesten paar Protokollnachrichten. Wenn Sie keine Units benennen, bekommen Sie einen Überblick über alle Units (gegebenenfalls eingeschränkt auf bestimmte Typen mit `-t`).


Der Protokollauszug ist normalerweise auf 10 Zeilen beschränkt und lange Zeilen werden gekürzt. Mit den Optionen `--lines` (`-n`) und `--full` (`-l`) können Sie das ändern.


»`status`« ist für den menschlichen Konsum gedacht. Wenn Sie Ausgabe wollen, die Sie mit einem Programm weiterverarbeiten können, dann benutzen Sie »`systemctl show`«.

cat Zeigt die Konfigurationsdatei(en) für eine oder mehrere Units an (inklusive Fragmente in Konfigurationsverzeichnissen speziell für diese Unit). Zur Verdeutlichung werden Kommentare mit den betreffenden Dateinamen eingestreut.

help Zeigt Dokumentation (zum Beispiel Man-Pages) für die betreffende(n) Unit(s) an:

```
$ systemctl help syslog
```

ruft zum Beispiel die Man-Page für den Protokolldienst auf, egal welches Programm konkret den Protokolldienst erbringt.


Bei den meisten Distributionen funktionieren Kommandos wie

```
# service example start
# service example stop
# service example restart
# service example reload
```

unabhängig davon, ob das System systemd oder System-V-Init benutzt.

Im nächsten Abschnitt sehen Sie noch einige Kommandos, die sich mit der Installation und Deinstallation von Units beschäftigen.

Andere Kommandos Hier sind noch ein paar Kommandos, die sich nicht speziell auf einzelne Units (oder Gruppen von Units) beziehen.

daemon-reload Dieses Kommando veranlasst systemd, seine Konfiguration neu einzulesen. Dazu gehört auch, dass Unit-Dateien neu generiert werden, die zur Laufzeit aus anderen Konfigurationsdateien des Systems erzeugt werden, und der Abhängigkeitsbaum neu aufgebaut wird.


Kommunikationskanäle, die systemd für Hintergrunddienste verwaltet, bleiben über den Ladevorgang hinweg erhalten.

daemon-reexec Startet das Programm `systemd` neu. Dabei wird der interne Zustand von systemd gespeichert und später wieder eingelesen.


Dieses Kommando ist vor allem nützlich, wenn eine neue Version von `systemd` eingespielt worden ist (oder zum Debugging von systemd). Auch hier bleiben Kommunikationskanäle, die systemd für Hintergrunddienste verwaltet, über den Ladevorgang hinweg erhalten.

is-system-running Gibt aus, in welchem Zustand sich das System befindet. Die möglichen Antworten sind:

initializing Das System ist im frühen Startvorgang (die Ziele `basic.target`, `rescue.target` oder `emergency.target` wurden noch nicht erreicht).

starting Das System ist im späten Startvorgang (es sind noch Jobs in der Warteschlange).

running Das System ist im normalen Betrieb.

degraded Das System ist im normalen Betrieb, aber eine oder mehrere Units sind in einem Fehlerzustand.

maintenance Eines der Ziele `rescue.target` oder `emergency.target` ist aktiv.

stopping Das System wird gerade angehalten.

Übungen

18.9 [!2] Verwenden Sie `systemctl`, um einen geeignet unverfänglichen Dienst (etwa `cups.service`) zu stoppen, zu starten, neu zu starten und die Konfiguration neu zu laden.

18.10 [2] Das Kommando `runlevel` von System-V-Init gibt aus, in welchem Runlevel der Rechner sich gerade befindet. Was wäre ein ungefähres Äquivalent für systemd?

18.11 [1] Was ist der Vorteil von

```
# systemctl kill example.service
```

gegenüber

```
# killall example
```

(oder »`pkill example`«)?

18.7 Installation von Units

Um mit systemd einen neuen Hintergrunddienst zur Verfügung zu stellen, brauchen Sie eine Unit-Datei, zum Beispiel `example.service`. (Ein System-V-Init-Skript würde es dank der Rückwärtskompatibilität auch tun, aber das interessiert uns jetzt mal nicht.) Diese platzieren Sie in einem geeigneten Verzeichnis (wir empfehlen `/etc/systemd/system`. Anschließend sollte sie auftauchen, wenn Sie »`systemctl list-unit-files`« aufrufen:

```
# systemctl list-unit-files
UNIT FILE                                   STATE
proc-sys-fs-binfmt_misc.automount           static
org.freedesktop.hostname1.busname           static
org.freedesktop.locale1.busname             static
◁◁◁◁◁
example.service                             disabled
◁◁◁◁◁
```

Der Zustand `disabled` bedeutet, dass die Unit zwar prinzipiell zur Verfügung steht, aber nicht automatisch gestartet wird.

Unit aktivieren

Sie können die Unit »aktivieren«, also dafür vormerken, bei Bedarf (etwa beim Systemstart oder wenn ein bestimmter Kommunikationskanal angesprochen wird) gestartet zu werden, indem Sie das Kommando »`systemctl enable`« geben:

```
# systemctl enable example
Created symlink from /etc/systemd/system/multi-user.target.wants/▷
 ◁example.service to /etc/systemd/system/example.service.
```

Was hier passiert, sehen Sie schon an der Ausgabe des Kommandos: Ein symbolisches Link auf die Unit-Datei des Dienstes aus dem Verzeichnis `/etc/systemd/system/multi-user.target.wants` sorgt dafür, dass die Unit als Abhängigkeit des `multi-user.target` gestartet wird.

Möglicherweise werden Sie sich fragen, woher systemd weiß, dass die Unit `example` ins `multi-user.target` integriert werden soll (und nicht etwa ein anderes Ziel). Die Antwort darauf lautet: Die Datei `example.service` hat einen Abschnitt

```
[Install]
WantedBy=multi-user.target
```

Nach einem `enable` macht systemd das Äquivalent eines »`systemctl daemon-reload`«. Allerdings werden keine Units neu gestartet (oder angehalten).

Sie können die symbolischen Links genausogut mit der Hand anlegen. Allerdings müssen Sie sich dann auch selber um das »`systemctl daemon-reload`« kümmern.

Wenn Sie möchten, dass die Unit gleich gestartet wird, können Sie entweder ein

```
# systemctl start example
```

hinterherschicken, oder Sie rufen »`systemctl enable`« mit der Option `--now` auf.

Sie können eine Unit auch direkt starten (mit »`systemctl start`«), ohne sie vorher mit »`systemctl enable`« aktiviert zu haben. Ersteres startet tatsächlich den Dienst, während letzteres nur dafür sorgt, dass der Dienst zu einem geeigneten Moment gestartet wird (etwa beim Systemstart oder wenn eine bestimmte Hardware angeschlossen wird).

Unit deaktivieren

Mit »`systemctl disable`« können Sie eine Unit wieder deaktivieren. Genau wie bei `enable` führt systemd ein implizites `daemon-reload` durch.

Auch hier gilt, dass die Unit nicht angehalten wird, wenn sie gerade läuft (Sie verhindern ja nur, dass sie aktiviert wird, wenn es mal wieder soweit ist). Verwenden Sie die Option `--now` oder ein explizites »`systemctl stop`«.

Das Kommando »`systemctl reenable`« entspricht einem »`systemctl disable`« unmittelbar gefolgt von einem »`systemctl enable`« für die betreffenden Units. Damit können Sie die Einbindung von Units auf den »Auslieferungszustand« zurücksetzen.

Unit »maskieren«

Mit dem Kommando »`systemctl mask`« können Sie eine Unit »maskieren«, also komplett blockieren. Damit wird sie nicht nur nicht automatisch gestartet, sondern kann nicht einmal mehr manuell gestartet werden. Mit »`systemctl unmask`« machen Sie diese Operation rückgängig.

Systemd implementiert das, indem der Name der Unit-Datei in `/etc/systemd/system` symbolisch auf `/dev/null` verlinkt wird. Damit werden gleichnamige Dateien in später durchsuchten Verzeichnissen (etwa `/lib/systemd/system`) völlig ignoriert.

Übungen

18.12 [!2] Was passiert, wenn Sie »`systemctl disable cups`« ausführen? (Betrachten Sie die ausgegebenen Kommandos.) Aktivieren Sie den Dienst anschließend wieder.

18.13 [2] Wie können Sie Units »maskieren«, deren Unit-Dateien in `/etc/systemd/system` stehen?

Kommandos in diesem Kapitel

`systemctl` Zentrales Steuerungsprogramm für systemd `systemctl`(1) 305, 315

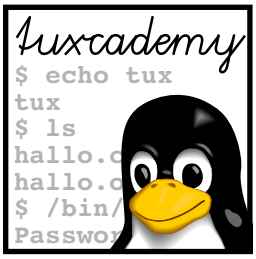
Zusammenfassung

- Systemd ist eine moderne Alternative zu System-V-Init.
- »Units« sind Systemkomponenten, die systemd verwaltet. Sie werden über Unit-Dateien konfiguriert.
- Unit-Dateien ähneln vage den INI-Dateien von Microsoft Windows.
- Systemd unterstützt flexible Mechanismen für lokale Konfiguration und für die automatische Erstellung sehr ähnlicher Unit-Dateien aus »Schablonen«.
- Mit systemd können Sie eine Vielzahl verschiedener Arten von Units verwalten – Dienste, Mountpunkte, Timer …
- Abhängigkeiten zwischen Units können auf verschiedenste Weise ausgedrückt werden.
- »Ziele« sind Units, die vage den Runlevels von System-V-Init entsprechen. Sie dienen zur Gruppierung verwandter Dienste und zur Synchronisation.
- Mit dem Kommando `systemctl` können Sie systemd steuern.
- Systemd enthält leistungsfähige Hilfsmittel zur Installation und Deinstallation von Units.

Literaturverzeichnis

systemd »systemd System and Service Manager«.
`http://www.freedesktop.org/wiki/Software/systemd/`

XDG-DS14 Preston Brown, Jonathan Blandford, Owen Taylor, et al. »Desktop Entry Specification«, April 2014. Version 1.1.
`http://standards.freedesktop.org/desktop-entry-spec/latest/`



19

Zeitgesteuerte Vorgänge – `at` und `cron`

Inhalt

19.1 Allgemeines . . . 322
19.2 Einmalige Ausführung von Kommandos . . . 322
19.2.1 `at` und `batch` . . . 322
19.2.2 `at`-Hilfsprogramme . . . 324
19.2.3 Zugangskontrolle . . . 325
19.3 Wiederholte Ausführung von Kommandos . . . 325
19.3.1 Aufgabenlisten für Benutzer . . . 325
19.3.2 Systemweite Aufgabenlisten . . . 327
19.3.3 Zugangskontrolle . . . 328
19.3.4 Das Kommando `crontab` . . . 328
19.3.5 Anacron . . . 328

Lernziele

- Kommandos mit `at` zu einem zukünftigen Zeitpunkt ausführen können
- Kommandos periodisch mit `cron` ausführen können
- `anacron` kennen und einsetzen können

Vorkenntnisse

- Umgang mit Linux-Kommandos
- Umgang mit einem Texteditor

19.1 Allgemeines

Ein wichtiger Bestandteil der Systemverwaltung besteht darin, wiederholt ablaufende Vorgänge zu automatisieren. Eine denkbare Aufgabe wäre, dass sich der Mailserver eines Firmennetzwerkes in regelmäßigen Abständen beim Internet-Provider einwählt und die eingegangenen Nachrichten abholt. Ferner könnten alle Mitglieder einer Projektgruppe eine halbe Stunde vor der wöchentlichen Besprechung eine schriftliche Erinnerung erhalten. Auch Administrationsaufgaben wie Dateisystemtests oder Datenarchivierung lassen sich automatisch in der Nacht ausführen, da zu dieser Zeit die Systembelastung naturgemäß deutlich geringer ist.

19.2 Einmalige Ausführung von Kommandos

19.2.1 at und batch

Mit Hilfe des Kommandos at lassen sich beliebige Shell-Befehle zu einem späteren Zeitpunkt, also zeitversetzt, einmalig ausführen. Wenn Kommandos hingegen in regelmäßigen Intervallen wiederholt ausgeführt werden sollen, ist die Verwendung von cron (Abschnitt 19.3) vorzuziehen.

Die Idee hinter at ist, einen Zeitpunkt vorzugeben, zu dem dann ein Kommando oder eine Folge von Kommandos ausgeführt wird. Etwa so:

```
$ at 01:00
warning: commands will be executed using /bin/sh
at> tar cvzf /dev/st0 $HOME
at> echo " Backup fertig" | mail -s Backup $USER
at> Strg+d
Job 123 at 2006-11-08 01:00
```

Hiermit würden Sie um 1 Uhr nachts eine Sicherheitskopie Ihres Heimatverzeichnisses auf Band schreiben (Band einlegen nicht vergessen!) und sich anschließend per Mail eine Vollzugsmeldung schicken lassen.

Zeitangabe

Das Argument von at ist eine Zeitangabe für die Ausführung der Kommandos. Zeiten im Format »⟨*HH*⟩:⟨*MM*⟩« bezeichnen den nächstmöglichen solchen Zeitpunkt: Wird das Kommando »at 14:00« um 8 Uhr früh angegeben, bezieht es sich auf denselben Tag; wird es um 16 Uhr angegeben, auf den Folgetag.

Sie können die Zeitpunkte durch Anhängen von today und tomorrow eindeutig machen: »at 14:00 today«, vor 14 Uhr gegeben, bezieht sich auf heute, »at 14:00 tomorrow« auf morgen.

Möglich sind auch angelsächsische Zeitangaben wie 01:00am oder 02:20pm sowie die symbolischen Namen midnight (0 bzw. 24 Uhr), noon (12 Uhr) und teatime (16 Uhr) (!); der ebenfalls erlaubte Zeitpunkt now ist vor allem in Verbindung mit relativen Zeitangaben (siehe unten) sinnvoll.

Datumsangabe

Neben Uhrzeiten versteht at auch Datumsangaben in der Form »⟨*MM*⟩⟨*TT*⟩⟨*JJ*⟩« und »⟨*MM*⟩/⟨*TT*⟩/⟨*JJ*⟩« (also nach amerikanischem Brauch, mit dem Monat vor dem Tag) sowie »⟨*TT*⟩.⟨*MM*⟩.⟨*JJ*⟩« (für uns Europäer). Daneben sind ausgeschriebene amerikanische Datumsangaben der Form »⟨*Monatsname*⟩ ⟨*Tag*⟩« und »⟨*Monatsname*⟩ ⟨*Tag*⟩ ⟨*Jahr*⟩« erlaubt. Wenn Sie nur ein Datum angeben, werden die Kommandos zur aktuellen Zeit am betreffenden Tag ausgeführt; Sie können auch Zeit- und Datumsangabe kombinieren, müssen dann aber das Datum nach der Zeit anführen:

```
$ at 11:11 November 11
warning: commands will be executed using /bin/sh
at> echo 'Helau!'
```

```
at> Strg+D
Job 124 at 2003-11-11 11:11
```

Außer »expliziten« Zeit- und Datumsangaben können Sie auch »relative« Angaben machen, indem Sie eine Differenz zu einem gegebenen Zeitpunkt bestimmen: »relative« Angaben

```
$ at now + 5 minutes
```

führt die Kommandos in fünf Minuten aus, während

```
$ at noon + 2 days
```

sich auf 12 Uhr am übermorgigen Tag bezieht (jedenfalls solange das `at`-Kommando vor 12 Uhr angegeben wurde). `at` unterstützt die Zeiteinheiten `minutes`, `hours`, `days` und `weeks`.

Eine einzige Verschiebung um eine einzige Zeiteinheit muss reichen: Kombinationen wie

```
$ at noon + 2 hours 30 minutes
```

oder

```
$ at noon + 2 hours + 30 minutes
```

sind leider nicht erlaubt. Natürlich können Sie jede beliebige Verschiebung in Minuten ausdrücken …

`at` liest die Kommandos von der Standardeingabe, also normalerweise der Tastatur; mit der Option »`-f` ⟨*Datei*⟩« können Sie stattdessen eine Datei angeben. Kommandos

`at` versucht, die Kommandos in einer Umgebung auszuführen, die der zum Zeitpunkt des `at`-Aufrufs möglichst ähnelt. Das aktuelle Arbeitsverzeichnis, die *umask* und die aktuellen Umgebungsvariablen (außer `TERM`, `DISPLAY` und `_`) werden gesichert und vor der Kommandoausführung wieder aktiviert.

Eine etwaige Ausgabe der per `at` gestarteten Kommandos – Standardausgabe- und Standardfehlerausgabekanal – bekommen Sie per Mail geschickt. Ausgabe

Wenn Sie vor dem Aufruf von `at` mit `su` eine andere Identität angenommen haben, werden die Kommandos mit dieser Identität ausgeführt. Die Ausgabemails gehen aber trotzdem an Sie.

Während Sie mit `at` Kommandos zu einem bestimmten Zeitpunkt ausführen können, macht der (ansonsten analog funktionierende) Befehl `batch` es möglich, eine Folge von Kommandos zum »nächstmöglichen Zeitpunkt« auszuführen. Wann das passiert, hängt von der aktuellen Systemlast ab; hat das System gerade viel zu tun, müssen `batch`-Jobs warten. Ausführung zum »nächstmöglichen Zeitpunkt«

Eine `at`-artige Zeitangabe ist bei `batch` erlaubt, aber nicht vorgeschrieben. Wird sie angegeben, so werden die Kommandos »irgendwann nach« dem genannten Zeitpunkt ausgeführt, so als wären sie gerade dann per `batch` eingereicht worden.

`batch` ist nicht geeignet für Umgebungen, wo die Benutzer um Ressourcen wie Rechenzeit konkurrieren. Hierfür müssen andere Systeme eingesetzt werden.

Übungen

19.1 [!1] Nehmen wir an, jetzt ist es der 1. März, 15 Uhr. Wann werden die mit den folgenden Kommandos eingereichten Aufträge ausgeführt?

1. `at 17:00`
2. `at 02:00pm`
3. `at teatime tomorrow`
4. `at now + 10 hours`

19.2 [1] Verwenden Sie das Programm `logger`, um in 3 Minuten eine Nachricht ins Systemprotokoll zu schreiben.

19.2.2 at-Hilfsprogramme

Warteschlange anschauen

Das System reiht die mit `at` registrierten Aufträge in eine Warteschlange ein. Diese können Sie mit `atq` inspizieren (Sie sehen aber nur Ihre eigenen Aufträge, es sei denn, Sie sind `root`):

```
$ atq
123     2003-11-08 01:00 a hugo
124     2003-11-11 11:11 a hugo
125     2003-11-08 21:05 a hugo
```

Das »a« in der Liste bezeichnet die »Auftragsklasse«, einen Buchstaben zwischen »a« und »z«. Sie können mit der Option `-q` für `at` eine Auftragsklasse bestimmen; Aufträge in Klassen mit »größeren« Buchstaben werden mit höheren `nice`-Werten ausgeführt. Standard ist »a« für `at`- und »b« für `batch`-Aufträge.

Ein gerade laufender Auftrag ist in der besonderen Auftragsklasse »=«.

Auftrag stornieren

Mit `atrm` können Sie einen Auftrag stornieren. Hierzu müssen Sie dessen Auftragsnummer angeben, die Sie bei der Einreichung genannt oder mit `atq` gezeigt bekommen haben. Wenn Sie nachschauen wollen, aus welchen Kommandos der Auftrag besteht, können Sie das mit »`at -c` ⟨*Auftragsnummer*⟩«.

Daemon

Zuständig für die tatsächliche Ausführung der `at`-Aufträge ist ein Daemon namens `atd`. Dieser wird üblicherweise beim Systemstart geladen und wartet im Hintergrund auf Arbeit. Beim Start von `atd` sind einige Optionen möglich:

`-b` (engl. *batch*, Stapel) Legt das minimale Intervall für den Start zweier `batch`-Aufträge fest. Voreinstellung ist hier 60 Sekunden.

`-l` (engl. *load*, Last) Legt einen Grenzwert für die Systembelastung fest, oberhalb der `batch`-Aufträge nicht ausgeführt werden. Voreinstellung ist hier ein Wert von 0,8.

`-d` (engl. *debug*, »Entwanzen«) aktiviert den Debug-Modus, d. h., alle Fehlermeldungen werden nicht per `syslogd` protokolliert, sondern auf die Standardfehlerausgabe geschrieben.

Der Daemon `atd` benötigt zu seiner Arbeit die folgenden Verzeichnisse:

- Im Verzeichnis `/var/spool/atjobs` werden die `at`-Aufträge abgelegt. Der Zugriffsmodus sollte 700 sein, der Eigentümer ist `at`.
- Das Verzeichnis `/var/spool/atspool` dient zur Zwischenspeicherung von Ausgaben. Auch hier sollten der Eigentümer `at` und der Zugriffsmodus 700 sein.

Übungen

19.3 [1] Registrieren Sie mit `at` einige Aufträge und lassen Sie sich eine Auftragsliste anzeigen. Entfernen Sie die Aufträge wieder.

19.4 [2] Wie würden Sie eine Liste von `at`-Aufträgen erzeugen, die nicht nach der Auftragsnummer, sondern nach dem vorgesehenen Ausführungszeitpunkt sortiert ist?

19.2.3 Zugangskontrolle

Die Dateien `/etc/at.allow` und `/etc/at.deny` bestimmen, wer mit `at` und `batch` Aufträge einreichen darf. Wenn die Datei `/etc/at.allow` existiert, sind nur die darin eingetragenen Benutzer berechtigt, Aufträge einzureichen. Gibt es die Datei `/etc/at.allow` nicht, aber die Datei `/etc/at.deny`, dann dürfen diejenigen Benutzer Aufträge einreichen, die *nicht* in der Datei stehen. Existiert weder die eine noch die andere, dann stehen `at` und `batch` nur `root` zur Verfügung.

`/etc/at.allow`
`/etc/at.deny`

Debian GNU/Linux liefert eine `/etc/at.deny`-Datei aus, in der die Namen diverser Systembenutzer stehen (etwa `alias`, `backup`, `guest` und `www-data`). Damit werden diese Benutzeridentitäten von der Verwendung von `at` ausgeschlossen.

Die Voreinstellung von Ubuntu entspricht auch hier der von Debian GNU/Linux.

Red Hat liefert eine leere `/etc/at.deny`-Datei aus; damit darf jeder Benutzer Aufträge einreichen.

Die OpenSUSE-Voreinstellung gleicht (interessanterweise) der von Debian GNU/Linux und Ubuntu – diverse Systembenutzer dürfen `at` nicht verwenden. (Den explizit ausgeschlossenen Benutzer `www-data` zum Beispiel gibt es bei OpenSUSE überhaupt nicht; Apache läuft mit der Identität von `wwwrun`.)

Übungen

19.5 [1] Wer darf auf Ihrem System `at` und `batch` benutzen?

19.3 Wiederholte Ausführung von Kommandos

19.3.1 Aufgabenlisten für Benutzer

Im Unterschied zu den `at`-Kommandos dient der Daemon `cron` zur automatischen Ausführung von sich regelmäßig wiederholenden Aufgaben. Gestartet werden sollte `cron` – wie auch der `atd` – beim Systemstart mit einem Init-Skript; Sie müssen sich darum normalerweise nicht kümmern, da `cron` und `atd` so wichtige Bestandteile eines Linux-Systems sind, dass keine der wichtigen Distributionen auf sie verzichtet.

Jeder Anwender hat seine eigene Aufgabenliste (vulgo `crontab`), die im Verzeichnis `/var/spool/cron/crontabs` (bei Debian GNU/Linux und Ubuntu; bei SUSE: `/var/spool/cron/tabs`, bei Red Hat: `/var/spool/cron`) unter dem jeweiligen Benutzernamen abgelegt ist. Die dort beschriebenen Kommandos werden mit den Rechten des betreffenden Anwenders ausgeführt.

Aufgabenliste

Auf Ihre Aufgabenliste im `cron`-Verzeichnis haben Sie keinen direkten Zugriff, sondern müssen das Programm `crontab` bemühen (siehe unten). Beachten Sie hierzu auch Übung 19.6.

Syntax

crontab-Dateien sind zeilenweise aufgebaut; jede Zeile beschreibt einen (wiederkehrenden) Zeitpunkt und ein Kommando, das zu dem betreffenden Termin ausgeführt werden soll. Leerzeilen und Kommentarzeilen (mit einem »#« am Anfang) werden ignoriert. Die übrigen Zeilen bestehen aus fünf Zeitfeldern und dem auszuführenden Kommando; die Zeitfelder beschreiben respektive die Minute (0–59), die Stunde (0–23), der Tag im Monat (1–31), der Monat (1–12 oder der englische Name) und den Wochentag (0–7, 0 und 7 stehen für Sonntag, oder der englische Name), zu denen das Kommando ausgeführt werden soll. Alternativ ist jeweils ein Sternchen (»*«) erlaubt, das für »egal« steht. Zum Beispiel bedeutet

Zeitfelder
Kommando

```
58 19 * * * echo "Gleich kommt die Tagesschau"
```

dass das Kommando täglich um 19.58 Uhr ausgeführt wird (Tag, Monat und Wochentag sind egal).

Das Kommando wird ausgeführt, wenn Stunde, Minute und Monat genau stimmen und *mindestens eine* der beiden Tagesangaben – Tag im Monat oder Wochentag – zutreffen. Die Spezifikation

```
1 0 13 * 5 echo "Kurz nach Mitternacht"
```

gibt also an, dass die Meldung an jedem Monats-Dreizehnten *und* an jedem Freitag ausgegeben wird, nicht nur am Freitag, dem 13.

Die letzte Zeile in einer crontab-Datei *muss* mit einem Zeilentrenner aufhören, sonst wird sie ignoriert.

Listen

cron akzeptiert in den Zeitfeldern nicht nur einzelne Zahlen, sondern erlaubt auch kommaseparierte Listen. Die Angabe »0,30« im Minutenfeld würde also dazu führen, dass das Kommando zu jeder »vollen halben« Stunde ausgeführt wird. Außerdem sind Bereiche erlaubt: »8-11« ist äquivalent zu »8,9,10,11«, »8-10,14-16« entspricht »8,9,10,14,15,16«. Ebenfalls gestattet ist die Angabe einer »Schrittweite« in Bereichen. Die Spezifikation »0-59/10« im Minutenfeld ist gleichbedeutend mit »0,10,20,30,40,50«. Wird – wie hier – der komplette Wertebereich abgedeckt, so könnten Sie auch »*/10« schreiben.

Schrittweite

Monats- und Wochentagsangaben

Die bei Monats- und Wochentagsangaben erlaubten Namen bestehen jeweils aus den ersten drei Buchstaben des englischen Monats- oder Wochentagsnamen (also zum Beispiel may, oct, sun oder wed). Bereiche und Listen von Namen sind nicht erlaubt.

Kommando

Der Rest der Zeile bestimmt das auszuführende Kommando, das von cron an /bin/sh (bzw. die in der Variablen SHELL benannte Shell, siehe unten) übergeben wird.

Prozentzeichen (%) im Kommando müssen mit einem Rückstrich versteckt werden (also »\%«), sonst werden sie in Zeilentrenner umgewandelt. Dann gilt das Kommando als am ersten Prozentzeichen beendet; die folgenden »Zeilen« werden dem Kommando auf der Standardeingabe verfüttert.

Übrigens: Wenn Sie als Systemadministrator möchten, dass bestimmte Kommandos nicht, wie sonst bei cron üblich, bei der Ausführung per syslogd protokolliert werden, können Sie dies durch die Angabe eines »-« als erstes Zeichen der Zeile unterdrücken.

Zuweisungen an Umgebungsvariablen

Außer Kommandos mit Wiederholungsangaben können die crontab-Zeilen auch Zuweisungen an Umgebungsvariablen enthalten. Diese haben die Form »⟨*Variablenname*⟩=⟨*Wert*⟩« (wobei im Gegensatz zur Shell vor und nach dem »=« auch Leerplatz stehen darf). Enthält der ⟨*Wert*⟩ Leerzeichen, sollte er mit Anführungszeichen eingefasst werden. Die folgenden Variablen werden automatisch voreingestellt:

SHELL Mit dieser Shell werden die eingegebenen Befehle ausgeführt. Voreingestellt ist dabei `/bin/sh`, aber auch die Angabe anderer Shells ist erlaubt.

LOGNAME Der Benutzername wird aus `/etc/passwd` übernommen und kann nicht geändert werden.

HOME Das Heimatverzeichnis wird ebenfalls aus `/etc/passwd` übernommen. Hier ist jedoch eine Änderung des Variablenwerts zulässig.

MAILTO An diese Adresse schickt `cron` Nachrichten mit der Ausgabe des aufgerufenen Kommandos (sonst gehen sie an den Eigentümer der `crontab`-Datei). Soll `cron` überhaupt keine Nachrichten verschicken, muss die Variable leer gesetzt sein, d. h. `MAILTO=""`.

19.3.2 Systemweite Aufgabenlisten

Neben den benutzerbezogenen Dateien existiert noch eine systemweite Aufgabenliste. Diese findet sich in `/etc/crontab` und gehört dem Systemadministrator `root`, der sie als einziger ändern darf. Die Syntax von `/etc/crontab` ist geringfügig anders als die der benutzereigenen `crontab`-Dateien; zwischen den Zeitangaben und dem auszuführenden Kommando steht hier noch der Name des Benutzers, mit dessen Rechten das Kommando ausgeführt werden soll.

`/etc/crontab`

Diverse Linux-Distributionen haben ein `/etc/cron.d`-Verzeichnis; in diesem Verzeichnis können Dateien stehen, die als »Erweiterungen« von `/etc/crontab` interpretiert werden. Per Paketmanagement installierte Softwarepakete können so leichter den `cron`-Dienst benutzen, als wenn sie Zeilen zur `/etc/crontab` hinzufügen müßten.

Populär sind auch Verzeichnisse `/etc/cron.hourly`, `/etc/cron.daily` und so weiter. In diesen Verzeichnissen können Softwarepakete (oder der Systemadministrator) Dateien ablegen, deren Inhalt dann stündlich, täglich, … ausgeführt wird. Diese Dateien sind keine `crontab`-Dateien, sondern »normale« Shellskripte.

`cron` liest die Aufgabenlisten – aus benutzereigenen `crontab`-Dateien, der systemweiten `/etc/crontab` und den Dateien in `/etc/cron.d`, sofern vorhanden – nur einmal beim Start ein und behält sie danach im Speicher. Das Programm schaut allerdings jede Minute nach, ob die `crontab`-Dateien geändert wurden. Zu diesem Zweck wird einfach die *mtime*, der Zeitpunkt der letzten Änderung, herangezogen. Wenn `cron` hier eine Veränderung bemerkt, wird die Aufgabenliste automatisch neu aufgebaut. In diesem Fall ist also kein expliziter Neustart des Daemons erforderlich.

`crontab`-Änderungen und `cron`

Übungen

19.6 [2] Wieso können Sie als Benutzer nicht auf Ihre Aufgabenliste in `/var/spool/cron/crontabs` (oder dem Äquivalent für Ihre Distribution) zugreifen? Wie wird arrangiert, dass `crontab` das kann?

19.7 [2] Wie können Sie dafür sorgen, dass eine Aufgabe nur am Freitag, dem 13., ausgeführt wird?

19.8 [3] Wie stellt das System sicher, dass die Aufgaben in `/etc/cron.hourly`, `/etc/cron.daily`, … tatsächlich einmal in der Stunde, einmal am Tag usw. ausgeführt werden?

19.3.3 Zugangskontrolle

Welche Anwender überhaupt mit cron arbeiten dürfen, ist ähnlich wie bei at in zwei Dateien festgelegt. In /etc/cron.allow (gelegentlich /var/spool/cron/allow) sind die Benutzer aufgeführt, die cron verwenden dürfen. Wenn die Datei nicht existiert, aber es dafür die Datei /etc/cron.deny (manchmal /var/spool/cron/deny) gibt, sind in letzterer diejenigen Benutzer aufgelistet, die *nicht* in den Genuss der automatisierten Befehlsausführung kommen dürfen. Sind beide Dateien nicht vorhanden, hängt es von der jeweiligen Konfiguration ab, ob nur root die Dienste von cron beanspruchen darf oder sozusagen »gleiches Recht für alle« herrscht und cron jedem Benutzer zur Verfügung steht.

19.3.4 Das Kommando crontab

Die einzelnen Benutzer können ihre crontab-Datei nicht von Hand ändern, da das System sie vor ihnen versteckt. Lediglich die systemweite Aufgabenliste /etc/crontab ist ein Fall für den Lieblingseditor von root.

Aufgabenlisten verwalten

Statt einen Editor direkt aufzurufen, sollten alle Benutzer das Kommando crontab verwenden. Hiermit können Aufgabenlisten erstellt, eingesehen, verändert und auch wieder entfernt werden. Mit

```
$ crontab -e
```

können Sie Ihre crontab-Datei mit dem Editor bearbeiten, dessen Name in den Umgebungsvariablen VISUAL bzw. EDITOR festgelegt ist – ersatzweise dem Editor vi. Nach dem Verlassen des Editors wird die modifizierte crontab-Datei automatisch installiert. Statt der Option -e können Sie auch den Namen einer Datei angeben, deren Inhalt dann als Aufgabenliste installiert wird. Der Dateiname »-« steht hierbei stellvertretend für die Standardeingabe.

Mit der Option -l gibt crontab Ihre crontab-Datei auf der Standardausgabe aus; mit der Option -r wird eine installierte Aufgabenliste ersatzlos gelöscht.

Mit der Option »-u ⟨*Benutzername*⟩« können Sie sich auf einen anderen Benutzer beziehen (wofür Sie in der Regel root sein müssen). Dies ist vor allem wichtig, wenn Sie su benutzen; in diesem Fall sollten Sie immer mit der -u-Option operieren, um sicherzustellen, dass Sie die richtige crontab-Datei erwischen.

Übungen

19.9 [1] Registrieren Sie mit dem Programm crontab eine Aufgabe, die jede Minute das aktuelle Datum an die Datei /tmp/date.log anhängt. Wie können Sie bequem erreichen, dass dasselbe alle zwei Minuten passiert?

19.10 [1] Verwenden Sie crontab, um den Inhalt Ihrer Aufgabenliste auf der Standardausgabe auszugeben und anschließend wieder zu löschen.

19.11 [2] (Für Administratoren.) Sorgen Sie dafür, dass der Benutzer hugo den cron-Dienst *nicht* verwenden darf. Vergewissern Sie sich, dass Ihre Maßnahme funktioniert.

19.3.5 Anacron

Mit cron können Sie Kommandos wiederholt zu vorgegebenen Zeitpunkten ausführen. Offensichtlich funktioniert das aber nur, wenn der Computer zum betreffenden Zeitpunkt eingeschaltet ist – es hat keinen großen Zweck, auf einem Arbeitsplatzrechner einen cron-Job für 2 Uhr morgens zu konfigurieren, wenn der Rechner außerhalb der üblichen Bürozeiten aus Stromspargründen ausgeschaltet ist. Auch mobile Rechner sind oft zu ungewöhnlichen Zeiten ein- oder ausgeschaltet, so dass es schwierig ist, die periodischen automatischen Aufräumungsarbeiten einzuplanen, die für ein Linux-System nötig sind.

Das Programm anacron (von Itai Tzur, aktuell gewartet von Pascal Hakim) kann ähnlich wie cron Jobs ausführen, die im täglichen, wöchentlichen oder monatlichen Rhythmus laufen sollen. (Tatsächlich gehen beliebige Abstände von *n* Tagen.) Dabei ist es nur wichtig, dass der Rechner am betreffenden Tag lange genug eingeschaltet ist, dass die Jobs ausgeführt werden können – es ist egal, wann am Tag. Allerdings wird anacron maximal einmal am Tag aktiv; wenn Sie eine höhere Frequenz brauchen (Stunden oder Minuten), führt an cron kein Weg vorbei.

Im Gegensatz zu cron ist anacron relativ primitiv, was die Verwaltung von Jobs angeht. Mit cron kann potentiell jeder Benutzer Jobs anlegen; bei anacron ist das das Privileg des Systemverwalters root.

Die Jobs für anacron werden in der Datei /etc/anacrontab festgelegt. Neben den üblichen Kommentar- und Leerzeilen (die ignoriert werden) enthält sie Zuweisungen an Umgebungsvariablen der Form

```
SHELL=/bin/sh
```

und Job-Beschreibungen der Form

```
7 10 weekly run-parts /etc/cron.weekly
```

Dabei steht die erste Zahl (hier 7) für den Zeitabstand (in Tagen), in dem einzelne Aufrufe des Jobs stattfinden sollen. Die zweite Zahl (10) gibt an, wieviele Minuten nach dem Start von anacron der Job gestartet werden soll. Als Nächstes kommen ein Name für den Job (hier weekly) und schließlich das auszuführende Kommando. Überlange Zeilen können mit einem »\« am Zeilenende umbrochen werden.

Der Jobname darf alle Zeichen enthalten außer Freiplatz und dem Schrägstrich. Er wird verwendet, um den Job in Protokollnachrichten zu identifizieren, und anacron benutzt ihn auch als Namen für die Datei, mit der er den Zeitstempel der letzten Ausführung protokolliert. (Diese Dateien stehen normalerweise in /var/spool/anacron.)

Wenn anacron gestartet wird, liest es /etc/anacrontab und prüft für jeden Job, ob er während der letzten *t* Tage ausgeführt wurde, wobei *t* der Zeitabstand aus der Jobdefinition ist. Wenn nein, dann wartet anacron die in der Jobdefinition angegebene Anzahl von Minuten ab und startet das Shell-Kommando.

Sie können auf der Kommandozeile von anacron einen Jobnamen angeben, um (gegebenenfalls) nur diesen Job auszuführen. Alternativ sind auf der Kommandozeile auch Shell-Suchmuster erlaubt, damit Sie Gruppen von (geschickt vergebenen) Jobnamen mit einem anacron-Aufruf ausführen können. Beim Aufruf von anacron gar keinen Jobnamen anzugeben ist dasselbe wie der Jobname »*«.

Den Zeitabstand zwischen Ausführungen eines Jobs können Sie auch symbolisch angeben; gültige Werte sind @daily, @weekly, @monthly, @yearly und @annually (die beiden letzten sind äquivalent).

In der Definition einer Umgebungsvariablen werden Leerzeichen links vom »=« ignoriert. Rechts vom »=« sind sie Teil des Wertes der Variablen. Definitionen gelten bis zum Dateiende oder bis zu einer neuen Definition derselben Variablen.

Einige »Umgebungsvariable« haben eine Sonderbedeutung für anacron. Mit RANDOM_DELAY können Sie eine zusätzliche zufällige Verzögerung für die Jobstarts festlegen: Wenn Sie die Variable auf eine Zahl *t* setzen, dann wird eine zufällige Anzahl von Minuten zwischen 0 und *t* zur in der Job-Beschreibung angegebenen Verzögerung addiert. Mit START_HOURS_RANGE können Sie einen Bereich von Stunden (auf der Uhr) angeben, während dem die Jobs gestartet werden sollen. Etwas wie

```
START_HOURS_RANGE=10-12
```

erlaubt Job-Starts nur zwischen 10 und 12 Uhr. Wie cron schickt anacron die Ausgabe von Jobs an die Adresse, die mit der Variablen MAILTO angegeben wurde, ersatzweise den Benutzer, der anacron ausführt (in der Regel root).

Normalerweise führt anacron die Jobs unabhängig voneinander und ohne Rücksicht auf Überlappungen aus. Mit der Option -s werden die Jobs »seriell« ausgeführt, das heißt, anacron startet einen neuen Job erst, wenn der vorige fertig ist.

Im Gegensatz zu cron ist anacron kein Hintergrunddienst, sondern wird beim Systemstart angestoßen, um allfällige liegengebliebene Jobs auszuführen (die Verzögerung in Minuten dient dazu, die Jobs zu verschieben, bis das System vernünftig läuft, um den Systemstart nicht noch weiter zu verlangsamen). Später können Sie anacron einmal am Tag mit cron ausführen, um dafür zu sorgen, das es auch funktioniert, wenn das System einmal unvorhergesehen lange läuft.

Sie können durchaus cron und anacron auf demselben System installiert haben. Während anacron normalerweise die eigentlich für cron gedachten Jobs in /etc/cron.daily, /etc/cron.weekly und /etc/cron.monthly ausführt, wird dafür gesorgt, dass anacron nichts macht, wenn cron aktiv ist. (Siehe hierzu Übung 19.13.)

Übungen

19.12 [!2] Überzeugen Sie sich, dass anacron so funktioniert wie behauptet. (*Tipp:* Wenn Sie nicht tagelang warten wollen, dann manipulieren Sie dazu kreativ die Zeitstempel-Dateien in /var/spool/anacron.)

19.13 [2] Wie vermeidet man, dass auf einem lang laufenden System, das sowohl cron und anacron installiert hat, anacron dem regulären cron in die Quere kommt? (*Tipp:* Untersuchen Sie den Inhalt von /etc/cron.daily & Co.)

Kommandos in diesem Kapitel

anacron Führt periodische Jobs aus, wenn der Computer nicht immer läuft anacron(8) 328

at Registriert Kommandos zur zeitversetzten Ausführung at(1) 322

atd Daemon für die zeitversetzte Ausführung von Kommandos über at atd(8) 324

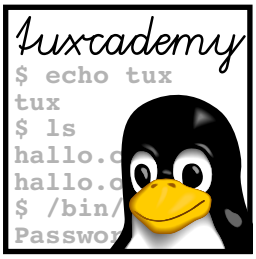
atq Programm zur Abfrage der Warteschlangen für zeitversetzte Kommandoausführung atq(1) 324

atrm Storniert zeitversetzt auszuführende Kommandos atrm(1) 324

crontab Programm zur Verwaltung von regelmäßig auszuführenden Kommandos crontab(1) 328

Zusammenfassung

- Mit `at` kann man Kommandos registrieren, die zu einem festen späteren Zeitpunkt ausgeführt werden.
- Das Kommando `batch` erlaubt die Ausführung von Kommandos, wenn das System dafür Kapazität frei hat.
- `atq` und `atrm` dienen zur Verwaltung der Auftragswarteschlangen. Der Daemon `atd` kümmert sich um die tatsächliche Ausführung der Aufträge.
- Der Zugang zu `at` und `batch` wird über die Dateien `/etc/at.allow` und `/etc/at.deny` gesteuert.
- Der `cron`-Daemon dient zur periodischen Wiederholung von Kommandos.
- Benutzer können eigene Aufgabenlisten (`crontabs`) haben.
- Eine systemweite Aufgabenliste existiert in `/etc/crontab` und – bei vielen Distributionen – im Verzeichnis `/etc/cron.d`.
- Der Zugriff auf `cron` wird über die Dateien `/etc/cron.allow` und `/etc/cron.deny` ähnlich geregelt wie der Zugriff auf `at`.
- Das Kommando `crontab` dient zur Verwaltung von `crontab`-Dateien.



20

Systemprotokollierung

Inhalt

20.1 Das Problem 334
20.2 Der Syslog-Daemon 334
20.3 Die Protokolldateien 337
20.4 Protokoll des Systemkerns 338
20.5 Erweiterte Möglichkeiten: Rsyslog 339
20.6 Die »nächste Generation«: Syslog-NG 343
20.7 Das Programm `logrotate` 348

Lernziele

- Den Syslog-Daemon kennen und konfigurieren können
- Protokolldateien mit `logrotate` verwalten können
- Verstehen, wie der Linux-Kernel mit Nachrichten umgeht

Vorkenntnisse

- Grundkenntnisse der Komponenten eines Linux-Systems
- Umgang mit Konfigurationsdateien

20.1 Das Problem

Anwendungsprogramme haben ihren Benutzern von Zeit zu Zeit etwas mitzuteilen. Der Vollzug einer Aufgabe, aber auch eine Fehlersituation oder Warnung müssen in geeigneter Form gemeldet werden. Textorientierte Programme geben entsprechende Nachrichten auf ihrem »Terminal« aus; Programme mit grafischer Oberfläche verwenden zum Beispiel »Alarmboxen« oder Statuszeilen mit wechselndem Inhalt.

Der Betriebssystemkern und die diversen im Hintergrund laufende System- und Netzwerkdienste sind dagegen mit keinem Benutzerterminal direkt verbunden. Wenn ein solcher Prozess eine Nachricht ausgeben will, dann tut er das normalerweise auf dem Bildschirm der Systemkonsole direkt; unter X11 erscheinen solche Nachrichten im `xconsole`-Fenster.

Im Mehrbenutzerbetrieb ist die Methode, eine Systemmeldung ausschließlich als Fehlermeldung auf die Systemkonsole zu schreiben, nicht ausreichend. Zum einen ist nicht sichergestellt, dass die Meldung auch von `root` gelesen werden, zum anderen lassen sich die Bildschirmmeldungen nicht sichern und gehen leicht verloren.

20.2 Der Syslog-Daemon

Die Lösung dieses Problems bietet der Syslog-Daemon oder `syslogd`. Anstelle der direkten Ausgabe einer Meldung können Systemmeldungen von spezieller Bedeutung mit der `syslog()`-Funktion ausgegeben werden, die Bestandteil der Standard-C-Bibliothek für Linux ist. Solche Meldungen werden vom `syslogd` entgegengenommen. Der `syslogd` erhält die Nachrichten über das Socket `/dev/log`.

Um Nachrichten vom Systemkern kümmert sich eigentlich ein anderes Programm namens `klogd`. Dieses Programm verarbeitet die Nachrichten vor und leitet sie normalerweise an den `syslogd` weiter. Siehe hierzu Abschnitt 20.4

Protokoll

Der `syslogd` erweist sich bei der Fehlersuche als sehr nützlich. Er protokolliert die verschiedenen Systemmeldungen und ist – wie der Name schon andeutet – ein Daemon-Programm. Der `syslogd` wird in der Regel beim Booten über ein Init-Skript gestartet. Erhält der `syslogd` Meldungen, so schreibt er sie z. B. in eine Datei oder sendet sie über das Netz weiter an einen Rechner, der ein zentrales Protokoll verwaltet.

Die gängigen Distributionen (Debian GNU/Linux, Ubuntu, Red Hat Enterprise Linux, Fedora, openSUSE, …) benutzen alle schon seit mehr oder weniger langer Zeit ein Paket namens »Rsyslog«, das eine modernere Implementierung eines `syslogd` mit mehr Konfigurationsmöglichkeiten enthält. Die zusätzlichen Möglichkeiten sind für den Einstieg und die Zwecke der LPI-Prüfung allerdings nicht wirklich wichtig. Wenn Sie den Anfang der Rsyslog-Konfiguration überlesen, entspricht das Ende im Wesentlichen dem, was in diesem Kapitel besprochen wird. Mehr über Rsyslog steht in Abschnitt 20.5.

Gewisse Versionen der SUSE-Distributionen, insbesondere des SUSE Linux Enterprise Servers (SLES) verwenden statt `syslogd` das Paket Syslog-NG, das substanziell anders konfiguriert wird. Für die LPI-Prüfung müssen Sie wissen, dass Syslog-NG existiert und was es in etwa macht; siehe dazu Abschnitt 20.6.

`/etc/syslog.conf`

Der Administrator bestimmt, was genau mit den einzelnen Meldungen geschehen soll. In der Konfigurationsdatei `/etc/syslog.conf` ist festgelegt, welche Meldungen wohin geschrieben werden sollen.

Tabelle 20.1: Kategorien für den `syslogd`

Kategorie	Bedeutung
`authpriv`	Vertrauliche Meldungen der Sicherheitsdienste
`cron`	Meldungen von `cron` und `at`
`daemon`	Meldungen von Daemon-Programmen ohne eigene Kategorie
`ftp`	Meldungen des FTP-Daemons
`kern`	Systemmeldungen aus dem Betriebssystemkern
`lpr`	Meldungen des Druckersystems
`mail`	Meldungen des Mailsystems
`news`	Meldungen des Usenet-News-Systems
`syslog`	Meldungen des `syslogd`
`user`	Meldungen, die mit Benutzern zu tun haben
`uucp`	Meldungen des UUCP-Systems
`local`*r*	($0 \leqslant r \leqslant 7$) Frei verwendbar für lokale Nachrichten

Der Rsyslog verwendet standardmäßig `/etc/rsyslog.conf` als Konfigurationsdatei. Diese ist in weiten Teilen kompatibel zu dem, was `syslogd` benutzen würde. Ignorieren Sie einfach alle Zeilen, die mit einem Dollarzeichen (`$`) anfangen.

Die Konfigurationsdatei besteht aus zwei Spalten und könnte folgendermaßen aussehen:

```
kern.warn;*.err;authpriv.none    /dev/tty10
kern.warn;*.err;authpriv.none   |/dev/xconsole
*.emerg                          *
*.=warn;*.=err                  -/var/log/warn
*.crit                           /var/log/warn
*.*;mail.none;news.none         -/var/log/messages
```

Die erste Spalte jeder Zeile bestimmt, welche Meldungen ausgewählt werden sollen, und die zweite Spalte gibt an, wohin die Meldungen geschrieben werden. Die erste Spalte hat das Format:

⟨*Kategorie*⟩.⟨*Priorität*⟩[;⟨*Kategorie*⟩.⟨*Priorität*⟩]…

Kategorien

Dabei bezeichnet die ⟨*Kategorie*⟩ das Systemprogramm oder die Komponente des Systems, die die Meldung verursacht. Das kann zum Beispiel der Mail-Daemon sein, der Kernel selbst oder auch Programme, die den Zugang zum System kontrollieren. Tabelle 20.1 zeigt die gültigen Kategorien. Wenn Sie anstelle der Kategorie einen Stern (»*«) setzen, steht dieser als Platzhalter für alle Kategorien. Es ist nicht ohne Weiteres möglich, eigene zusätzliche Kategorien zu definieren; die »lokalen« Kategorien `local0` bis `local7` sollten aber für die meisten Zwecke ausreichen.

Prioritäten

Die ⟨*Priorität*⟩ gibt an, wie schwerwiegend die Meldung ist. Die gültigen Prioritäten stehen in Tabelle 20.2.

Wer legt eigentlich fest, in welche Kategorie und mit welcher Priorität eine Meldung verschickt wird? Die Lösung ist simpel: Derjenige, der die `syslog()`-Funktion verwendet, nämlich der Entwickler der jeweiligen Software, muss Kategorie und Priorität der Meldungen seiner Routinen festlegen. Viele Programme lassen es zu, dass der Administrator zumindest die Kategorie der Meldungen umdefiniert.

Auswahlkriterien

Ein Auswahlkriterium der Form `mail.info` bedeutet »alle Meldungen des Mail-Daemon mit der Priorität `info` und höher«. Möchten Sie nur die Meldungen einer einzigen Prioritätsstufe erfassen, dann können Sie das über ein Auswahlkriterium wie `mail.=info` tun. Der Stern (»*«) steht für alle Prioritäten (Sie könnten

Tabelle 20.2: Prioritäten für den `syslogd` (nach aufsteigender Dringlichkeit)

Priorität	Bedeutung
`none`	Keine Priorität im eigentlichen Sinne – dient dazu, alle Nachrichten einer Herkunftskategorie auszuschließen
`debug`	Mitteilung innerer Programmzustände bei der Fehlersuche
`info`	Protokollierung des normalen Betriebsgeschehens
`notice`	Dokumentation besonders bemerkenswerter Situationen im Rahmen des normalen Betriebs
`warning`	(oder `warn`) Warnung über Zustände, die nicht gravierend sind, aber nicht mehr zum normalen Betrieb gehören
`err`	Fehlermeldungen aller Art
`crit`	Kritische Fehlermeldungen (die Grenze zur Priorität `err` ist nicht definiert)
`alert`	»Alarmierende« Nachrichten, die sofortiges Eingreifen erfordern
`emerg`	Die letzten Meldungen vor dem Absturz

auch »`debug`« angeben). Ein vorgestelltes `!` bedeutet Negation: `mail.!info` deselektiert Meldungen des Mail-Daemons mit einer Priorität von `info` und höher; dies ist vor allem in Kombinationen wie `mail.*;mail.!err` sinnvoll, um gewisse Meldungen niedriger Prioritätsstufen auszuwählen. `!` und `=` können kombiniert werden; `mail.!=info` deselektiert (genau) die Nachrichten des Mail-Daemons mit der Priorität `info`.

Mehrere Kategorien – selbe Priorität

Sie können auch mehrere Kategorien mit derselben Priorität in der Form `mail,news.info` angeben; dieser Ausdruck selektiert Meldungen der Priorität `info` und höher, die zu den Kategorien `mail` oder `news` gehören.

Aktionen

Nun zur rechten Spalte, dem Ziel der Meldungen. Die Nachrichten können auf verschiedene Weise verarbeitet werden:

- Sie können in eine *Datei* geschrieben werden. Dazu wird der absolute Dateiname angegeben. Normalerweise schreibt `syslogd` seine Meldungen *synchron*, also ohne Zwischenspeicherung im Platten-Cache des Linux-Kerns, auf die Platte, damit bei einem Systemabsturz möglichst viele Indizien gesichert sind. Steht vor der Pfadangabe ein »`-`«, ist eine Pufferung im Platten-Cache möglich. Das geht schneller, aber heißt, dass Sie im Fall eines Absturzes möglicherweise ausstehende Protokollnachrichten verlieren – für relativ unwichtige Nachrichten, etwa solche der Priorität `notice` und darunter, oder für Nachrichten »geschwätziger« Kategorien wie Mail und News ist das aber vielleicht nicht wirklich ein Problem.

  Der Dateiname darf sich auch auf ein Gerät (etwa `/dev/tty10` im obigen Beispiel) beziehen.

- Protokollnachrichten können in eine *benannte Pipe* (FIFO) geschrieben werden. Dazu muss der FIFO-Name als absoluter Pfad mit einem davorstehenden »`|`« angegeben werden. Ein solcher FIFO ist `/dev/xconsole`.

- Sie können *über das Netz* an einen anderen `syslogd` weitergeleitet werden. Dazu wird der Name oder die IP-Adresse des Zielrechners mit einem vorgestellten `@`-Zeichen angegeben. Das ist besonders sinnvoll, wenn es zu einem kritischen Systemzustand kommt, bei dem der Zugriff auf die lokale Nachrichtendatei nicht mehr möglich ist; um das Protokoll dem Zugriff allfälliger Cracker zu entziehen, die gerne ihre Spuren verwischen würden; oder um die Protokollnachrichten aller Rechner im Netz auf einem Rechner zu sammeln und gemeinsam auszuwerten.

  Auf dem Zielrechner muss der `syslogd` mit der Option `-r` (engl. *remote*) gestartet worden sein, damit er Meldungen anderer Rechner annimmt. Wie Sie das am geschicktesten machen, hängt von Ihrer Distribution ab.

- Sie können *direkt an Benutzer* geschickt werden. Die Benutzernamen müssen dazu in einer durch Kommata getrennten Liste aufgeführt werden. Die Nachricht wird den aufgelisteten Benutzern auf dem Terminal angezeigt, sofern diese im Augenblick des Eingangs der Nachricht angemeldet sind.
- Sie können *an alle eingeloggten Benutzer* geschickt werden, indem ein Stern »*« anstelle des Loginnamens angegeben wird.

In der Regel enthält Ihr System nach der Installation bereits einen laufenden `syslogd` und die Datei `/etc/syslog.conf` in einer brauchbaren Konfiguration. Falls Sie weitere Meldungen protokollieren lassen möchten, z. B. weil spezielle Probleme auftreten, sollten Sie die Datei `syslog.conf` editieren und dann den `syslogd` mit dem Signal `SIGHUP` anweisen, seine Konfigurationsdatei erneut zu lesen. Konfiguration ändern

Testen können Sie den `syslogd`-Mechanismus übrigens mit dem Programm `logger`. Ein Aufruf der Form

```
$ logger -p local0.err -t TEST "Hallo Welt"
```

produziert eine Protokollnachricht der Form

```
Aug  7 18:54:34 red TEST: Hallo Welt
```

Die meisten modernen Programmiersprachen enthalten eine Zugriffsmöglichkeit auf die `syslog()`-Funktion.

Übungen

20.1 [!2] Finden Sie heraus, wann zuletzt jemand per `su` auf Ihrem Rechner die Identität von `root` angenommen hat.

20.2 [!2] Konfigurieren Sie den `syslogd` so, dass er zusätzlich zur aktuellen Konfiguration alle (!) Meldungen in eine neue Datei `/var/log/test` protokolliert. Testen Sie das Ergebnis.

20.3 [3] (Braucht zwei Rechner und eine funktionierende Netzverbindung.) Konfigurieren Sie den `syslogd` auf einem Rechner so, dass er Protokollnachrichten über das Netz entgegennimmt. Konfigurieren Sie den `syslogd` auf dem anderen Rechner so, dass er Nachrichten der Kategorie `local0` auf den ursprünglichen Rechner schickt. Testen Sie die Konfiguration.

20.4 [2] Wie können Sie einen Protokollmechanismus implementieren, der sicher vor Angreifern ist, die den protokollierenden Rechner unter ihre Kontrolle bringen? (Ein Angreifer kann immer verhindern, dass weitere Protokollnachrichten geschrieben werden. Wir möchten gerne erreichen, dass der Angreifer bereits geschriebene Protokollnachrichten nicht verfälschen oder löschen kann.)

20.3 Die Protokolldateien

Protokolldateien werden in der Regel unter `/var/log` abgelegt. Die Namen der einzelnen Dateien variieren – prüfen Sie gegebenenfalls die `syslog.conf`-Datei. Hier sind einige Beispiele: /var/log

Debian GNU/Linux sammelt sämtliche Nachrichten außer denen, die mit Authentisierung zu tun haben, in der Datei `/var/log/syslog`. Für die Kategorien `auth`, `daemon`, `kern`, `lpr`, `mail`, `user` und `uucp` gibt es jeweils eigene Protokolldateien `auth.log` usw. Das Mailsystem verwendet außerdem die Dateien

mail.info, mail.warn und mail.err, die jeweils nur die Nachrichten der Prioritäten info usw. (und darüber) enthalten. Debug-Nachrichten aller Kategorien außer authpriv, news und mail landen in /var/log/debug und Nachrichten der Prioritäten info, notice und warn in allen Kategorien außer den eben genannten sowie cron und daemon in /var/log/messages.


Die Voreinstellungen von Ubuntu entsprechen denen von Debian GNU/Linux.


Bei den Red-Hat-Distributionen kommen alle Meldungen mit der Priorität info oder darüber außer denen von authpriv und cron in die Datei /var/log/messages, die Meldungen von authpriv in die Datei /var/log/secure und die von cron nach /var/log/cron. Alle Nachrichten des Mailsystems landen in /var/log/maillog.


OpenSUSE protokolliert alle Nachrichten außer denen von iptables und den Kategorien news und mail nach /var/log/messages. Nachrichten von iptables landen in /var/log/firewall. Nachrichten, die nicht von iptables kommen und außerdem die Priorität warn, err oder crit haben, werden ferner nach /var/log/warn geschrieben. Außerdem gibt es die Protokolldateien /var/log/localmessages für Nachrichten aus den local*-Kategorien, /var/log/NetworkManager für Nachrichten des NetworkManager-Programms und /var/log/acpid für Nachrichten des ACPI-Daemons. Das Mailsystem schreibt sein Protokoll sowohl nach /var/log/mail (alle Nachrichten) sowie außerdem in die Dateien mail.info, mail.warn und mail.err (letztere für die Prioritäten err und crit), das Newssystem schreibt sein Protokoll hingegen nach news/news.notice, news/news.err und news/news.crit (je nach Priorität) – eine Gesamtprotokolldatei gibt es bei News nicht. (Wenn Sie das inkonsistent und verwirrend finden, sind Sie nicht alleine.)


Einige Protokolldateien enthalten auch Meldungen, die die Privatsphäre der Benutzer betreffen, sie sollten deshalb nur für root lesbar sein. In den meisten Fällen sind die Distributionen lieber vorsichtig und behandeln die Zugriffsrechte auf Protokolldateien eher restriktiv.

Protokolldateien anschauen

Die vom syslogd erzeugten Logdateien können Sie mit less anschauen. Bei langen Dateien bietet sich tail an (evtl. mit der Option -f). Außerdem gibt es spezielle Programme zum Beobachten von Logdateien, die bekanntesten sind logsurfer und xlogmaster.

Meldungen

Die von syslogd protokollierten Meldungen enthalten normalerweise das Datum, den Rechnernamen, einen Hinweis auf den Prozess oder die Komponente, die die Meldung verursacht hat, sowie die Meldung selbst. Typische Meldungen sehen beispielsweise so aus:

```
Mar 31 09:56:09 red modprobe: modprobe: Can't locate ...
Mar 31 11:10:08 red su: (to root) user1 on /dev/pts/2
Mar 31 11:10:08 red su: pam-unix2: session started for ...
```

Eine zu groß gewordene Logdatei können Sie mit rm löschen oder durch Umbenennen mit der Erweiterung .old erst einmal sichern. Beim nächsten Neustart von syslogd wird gegebenenfalls eine neue Protokolldatei angelegt. Es geht aber auch komfortabler, wie der nächste Abschnitt zeigt.

20.4 Protokoll des Systemkerns

Der Kernel schickt seine Nachrichten nicht an den syslogd, sondern stellt sie in einen internen »Ringpuffer«. Von dort können sie auf verschiedene Weise ausgelesen werden – über einen speziellen Systemaufruf oder die »Datei« /proc/kmsg. Traditionell kümmert sich ein Programm namens klogd darum, /proc/kmsg zu lesen und die Nachrichten an den syslogd weiterzuleiten.

Der rsyslog kommt ohne ein separates `klogd`-Programm aus, weil er sich um Protokollnachrichten des Kernels direkt selber kümmern kann. Wenn Sie auf Ihrem System also keinen `klogd` finden können, dann könnte das daran liegen, dass es rsyslog benutzt.

Beim Systemstart stehen `syslogd` und gegebenenfalls `klogd` nicht sofort zur Verfügung – sie müssen ja als Programme gestartet werden und können so die Meldungen des Kernels beim Start nicht direkt verarbeiten. Das Kommando `dmesg` macht es möglich, rückwirkend auf den Inhalt des Puffers zuzugreifen und das Startprotokoll einzusehen. Mit einem Kommando wie

```
# dmesg >boot.msg
```

können Sie die Nachrichten in eine Datei schreiben und sie einem Kernelentwickler zuschicken.

Mit dem `dmesg`-Kommando können Sie den Kernel-Ringpuffer auch löschen (Option `-c`) und die Priorität setzen, ab der Nachrichten direkt auf die Konsole ausgegeben werden (Option `-n`). Kernel-Nachrichten haben Prioritäten von 0 bis 7, die mit den `syslogd`-Prioritäten `emerg` bis `debug` korrespondieren. Mit dem Kommando

```
# dmesg -n 1
```

werden zum Beispiel nur noch `emerg`-Nachrichten direkt auf die Konsole ausgegeben. Über `/proc/kmsg` werden auf jeden Fall immer alle Nachrichten protokolliert – hier ist es die Aufgabe von nachgeschalteter Software wie `syslogd`, die gewünschten Nachrichten auszusortieren.

Übungen

20.5 [2] Was verrät die Ausgabe von `dmesg` Ihnen über die Hardware in Ihrem Rechner?

20.5 Erweiterte Möglichkeiten: Rsyslog

Rsyslog von Rainer Gerhards hat bei den meisten gängigen Linux-Distributionen inzwischen den traditionellen BSD-`syslogd` ersetzt. Das Ziel von rsyslog ist neben größerer Effizienz auch die Unterstützung von diversen Quellen und Senken für Protokollnachrichten. Zum Beispiel schreibt er Nachrichten nicht nur in Textdateien und auf Terminals, sondern auch in eine breite Auswahl von Datenbanken.

Laut seiner eigenen Webseite steht »rsyslog« für *rocket-fast syslog*. Natürlich darf man auf solche Selbstanpreisungen nicht unbedingt viel geben, aber in diesem Fall ist das Eigenlob nicht völlig unangebracht.

Die grundlegenden Ideen hinter rsyslog sind in etwa wie folgt:

- »Quellen« reichen Nachrichten an »Regelsätze« weiter. Es gibt standardmäßig einen eingebauten Regelsatz (`RSYSLOG_DefaultRuleset`), aber Sie als Benutzer können weitere definieren.
- Jeder Regelsatz enthält beliebig viele Regeln (auch keine, selbst wenn das keinen großen Sinn ergibt).
- Eine Regel besteht aus einem »Filter« und einer »Aktionsliste«. Filter treffen Ja-Nein-Entscheidungen darüber, ob die zugehörige Aktionsliste ausgeführt wird.

- Für jede Nachricht werden alle Regeln im Regelsatz in ihrer Reihenfolge von der ersten zur letzten ausgeführt (und keine anderen). Es werden immer alle Regeln ausgeführt, egal wie die Filter-Entscheidungen ausfallen, wobei es auch die Aktion »Bearbeitung abbrechen« gibt.
- Eine Aktionsliste kann viele Aktionen enthalten (mindestens eine). Innerhalb einer Aktionsliste sind keine weiteren Filter möglich. Die Aktionen bestimmen, was mit passenden Protokollnachrichten geschieht.
- Das genaue Aussehen von Protokollnachrichten in der Ausgabe läßt sich über »Templates« steuern.

Die Konfiguration von rsyslog steht in der Datei `/etc/rsyslog.conf`. Innerhalb dieser Datei können Sie parallel drei verschiedene Arten von Konfigurationseinstellungen verwenden:

- Die traditionelle Syntax von `/etc/syslog.conf` (»sysklogd«).
- Eine veraltete Syntax von rsyslog (»legacy rsyslog«). Diese erkennen Sie daran, dass Kommandos mit einem Dollarzeichen (`$`) anfangen.
- Die aktuelle Syntax von rsyslog (»RainerScript«). Sie ist für komplexe Situationen am besten geeignet.

Die ersten beiden Geschmacksrichtungen sind zeilenbasiert. In der aktuellen Syntax sind Zeilenumbrüche irrelevant.

Für sehr einfache Anwendungen können – und sollten! – Sie die sysklogd-Syntax verwenden (wie wir sie in den vorigen Abschnitten besprochen haben). Wenn Sie Konfigurationsparameter einstellen oder aufwendigen Kontrollfluss ausdrücken wollen, ist RainerScript empfehlenswerter. Um die veraltete rsyslog-Syntax sollten Sie einen Bogen machen (auch wenn diverse Linux-Distributionen das in ihren Voreinstellungen nicht tun), bis darauf, dass manche Eigenschaften von rsyslog nur in dieser Syntax zugänglich sind.

Wie üblich werden Leerzeilen und Kommentarzeilen ignoriert. Als Kommentarzeilen gelten sowohl Zeilen (oder Teile von Zeilen), die mit `#` anfangen (der Kommentar geht dann bis zum jeweiligen Zeilenende) als auch C-artige Kommentare, die von einem `/*` ohne Ansehen von Zeilenumbrüchen bis zu einem `*/` reichen.

C-artige Kommentare dürfen Sie nicht verschachteln[1], aber innerhalb von C-artigen Kommentaren können `#`-Kommentare auftauchen. Damit sind C-artige Kommentare vor allem dazu nützlich, große Stücke einer Konfigurationsdatei »auszukommentieren«, also für rsyslog unsichtbar zu machen.

Rsyslog bietet diverse Möglichkeiten, die über die des BSD-`syslogd` hinausgehen. Sie können zum Beispiel erweiterte Filterausdrücke für Nachrichten verwenden:

```
:msg, contains, "FOO"    /var/log/foo.log
```

Die erweiterten Filterausdrücke bestehen immer aus einem Doppelpunkt am linken Rand, einer »Property«, die rsyslog der Nachricht entnimmt, einem Filteroperator (hier `contains`) und einem Suchbegriff. In unserem Beispiel werden alle Protokollnachrichten, deren Text die Zeichenkette `FOO` enthält, in die Datei `/var/log/foo.log` geschrieben.

Zu den »Properties«, die Sie verwenden können, gehören außer `msg` (der Protokollnachricht im eigentlichen Sinne) zum Beispiel `hostname` (der Name des Rechners, von dem die Nachricht ausging), `fromhost` (der Name des Rechners, von dem rsyslog die Nachricht bekommen hat), `pri` (die Kategorie und

[1] Das dürfen Sie in C auch nicht, also sollte es Sie nicht übermäßig belästigen.

Priorität der Nachricht als undekodierte Zahl), `pri-text` (Kategorie und Priorität als Text, mit der Zahl dahinter, etwa »`local0.err<133>`«), `syslogfacility` und `syslogseverity` sowie `syslogfacility-text` und `syslogseverity-text` zum gezielten Zugriff auf Kategorie und Priorität, `timegenerated` (wann die Nachricht empfangen wurde) oder `inputname` (der rsyslog-Modulname der Quelle der Nachricht). Es gibt noch einige mehr; schauen Sie sich die rsyslog-Dokumentation an.

Die erlaubten Vergleichsoperationen sind `contains`, `isequal`, `startswith`, `regex` und `ereregex`. Diese sprechen für sich selbst, bis auf die beiden letzten – `regex` betrachtet seinen Parameter als einfachen und `ereregex` als »erweiterten« regulären Ausdruck gemäß POSIX. Groß- und Kleinschreibung wird bei allen Vergleichsoperationen beachtet.

Der `startswith`-Vergleich ist nützlich, weil er deutlich effizienzer ist als ein am Anfang der Nachricht verankerter regulärer Ausdruck (jedenfalls solange Sie nur nach einer konstanten Zeichenkette suchen). Allerdings sollten Sie aufpassen, denn was Sie für den Anfang der Nachricht halten und was rsyslog darüber denkt, können durchaus zwei Paar Schuhe sein. Wenn rsyslog über den syslog-Dienst eine Nachricht empfängt, sieht diese zum Beispiel aus wie

```
<131>Jul 22 14:25:50 root: error found
```

Für rsyslog beginnt `msg` aber nicht, wie man naiverweise denken könnte, mit dem `e` von `error`, sondern mit dem Leerzeichen davor. Wenn Sie also nach Nachrichten suchen, die mit `error` anfangen, sollten Sie

```
:msg, startswith, " error"   /var/log/error.log
```

schreiben.

Es gibt eine nette Erweiterung auf der »Aktionsseite« einfacher Regeln: Sie hatten ja schon beim traditionellen `syslogd` gesehen, dass Sie mit einem Eintrag wie

```
local0.*       @red.example.com
```

Protokollnachrichten über das (UDP-basierte) syslog-Protokoll an einen entfernten Rechner weiterreichen können. Bei rsyslog können Sie auch

```
local0.*       @@red.example.com
```

schreiben, damit die Übertragung per TCP erfolgt. Das ist potentiell zuverlässiger, vor allem, wenn Firewalls im Spiel sind.

Am anderen Ende der TCP-Verbindung muss natürlich ein geeignet konfigurierter rsyslog lauschen. Das können Sie zum Beispiel über

```
module(load="imtcp" MaxSessions="500")
input(type="imtcp" port="514")
```

ermöglichen. In der veralteten Syntax ist das

```
$ModLoad imtcp
$InputTCPMaxSessions 500
$InputTCPServerRun 514
```

Achten Sie darauf, dass nur der UDP-Port 514 offiziell für das syslog-Protokoll reserviert ist. Der TCP-Port 514 ist eigentlich anderweitig vergeben[2]. Sie können gegebenenfalls einen anderen Port festlegen:

```
local0.*       @@red.example.com:10514
```

(und das funktioniert, wenn nötig, auch für UDP). Die serverseitig nötigen Änderungen finden Sie sicher leicht selber raus.

Die nächste Komplexitätsstufe sind Filter auf der Basis von Ausdrücken, die beliebige Boolesche, arithmetische und Zeichenketten-Operationen umfassen können. Diese beginnen immer mit einem `if` ganz am linken Rand einer neuen Zeile:

```
if $syslogfacility-text == "local0" and $msg startswith " FOO" ▷
 ◁ and ($msg contains "BAR" or $msg contains "BAZ") ▷
 ◁ then /var/log/foo.log
```

(in Ihrer Datei sollte das alles in einer Zeile stehen). Mit dieser Regel werden Nachrichten der Kategorie `local0` in `/var/log/foo.log` geschrieben, die mit `FOO` anfangen und außerdem `BAR` oder `BAZ` (oder beides) enthalten. (Achten Sie auf die Dollarzeichen vor den Namen der Properties.)

Rsyslog unterstützt eine große Anzahl von Modulen, die bestimmen, was mit Protokollnachrichten passieren soll. Sie könnten beispielsweise wichtige Nachrichten per E-Mail verschicken. Dazu können Sie in `/etc/rsyslog.conf` etwas schreiben wie

```
module(load="ommail")
template(name="mailBody" type="string" string="ALERT\\r\\n%msg%")
if $msg contains "disk error" then {
    action(type="ommail" server="mail.example.com" port="25"
           mailfrom="rsyslog@example.com" mailto="admins@example.com"
           subject.text="disk error detected"
           body.enable="on" template="mailBody"
           action.execonlyonceeveryinterval="3600")
}
```

Wenn Sie eine ältere Version von rsyslog haben (vor 8.5.0), müssen Sie die veraltete Syntax nehmen, um das `ommail`-Modul zu konfigurieren. Das sieht dann zum Beispiel so aus:

```
$ModLoad ommail
$ActionMailSMTPServer mail.example.com
$ActionMailFrom rsyslog@example.com
$ActionMailTo admins@example.com
$template mailSubject,"disk error detected"
$template mailBody,"ALERT\\r\\n%msg%"
$ActionMailSubject mailSubject
$ActionExecOnlyOnceEveryInterval 3600
if $msg contains "disk error" then :ommail:;mailBody
$ActionExecOnlyOnceEveryInterval 0q
```

Die SMTP-Implementierung von rsyslog ist einigermaßen primitiv, da sie keine Verschlüsselung oder Authentisierung zuläßt. Das heißt, der Mailserver, den Sie in der rsyslog-Konfiguration angeben, muss in der Lage sein, auch ohne Verschlüsselung oder Authentisierung Mail von rsyslog anzunehmen.

[2] … auch wenn sich heutzutage niemand mehr für den Remote-Shell-Dienst interessiert. Niemand Vernünftiges jedenfalls.

Ach ja: Rsyslog kümmert sich direkt um Protokollnachrichten aus dem Linux-Kern. Dazu müssen Sie lediglich das `imklog`-Eingabemodul aktivieren:

```
module(load="imklog")
```

oder (veraltete Syntax)

```
$ModLoad imklog
```

Ein separater `klogd`-Prozess ist nicht erforderlich.

Detaillierte Informationen über rsyslog finden Sie zum Beispiel in der Online-Dokumentation [rsyslog].

Übungen

20.6 [!3] (Falls Ihre Distribution nicht sowieso schon rsyslog benutzt.) Installieren Sie rsyslog und erstellen Sie eine Konfiguration, die Ihrer bisherigen `syslogd`-Konfiguration möglichst nahe kommt. Testen Sie sie zum Beispiel mit `logger`. Wo sehen Sie Verbesserungsmöglichkeiten?

20.7 [2] PAM, das Anmelde- und Authentisierungssystem, protokolliert An- und Abmeldevorgänge in der folgenden Form:

```
kdm: :0[5244]: (pam_unix) session opened for user hugo by (uid=0)
◁◁◁◁◁
kdm: :0[5244]: (pam_unix) session closed for user hugo
```

Konfigurieren Sie rsyslog so, dass jedesmal, wenn sich ein bestimmter Benutzer (etwa Sie) an- oder abmeldet, eine Meldung auf dem Terminal des Systemverwalters (`root`) erscheint, falls dieser angemeldet ist. (*Tipp:* Die PAM-Nachrichten erscheinen in der Kategorie `authpriv`.)

20.8 [3] (Arbeiten Sie für diese Aufgabe gegebenenfalls mit einem anderen Kursteilnehmer zusammen.) Konfigurieren Sie rsyslog so, dass alle Protokollnachrichten von einem Rechner über eine TCP-Verbindung an einen anderen Rechner weitergeleitet werden. Testen Sie die Verbindung mit `logger`.

20.6 Die »nächste Generation«: Syslog-NG

Syslog-NG (»NG« für »New Generation«) ist eine kompatible, aber erweiterte Neuimplementierung eines Syslog-Daemons von Balazs Scheidler. Einige der Hauptvorteile von Syslog-NG gegenüber dem herkömmlichen `syslogd` sind: Hauptvorteile

- Filterung von Nachrichten auf Inhaltsbasis (nicht nur Kategorien und Prioritäten)
- Verknüpfung mehrerer Filter möglich
- Ausgefeilteres Ein-/Ausgabesystem, inklusive Weiterleitung über TCP und in Unterprozesse

Das Programm selbst heißt `syslog-ng`.

Für die Syslog-Clients ändert sich zunächst nichts; Sie können einen `syslogd` also problemlos durch Syslog-NG ersetzen.

Sie finden Informationen über Syslog-NG in den Handbuchseiten sowie unter [syslog-ng]. Dort gibt es Dokumentation und auch eine sehr nützliche FAQ-Sammlung.

Konfigurationsdatei Syslog-NG liest seine Konfiguration aus einer Datei, normalerweise `/etc/syslog-ng/syslog-ng.conf`. Im Gegensatz zum `syslogd` unterscheidet Syslog-NG verschiedene Arten von Einträgen in seiner Konfigurationsdatei:

Arten von Einträgen

Globale Optionen Diese Einstellungen gelten für alle Nachrichtenquellen oder den Syslog-NG-Daemon selbst.

Quellen von Nachrichten Syslog-NG kann Nachrichten auf diverse Weisen lesen: per Unix-Domain-Socket und UDP wie `syslogd`, aber auch zum Beispiel aus Dateien, FIFOs oder TCP-Sockets. Jede Nachrichtenquelle bekommt einen Namen.

Filter Filter sind Boolesche Ausdrücke auf der Basis interner Funktionen, die sich zum Beispiel auf die Herkunft, Kategorie, Priorität oder auch den textuellen Inhalt einer Protokollnachricht beziehen können. Auch Filter werden benannt.

Senken für Nachrichten Syslog-NG erlaubt alle Protokollierungsmethoden von `syslogd` und noch ein paar mehr.

Protokoll-Pfade Ein »Protokoll-Pfad« verbindet eine oder mehrere Nachrichtenquellen, Filter und Senken zu einem Ganzen: Treffen aus den Nachrichtenquellen Protokollmeldungen ein, auf die der oder die Filter passen, werden sie an die angegebene(n) Senke(n) weitergeleitet. Die Konfigurationsdatei besteht letzten Endes aus einer Reihe solcher Protokoll-Pfade.

Optionen Sie können diverse »globale« Optionen angeben, die das allgemeine Verhalten von Syslog-NG beeinflussen oder Standardwerte für einzelne Nachrichtenquellen oder -senken vorgeben (spezielle Angaben bei den Quellen oder Senken haben Vorrang). Eine komplette Liste steht in der Syslog-NG-Dokumentation. Zu den allgemeinen Optionen gehören diverse Einstellungen für den Umgang mit DNS und das Weiterreichen oder Umschreiben der Absender-Rechnernamen von Nachrichten.

Wenn der Syslog-NG auf Recher *A* eine Nachricht von Rechner *B* empfängt, prüft er die Option `keep_hostnames()`. Hat diese den Wert `yes`, wird *B* als Rechnername für das Protokoll beibehalten. Sonst kommt es auf die Einstellung `chain_hostnames()` an; ist diese `no`, wird *A* als Rechnername ins Protokoll geschrieben, ist sie `yes`, protokolliert Syslog-NG *B*/*A*. Dies ist insbesondere wichtig, wenn das Protokoll an noch einen anderen Rechner weitergeleitet wird.

Quellen von Nachrichten Nachrichtenquellen definieren Sie bei Syslog-NG über das Schlüsselwort `source`. Eine Nachrichtenquelle fasst einen oder mehrere »Treiber« zusammen. Um dasselbe zu erreichen wie ein »normaler« `syslogd`, würden Sie zum Beispiel die Zeile

```
source src { unix-stream("/dev/log"); internal(); };
```

in Ihre Konfiguration aufnehmen; dies weist Syslog-NG an, auf dem Unix-Domain-Socket `/dev/log` zu lauschen. `internal()` bezieht sich auf Nachrichten, die der Syslog-NG selbst erzeugt.

Eine Syslog-NG-Nachrichtenquelle, die der `syslogd`-Option `-r` entspricht, sähe zum Beispiel so aus:

```
source s_remote { udp(ip(0.0.0.0) port(514)); };
```

Da das der Standardwert ist, reicht auch ein

Tabelle 20.3: Filterfunktionen für Syslog-NG

Syntax	Beschreibung
`facility(`⟨*Kategorie*⟩`[,`⟨*Kategorie*⟩ …`])`	Passt auf Nachrichten mit einer der angegebenen Kategorien
`level(`⟨*Priorität*⟩`[,`⟨*Priorität*⟩ …`])`	Passt auf Nachrichten mit einer der angegebenen Prioritäten
`priority(`⟨*Priorität*⟩`[,`⟨*Priorität*⟩ …`])`	Äquivalent zu `level()`
`program(`⟨*reg. Ausdruck*⟩`)`	Passt auf Nachrichten, bei denen der Name des Absenderprogramms auf den ⟨*reg. Ausdruck*⟩ passt
`host(`⟨*reg. Ausdruck*⟩`)`	Passt auf Nachrichten, deren Absenderrechner auf den ⟨*reg. Ausdruck*⟩ passt
`match(`⟨*reg. Ausdruck*⟩`)`	Passt auf Nachrichten, die selbst auf den ⟨*reg. Ausdruck*⟩ passen
`filter(`⟨*Name*⟩`)`	Ruft eine andere Filterregel auf und liefert ihren Wert zurück
`netmask(`⟨*IP-Adresse*⟩`/`⟨*Netzmaske*⟩`)`	Prüft, ob die IP-Adresse im angegebenen Netz liegt

```
source s_remote { udp(); };
```

Mit `ip()` können Sie den Syslog-NG nur auf bestimmten lokalen IP-Adressen lauschen lassen. Mit `syslogd` geht das nicht.

Die folgende Quellenangabe erlaubt es Syslog-NG, das `klogd`-Programm zu ersetzen:

```
source kmsg { file("/proc/kmsg" log_prefix("kernel: ")); };
```

Alle Nachrichtenquellen unterstützen einen weiteren Parameter, `log_msg_size()`, der die maximale Länge einer Nachricht in Bytes angibt.

Filter Filter dienen dazu, Protokollnachrichten auszusortieren oder auf verschiedene Senken zu verteilen. Sie beruhen auf internen Funktionen, die bestimmte Aspekte der Nachrichten betrachten; diese Funktionen können über die logischen Funktionen `and`, `or` und `not` miteinander verknüpft werden. Eine Liste der möglichen Funktionen finden Sie in Tabelle 20.3.

Beispielsweise könnten Sie einen Filter definieren, der auf alle Nachrichten vom Rechner `green` passt, die den Text `error` enthalten:

```
filter f_green { host("green") and match("error"); };
```

Bei der Funktion `level()` (bzw. `priority()`) können Sie entweder eine oder mehrere durch Kommas getrennte Prioritäten oder aber einen Bereich von Prioritäten in der Form »`warn .. emerg`« angeben.

Senken für Nachrichten Genau wie Quellen bestehen auch Senken aus verschiedenen »Treibern« für Protokollierungsmethoden. Zum Beispiel könnten Sie Nachrichten in eine Datei schreiben:

```
destination d_file { file("/var/log/messages"); };
```

Sie können auch eine »Schablone« vorgeben, die beschreibt, in welchem Format die Nachricht an die betreffende Senke geschrieben werden soll. Dabei können Sie auf »Makros« zurückgreifen, die diverse Bestandteile der Nachricht zugänglich machen. Zum Beispiel:

```
destination d_file {
    file("/var/log/$YEAR.$MONTH.$DAY/messages"
         template("$HOUR:$MIN:$SEC $TZ $HOST [$LEVEL] $MSG\n")
         template_escape(no)
         create_dirs(yes)
    );
};
```

Die Makros `$YEAR`, `$MONTH` usw. werden dabei durch ihre naheliegenden Werte ersetzt. `$TZ` ist die aktuelle Zeitzone, `$LEVEL` die Priorität der Nachricht und `$MSG` die Nachricht selbst (inklusive der Prozess-ID des Absenders). Eine komplette Liste der Makros finden Sie in der Dokumentation zu Syslog-NG.

Der Parameter `template_escape()` steuert, ob die Anführungszeichen (»'« und »"«) in der Ausgabe »versteckt« werden sollen. Dies ist wichtig, falls Sie Protokollnachrichten zum Beispiel an einen SQL-Server verfüttern wollen.

Im Gegensatz zum `syslogd` erlaubt Syslog-NG die Weiterleitung von Nachrichten per TCP. Dies ist nicht nur bequemer, wenn Firewalls im Spiel sind, sondern stellt auch sicher, dass keine Protokollnachrichten verloren gehen (was bei UDP passieren könnte). Eine TCP-Weiterleitung könnten Sie zum Beispiel so definieren:

```
destination d_tcp { tcp("10.11.12.13" port(514); localport(514)); };
```

Sehr nützlich ist auch die Weiterleitung in Programme mit `program()`. Syslog-NG startet das Programm, wenn er selbst hochfährt, und lässt es weiterlaufen bis zu seinem eigenen Ende oder bis er ein `SIGHUP` empfängt. Dies dient nicht nur der Effizienz, sondern ist eine Vorsichtsmaßnahme gegen Denial-of-Service-Angriffe – wenn für jede Meldung ein Prozess gestartet würde, könnte ein Angreifer die Protokollierung außer Gefecht setzen, indem er jede Menge passende Protokollnachrichten schickt. (Andere Protokollnachrichten, die über seine Machenschaften Aufschluss geben würden, fallen dann vielleicht herunter.)

Protokoll-Pfade Protokoll-Pfade dienen dazu, Quellen, Filter und Senken zusammenzubringen und tatsächlich Nachrichten auszuwerten. Sie beginnen immer mit dem Schlüsselwort `log`. Hier sind ein paar Beispiele für Protokoll-Pfade auf der Basis von Regeln, die Sie schon aus der Diskussion von `/etc/syslog.conf` kennen:

```
# Vorbedingungen
source s_all { internal(); unix-stream("/dev/log"); };
filter f_auth { facility(auth, authpriv); };
destination df_auth { file("/var/log/auth.log"); };

# auth,authpriv.*   /var/log/auth.log
log {
        source(s_all);
        filter(f_auth);
        destination(df_auth);
};
```

Bei dieser Regel werden alle Nachrichten, die mit Authentisierung zu tun haben, in die Datei `/var/log/auth.log` geschrieben. Natürlich ist das mit dem `syslogd` in einer Zeile zu erledigen …

Hier ist ein etwas komplexeres Beispiel:

```
# kern.warn;*.err;authpriv.none    /dev/tty10
filter f_nearly_all {
    (facility(kern) and priority(warn .. emerg))
    or (not facility(authpriv,kern));
};
destination df_tty { file("/dev/tty10"); };
log {
        source(s_all);
        filter(f_nearly_all);
        destination(df_tty);
};
```

Auch hier ist die Schreibweise des `syslogd` etwas kompakter, dafür haben Sie hier eher die Chance, nachzuvollziehen, was passiert.

Jede Nachricht durchläuft alle Protokoll-Pfade und wird auch von allen passenden protokolliert (dieses Verhalten entspricht dem von `syslogd`). Wenn Sie wollen, dass eine Nachricht, die einen bestimmten Protokoll-Pfad durchlaufen hat, nicht mehr weiter betrachtet wird, können Sie diesem Pfad die Option `flags(final)` hinzufügen.

`flags(final)` bedeutet nicht, dass die Nachricht nur einmal protokolliert wird; sie könnte vor dem betreffenden Pfad noch von anderen Pfaden protokolliert worden sein.

Mit `flags(fallback)` können Sie einen Pfad zum »Ersatzpfad« erklären. Dann wird er nur für Nachrichten betrachtet, auf die keine nicht mit `flags(fallback)` gekennzeichneten Pfade gepasst haben.

Übungen

20.9 [!3] Installieren Sie Syslog-NG und erstellen Sie eine Konfiguration, die Ihrer bisherigen `syslogd`-Konfiguration möglichst nahe kommt. Testen Sie sie zum Beispiel mit `logger`. Wo sehen Sie Verbesserungsmöglichkeiten?

20.10 [2] PAM, das Anmelde- und Authentisierungssystem, protokolliert An- und Abmeldevorgänge in der folgenden Form:

```
kdm: :0[5244]: (pam_unix) session opened for user hugo by (uid=0)
◁◁◁◁◁
kdm: :0[5244]: (pam_unix) session closed for user hugo
```

Konfigurieren Sie Syslog-NG so, dass jedesmal, wenn sich ein bestimmter Benutzer (etwa Sie) an- oder abmeldet, eine Meldung auf dem Terminal des Systemverwalters (`root`) erscheint, falls dieser angemeldet ist. (*Tipp:* Die PAM-Nachrichten erscheinen in der Kategorie `authpriv`.)

20.11 [3] (Arbeiten Sie für diese Aufgabe gegebenenfalls mit einem anderen Kursteilnehmer zusammen.) Konfigurieren Sie Syslog-NG so, dass alle Protokollnachrichten von einem Rechner über eine TCP-Verbindung an einen anderen Rechner weitergeleitet werden. Testen Sie die Verbindung mit `logger`. Experimentieren Sie mit verschiedenen Einstellungen für `keep_hostnames()` und `chain_hostnames()`.

```
/var/log/syslog
{
    rotate 7
    daily
    missingok
    notifempty
    delaycompress
    compress
    postrotate
        invoke-rc.d rsyslog rotate >/dev/null
    endscript
}
```

Bild 20.1: Beispiel-Konfiguration für `logrotate` (Debian GNU/Linux 8.0)

20.7 Das Programm `logrotate`

Je nach der Anzahl der Benutzer und der Anzahl und Art der laufenden Dienste können die Protokolldateien ziemlich schnell ziemlich groß werden. Um ein »Zumüllen« des Systems zu verhindern, sollten Sie einerseits zumindest auf vielbeschäftigten Serversystemen versuchen, die entsprechenden Verzeichnisse (z. B. `/var/log` oder `/var`) auf eine eigene Partition zu legen. Andererseits gibt es Software, die die Protokolldateien regelmäßig nach verschiedenen Kriterien, etwa der Größe, überwacht, kürzt und alte Protokolle löscht oder archiviert. Dieser Vorgang heißt »Rotieren«, und ein solches Programm ist `logrotate`.

`logrotate` ist kein Daemon, sondern wird zum Beispiel einmal täglich über `cron` – oder einen ähnlichen Dienst – gestartet.

`logrotate` weigert sich, eine Protokolldatei mehr als einmal am Tag zu modifizieren, es sei denn, die Entscheidung hängt von der Größe der Protokolldatei ab, Sie verwenden das Kriterium `hourly`, oder beim Aufruf wurde die Option `--force` (kurz `-f`) angegeben.

`/etc/logrotate.conf` `/etc/logrotate.d`

`logrotate` wird nach Konvention über die Datei `/etc/logrotate.conf` und über die Dateien in `/etc/logrotate.d` konfiguriert. Die Datei `/etc/logrotate.conf` setzt allgemeine Parameter, die aber von den Dateien in `/etc/logrotate.d` wieder überschrieben werden können. In `/etc/logrotate.conf` steht unter anderem der Parameter `include /etc/logrotate.d`, der bewirkt, dass die dort befindlichen Dateien an der entsprechenden Stelle als Teil der Datei `/etc/logrotate.conf` integriert werden.

Prinzipiell liest `logrotate` alle auf der Kommandozeile übergebenen Dateinamen als Konfigurationsdateien ein, wobei die Inhalte später genannter Dateien die von weiter vorne genannten überschreiben. Die Sache mit `/etc/logrotate.conf` ist nur eine (sinnvolle) Vereinbarung, die durch einen entsprechenden Aufruf von `logrotate` in `/etc/cron.daily/logrotate` (oder Äquivalent) realisiert wird.

Wir erwähnen das hier, weil Ihnen das grundsätzlich die Möglichkeit eröffnet, ohne große Mühe separate `logrotate`-Läufe für Protokolldateien zu machen, die nicht Bestandteil der regulären Konfiguration sind. Wenn Sie zum Beispiel eine extrem schnell wachsende Protokolldatei etwa eines populären Web-Servers haben, können Sie diese über eine eigene Instanz von `logrotate` verwalten, die öfter als einmal täglich läuft.

`logrotate` überwacht alle Dateien, die ihm über die genannten Konfigurationsdateien mitgeteilt werden, also nicht nur die Ausgabe von `syslogd`. Bild 20.1 zeigt als Beispiel einen Ausschnitt aus der Konfigurationsdatei für rsyslog von Debian GNU/Linux 8.

Die erste Zeile des Beispiels gibt an, für welche Dateien die Konfiguration gilt (hier /var/log/syslog). Sie können mehrere Dateien aufzählen oder auch Shell-Suchmuster verwenden. Anschließend steht in geschweiften Klammern ein Block von Direktiven, die beschreiben, wie logrotate mit den benannten Dateien umgehen soll.

Typischerweise finden sich in /etc/logrotate.conf Direktiven, die außerhalb eines Blocks stehen. Diese Direktiven dienen als Voreinstellungen, die für alle Protokolldateien in der Konfiguration gelten, sofern in deren Blöcken nicht etwas Spezifischeres verfügt wird.

alte Versionen

»rotate 7« heißt, dass maximal sieben alte Versionen jeder Protokolldatei aufgehoben werden. Wenn dieses Maximum erreicht ist, wird die älteste Version der Protokolldatei gelöscht.

Wenn Sie mit mail eine Adresse angeben, werden Dateien nicht gelöscht, sondern an die betreffende Adresse geschickt.

»rotate 0« wirft »rotierte« Protokolldateien sofort weg, ohne sie zu aufzuheben.

Die rotierten Dateien werden einfach fortlaufend durchnummeriert, das heißt, wenn die aktuelle Version der Datei /var/log/syslog heißt, dann heißt die unmittelbar vorhergehende Version /var/log/syslog.1, die Version unmittelbar davor /var/log/syslog.2 und so weiter.

Sie können statt einer fortlaufenden Nummer auch das Datum als Dateiendung benutzen. Das heißt, wenn heute der 20. Juli 2015 ist und Ihr logrotate-Lauf täglich in den frühen Morgenstunden stattfindet, dann heißt die unmittelbar vorhergehende Version der Datei nicht /var/log/syslog.1, sondern /var/log/syslog-20150720, die Version unmittelbar davor entsprechend /var/log/syslog-20150719 und so fort. Um das zu erreichen, müssen Sie die Direktive »dateext« angeben.

Mit »dateformat« können Sie bestimmen, wie genau die Datums-Dateiendung aussehen kann. Sie müssen dazu eine Zeichenkette übergeben, die die Schlüssel %Y, %m, %d und %s enthalten darf. Sie stehen respektive für das (vierstellige) Jahr, Kalendermonat und Kalendertag (jeweils zweistellig und gegebenfalls mit führender Null) und die Sekunden seit dem 1. Januar 1970, 0 Uhr UTC. Die Standardvorgabe ist – wie Sie aus dem vorigen Absatz schließen können – »-%Y%m%d«.

Wenn Sie dateformat benutzen, sollten Sie darauf achten, dass logrotate beim Rotieren die Dateinamen lexikografisch sortiert, um zu entscheiden, was die älteste Datei ist. Mit »-%Y%m%d« klappt das, aber mit »-%d%m%Y« nicht.

Zeiträume

»daily« heißt, dass Protokolldateien täglich rotiert werden sollen. Im Zusammenspiel mit »rotate 7« bedeutet das, dass Sie immer Zugriff auf die Protokolle der letzten Woche haben.

Es gibt auch noch weekly, monthly and yearly. Mit weekly wird die Datei rotiert, wenn der aktuelle Wochentag vor dem Wochentag der letzten Rotation liegt oder mehr als eine Woche seit der letzten Rotation vergangen ist (unter dem Strich heißt das, dass am ersten Wochentag rotiert wird, wobei das gemäß US-amerikanischer Gepflogenheit der Sonntag ist). Mit monthly wird die Datei beim ersten Lauf von logrotate in jedem Monat rotiert (normalerweise am Monatsersten). Mit yearly erfolgt die Rotation beim ersten Lauf von logrotate in jedem Jahr. Theoretisch rotiert hourly die Protokolldatei in stündlichem Rhythmus, aber da logrotate normalerweise nur einmal am Tag aufgerufen wird, müssen Sie gegebenenfalls selber dafür sorgen, dass es tatsächlich so oft läuft.

Ein alternatives Kriterium ist »size«. Damit können Sie eine Protokolldatei rotieren, wenn sie eine bestimmte Größe überschritten haben. Die Dateigröße wird als Parameter angegeben – ohne Einheit wird sie als Bytes interpretiert, während die Einheiten k (oder K), M, G respektive für Kibibytes (2^{10} Bytes), Mebibytes (2^{20} Bytes) oder Gibibytes (2^{30} Bytes) stehen.

»size« und die zeitbasierten Kriterien schließen einander aus, das heißt, wenn Sie ein »size«-Kriterium angeben, entscheidet allein die Dateigröße über das Rotieren, egal wann die Datei zuletzt rotiert wurde.

Dateigröße und Zeit als Kriterien verbinden können Sie über die Direktiven »maxsize« und »minsize«. Mit »maxsize« können Sie eine Größe angeben, deren Überschreiten logrotate die Datei rotieren läßt, selbst wenn der nächste offizielle Zeitpunkt dafür noch nicht erreicht wurde. Bei »minsize« wird die Datei zum per Zeit-Kriterium vorgesehenen Zeitpunkt nur rotiert, wenn sie die angegebene Größe überschritten hat (kleine Dateien werden übergangen).

Fehlermeldungen

»missingok« unterdrückt Fehlermeldungen, wenn eine Protokolldatei nicht gefunden wird. (Der Standard ist »nomissingok«.) »notifempty« rotiert eine Datei nicht, wenn sie leer ist (die Vorgabe ist hier »ifempty«).

Mit »compress« können Sie verfügen, dass rotierte Versionen der Protokolldatei komprimiert werden sollen.

Standardmäßig passiert das mit gzip, wenn Sie nicht mit »compresscmd« ein anderes Kommando benennen. Mit »compressoptions« können Sie Optionen für dieses Kommando angeben (die Sie sonst auf der Kommandozeile übergeben würden). Der Standard für gzip ist »-6«.

Die Direktive »delaycompress« sorgt dafür, dass eine frisch rotierte Datei nicht unmittelbar nach dem Rotieren komprimiert wird, sondern erst im nächsten Durchgang. Während normalerweise die Folge der Dateien aussähe wie

```
/var/log/syslog  /var/log/syslog.1.gz  /var/log/syslog.2.gz  …
```

bekommen Sie mit »delaycompress« die Folge

```
/var/log/syslog  /var/log/syslog.1  /var/log/syslog.2.gz …
```

(mit anderen Worten, /var/log/syslog.1 bleibt noch unkomprimiert). Diese Einstellung brauchen Sie, wenn die Möglichkeit besteht, dass das schreibende Programm (hier rsyslog) nach dem Umbenennen (Rotieren) der Datei noch Daten an sie anhängt – das kann passieren, weil rsyslog die Protokolldatei dauerhaft offenhält und Umbenennungen für das Schreiben in die Datei irrelevant sind.

Daraus folgt, dass Sie rsyslog benachrichtigen müssen, dass es eine neue Protokolldatei gibt. Dafür ist die Direktive

```
postrotate
  invoke-rc.d rsyslog rotate >/dev/null
endscript
```

gedacht. Die Shell-Kommandos zwischen »postrotate« und »endscript« führt logrotate jedesmal aus, nachdem die Protokolldatei rotiert wurde.

Das Kommando selbst ist für uns im Grunde egal (die Idee zählt), aber was hier unter dem Strich passiert, ist, dass das Init-Skript von rsyslog aufgerufen wird und dem Programm ein SIGHUP schickt. Andere Distributionen haben dafür auch Mittel und Wege.

Das SIGHUP bewegt rsyslog dann dazu, seine Konfigurationsdatei neu einzulesen und alle Protokolldateien zu schließen und wieder zu öffnen. Da /var/log/syslog inzwischen umbenannt wurde, öffnet rsyslog unter diesem Namen eine neue Protokolldatei. – An dieser Stelle könnte logrotate die Datei /var/log/syslog.1 komprimieren, aber es hat keine Möglichkeit, zu erfahren, wann rsyslog tatsächlich mit der Datei »fertig hat« (G. Trapattoni). Darum wird das beim nächsten Rotieren der Datei nachgeholt.

Zwischen »postrotate« und »endscript« dürfen mehrere Zeilen mit Kommandos stehen. logrotate hängt sie alle aneinander und übergibt sie als Ganzes der Shell (/bin/sh) zur Ausführung. Das Kommando bekommt den Namen der Protokolldatei als Parameter übergeben; dieses steht dort wie üblich als »$1« zur Verfügung.

Die postrotate-Kommandos werden einzeln für jede Protokolldatei ausgeführt, die am Anfang des Konfigurationsblocks aufgezählt wurde. Das heißt, dass die Kommandos möglicherweise mehrmals aufgerufen werden. Mit der Direktive »sharedscripts« können Sie dafür sorgen, dass die Kommandos für alle Dateien, auf die das Suchmuster passt, insgesamt nur einmal ausgeführt werden (oder gar nicht, wenn von den Dateien keine rotiert werden musste).

Mit »create« können Sie dafür sorgen, dass unmittelbar nach dem Rotieren und vor der Ausführung der postrotate-Kommandos die Protokolldatei neu angelegt wird. Dabei wird der Name der alten Datei verwendet. Zugriffsmodus, Eigentümer und Gruppe richten sich nach den Parametern von create, es gibt die drei Möglichkeiten

`create 600 root adm`	*Zugriffsmodus, Eigentümer und Gruppe*
`create root adm`	*Nur Eigentümer und Gruppe*
`create`	*Gar nichts*

Nicht angegebene Dateieigenschaften werden von der vorigen Version der Datei übernommen.

Dies ist nur eine Auswahl der wichtigsten Konfigurationsparameter. Studieren Sie logrotate(8), um die komplette Liste zu sehen.

Übungen

20.12 [!1] Welche systemweiten Voreinstellungen gelten für logrotate bei Ihrer Distribution?

20.13 [2] Konfigurieren Sie logrotate so, dass Ihre neue Protokolldatei /var/log/test rotiert wird, sobald sie eine Größe von 100 Byte überschreitet. Es sollen 10 Backups der rotierten Dateien aufgehoben werden, außerdem sollen diese Backups komprimiert werden und einen Namen tragen, der das Datum ihrer Erstellung enthält. Testen Sie Ihre Konfiguration.

Kommandos in diesem Kapitel

dmesg Gibt den Inhalt des Kernel-Nachrichtenpuffers aus dmesg(8) 339
klogd Akzeptiert Protokollnachrichten des Systemkerns klogd(8) 334, 338
logger Macht Einträge ins Systemprotokoll logger(1) 337
logrotate Verwaltet, kürzt und „rotiert“ Protokolldateien logrotate(8) 347
logsurfer Programm zum Durchsuchen des Systemprotokolls nach wichtigen Ereignissen www.cert.dfn.de/eng/logsurf/ 338
syslog-ng Bearbeitet Systemprotokoll-Meldungen (neuer und besser) syslog-ng(8) 343
syslogd Bearbeitet Systemprotokoll-Meldungen syslogd(8) 334
tail Zeigt das Ende einer Datei an tail(1) 338
xconsole Zeigt Systemmeldungen in einem X-Fenster an xconsole(1) 334
xlogmaster X11-basiertes Systembeobachtungsprogramm xlogmaster(1), www.gnu.org/software/xlogmaster/ 338

Zusammenfassung

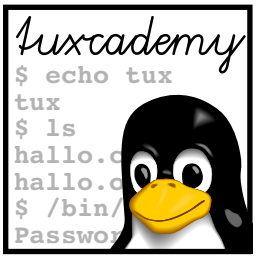
- Der syslogd kann Protokollnachrichten verschiedener Systemkomponenten annehmen, in Dateien schreiben, an Benutzer oder andere Rechner weiterreichen.
- Protokollnachrichten können aus verschiedenen Kategorien kommen und unterschiedliche Prioritäten haben.
- Mit dem Kommando logger kann man Nachrichten an den syslogd schicken.
- Protokolldateien stehen für gewöhnlich im Verzeichnis /var/log.
- Syslog-NG ist eine kompatible, aber erweiterte Neuimplementierung eines Syslog-Daemons.
- Mit logrotate können Protokolldateien verwaltet und archiviert werden.

Literaturverzeichnis

RFC3164 C. Lonvick. »The BSD syslog Protocol«, August 2001.
http://www.ietf.org/rfc/rfc3164.txt

rsyslog »Welcome to Rsyslog«. Online-Dokumentation für rsyslog 8.*x*.
http://www.rsyslog.com/doc/v8-stable/index.html

syslog-ng »syslog-ng – Log Management Software«.
http://www.balabit.com/products/syslog_ng/



21

Systemprotokollierung mit systemd und »dem Journal«

Inhalt

21.1 Grundlagen 354
21.2 Systemd und journald 355
21.3 Protokollauswertung 358

Lernziele

- Grundprinzipien von journald verstehen
- Journald konfigurieren können
- Einfache Anfragen an die Protokolldatenbank formulieren können
- Verstehen, wie journald mit Protokolldateien umgeht

Vorkenntnisse

- Grundkenntnisse der Komponenten eines Linux-Systems
- Umgang mit Konfigurationsdateien
- Kenntnisse des traditionellen Systemprotokolldiensts (Kapitel 20)
- Kenntnisse über systemd

21.1 Grundlagen

Systemd ist eine durchgreifende Erneuerung der Software, die den grundlegenden Systembetrieb eines Linux-Rechners sichert. Im engeren Sinne kümmert systemd sich um das Starten und Nachverfolgen von Diensten und die Verwaltung von Ressourcen. Allerdings enthält systemd auch einen vom traditionellen syslogd-Verfahren deutlich verschiedenen Ansatz für die Systemprotokollierung, das »Journal«, und die dafür nötigen Systemkomponenten.

Während im traditionellen Ansatz der syslog-Daemon Protokollnachrichten auf dem UDP-Port 514 oder dem Socket `/dev/log` entgegennimmt und (typischerweise) in Textdateien schreibt (oder sie an andere Rechner weitergeleitet werden, wo sie dann in Textdateien geschrieben werden), können in der systemd-Welt Hintergrunddienste ihre Protokollnachrichten einfach auf die Standardfehlerausgabe schreiben, und systemd kümmert sich darum, sie an den Protokolldienst weiterzuleiten[1]. Protokolldateien sind bei systemd keine Textdateien (wobei jede Meldung möglicherweise in mehreren Dateien abgelegt wird), sondern Meldungen werden in eine (binäre) Datenbank geschrieben, die anschließend nach diversen Kriterien durchsucht werden kann.

Es ist zum Beispiel sehr einfach möglich, alle Nachrichten anzuzeigen, die ein bestimmter Dienst in einem bestimmten durch Anfangs- und Enddatum und -uhrzeit gegebenen Zeitraum protokolliert hat. Im traditionellen System ist das ziemlich aufwendig.

Fairerweise sollten wir nochmal hervorheben, dass auch die modernen syslog-Implementierungen wie Rsyslog oder Syslog-NG prinzipiell in der Lage sind, Nachrichten in eine Datenbank zu schreiben. Allerdings sind Sie dann selbst dafür verantwortlich, ein entsprechendes Datenbankschema zu entwerfen, Rsyslog bzw. Syslog-NG passend zu konfigurieren, und Software zu entwickeln, die Ihnen bequemen Zugang zu den Protokollnachrichten verschafft. Bei systemd ist das alles »ab Werk« dabei.

Das Journal ist nicht auf textuelle Protokollnachrichten beschränkt. Zum Beispiel ist es absolut möglich, Hauptspeicherabzüge (*core dumps*) abgestürzter Programme ins Journal zu schreiben (jedenfalls solange sie nicht übergroß sind). Ob das eine uneingeschränkt geniale Idee ist, ist natürlich debattierbar, und die systemd-Entwickler haben sich auch schon eine alternative Methode überlegt.

Wie üblich kann auch das Protokollsystem von systemd mit dem traditionellen Ansatz interoperieren. Es protokolliert auf Wunsch auch Nachrichten, die über `/dev/log` oder den UDP-Port 512 eingehen, und kann Nachrichten an einen traditionellen syslog-Daemon (oder eine moderne Neuimplementierung) weiterreichen.

Dem Journal verdanken Sie übrigens auch den (extrem bequemen) Umstand, dass Sie bei »`systemctl status`« die letzten Protokollnachrichten des betreffenden Dienstes angezeigt bekommen:

```
# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since Mo 2015-07-27 13:37:22 CEST; 8h ago
 Main PID: 428 (sshd)
   CGroup: /system.slice/ssh.service
           └─428 /usr/sbin/sshd -D

Jul 27 13:37:23 blue sshd[428]: Server listening on 0.0.0.0 port 22.
Jul 27 13:37:23 blue sshd[428]: Server listening on :: port 22.
Jul 27 13:56:50 blue sshd[912]: Accepted password for hugo from ...sh2
```

[1] Systemd hat außerdem noch ein eigenes API für Protokollnachrichten.

```
Jul 27 13:56:50 blue sshd[912]: pam_unix(sshd:session): session ...=0)
Hint: Some lines were ellipsized, use -l to show in full.
```

Wie die letzte Zeile der Ausgabe andeutet, werden überlange Zeilen so abgekürzt, dass sie gerade noch auf dem Bildschirm passen. Wenn Sie sie komplett sehen wollen, müssen Sie `systemctl` mit der Option `-l` aufrufen.

Übungen

21.1 [2] Welche Vor- und Nachteile hat der traditionelle Ansatz (Textdateien in `/var/log`) gegenüber dem datenbankbasierten Vorgehen des Journals?

21.2 Systemd und journald

Das Journal ist ein integraler Bestandteil von systemd. Im einfachsten Fall verwendet systemd einen Ringpuffer begrenzter Größe in `/run/log/journal`, um eine gewisse Anzahl von Protokollnachrichten im RAM zwischenzuspeichern (wenn Sie die Daten ohnehin an einen traditionellen Protokolldienst weiterleiten wollen, dann reicht das aus). Um alle Vorteile des Journals auszunutzen, sollten Sie aber dafür sorgen, dass die Protokollnachrichten dauerhaft auf der Platte gespeichert werden. Dazu genügt es, das Verzeichnis für die Speicherung anzulegen:

```
# mkdir -p /var/log/journal
# systemctl --signal=USR1 kill systemd-journald
```

(durch das `SIGUSR1` wird systemd dazu gebracht, das bisher im RAM angelegte Journal in die neue Datei auf Platte umzutragen).

Die Komponente von systemd, die sich um das Journal kümmert, heißt `systemd-journald` (oder unter Freunden journald).

Konfiguriert wird das Journal in der Datei `/etc/systemd/journald.conf`. Im `[Journal]`-Abschnitt der Datei (dem einzigen) steht zum Beispiel der Parameter `Storage`, der die folgenden Werte haben kann:

`volatile` Protokollnachrichten werden nur im RAM gespeichert (in `/run/log/journal`), auch wenn `/var/log/journal` existiert.

`persistent` Protokollnachrichten werden bevorzugt auf Platte gespeichert (in `/var/log/journal`). Das Verzeichnis wird angelegt, falls es nicht existiert. Während des frühen Startvorgangs und falls die Platte nicht schreibbar sein sollte, fällt systemd auf `/run/log/journal` zurück.

`auto` Ähnelt `persistent`, aber hier entscheidet die Existenz des Verzeichnisses `/var/log/journal` darüber, ob ein persistentes Journal geschrieben wird – wenn das Verzeichnis nicht existiert, bleibt es beim flüchtigen Journal im RAM.

`none` Im Journal werden überhaupt keine Protokollnachrichten gespeichert. Sie können die Nachrichten aber trotzdem zum Beispiel an den traditionellen syslog-Dienst weiterleiten.

Es gibt noch ein paar andere interessante Parameter. `Compress` gibt an, ob die Protokolldaten (jedenfalls die, die eine gewisse Größe überschreiten) transparent komprimiert werden sollen; der Standardwert ist `yes`. Mit `Seal` können Sie dafür sorgen, dass persistente Journaldateien über eine kryptografische Signatur gegen unbemerkte Manipulationen abgesichert werden. Dafür müssen Sie nur einen Schlüssel zur Verfügung stellen (die Dokumentation erklärt, wie).

Die Parameter `RateLimitInterval` und `RateLimitBurst` sollen es schwieriger machen, das Protokoll mit Nachrichten zu überfluten. Wenn ein Dienst in der durch `RateLimitInterval` gegebenen Zeitspanne mehr als `RateLimitBurst` Nachrichten einreicht, dann werden bis zum Ablauf der Zeitspanne alle weiteren Nachrichten ignoriert (im Protokoll steht dann nur eine Nachricht über die Anzahl der ignorierten Nachrichten. Standardmäßig ist die Grenze bei 1000 Nachrichten in 30 Sekunden; wenn Sie irgendeinen der Parameter auf 0 setzen, wird die Begrenzung aufgehoben.

`SyncIntervalSec` gibt an, in welchen Zeitabständen das Journal auf Platte gesichert wird. Eine Sicherung erfolgt immer unmittelbar nachdem eine Nachricht der Priorität `crit` (oder höher) geschrieben wurde; solange keine solche Nachricht eingeht, wird die mit `SyncIntervalSec` angegebene Zeitspanne abgewartet. Der Standardwert ist »5 Minuten«.

Anschauen können Sie das Protokoll mit dem Kommando `journalctl`:

```
# journalctl
-- Logs begin at Mo 2015-07-27 13:37:14 CEST, end at Mo 2015-07-27▷
 ◁ 22:20:47 CEST. --
Jul 27 13:37:14 blue systemd-journal[138]: Runtime journal is using 4.
Jul 27 13:37:14 blue systemd-journal[138]: Runtime journal is using 4.
Jul 27 13:37:14 blue kernel: Initializing cgroup subsys cpuset
Jul 27 13:37:14 blue kernel: Initializing cgroup subsys cpu
Jul 27 13:37:14 blue kernel: Initializing cgroup subsys cpuacct
Jul 27 13:37:14 blue kernel: Linux version 3.16.0-4-amd64 (debian-kern
Jul 27 13:37:14 blue kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-3.
```

Die Ausgabe erinnert stark an das, was Sie in `/var/log/messages` finden würden, aber hat in der Tat einige Verbesserungen (die hier in der Schulungsunterlage teils nur eingeschränkt zum Tragen kommen):

- Das Protokoll wird mit Ihrem Lieblings-Anzeigeprogramm für Textdateien dargestellt (typischerweise `less`). Mit `less` können Sie die Enden von überlangen Zeilen anschauen, indem Sie die horizontalen Pfeiltasten verwenden.

  Entscheidend dafür ist der Wert der Umgebungsvariable `SYSTEMD_PAGER`, ersatzweise der Wert von `PAGER`, ersatzweise `less`. Mit der Umgebungsvariable `SYSTEMD_LESS` können Sie Optionen für `less` angeben (wenn Sie nicht die Standard-Vorgabe benutzen, wird diese Variable ignoriert, aber Sie können die Optionen dann ja direkt in `SYSTEMD_PAGER` schreiben).

  Wenn Sie `journalctl` mit der Option `--no-pager` aufrufen oder `SYSTEMD_PAGER` auf den Wert `cat` oder eine leere Zeichenkette setzen, wird die Ausgabe nicht seitenweise dargestellt.

- Die Ausgabe umfasst alle zugänglichen Protokolldateien, auch rotierte (dazu später mehr).
- Zeitstempel sind in Zonenzeit (nicht UTC).
- Protokollnachrichten der Priorität `notice` oder `warning` werden im Fettdruck dargestellt.
- Protokollnachrichten der Priorität `error` (oder höher) erscheinen in rot.

`systemd-journald` versucht, den zur Verfügung stehenden Platz sinnvoll auszunutzen. Das heißt, normalerweise werden neue Nachrichten hinten ans Journal angehängt, aber wenn eine gewisse Obergrenze für die Größe des Journals erreicht ist, wird versucht, alte Protokollnachrichten zu entsorgen.


In `/etc/systemd/journald.conf` können Sie die Parameter `SystemMaxUse` und `RuntimeMaxUse` angeben, die beschreiben, wieviel Platz das Journal jeweils unter `/var/log/journal` und `/run/log/journal` einnehmen darf. Die Parameter `SystemKeepFree` und `RuntimeKeepFree` bestimmen dagegen, wieviel Platz in den entsprechenden Dateisystemen frei bleiben soll. `systemd-journald` beachtet beide Werte (`…MaxUse` und `…KeepFree`) und beschränkt sich auf das Minimum.


Die `Runtime…`-Werte kommen beim Systemstart zum Tragen oder wenn kein persistentes Journal verwendet wird. Die `System…`-Werte gelten für das persistente Journal, wenn das System weit genug gebootet ist. Bei der Bestimmung des vom Journal belegten Platzes werden nur Dateien berücksichtigt, deren Namen auf `.journal` endet.


Sie können die Werte in Bytes angeben oder eine der (binären) Einheiten `K`, `M`, `G`, `T`, `P` oder `E` anhängen[2].


Standardwert für `…MaxUse` ist 10% und für `…KeepFree` 15% der Größe des betreffenden Dateisystems. Wenn beim Start von `systemd-journald` schon weniger Platz auf dem Dateisystem frei ist, als der `…KeepFree`-Wert angibt, dann wird die Grenze auch noch weiter reduziert, so dass Platz für andere Sachen bleibt.

Ähnlich wie `logrotate` »rotiert« systemd das Journal, um Platz für neue Nachrichten zu schaffen. Dazu wird der verfügbare Platz in eine Reihe von Dateien aufgeteilt, damit von Zeit zu Zeit die älteste Protokolldatei verworfen werden kann. Diese Rotation ist für Benutzer transparent, da `systemd-journald` die Rotation selbsttätig bei Bedarf vornimmt und `journalctl` immer das komplette Journal auswertet, egal auf wieviele Dateien es verteilt ist.


Maßgeblich für die Aufteilung sind die Parameter `SystemMaxFileSize` und `RuntimeMaxFileSize` in der Datei `/etc/systemd/journald.conf`. Sie geben an, wie groß individuelle Journal-Dateien werden dürfen – der Standard ist »ein Achtel des für das Journal verfügbaren Platzes«, so dass Sie immer eine »Vorgeschichte« von sieben Dateien und die aktuelle Datei zur Verfügung haben.


Sie können das Rotieren von Protokolldateien auch von der Zeit abhängig machen: `MaxFileSec` gibt die maximale Zeitspanne an, bevor systemd eine neue Protokolldatei anfängt. (Normalerweise reicht die größenbasierte Rotation aber völlig aus.) Mit `MaxRetentionSec` können Sie eine Obergrenze für die Haltezeit alter Protokollnachrichten angeben. Der Standardwert für `MaxFileSec` ist `1month` (`0` steht ggf. für »beliebig lange«) und der für `MaxRetentionSec` ist 0 (der Mechanismus ist also ausgeschaltet).

Weiterleitung an syslog

Ebenfalls in `/etc/systemd/journald.conf` können Sie die Weiterleitung an ein traditionelles syslog-System konfigurieren. Dazu setzen Sie einfach

```
[Journal]
ForwardToSyslog=yes
```

Übungen


21.2 [!2] Konfigurieren Sie Ihren Rechner so, dass das Journal persistent auf der Platte liegt. Vergewissern Sie sich, dass das tatsächlich funktioniert hat (etwa indem Sie mit `logger` eine Nachricht ins Protokoll schreiben, den Rechner neu starten und dann prüfen, dass die Nachricht noch da ist).

[2] Wir nehmen mal an, dass es noch ein bisschen dauern wird, bis Sie eine Größenbeschränkung für das Journal in Exbibyte (2^{60} Bytes) angeben müssen, aber es ist beruhigend, dass die systemd-Entwickler anscheinend für die Zukunft planen.

21.3 [2] Hat Ihr Rechner noch einen traditionellen syslog-Daemon? Wenn nein, dann installieren Sie einen (BSD-`syslogd` oder Rsyslog bieten sich an) und sorgen Sie dafür, dass Protokollnachrichten an diesen weitergeleitet werden. Überzeugen Sie sich (etwa mit `logger`), dass das klappt.

21.3 Protokollauswertung

Mit `journalctl` können Sie sehr detaillierte Anfragen an das Journal richten. In diesem Abschnitt beleuchten wir das etwas genauer, aber zuerst noch ein paar Vorbemerkungen.

Zugriffsrechte Während Sie als `root` das komplette Protokoll zu sehen bekommen, steht Ihnen als normaler Benutzer nur Ihr eigenes Protokoll offen, also die Nachrichten von Programmen, die Sie selbst (oder der Computer in Ihrem Namen) gestartet hat. Wenn Sie auch als normaler Benutzer den Vollzugriff haben wollen – wir empfehlen ja, dass Sie auch als Administrator so viel wie möglich mit einem nicht privilegierten Benutzerkonto zu arbeiten –, dann müssen Sie dafür sorgen, dass Sie in der Gruppe `adm` sind:

```
# usermod -a -G adm hugo
```

Sie müssen sich abmelden und wieder anmelden, damit diese Einstellung zum Tragen kommt.

Protokollbeobachtung in Echtzeit In Analogie zum populären »`tail -f`«-Kommando können Sie beobachten, wie neue Nachrichten im Journal abgelegt werden:

```
$ journalctl -f
```

Auch hier bekommen Sie 10 Zeilen angezeigt, und anschließend wartet `journalctl` darauf, dass mehr Meldungen eingehen. Die Anzahl der Zeilen können Sie (wie beim guten alten `tail`) mit der Option `-n` einstellen, und das funktioniert auch ohne `-f`.

Dienste und Prioritäten Mit der Option `-u` können Sie die Ausgabe auf diejenigen Protokollnachrichten beschränken, die eine bestimmte systemd-Unit geschrieben hat:

```
$ journalctl -u ssh
-- Logs begin at Mo 2015-07-27 13:37:14 CEST, end at Di 2015-07-28 ▷
 ◁ 09:32:08 CEST. --
Jul 27 13:37:23 blue sshd[428]: Server listening on 0.0.0.0 port 22.
Jul 27 13:37:23 blue sshd[428]: Server listening on :: port 22.
Jul 27 13:56:50 blue sshd[912]: Accepted password for hugo from 192.16
Jul 27 13:56:50 blue sshd[912]: pam_unix(sshd:session): session opened
```

Statt eines gezielten Unit-Namens können Sie auch ein Shell-Suchmuster angeben, um mehrere Units zu erfassen. Oder Sie geben einfach mehrere `-u`-Optionen an.

Um nur Nachrichten bestimmter Prioritäten angezeigt zu bekommen, können Sie die Option `-p` benutzen. Diese übernimmt entweder eine einzelne numerische oder textuelle Priorität (`emerg` hat den Wert 0, `debug` hat den Wert 7) und beschränkt die Ausgabe dann auf Nachrichten der entsprechenden Priorität oder darüber (darunter, wenn Sie nach den numerischen Werten gehen wollen). Oder Sie geben einen Bereich in der Form

```
$ journalctl -p warning..crit
```

an, um nur die Nachrichten zu sehen, deren Priorität in diesem Bereich ist.

Selbstverständlich können Sie die Optionen `-u` und `-p` auch kombinieren:

```
$ journalctl -u apache2 -p err
```

zeigt Ihnen alle Fehlermeldungen (oder schlimmer) von Apache.

Die Option `-k` beschränkt die Ausgabe auf Nachrichten, die der Systemkern geschickt hat. Dabei werden nur Nachrichten seit dem letzten Systemstart berücksichtigt.

Zeit Wenn Sie sich nur für Nachrichten in einem bestimmten Zeitraum interessieren, können Sie die Ausgabe entsprechend einschränken. Die Optionen `--since` und `--until` gestatten die Angabe eines Datums oder einer Uhrzeit in der Form »`2015-07-27 15:36:11`«, und es werden nur Nachrichten ausgegeben, die seit oder bis zu dem gegebenen Zeitpunkt protokolliert wurden.

Sie können die Uhrzeit komplett weglassen, dann wird »`00:00:00`« angenommen. Oder Sie lassen nur die Sekunden weg, dann ist »`…:00`« impliziert. Wenn Sie das Datum weglassen (dann ist natürlich eine Uhrzeit nötig, mit oder ohne Sekunden), dann nimmt `journalctl` »heute« an.

Die Schlüsselwörter `yesterday`, `today` und `tomorrow` stehen respektive für »`00:00:00`« gestern, heute oder morgen.

Relative Zeitangaben sind auch erlaubt: »`-30m`« steht für »vor einer halben Stunde«. (»`+1h`« steht für »in einer Stunde«, aber es ist unwahrscheinlich, dass Ihr Systemprotokoll Einträge aus der Zukunft enthält[3].)

Jeder Systemstartvorgang bekommt eine eindeutige Kennung, und Sie können Ihre Suche auf den Teil des Journals beschränken, der zwischen einem bestimmten Systemstart und dem nächsten liegt. Im einfachsten Fall betrachtet »`journalctl -b`« nur die Meldungen des aktuellen Rechnerlaufs:

```
$ journalctl -b -u apache2
```

Mit der Option `--list-boots` gibt `journald` eine Liste der Startvorgänge aus, die im aktuellen Journal zu finden sind, zusammen mit den Zeiträumen, für die es Protokolleinträge gibt:

```
$ journalctl --list-boots
-1 30eb83c06e054feba2716a1512f64cfc Mo 2015-07-27 22:45:08 CEST–▷
 ◁ Di 2015-07-28 10:03:31 CEST
 0 8533257004154353b45e99d916d66b20 Di 2015-07-28 10:04:22 CEST–▷
 ◁ Di 2015-07-28 10:04:27 CEST
```

Sie können sich auf bestimmte Startvorgänge beziehen, indem Sie bei `-b` deren Index angeben (`1` steht für den chronologisch ersten Startvorgang im Protokoll, `2` für den zweiten und so weiter) oder den negativen Versatz in der ersten Spalte der Ausgabe von »`journalctl --list-boots`« (`-0` steht für den aktuellen Startvorgang, `-1` für den vorigen und so weiter).

[3] Es sei denn, Sie sind der Doktor und durchsuchen das Journal der TARDIS.

```
Mo 2015-07-27 13:37:23.580820 CEST [s=89256633e44649848747d32096fb42▷
 ◁ 68;i=1ca;b=30eb83c06e054feba2716a1512f64cfc;m=11a1309;t=51bd9c6f▷
 ◁ 8812e;x=f3d8849a4bcc3d87]
    PRIORITY=6
    _UID=0
    _GID=0
    _SYSTEMD_SLICE=system.slice
    _BOOT_ID=30eb83c06e054feba2716a1512f64cfc
    _MACHINE_ID=d2a0228dc98041409d7e68858cac6aba
    _HOSTNAME=blue
    _CAP_EFFECTIVE=3fffffffff
    _TRANSPORT=syslog
    SYSLOG_FACILITY=4
    SYSLOG_IDENTIFIER=sshd
    SYSLOG_PID=428
    MESSAGE=Server listening on 0.0.0.0 port 22.
    _PID=428
    _COMM=sshd
    _EXE=/usr/sbin/sshd
    _CMDLINE=/usr/sbin/sshd -D
    _SYSTEMD_CGROUP=/system.slice/ssh.service
    _SYSTEMD_UNIT=ssh.service
    _SOURCE_REALTIME_TIMESTAMP=1437997043580820
```

Bild 21.1: Vollständige Protokollausgabe mit `journalctl`

Sie können auch die 32 Zeichen lange alphanumerische Boot-ID angeben, die in der zweiten Spalte der Ausgabe von »`journalctl --list-boots`« steht, um das Journal nur für den betreffenden Startvorgang zu durchsuchen. Auch dabei können Sie einen positiven oder negativen Versatz anhängen, um Startvorgänge davor oder danach zu identifizieren: Im obigen Beispiel ist

```
$ journalctl -b 8533257004154353b45e99d916d66b20-1
```

eine umständliche Methode,

```
$ journalctl -b 1
```

zu sagen.

Beliebige Suchoperationen Wenn Sie einen Pfadnamen als Parameter angeben, versucht `journalctl` etwas Sinnvolles damit anzufangen:

- Wenn er sich auf eine ausführbare Datei bezieht, dann sucht es nach Protokolleinträgen, die das betreffende Programm gemacht hat.
- Wenn er sich auf eine Gerätedatei bezieht, sucht es nach Einträgen, die das angegebene Gerät betreffen.

Diese Suchoperationen sind Spezialfälle eines allgemeineren Suchmechanismus, den das Journal anbietet. Systemd protokolliert nämlich weitaus mehr Informationen, als der traditionelle syslog-Mechanismus das tut[4]. Sie sehen das, wenn Sie `journalctl` mit der Option `--output=verbose` aufrufen (siehe Bild 21.1).

[4] Auch hier muss man zur Ehrenrettung der modernen Implementierungen erwähnen, dass diese durchaus mehr machen, als sie müssen – auch wenn sie diese Funktionalität mitunter erst sehr kürzlich erworben haben, um mit dem Journal von systemd gleichzuziehen.

Die erste Zeile in Bild 21.1 ist ein Zeitstempel für die Nachricht zusammen mit einem »Cursor«. Der Cursor identifiziert die Nachricht innerhalb des Journals und wird zum Beispiel gebraucht, um Protokolle auf entfernten Rechnern abzulegen.

Die Folgezeilen sind Felder im Journal, die sich auf die betreffende Nachricht beziehen. Feldnamen ohne Unterstreichung am Anfang gehen auf Informationen zurück, die das protokollierende Programm geliefert hat, und sind darum nicht notwendigerweise vertrauenswürdig (das Programm könnte zum Beispiel versuchen, über seine PID oder seinen Namen – in `SYSLOG_IDENTIFIER` – Unfug zu erzählen). Feldnamen mit einer Unterstreichung am Anfang werden von systemd geliefert und können vom protokollierenden Programm nicht verfälscht werden.

`PRIORITY`, `SYSLOG_FACILITY`, `SYSLOG_IDENTIFIER`, `SYSLOG_PID` und `MESSAGE` stammen aus dem syslog-Protokoll und sind ziemlich selbsterklärend. `_UID`, `_GID`, `_HOSTNAME`, `_PID` und `_SYSTEMD_UNIT` erklären sich ebenfalls selber. `_BOOT_ID` ist die Kennung des aktuellen Startvorgangs und `_MACHINE_ID` identifiziert den protokollierenden Rechner aufgrund seiner Kennung in `/etc/machine-id`. `_CAP_EFFECTIVE` gibt an, welche Sonderrechte (*capabilities*) der protokollierende Prozess hat, und `_TRANSPORT` beschreibt, auf welchem Weg die Nachricht systemd erreicht hat (gängig sind außer `syslog` zum Beispiel `stdout` für Nachrichten, die das Programm auf seiner Standardausgabe oder Standardfehlerausgabe geschrieben hat oder `kernel` für Nachrichten, die der Systemkern über `/dev/klog` eingeliefert hat). `_COMM`, `_EXE` und `_CMDLINE` beschreiben alle das Kommando, das gerade ausgeführt wird. `_SYSTEMD_SLICE` und `_SYSTEMD_CGROUP` geben an, wo in der internen Prozessverwaltung von systemd der protokollierende Prozess zu finden ist. Eine genauere Erklärung finden Sie in `systemd.journal-fields`(7).

Nach allen diesen Feldern können Sie suchen, einfach indem Sie sie auf der Kommandozeile von `journalctl` angeben:

```
$ journalctl _HOSTNAME=red _SYSTEMD_UNIT=apache2.service
```

Suchbegriffe mit verschiedenen Feldern werden implizit UND-verknüpft. Taucht dasselbe Feld in mehreren Suchbegriffen auf, werden diese implizit ODER-verknüpft.

Es gibt auch ein explizites ODER:

```
$ journalctl _HOSTNAME=red _UID=70 + _HOSTNAME=blue _UID=80
```

zeigt alle Nachrichten von Prozessen mit der UID 70 auf dem Rechner `red` sowie von Prozessen mit der UID 80 auf dem Rechner `blue`. (Das funktioniert selbstredend nur, wenn Sie die Journale beider Rechner auf Ihrem Rechner konsolidieren.)

Natürlich können Sie diese Suchbegriffe beliebig mit Optionen kombinieren, etwa um zeitliche Einschränkungen zu machen oder sich Tipparbeit zu sparen:

```
$ journalctl -u apache2 _HOSTNAME=red
```

Wenn Sie sich (wie wir) nicht merken können, welche Werte für einen Suchbegriff in Frage kommen, können Sie einfach das Journal fragen:

```
$ journalctl -F _SYSTEMD_UNIT
session-2.scope
```

```
udisks2.service
session-1.scope
polkitd.service
dbus.service
user@1000.service
◁◁◁◁◁
```

Als weitere Vereinfachung funktioniert sogar Kommandozeilenvervollständigung für Feldnamen und -Werte:

```
$ journalctl _SYS⟨Tab⟩                                                    wird zu
$ journalctl _SYSTEMD_⟨Tab⟩
_SYSTEMD_CGROUP= _SYSTEMD_OWNER_UID= _SYSTEMD_SESSION= _SYSTEMD_UNIT=
$ journalctl _SYSTEMD_U⟨Tab⟩                                              wird zu
$ journalctl _SYSTEMD_UNIT=⟨Tab⟩⟨Tab⟩
acpid.service       lightdm.service       ssh.service
anacron.service     networking.service    systemd-journald.service
◁◁◁◁◁
$ journalctl _SYSTEMD_UNIT=ss⟨Tab⟩                                        wird zu
$ journalctl _SYSTEMD_UNIT=ssh.service
```

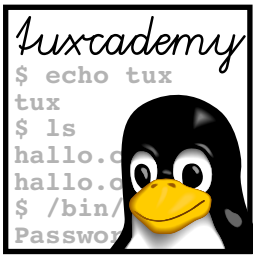
Das Journal und `journald` sind ungeheuer flexibel und leistungsfähig und lassen die traditionelle Methode (Textdateien in `/var/log`) im Vergleich ziemlich primitiv aussehen.

Übungen

21.4 [!2] Experimentieren Sie mit `journalctl`. Wie viele verschiedene Benutzeridentitäten haben auf Ihrem Rechner Nachrichten ans Journal geschickt? Ist gestern zwischen 12 und 13 Uhr etwas Interessantes passiert? Was waren die letzten 10 Meldungen der Priorität `warning`? Überlegen Sie sich selber einige interessante Fragen und beantworten Sie sie.

Zusammenfassung

- Das »Journal« ist ein moderner Systemprotokolldienst, der von systemd zur Verfügung gestellt wird. Es beruht auf binären, datenbankartigen Protokolldateien.
- Das Journal steht entweder in `/run/log/journal` oder (für persistente Protokollierung auf Platte) in `/var/log/journal`.
- Innerhalb von systemd kümmert sich `systemd-journald` um das Journal. Mit `journalctl` können Sie auf das Journal zugreifen.
- `journalctl` erlaubt sehr ausgefeilte Abfragen des Journals.



22

Grundlagen von TCP/IP

Inhalt

22.1 Geschichte und Grundlagen 364
22.1.1 Die Geschichte des Internet 364
22.1.2 Verwaltung des Internet 365
22.2 Technik 366
22.2.1 Überblick. 366
22.2.2 Protokolle 367
22.3 TCP/IP 370
22.3.1 Überblick. 370
22.3.2 Kommunikation von Ende zu Ende: IP und ICMP 371
22.3.3 Die Basis für Dienste: TCP und UDP 374
22.3.4 Die wichtigsten Anwendungsprotokolle 378
22.4 Adressen, Wegleitung und Subnetting 379
22.4.1 Grundlagen 379
22.4.2 Wegleitung 380
22.4.3 IP-Netzklassen 381
22.4.4 Subnetting 382
22.4.5 Private IP-Adressen 382
22.4.6 Masquerading und Portweiterleitung 383
22.5 IPv6. 384
22.5.1 Überblick. 384
22.5.2 IPv6-Adressierung 385

Lernziele

- Den grundlegenden Aufbau der TCP/IP-Protokollfamilie kennen
- Grundlagen der IP-Adressierung kennen
- Konzepte von Subnetting und Routing verstehen
- Die wichtigsten Eigenschaften und Unterschiede von TCP, UDP und ICMP kennen
- Die wichtigsten TCP- und UDP-Dienste kennen
- Die wesentlichen Unterschiede zwischen IPv4 und IPv6 kennen

Vorkenntnisse

- Grundlegende Kenntnisse von Rechnernetzen und TCP/IP-Diensten als Anwender sind hilfreich

22.1 Geschichte und Grundlagen

22.1.1 Die Geschichte des Internet

Die Geschichte der Vernetzung von Computern reicht zurück bis fast an den Anfang des »Computerzeitalters«. Die meisten Techniken der Frühzeit sind heute längst vergessen – das »Internet« hat sich weithin durchgesetzt. Aber was ist »das Internet« überhaupt und wo kommt es her? In diesem Abschnitt geben wir einen kurzen Überblick über seine Geschichte und die Entwicklung der weltweiten Rechnerkommunikation. Wenn Sie das schon woandersher wissen, können Sie gleich zum nächsten Abschnitt weiterblättern. Vielen Dank.

ARPAnet Ursprung des heutigen Internet ist das ARPAnet, dessen Entwicklung vom amerikanischen Verteidigungsministerium finanziert wurde. Wir schreiben die späten 1960er Jahre.

Ziel der Sache war nicht, wie gerne behauptet wird, die Konstruktion einer Kommunikationsinfrastruktur für den Fall eines Atomkriegs, sondern lediglich die Erforschung der Datenübertragung, wobei man gleichzeitig die Kommunikation zwischen den an der Rüstungsforschung beteiligten Firmen und Universitäten vereinfachen wollte.

1969 bestand das ARPAnet aus 4 Knoten; 1970 bis 1972 wurde das *Network Control Protocol* (NCP) als grundlegender Standard für die Kommunikation auf dem ARPAnet implementiert. Der wichtigste Dienst damals war elektronische Post.

NCP

In den 1970er Jahren gewann die Idee eines »Internet«, das verschiedene schon existierende Netze verbinden sollte, an Bedeutung. Man versuchte sich an der Implementierung von »TCP«, einem zuverlässigen Kommunikationsprotokoll auf der Basis eines unzuverlässigen Übertragungsprotokolls (die Idee, mit UDP auch ein unzuverlässiges Kommunikationsprotokoll zur Verfügung zu stellen, kam erst einige Zeit später dazu, was den Namen »TCP/IP« (anstelle von »TCP/UDP/IP« oder ähnlichem) erklären dürfte). Die ersten TCP-Implementierungen erschienen in den frühen 1970ern auf »großen« Systemen wie TOPS-20 oder Tenex; wenig später wurde bewiesen, dass die Implementierung von TCP auf Workstationrechnern wie dem Xerox Alto möglich war, dass also auch solche Rechner am Internet teilnehmen konnten. Das erste Ethernet wurde ebenfalls 1973 am Xerox PARC entwickelt.

Die heutigen grundlegenden TCP/IP-Standards kamen Anfang der 1980er Jahre heraus. Sie wurden testhalber im damaligen BSD – der an der *University of California at Berkeley* entwickelten Unix-Variante – implementiert, das dadurch das Fundament für seine Popularität bei Benutzern und Rechnerherstellern legte. Am 1. Januar 1983 erfolgte die Umstellung des ARPAnet von NCP auf TCP/IP. Wenig später wurde das ursprüngliche ARPAnet administrativ in die beiden Komponenten MILnet (für die militärischen Anwendungen) und ARPAnet (für die Rüstungsforschung) aufgeteilt. Ebenfalls in 1983 wurde mit der Entwicklung des DNS der Grundstein für die künftige Expansion gelegt. In den künftigen Jahren – 1984 bis 1986 – entstanden weitere Netze auf der Basis von TCP/IP, etwa das NSFNET der *National Science Foundation* der USA, und der Begriff »Internet« für die Zusammenfassung aller miteinander verbundenen TCP/IP-basierten Netze begann Fuß zu fassen.

TCP/IP Flag Day
MILnet
ARPAnet
NSFNET

Ende 1989 waren Australien, Deutschland, Großbritannien, Israel, Italien, Japan, Mexiko, die Niederlande und Neuseeland ans Internet angeschlossen. Man zählte jetzt über 160.000 Knoten.

1990 wurde das ARPAnet offiziell außer Dienst gestellt (es war längst im »Internet« aufgegangen) und 1991 das NSFNET für die kommerzielle Nutzung freigegeben. Kommerzielle Anbieter expandierten. Heute ist der überwiegende Teil der Netzinfrastruktur in privater Hand.

Heute existiert ein globales Leitungsnetz mit einem einheitlichen Adressraum. Verwendet werden offene Protokolle und einheitliche Kommunikationsverfahren, so dass sich jeder an der Entwicklung beteiligen kann und das Netz jedem zur

Verfügung steht. Die Weiterentwicklung des Internet ist noch lange nicht abgeschlossen; künftige Neuerungen werden versuchen, anstehende Probleme wie die Verknappung der Adressen und die gestiegenen Sicherheitsbedürfnisse zu beheben.

22.1.2 Verwaltung des Internet

Ein globales Netz wie das Internet kann nicht ohne Verwaltungsstrukturen funktionieren. Diese Aufgabe wurde in den USA angesiedelt, da hier zu Beginn die meisten Netze miteinander verbunden waren. Dort – genauer gesagt beim US-Handelsministerium – liegt die Verwaltung auch noch heute.

Diversen Leuten ist die Dominanz der USA in Fragen des Internets ein Dorn im Auge. Leider weiß man aber nicht so recht, was man dagegen unternehmen soll, da die Amerikaner das Heft nicht formell aus der Hand geben wollen. Auf der anderen Seite verfolgt das US-Handelsministerium einen ausgeprägten Laissez-faire-Ansatz, der die Kritiker bis zu einem gewissen Grad mit dem *status quo* versöhnt.

Internet Society

Theoretisch liegt die Kontrolle über das Internet in der Hand der »Internet Society« (ISOC), einer 1992 gegründeten internationalen nicht gewinnorientierten Organisation. Ihre Mitglieder sind Regierungen, Firmen, Universitäten, andere Organisationen und auch Einzelpersonen (jeder darf mitmachen).

Ziel der ISOC war, den bis dahin juristisch sehr schwammigen Institutionen wie der IETF (siehe unten) einen formellen Rahmen zu geben und auch deren finanzielle Unterstützung zu sichern. Ferner hält die ISOC das Urheberrecht an den RFCs, den Standarddokumenten für das Internet, die allen Interessenten frei zur Verfügung stehen.

Die Aktivitäten der ISOC teilen sich grob auf drei Kategorien auf:

Normen Die ISOC ist der Überbau für eine Reihe von Gremien, die sich um die technische Weiterentwicklung des Internet bemühen. Unter anderem:

- Das *Internet Architecture Board* (IAB) ist das Komitee, das die Aufsicht über die technische Weiterentwicklung des Internet hat. Das IAB kümmert sich zum Beispiel um die Veröffentlichung der RFCs und berät die Leitung der ISOC in technischen Fragen.

  Das IAB hat im Moment ein gutes Dutzend Mitglieder (Menschen), die vom »IETF-Nominierungskomitee« ausgewählt wurden, einen ebenfalls vom IETF-Nominierungskomitee eingesetzten Vorsitzenden und ein paar Ex-Officio-Mitglieder und Vertreter anderer Organisationen.

- Die *Internet Engineering Task Force* (IETF) kümmert sich um die tatsächliche Entwicklung von Internet-Standards und arbeitet dabei eng mit Institutionen wie ISO/IEC und dem World Wide Web Consortium (W3C) zusammen. Die IETF ist eine offene Organisation ohne Mitgliedschaft, die von »Freiwilligen« betrieben wird (deren Arbeitgeber in der Regel für die Kosten aufkommen). Innerhalb der IETF gibt es eine große Anzahl von »Arbeitsgruppen«, die sich gemäß ihrer Thematik in »Gebiete« (engl. *areas*) aufteilen. Jedes Gebiet hat einen oder zwei *area directors*, die zusammen mit dem Vorsitzenden der IETF die *Internet Engineering Steering Group* (IESG) bilden. Diese ist für die Arbeit der IETF verantwortlich.

  Wegen ihrer amorphen Struktur ist es schwer zu sagen, wie groß die IETF zu einem gegebenen Zeitpunkt tatsächlich ist. In den ersten Jahren seit ihrer Gründung 1986 schwankte die Anwesenheit bei den regelmäßigen Treffen zwischen 30 und 120 Personen. Seit

dem explosionsartigen Wachstum des Internet in den 1990er Jahren ist der Kreis etwas größer geworden, auch wenn er nach dem Platzen der »Dot-Com«-Blase von bis zu 3000 Personen im Jahre 2000 wieder auf um die 1200 abgesunken ist.

Das Mantra der IETF ist »Grober Konsens und laufender Code« (*rough consensus and running code*) – man besteht bei Entscheidungen nicht auf Einstimmigkeit, aber möchte den größten Teil der Gruppe hinter sich wissen. Außerdem wird sehr viel Wert auf Lösungen gelegt, die tatsächlich in der Praxis funktionieren. Dies und die Tatsache, dass die Arbeit zum größten Teil von Freiwilligen geleistet wird, können dazu führen, dass es mitunter lange dauert, bis eine IETF-Arbeitsgruppe Ergebnisse liefert – vor allem, wenn es zu wenige oder zu viele Interessenten gibt, die dazu beitragen möchten.

- Die *Internet Corporation for Assigned Names and Numbers*, kurz ICANN, ist eine andere nicht gewinnorientierte Organisation, die 1998 gegründet wurde, um einige Dinge zu übernehmen, die vorher andere Organisationen (vor allem die IANA, siehe nächster Punkt) direkt im Namen der US-Regierung erledigt haben. Insbesondere gemeint ist die Verteilung von IP-Adressen und DNS-Top-Level-Domainnamen. Gerade letztere ist ein extrem politisches Thema und immer wieder Quelle von Konflikten.
- Die *Internet Assigned Numbers Authority* (IANA) kümmert sich um die tatsächliche Zuteilung von IP-Adressen und die Verwaltung der DNS-Root-Server. Administrativ ist die IANA Teil der ICANN. Außerdem verwaltet die IANA alle global eindeutigen Namen und Zahlen in Internet-Protokollen, die als RFCs veröffentlicht sind. Dabei arbeitet sie eng mit der IETF und der RFC-Redaktion zusammen.

Die Zuteilung von IP-Adressen delegiert die IANA weiter an sogenannte *Regional Internet Registries* (RIRs), die jeweils für einen Teil der Welt den »Vertrieb« (typischerweise) an Provider übernehmen. Im Moment gibt es fünf RIRs, zuständig für Deutschland ist das RIPE NCC.

Fortbildung Die ISOC veranstaltet Konferenzen, Seminare und Kurse über wichtige Internet-Themen, unterstützt lokale Internet-Organisationen und gibt Internet-Experten in Entwicklungsländern durch finanzielle Hilfen die Möglichkeit, an der Weiterentwicklung und Diskussion teilzunehmen.

Politische Willensbildung Die ISOC kooperiert mit Regierungen und nationalen und internationalen Organisationen, um die Ideen und Werte der ISOC voranzubringen. Erklärtes Ziel der ISOC ist »eine Zukunft, in der Menschen in allen Teilen der Welt das Internet verwenden können, um ihre Lebensqualität zu verbessern«.

22.2 Technik

22.2.1 Überblick

Computer verarbeiten digitale Informationen. Diese Informationen werden in der »wirklichen Welt« allerdings durch physikalische Phänomene wie Spannung, Ladung, Licht und ähnliches dargestellt, und die wirkliche Welt ist nun mal fundamental »analog«. Die erste Herausforderung der Datenkommunikation ist also, die digitalen Informationen im Computer für die Übertragung zu einem anderen Computer in etwas Analoges umzuwandeln – zum Beispiel eine Reihenfolge von elektrischen Impulsen auf einem Kabel – und am anderen Ende daraus wieder digitale Informationen zu machen. Die nächste Herausforderung besteht darin, das

zum Funktionieren zu bringen, wenn der eine Computer in Berlin und der andere in Neuseeland steht.

Man kann Datennetze ganz grob und ohne nähere Beleuchtung der verwendeten Technik in zwei Gruppen einteilen: **Lokale Netze** (engl. *Local Area Networks* oder »LANs«) verbinden eine kleine Anzahl von Stationen in einem räumlich begrenzten Areal, **Weitverkehrsnetze** (engl. *Wide Area Networks* oder »WANs«) eine potentiell große Anzahl von Stationen in einem räumlich sehr großen Areal.

Lokale Netze

Weitverkehrsnetze

Bei LANs ist der Eigentümer (eine Firma oder andere Organisation oder – heute oft – ein Haushalt) in der Regel auch der Betreiber und der einzige Nutzer, und das Netz verfügt über eine hohe Bandbreite (100 MBit/s und mehr). WANs dagegen verbinden eine Vielzahl von Nutzern, die normalerweise nicht Eigentümer des Netzes sind; die Bandbreiten sind kleiner und die Nutzung teurer.

Es gibt eine Vielzahl verschiedener Netzwerktechniken für die unterschiedlichsten Anforderungen – von drahtlosen Verbindungen über sehr kurze Strecken (Bluetooth) über typische LAN-Technik wie Ethernet bis zu Glasfaserverbindungen mit ATM für WANs. Als Programmierer und Systemadministratoren möchten wir mit deren ekligen elektro- und nachrichtentechnischen Details allerdings so wenig wie möglich konfrontiert werden. Deswegen sprechen wir von einem »Protokollstapel« und versuchen, die einzelnen Bestandteile – den »elektrischen« Teil, die prinzipielle Kommunikation zwischen Rechnern im selben Netz, die prinzipielle Kommunikation zwischen Rechnern in verschiedenen Netzen und schließlich konkrete »Dienste« wie E-Mail oder das World Wide Web sauber zu trennen. Aber der Reihe nach.

22.2.2 Protokolle

Ein »Protokoll« ist eine Vereinbarung dafür, wie zwei (oder mehr) Stationen in einem Netz sich unterhalten. Das Spektrum der möglichen Protokolle reicht von Regeln für elektrische Signale auf einem Ethernet-Kabel oder Funksignale in einem WLAN bis hin zu (beispielsweise) Protokollen, die den Zugriff auf einen SQL-Datenbankserver regeln. Protokolle lassen sich grob in drei Klassen einteilen:

Übertragungsprotokolle (oft auch »Zugriffsverfahren« genannt) regeln die Datenübertragung grob gesagt auf der Ebene von Netzwerkkarten und Leitungen. Ihre Ausgestaltung hängt von den elektro- und nachrichtentechnischen Eigenschaften und Einschränkungen ab, die aus der Realisierung in »Hardware« folgen. Die Kommunikation zwischen zwei Rechnern über ein serielles Nullmodem-Kabel läuft zum Beispiel ganz anders ab als die Übertragung von Daten über eine Funkverbindung im WLAN, und es existieren völlig andere Anforderungen an die verwendeten Zugriffsverfahren.

Das gängigste Zugriffsverfahren im LAN-Bereich ist Ethernet, auch wenn das heutige Ethernet mit dem gleichnamigen Original von 1973 so gut wie nichts mehr zu tun hat (OK, beide involvieren Elektrizität, aber da hört die Ähnlichkeit auch fast schon auf.) Andere Standards wie Token-Ring oder Feldbussysteme tauchen nur noch in Spezialanwendungen auf. Heute populär sind auch WLAN-Zugriffsverfahren wie IEEE 802.11.

Kommunikationsprotokolle dienen dazu, die Kommunikation zwischen Rechnern in verschiedenen Netzen zu regeln, ohne dass dafür eine genaue Kenntnis der verwendeten Zugriffsverfahren notwendig ist. Damit Sie auf Ihrem Heim-PC in Deutschland eine Webseite über Kängurus auf einem Server einer Universität in Australien anschauen können, möchten Sie nicht wissen müssen, dass Ihr PC über Ethernet mit Ihrem Heim-Router redet,

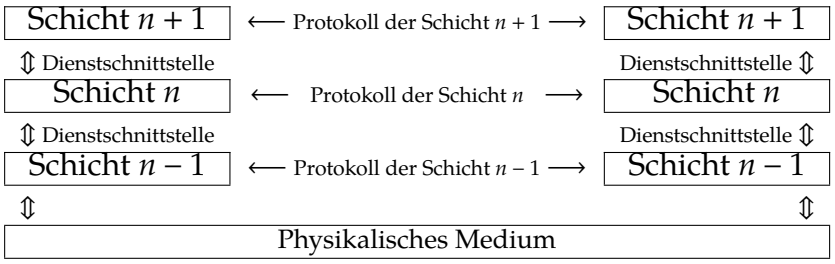


Bild 22.1: Protokolle und Dienstschnittstellen

dieser wiederum über ATM mit dem DSLAM draußen im Telekom-Häuschen auf der anderen Straßenseite, von da geht es dann via Glasfaser und um einige Ecken herum nach Australien und so weiter und so fort – Sie geben einfach `www.kangaroos-r-us.au` in Ihrem Browser ein. Dass Ihr Heim-PC den entfernten Web-Server findet und mit ihm Daten austauschen kann, verdanken Sie den Kommunikationsprotokollen.

Kommunikationsprotokolle sind dafür da, dass Sie sich nicht mit den Übertragungsprotokollen herumärgern müssen, aber sie können ohne diese natürlich nicht funktionieren. Ziel der Kommunikationsprotokolle ist, die unappetitlichen Details der Übertragungsprotokolle vor Ihnen zu verstecken – etwa so, wie das Gaspedal Ihres Autos dazu dient, die Kennlinie der elektronischen Benzineinspritzung vor Ihnen zu verstecken.

Die Kommunikationsprotokolle, die uns interessieren, sind natürlich IP, TCP und UDP. Dazu kommt noch ICMP als »Infrastrukturprotokoll«, das zur Diagnose, Steuerung und Fehlermeldung dient.

Anwendungsprotokolle realisieren auf der Basis der Kommunikationsprotokolle tatsächliche Dienste wie elektronische Post, Dateiübertragung oder Internet-Telefonie. Wenn Kommunikationsprotokolle dafür gut sind, beliebige Bits und Bytes nach Australien zu schicken und andere von dort zurückzubekommen, dann sorgen Anwendungsprotokolle dafür, dass Sie mit diesen Bits und Bytes tatsächlich etwas anfangen können.

Typische Anwendungsprotokolle, mit denen Sie als Linux-Administrator konfrontiert werden könnten, sind zum Beispiel SMTP, FTP, SSH, DNS, HTTP, POP3 oder IMAP, zum Teil auch mit »sicheren«, also authentisierten und verschlüsselten Ablegern. Alle diese Protokolle werden von Anwendungsprogrammen wie Mail-Clients oder Web-Browsern verwendet und bauen auf Kommunikationsprotokollen wie TCP oder UDP auf.

Protokolldateneinheiten

Die Daten, die über ein Protokoll ausgetauscht werden, nennen wir abstrakt Protokolldateneinheiten – je nach Protokoll können sie spezifischere Namen haben, etwa »Pakete«, »Datagramme«, »Segmente« oder »Frames«.

Schichtenmodell

Aus der Tatsache, dass Kommunikationsprotokolle dafür gedacht sind, die Details der Übertragungsprotokolle zu verstecken, und Anwendungsprotokolle wiederum dazu dienen, die Details der Kommunikationsprotokolle zu verstecken, kann man ein »Schichtenmodell« (Bild 22.1) konstruieren, bei dem die Übertragungsprotokolle die unterste und die Anwendungsprotokolle die oberste Ebene einnehmen. (Daher kommt auch der Begriff »Protokollstapel«.) Jede Schicht auf der Absenderseite empfängt Daten »von oben« und gibt sie »nach unten« weiter; auf der Empfängerseite werden sie »von unten« empfangen und »nach oben« weitergegeben. Konzeptuell sagt man trotzdem, dass zwei Stationen zum Beispiel »über HTTP« kommunizieren, auch wenn die HTTP-Daten in Wirklichkeit über

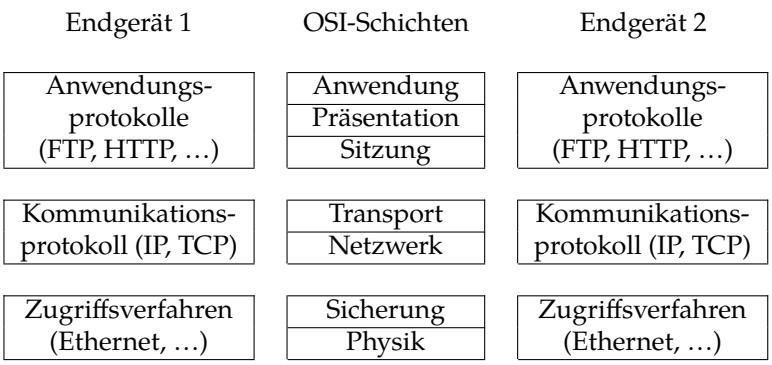


Bild 22.2: ISO/OSI-Referenzmodell

TCP, IP und einen ganzen Zoo von Übertragungsprotokollen von einer Station zur anderen fließen und dort wiederum die IP- und TCP-Schichten passieren müssen, bis sie wieder als HTTP-Daten »sichtbar« werden.

Technisch empfängt auf der Absenderseite auf jeder Ebene das entsprechende Protokoll an seiner Dienstschnittstelle ein »Datenpaket« von der darüberliegenden Schicht und fügt ihm einen »Kopf« mit allen für seine Aufgabe wichtigen Informationen hinzu, bevor es über die Dienstschnittstelle des Protokolls der darunterliegenden Schicht weitergegeben wird. Alles, was an der Dienstschnittstelle übergeben wird, betrachtet die nächstuntere Ebene als Daten; die Kopfdaten des vorherigen Protokolls sind auf der tieferen Ebene nicht von Interesse. Auf der Empfängerseite der Kommunikation durchlaufen die Pakete die gleichen Schichten in umgekehrter Reihenfolge, wobei jede Ebene »ihren« Kopf wieder entfernt und die »Nutzdaten« nach oben weitergibt.

Kopf

Das bekannteste Schichtenmodell ist das »ISO/OSI-Referenzmodell« (Bild 22.2). ISO/OSI (kurz für *International Organisation for Standardisation/Open Systems Interconnection*) war die Basis einer Protokollfamilie, die vom CCITT, der Weltorganisation der Telekommunikationsbehörden und -unternehmen, vorgeschlagen wurde.

ISO/OSI-Referenzmodell

Die ISO/OSI-Netzwerkstandards haben sich nie durchsetzen können – sie waren zu kompliziert und praxisfern, um nützlich zu sein, und die Standarddokumente waren nur schwer zugänglich –, das Referenzmodell mit seinen sieben (!) Schichten ist aber übriggeblieben und wird gerne herangezogen, um die Abläufe bei der Datenübertragung zu erklären.

Viele Protokollstapel lassen sich nicht exakt auf das ISO/OSI-Referenzmodell abbilden. Dies kommt zum einen daher, dass sich nicht jeder Hersteller an die Definitionen des Modells hält, zum anderen aber auch daher, dass einige Protokollstapel schon älter als das OSI-Modell sind. Sie sollten auch nicht den Fehler machen, das ISO/OSI-Referenzmodell mit einer verbindlichen »Norm« für die Struktur von Netzwerksoftware oder gar mit einer Implementierungsanleitung zu verwechseln. Das ISO/OSI-Referenzmodell ist lediglich eine Klarstellung der verschiedenen beteiligten Konzepte und macht es bequemer, über sie zu reden. Trotzdem hier ein kurzer Überblick über die beteiligten Schichten:

- Die Schichten 1 und 2 (physikalische Schicht und Sicherungsschicht) beschreiben, wie die Daten auf das Kabel geschickt werden. Dies beinhaltet neben dem Zugriffsverfahren auch die Codierung der Daten.
- Die Schicht 3 (Netzwerkschicht) definiert die Funktionen, die für die Wegleitung der Daten benötigt werden. Dies umfasst auch die dafür erforderliche Adressierung.
- Der Transport der Applikationsdaten wird in der Schicht 4 (Transportschicht) beschrieben. Hier unterscheidet man zwischen verbindungsorientierten und verbindungslosen Diensten.

- Die Schichten 5, 6 und 7 (Sitzungs-, Präsentations- und Anwendungsschicht) werden in der Praxis nicht so häufig explizit unterschieden (z. B. nicht bei den TCP/IP-Protokollen). Sie beschreiben die systemunabhängige Darstellung von Daten im Netzwerk sowie die Schnittstelle zu den Anwendungsprotokollen.
- Andy Tanenbaum [Tan02] postuliert zusätzlich noch die Schichten 8 und 9 (die finanzielle und die politische Schicht). Während diese Schichten in der Praxis wohlbekannt sind, haben sie ins offizielle ISO/OSI-Referenzmodell noch keinen Eingang gefunden.

Übungen

22.1 [2] Rekapitulieren Sie kurz die Unterschiede zwischen Übertragungs-, Kommunikations- und Anwendungsprotokollen. Nennen Sie Beispiele für die jeweiligen Sorten. (Kennen Sie welche, die nicht aus dem TCP/IP-Umfeld stammen?)

22.2 [1] Was ist der wesentliche Unterschied zwischen den ISO/OSI-Schichten 2 und 3?

22.3 TCP/IP

22.3.1 Überblick

TCP/IP steht für *Transmission Control Protocol/Internet Protocol* und ist die heute am häufigsten eingesetzte Methode für den Datentransfer in Rechnernetzen, seien es nur zwei Rechner in einem lokalen Netz oder aber das ganze Internet. Bei TCP/IP handelt es sich nicht nur um ein einzelnes Protokoll, sondern um eine Fülle unterschiedlicher aufeinander aufbauender Protokolle mit teils sehr unterschiedlichen Aufgaben. Man spricht von einer »Protokollfamilie«.

Die Protokolle der TCP/IP-Protokollfamilie lassen sich zumindest ungefähr in das ISO/OSI-Schichtenmodell aus Bild 22.2 einordnen. Die wichtigsten sind hier kurz aufgelistet:

Netzzugangsschicht Ethernet, IEEE 802.11, PPP (dies sind strenggenommen keine TCP/IP-Protokolle)

Internetschicht IP, ICMP, ARP

Transportschicht TCP, UDP, …

Anwendungsschicht HTTP, DNS, FTP, SSH, NIS, NFS, LDAP, …

Um den Ablauf der Datenübertragung besser zu verstehen und Fehler, die hierbei auftreten, eingrenzen und auffinden zu können, ist es sehr nützlich, die Struktur der wichtigsten Protokolle und den Aufbau der dazugehörigen Protokolldateneinheiten zu kennen. Wir erklären im folgenden kurz die wichtigsten TCP/IP-Protokolle aus der Internet- und Transportschicht.

Übungen

22.3 [2] Welche anderen Protokolle aus der TCP/IP-Protokollfamile fallen Ihnen ein? Zu welchen der vier Schichten gehören sie?

22.3.2 Kommunikation von Ende zu Ende: IP und ICMP

IP IP stellt die Verbindung zwischen zwei Systemen her. Es ist als Protokoll der ISO/OSI-Schicht 3 dafür verantwortlich, dass die Daten durch das Internet den Weg vom Sender zum Empfänger finden. Der Haken an der Sache ist, dass dieser Weg weite Strecken umfassen kann, die aus diversen unabhängigen Abschnitten mit deutlich verschiedener Netzwerktechnik bestehen und die deutlich unterschiedliche Kommunikationsparameter aufweisen. Stellen wir uns vor, ein Benutzer »surft« zu Hause im Internet. Sein Rechner ist über ein analoges Modem und das Telefonnetz per PPP mit einem Einwahlrechner bei einem Provider verbunden, der die Verbindung ins eigentliche Internet herstellt. Die Web-Anfragen werden dann zum Beispiel über Glasfaserleitungen mit ATM um die halbe Welt geschickt, bis sie im Rechenzentrum einer Universität ankommen, wo die Daten per FDDI-basiertem Campusnetz zu einem Institutsrouter fließen, der sie dann an den über Ethernet angeschlossenen Web-Server leitet. Die Web-Seiten nehmen den umgekehrten Weg zurück. Alle diese verschiedenen Teilstrecken verwenden nicht nur unterschiedliche Netzwerktechnik, sondern auch unterschiedliche »lokale« Adressen – während bei der PPP-Übertragung überhaupt keine Adressierung nötig ist (es gibt ja nur zwei Kommunikationspartner), funktioniert ein Ethernet beispielsweise auf der Basis von 48 Bit breiten »MAC«-Adressen.

Provider

Eine der Leistungen von IP besteht darin, einen »globalen« Adressraum zur Verfügung zu stellen, der jedem am Internet beteiligten System eine eindeutige Adresse gibt, über die es identifiziert werden kann. Ferner sorgt es für die Wegleitung (engl. *routing*) von einem System zum anderen ohne Berücksichtigung der tatsächlich verwendeten Netzwerktechnik.

Adressraum

Wegleitung

IP ist ein **verbindungsloses Protokoll**, das heißt, im Gegensatz z. B. zum traditionellen Telefonnetz wird keine feste Verbindung (»Draht«) für die Kommunikation zweier Systeme zur Verfügung gestellt[1], sondern die zu übertragenden Daten werden in Häppchen, sogenannte **Datagramme**, eingeteilt, die unabhängig voneinander adressiert und zugestellt werden. Prinzipiell kann jedes Datagramm einen anderen Weg zum Empfänger nehmen als das vorige; dies macht IP unempfindlich gegen Ausfälle von Leitungen oder Vermittlungsrechnern, solange sich noch irgendein Weg vom Quell- zum Zielsystem finden läßt. IP gibt keine Garantie, dass alle abgeschickten Daten auch tatsächlich beim Zielsystem ankommen, und genausowenig wird garantiert, dass die Daten, die tatsächlich ankommen, in derselben Reihenfolge ankommen, in der sie abgeschickt wurden. Es ist die Aufgabe »höhergelegener« Protokolle, hier für Ordnung zu sorgen, falls die Anwendung das erfordert.

verbindungsloses Protokoll

Datagramme

Stellen Sie sich vor, Sie möchten Ihrer Tante in Australien einen langen Text mit der Post schicken. Wenn Sie das »à la IP« machen wollten, würden Sie den Text auf eine Menge von einzelnen Postkarten schreiben. Die Chancen stehen gut, dass Ihre Postkarten auf dem Weg nach Känguru-Land durcheinander geraten und der dortige Postbote sie nicht genau in derselben Reihenfolge in den Briefkasten Ihrer Tante steckt, in der Sie sie hier eingeworfen haben. Es ist auch durchaus möglich, dass die eine oder andere Postkarte irgendwo liegenbleibt oder ganz verlorengeht.

Warum ist das ein Vorteil? Das traditionelle Telefonnetz mit seinen von Ende zu Ende durchgeschalteten Drahtverbindungen war sehr anfällig gegenüber Störungen – fiel irgendein Leitungsabschnitt aus, brach das Gespräch komplett zusammen und musste neu aufgebaut werden (im Zeitalter der Handvermittlung eine größere Sache). Ergibt sich bei verbindungsloser Übertragung eine Störung oder Unterbrechung, kann das Netz für spätere Datagramme alternative Routen suchen, die die gestörte Stelle umgehen. Verfahren wie TCP erlauben es, zu erkennen, welche Daten durch die Störung verloren gegangen sind, und sorgen dafür, diese nochmals zu versenden.

[1] Auch das Telefonnetz – im Fachjargon POTS (für *plain old telephone system*) genannt – funktioniert längst nicht mehr so.

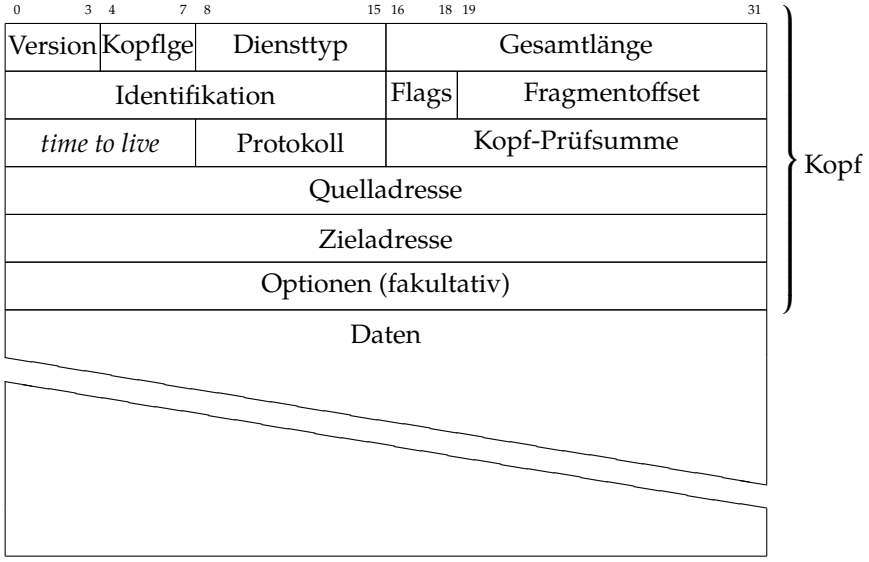


Bild 22.3: Aufbau eines IP-Datagramms. Jede Zeile entspricht 32 Bit.

Fragmentierung

Außerdem kümmert IP sich um **Fragmentierung**. IP-Datagramme dürfen bis zu 65535 Bytes lang sein, aber bei den meisten Transportprotokollen sind nur wesentlich kürzere Protokolldateneinheiten erlaubt – bei Ethernet beispielsweise nur bis zu 1500 Bytes. Längere Datagramme müssen darum »fragmentiert« übertragen werden – am Anfang einer entsprechenden Teilstrecke wird das Datagramm auseinandergenommen, in nummerierte Fragmente aufgeteilt und später wieder zusammengesetzt. IP sorgt dafür, dass nur solche Datagramme als offiziell empfangen gelten, in denen kein Fragment fehlt.

Die offizielle Spezifikation von IP ist [RFC0791]. Lesen müssen Sie das nicht, aber es hilft vielleicht gegen Schlafstörungen.

Bild 22.3 zeigt den Aufbau eines IP-Datagramms. Zumindest zwei Felder sollten wir noch kurz erklären:

- Die *time to live* (oder TTL) gibt die maximale »Lebensdauer« des Datagramms an. Sie wird vom Absender des Datagramms gesetzt und von jeder Station, die das Datagramm auf dem Weg zum Empfänger durchläuft, um 1 vermindert. Wenn die TTL den Wert 0 erreicht, wird das Datagramm verworfen und der Absender benachrichtigt. Das Ziel ist die Vermeidung von »fliegenden Holländern« – Datagrammen, die aufgrund von Fehlern bei der Wegleitung ziellos im Internet umherirren, ohne jemals ihren Bestimmungsort zu erreichen. (Gerne als Standard verwendet werden Werte wie 64, die bei der aktuellen Ausdehnung des Internet absolut jenseits von Gut und Böse sind.)
- Der Diensttyp (engl. *type of service*, TOS) gibt an, welche Dienstgüte für das Datagramm erwünscht ist. Hier können Sie theoretisch neben einer von sieben Vorrangstufen (die ignoriert werden) noch die Attribute »niedrige Verzögerung«, »hoher Durchsatz«, »hohe Zuverlässigkeit« und »niedriger Preis« angeben. Ob das allerdings in irgendeiner Form für einen Unterschied bei der Zustellung sorgt, ist äußerst zweifelhaft, da diese Angaben nicht verbindlich sind und Router sie gerne ignorieren. (Wenn das nicht so wäre, würden wahrscheinlich alle Datagramme *alle* diese wünschenswerten Optionen einschalten.)

ICMP Ein weiteres wichtiges Protokoll ist das *Internet Control Message Protocol* [RFC0792], kurz ICMP genannt (siehe Bild 22.4). Es wird zur Netzverwaltung

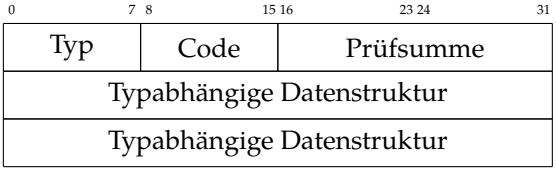


Bild 22.4: Aufbau eines ICMP-Pakets

benutzt und dazu, Probleme mit dem Netz zu melden, etwa wenn eine Verbindung nicht zustande kommt oder ein Teilnetz nicht erreichbar ist. Das sehr bekannte Programm `ping` zum Beispiel verwendet zwei spezielle ICMP-Meldungen (`echo request` und `echo reply`). Das ICMP-Paket wird als »Daten« in einem IP-Datagramm transportiert und enthält abhängig vom Code verschiedene weitere Datenfelder.

IP und Übertragungsprotokolle Damit wir über IP Daten ungeachtet der tatsächlich verwendeten Netzwerktechnik übertragen können, müssen wir von Fall zu Fall definieren, wie IP-Datagramme über das betreffende Netz – sei es Ethernet, PPP über eine analoge Telefonleitung, ATM, WLAN, … – transportiert werden.

In einem Ethernet zum Beispiel sind alle Stationen zumindest konzeptuell an ein gemeinsames Medium angeschlossen – im »klassischen« Ethernet an ein einziges langes Koaxialkabel, das von einer Station zur nächsten läuft, heute zumeist mit Twisted-Pair-Kabeln an einen gemeinsamen Sternverteiler oder Switch. Alles, was eine Station sendet, wird von allen anderen Stationen empfangen, wobei diese sich in der Regel nur diejenigen Protokolldateneinheiten heraussuchen, die wirklich für sie gemeint sind (heutzutage helfen auch die Switches, indem sie den Verkehr »vorsortieren«). Senden zwei Stationen gleichzeitig, kommt es zu einer Kollision, die dadurch behoben wird, dass beide ihren Sendevorgang abbrechen, eine zufällige Zeitspanne warten und es dann wieder versuchen. Ein solches gemeinsames Medium im Ethernet heißt auch »Segment«.

Kollision

Segment

Jede Ethernet-Schnittstelle hat eine eindeutige Adresse, die 48 Bit lange »MAC-Adresse« (kurz für *medium access control* – Medienzugangssteuerung). Protokolldateneinheiten im Ethernet, die sogenannten Frames, können Sie entweder an spezielle andere Stationen im selben Segment verschicken, indem Sie deren MAC-Adresse als Empfänger eintragen – das Frame wird zwar von allen Stationen gesehen, aber von allen außer der adressierten Station ignoriert –, oder als Rundruf (engl. *broadcast*) an *alle* Stationen im Segment senden.

MAC-Adresse

Frames

Ethernet-Netzwerkkarten unterstützen in der Regel auch den sogenannten *promiscuous mode*, in dem sie *alle* Frames – auch die, die sie eigentlich gar nichts angehen – ans System weiterreichen. Dies ist die Basis für interessante Software wie Netzwerkanalyseprogramme und Cracker-Werkzeuge.

Dies macht man sich für die Integration von IP und Ethernet zunutze. Will eine Station (nennen wir sie einmal *A*) mit einer anderen Station (*B*) kommunizieren, deren IP-Adresse sie kennt, deren MAC-Adresse ihr aber nicht bekannt ist, fragt sie zunächst per Ethernet-Broadcast alle angeschlossenen Stationen:

> Station *A*: Wer hat hier die IP-Adresse `203.177.8.4`?
> Station *B*: Ich, und meine MAC-Adresse ist `00:06:5B:D7:30:6F`

Dieser Vorgang folgt dem *Address Resolution Protocol* (ARP, [RFC0826]). Hat Station *A* die MAC-Adresse von Station *B* erhalten, speichert sie diese für eine gewisse Zeit in ihrem »ARP-Cache«, um die Anfrage nicht für jedes Frame wiederholen zu müssen; IP-Datagramme an Stationen, deren IP- und MAC-Adressen im ARP-Cache stehen, können Sie auf Ethernet-Ebene direkt adressieren, indem Sie sie als »Nutzdaten« in Ethernet-Frames einbetten. Auf den ARP-Cache können Sie mit

ARP

ARP-Cache

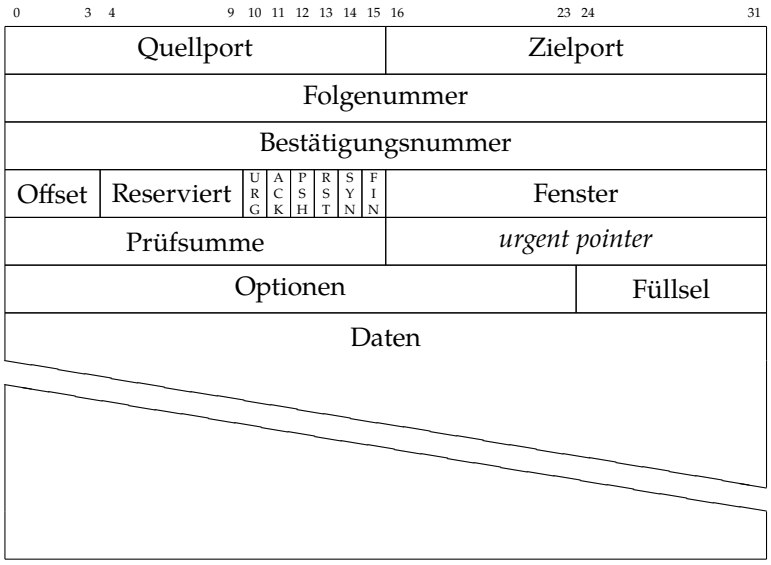


Bild 22.5: Aufbau eines TCP-Segments

dem Kommando `arp` zugreifen – nicht nur lesen, sondern auch Einträge schreiben. Die Ausgabe von `arp` kann beispielsweise so aussehen:

```
# arp
Address              Hwtype  Hwaddress           Flags  Mask  Iface
server.example.org   ether   00:50:DB:63:62:CD   C             eth0
```

Routing

Datagramme an IP-Adressen, die nicht zu Stationen im selben Ethernet-Segment gehören, müssen **geroutet** werden (Abschnitt 22.4.2).

Übungen

22.4 [3] Schätzen Sie die minimale TTL, die notwendig ist, damit Sie von Ihrem Rechner aus alle anderen Stationen auf dem Internet erreichen können. Wie könnten Sie die minimale TTL ermitteln, die zum Erreichen einer gegebenen Station nötig ist? Ist diese Zahl eine Konstante?

22.3.3 Die Basis für Dienste: TCP und UDP

TCP Das »Transmission Control Protocol« (TCP) ist ein zuverlässiges, verbindungsorientiertes Protokoll, das u. a. in [RFC0793] definiert ist. Im Gegensatz zum verbindungslosen IP kennt TCP Operationen zum Verbindungsauf- und -abbau, mit denen zumindest eine »virtuelle« Verbindung zwischen Quell- und Zielsystem geschaltet wird – da TCP-Daten wie alle anderen Daten auch über IP übertragen werden, erfolgt die tatsächliche Datenübertragung nach wie vor verbindungslos und unzuverlässig. TCP erreicht Zuverlässigkeit, indem die Gegenstelle die Ankunft jedes Pakets (im TCP-Jargon »Segment«) bestätigt. Jede der beiden kommunizierenden Stationen versieht ihre Segmente mit Folgenummern (engl. *sequence numbers*), die die Gegenstelle in einem ihrer nächsten Segmente als »angekommen« quittiert. Kommt innerhalb einer gewissen definierten Zeitspanne keine solche Quittung, versucht die sendende Station, das Segment erneut zu schicken, um es vielleicht diesmal bestätigt zu bekommen. Damit die Effizienz darunter nicht zu sehr leidet, wird ein *sliding-window*-Protokoll eingesetzt, so dass eine gewisse Anzahl von Segmenten gleichzeitig unbestätigt bleiben darf. Trotzdem ist TCP deutlich langsamer als IP.

Folgenummer

Eigentlich basieren die Bestätigungen von TCP auf Oktetten (vulgo Bytes),

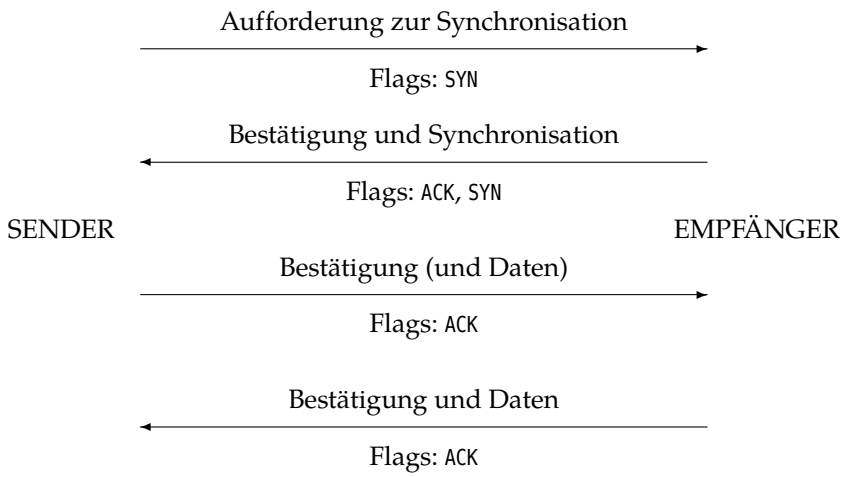


Bild 22.6: Aufbau einer TCP-Verbindung: Der Drei-Wege-Handshake

nicht auf Segmenten – aber für unsere Zwecke ist der Unterschied zunächst akademisch.

Jedes TCP-Segment hat einen mindestens 20 Byte großen Kopf (Bild 22.5) zusätzlich zum IP-Kopf. (Sie erinnern sich: das TCP-Segment *inklusive* TCP-Kopf gilt aus der Sicht von IP, dem Protokoll der darunterliegenden Schicht, als »Daten«.) Anhand einer Prüfsumme können die Daten auf Fehler überprüft werden. Jedes System unterstützt viele unabhängige, gleichzeitige TCP-Verbindungen, zwischen denen anhand von **Portnummern** unterschieden wird.

Portnummern

Die Kombination aus einer IP-Adresse und einer Portnummer zusammen mit der IP-Adresse und Portnummer der Gegenstelle bezeichnet man auch als *socket*. (Derselbe TCP-Port auf einer Station darf gleichzeitig an mehreren TCP-Verbindungen mit unterschiedlichen Gegenstellen – definiert durch IP-Adresse und Portnummer – beteiligt sein.)

Der Aufbau der virtuellen Verbindung erfolgt über den sogenannten **Drei-Wege-Handshake** (engl. *three-way handshake*, siehe Bild 22.6). Im Drei-Wege-Handshake einigen die Kommunikationspartner sich über die zu verwendenden Folgenummern. Hierbei spielen zwei **Flags** im TCP-Kopf, `SYN` und `ACK`, eine entscheidende Rolle. Im ersten Datensegment, das der Absender dem Empfänger schickt, ist das `SYN`-Flag gesetzt und das `ACK`-Flag nicht. Ein solches Segment signalisiert den Verbindungswunsch. Der Empfänger bestätigt das mit einem TCP-Segment, bei dem sowohl das `SYN`- als auch das `ACK`-Flag gesetzt sind. Dieses Segment bestätigt der Absender wiederum mit einem Segment, das das `ACK`-Flag, aber *nicht* das `SYN`-Flag gesetzt hat. An diesem Punkt steht die Verbindung. Darauffolgende TCP-Segmente haben ebenfalls nur noch das `ACK`-Flag gesetzt. – Am Ende der Kommunikation wird die Verbindung über einen Zwei-Wege-Handshake mit dem `FIN`-Flag wieder abgebaut.

Drei-Wege-Handshake

Flags

Am Anfang der Verbindung müssen die beiden Stationen sich einig werden, aber abgebaut werden kann die Verbindung auch unilateral. Das ist sogar dringend nötig, damit Kommandos wie das folgende funktionieren:

```
$ cat bla | ssh blue sort
```

Hier starten wir über die Secure Shell (siehe Kapitel 25) auf dem Rechner `blue` das Kommando `sort` und versorgen es über die Standardeingabe mit

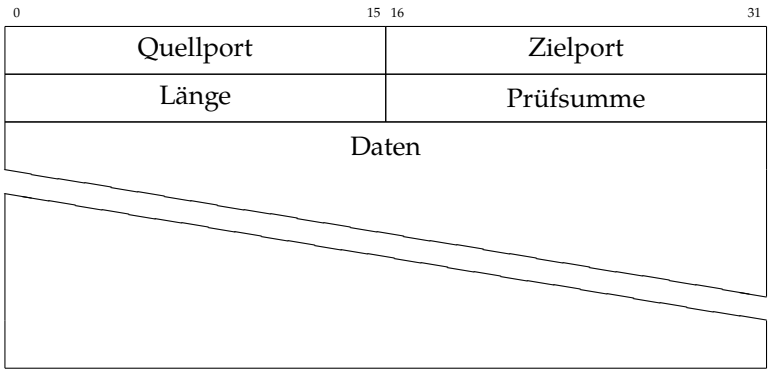


Bild 22.7: Aufbau eines UDP-Datagramms

Daten. (`ssh` liest seine Standardeingabe auf dem lokalen Rechner, leitet sie über das Netz an den entfernten Rechner weiter und übergibt sie dort dem Programm `sort` auf dessen Standardeingabe.) `sort` funktioniert aber so, dass es seine Standardeingabe bis zum Ende liest, dann erst die gelesenen Daten sortiert und sie auf die Standardausgabe schreibt, die dann per `ssh` wieder auf den lokalen Rechner (und den Bildschirm) geleitet wird. – Das Problem besteht nun darin, dass die `ssh` dem entfernt laufenden `sort` signaliseren können muss, dass die Standardeingabe fertig gelesen wurde und es mit dem Sortieren und der Ausgabe anfangen kann. Das geschieht dadurch, dass die Verbindung »zum« entfernten Rechner abgebaut wird. Derjenige Teil der Verbindung, der »vom« entfernten Rechner liest, bleibt jedoch erhalten und kann die Ausgabe von `sort` zurücktransportieren – würde ein Verbindungsabbau automatisch für beide Richtungen gelten, könnte diese Anwendung nicht funktionieren.

Nach einem unilateralen Verbindungsabbau fließen natürlich nach wie vor Daten in beide Richtungen zwischen den Stationen, denn die Station, die die Verbindung abgebaut hat, muss die Daten, die sie über den noch stehenden Teil der Verbindung empfängt, ja auch bestätigen. Nur Nutzdaten kann sie nicht mehr über die Verbindung schicken.

UDP Im Gegensatz zu TCP ist das »User Datagram Protocol« (UDP) [RFC0768] ein verbindungsloses und unzuverlässiges Protokoll. Tatsächlich ist es nicht viel mehr als »IP mit Ports«, denn wie bei TCP können auf einer Station maximal 65535 Kommunikationsendpunkte unterschieden werden (UDP und TCP können dieselbe Portnummer gleichzeitig für unterschiedliche Zwecke verwenden). Bei UDP entfällt der Verbindungsaufbau von TCP genau wie die Bestätigungen, so dass das Protokoll viel »schneller« ist – allerdings um den Preis, dass wie bei IP Daten verloren gehen oder durcheinander geraten können.

UDP verwendet man da, wo entweder nur wenige Daten übertragen werden müssen, so dass die Kosten des Verbindungsaufbaus bei TCP im Vergleich sehr hoch sind – Stichwort DNS –, oder dort, wo es nicht auf jedes Bit ankommt, aber Zeitverzögerungen unerwünscht sind. Bei Internet-Telefonie oder Videoübertragung machen verlorene Datagramme sich höchstens durch Knacklaute oder Schneegriesel im Bild bemerkbar; eine längere Kunstpause, wie sie bei TCP durchaus mal vorkommen kann, wäre im Vergleich viel störender.

Ports **Ports** TCP und UDP unterstützen das Konzept von Ports, über die ein System mehr als eine Netzwerkverbindung gleichzeitig verwalten kann (gut, bei UDP gibt es keine »Verbindungen«, aber trotzdem …). Getrennt für TCP und UDP gibt es

```
# Network services, Internet style

echo       7/tcp
echo       7/udp
discard    9/tcp    sink null
discard    9/udp    sink null
systat     11/tcp   users
daytime    13/tcp
daytime    13/udp
netstat    15/tcp
qotd       17/tcp   quote
chargen    19/tcp   ttytst source
chargen    19/udp   ttytst source
ftp-data   20/tcp
ftp        21/tcp
fsp        21/udp   fspd
ssh        22/tcp                  # SSH Remote Login Protocol
ssh        22/udp                  # SSH Remote Login Protocol
telnet     23/tcp
smtp       25/tcp   mail
◁◁◁◁◁
```

Bild 22.8: Die Datei `/etc/services` (Auszug)

jeweils 65536 Ports, die allerdings nicht alle sinnvoll benutzt werden können: Die Portnummer 0 ist ein Signal an die TCP/IP-Implementierung des Systems, einen ansonsten unbenutzten Port auszusuchen.

Die meisten Ports stehen den Anwendern des Systems frei zur Verfügung, aber diverse Ports sind fest bestimmten Diensten zugeordnet. Man spricht von den ***well-known ports*** und den ***registered ports***. So ist zum Beispiel festgelegt, dass der TCP-Port 25 auf einem System für dessen Mail-Server reserviert ist, der dort auf Verbindungen gemäß dem *Simple Mail Transfer Protocol* (SMTP) wartet. Entsprechend ist der TCP-Port 21 dem *File Transfer Protocol*-Server (FTP) vorbehalten und so weiter. Diese Zuordnungen werden in regelmäßigen Abständen von der IANA veröffentlicht und sind zum Beispiel unter `http://www.iana.org/assignments/port-numbers` zu finden.

well-known ports

registered ports

Die *well-known ports* sind laut IANA die Ports von 0 bis 1023, die *registered ports* diejenigen von 1024 bis 49151. Wenn Sie ein Programm in Umlauf bringen wollen, das einen neuen Dienst anbietet, sollten Sie sich von der IANA eine oder mehrere Portnummern zusprechen lassen.

Die übrigen Ports – von 49152 bis 65535 – heißen im IANA-Jargon *dynamic and/or private ports*. Diese kommen für die Client-Seite von Verbindungen in Frage (es ist unwahrscheinlich, dass Ihr Rechner über 16.000 Verbindungen zu TCP-Servern gleichzeitig halten muss) oder für die Implementierung »privater« Server.

Die IANA neigt dazu, eine für ein TCP-basiertes Protokoll reservierte Portnummer auch für UDP zu reservieren, selbst wenn das betreffende TCP-Protokoll mit UDP keinen Sinn ergibt, und umgekehrt. Port 80 ist zum Beispiel sowohl als TCP- wie auch als UDP-Port für HTTP reserviert, auch wenn UDP-basiertes HTTP kein aktuell interessantes Thema ist. Man läßt sich so Ellbogenfreiheit für künftige Erweiterungen.

Auf einem Linux-System steht eine Zuordnungstabelle in der Datei `/etc/services` (Bild 22.8). Diese Zuordnung wird zum Beispiel vom Internet-Daemon

Tabelle 22.1: Gängige Anwendungsprotokolle auf TCP/IP-Basis

Port	K-Prot	Name	Bedeutung
20	TCP	FTP	Dateiübertragung (Datenverbindungen)
21	TCP	FTP	Dateiübertragung (Steuerverbindungen)
22	TCP	SSH	Sicheres (authentisiertes und verschlüsseltes) Anmelden auf entfernten Rechnern; sichere Dateiübertragung
23	TCP	TELNET	Anmelden auf entfernten Rechnern (unsicher und veraltet)
25	TCP	SMTP	E-Mail-Übertragung
53	UDP/TCP	DNS	Namens- und Adressenauflösung und verwandte Verzeichnisdienste
80	TCP	HTTP	Ressourcenzugriff im World Wide Web
110	TCP	POP3	Zugriff auf entfernte E-Mail-Postfächer
123	UDP/TCP	NTP	Network Time Protocol (Zeitsynchronisation)
137	UDP	NETBIOS	NetBIOS-Namensdienst
138	UDP	NETBIOS	NetBIOS-Datagrammdienst
139	TCP	NETBIOS	NetBIOS-Sitzungsdienst
143	TCP	IMAP	Zugriff auf Mail auf einem entfernten Server
161	UDP	SNMP	Netzwerkverwaltung
162	UDP	SNMP	Traps für SNMP
389	TCP	LDAP	Verzeichnisdienst
443	TCP	HTTPS	HTTP über SSL (authentisiert/verschlüsselt)
465	TCP	SSMTP	SMTP über SSL (veraltet – nicht benutzen!)*
514	UDP	Syslog	Protokolldienst
636	TCP	LDAPS	LDAP über SSL (authentisiert/verschlüsselt)*
993	TCP	IMAPS	IMAP über SSL (authentisiert/verschlüsselt)*
995	TCP	POP3S	POP3 über SSL (authentisiert/verschlüsselt)*

* Diese Dienste können auch über Verbindungen abgewickelt werden, die zunächst unverschlüsselt aufgebaut und dann zu authentisierten und verschlüsselten Verbindungen »aufgewertet« werden.

(`inetd` oder `xinetd`) oder von der C-Bibliotheksfunktion `getservbyname()` verwendet, um zu einem gegebenen Dienstnamen den passenden Port zu finden.

Sie können `/etc/services` selbst ändern, etwa um Ihre selbsterfundenen Dienste zu unterstützen. Achten Sie aber auf Aktualisierungen der Datei durch Ihre Distribution.

reservierte Ports

Die Ports 0 bis 1023 sind auf unixartigen Systemen reserviert, das heißt, nur `root` darf sie öffnen. Dies ist eine Sicherheitsvorkehrung dagegen, dass beliebige Benutzer zum Beispiel einen eigenen Webserver auf einem ansonsten unbenutzten Port 80 starten und damit offiziell wirken.

22.3.4 Die wichtigsten Anwendungsprotokolle

Im vorigen Abschnitt haben wir das Konzept eines »Diensts« angesprochen. Während Kommunikationsprotokolle wie TCP und UDP sich damit befassen, Daten von einer Station zur anderen zu befördern, basieren »Dienste« in der Regel auf Anwendungsprotokollen, die den per Kommunikationsprotokoll ausgetauschten Daten eine Bedeutung zuordnen. Wenn Sie zum Beispiel eine E-Mail per SMTP verschicken, nimmt Ihr Rechner mit dem entfernten SMTP-Server Kontakt auf (über TCP auf Port 25), identifiziert sich, schickt Ihre Adresse sowie die des oder der Adressaten und danach die eigentliche Mail – jeweils nach Aufforderung durch den entfernten Server. Die Details dieser Konversation regelt das Anwendungsprotokoll SMTP.

»Dienste« und »Protokolle« sind nicht ganz dasselbe. Ein »Dienst« ist etwas, wofür Sie den Computer verwenden möchten, etwa E-Mail, Web-Zugriff

oder Drucken auf einem entfernten Druckserver. Für viele Dienste gibt es im Internet »kanonische« Protokolle, die sich einfach anbieten – für E-Mail gibt es zum Beispiel kaum Alternativen zu SMTP –, aber manche Dienste verwenden auch dasselbe unterliegende Protokoll wie andere. Web-Zugriff beispielsweise erfolgt in der Regel über HTTP und der Zugriff auf einen entfernten Druckserver über das »Internet Printing Protocol« (IPP). Wenn Sie genau nachschauen, werden Sie aber feststellen, dass IPP, so wie es heute verwendet wird, auch nichts anderes ist als glorifiziertes HTTP. Der einzige Unterschied ist, dass HTTP den TCP-Port 80 verwendet und IPP den TCP-Port 631.

Tabelle 22.1 zeigt eine Zusammenfassung einiger wichtiger Anwendungsprotokolle. Einige davon werden uns später in dieser Unterlage noch einmal begegnen; mit weiteren befassen sich andere Linup-Front-Schulungsunterlagen.

Schlechte Nachrichten für LPIC-1-Kandidaten: Das LPI möchte, dass Sie die Portnummern aus Tabelle 22.1 und die jeweiligen dazugehörigen Dienste auswendig wissen (LPI-Lernziel 109.1). Viel Spaß beim Büffeln.

22.4 Adressen, Wegleitung und Subnetting

22.4.1 Grundlagen

Jede Netzwerk-Schnittstelle eines Systems in einem TCP/IP-Netz hat mindestens eine IP-Adresse. Als »Schnittstelle« wird in diesem Fall der Teil eines Systems bezeichnet, der in der Lage ist, IP-Datagramme zu senden und zu empfangen. Ein einzelnes System kann mehr als eine solche Schnittstelle haben und hat dann meist auch mehr als eine IP-Adresse. Mit

```
$ /sbin/ifconfig
```

oder

```
$ /sbin/ip addr show
```

können Sie die konfigurierten Schnittstellen bzw. Netzwerkgeräte auflisten.

IP-Adressen

IP-Adressen sind 32 Bit lang und werden normalerweise als *dotted quads* notiert – man betrachtet sie als Folge von vier 8-Bit-Zahlen, die man dezimal als Werte zwischen 0 und 255 hinschreibt, etwa als »`203.177.8.4`«[2]. Jede IP-Adresse wird weltweit eindeutig vergeben und bezeichnet eine Station in einem bestimmten Teilnetz des Internet. Dazu werden IP-Adressen in einen Netzwerk- und einen Stationsanteil aufgeteilt. Der Netzwerk- und Stationsanteil ist variabel und kann der Anzahl der in einem Netz benötigten Stationsadressen angepasst werden. Wenn der Stationsanteil n Bit beträgt, bleiben für den Netzwerkanteil $32 - n$ Bit. Die Verteilung dokumentiert die **Netzmaske**, die für jedes Bit der IP-Adresse, das zum Netzwerkanteil gehört, eine binäre 1 und für jedes Bit des Stationsanteils eine binäre 0 enthält. Die Netzmaske wird entweder als *dotted quad* oder – heutzutage oft – einfach als Anzahl der Einsen notiert. »`203.177.8.4/24`« wäre also eine Adresse in einem Netz mit der Netzmaske »`255.255.255.0`«.

Netzmaske

Nehmen wir als Beispiel ein Netzwerk mit 28 Geräten an. Die nächsthöhere Potenz von 2 ist $32 = 2^5$. Das bedeutet, dass für die Numerierung der verschiedenen Systeme 5 Bits gebraucht werden. Die restlichen 27 Bits (32 – 5) identifizieren das Netzwerk und sind bei jedem System in diesem Netzwerk gleich. Die Netzmaske ist `255.255.255.224`, denn im letzten *quad* sind die obersten 3 Bits – die mit den Werten 128, 64 und 32, zusammen 224 –, gesetzt.

[2] Es ist übrigens durchaus legitim und wird von den meisten Programmen verstanden, eine IP-Adresse als ausmultiplizierte Dezimalzahl anzugeben – in unserem Beispiel statt `203.177.8.4` etwa 3417376772. Dies ist die Basis für »Trick-URLs« der Form `http://www.microsoft.com@3417376722/bla.html`.

Tabelle 22.2: Beispiel für Adressenvergabe

Bedeutung	IP-Adresse binär	IP-Adresse dezimal
Netzmaske	11111111 11111111 11111111 11100000	255.255.255.224
Netzwerkadresse	11001011 10110001 00001000 00000000	203.177.8.0
Stationsadressen	11001011 10110001 00001000 00000001	203.177.8.1
⋮	⋮	⋮
	11001011 10110001 00001000 00011110	203.177.8.30
Broadcast-Adresse	11001011 10110001 00001000 00011111	203.177.8.31

Netzwerkadresse
Broadcast-Adresse

Die erste und die letzte IP-Adresse in einem Netzwerk werden verabredungsgemäß für spezielle Zwecke reserviert: Die erste Adresse (Stationsanteil nur binäre Nullen) ist die **Netzwerkadresse**, die letzte Nummer (Stationsanteil nur binäre Einsen) die **Broadcast-Adresse**. Im obigen Beispiel wäre also `203.177.8.0` die Netzwerkadresse und `203.177.8.31` die Broadcast-Adresse. Für die Stationen stehen dann die Nummern von 1 bis 30 zur Verfügung (Tabelle 22.2).

Die Adresse `255.255.255.255` ist eine Broadcast-Adresse, aber nicht für das gesamte Internet, sondern für das lokale Netzsegment (also zum Beispiel alle Stationen auf demselben Ethernet). Diese Adresse wird benutzt, wenn keine genauere Adresse bekannt ist, etwa wenn eine Station sich über DHCP eine IP-Adresse und eine Netzmaske holen möchte.

22.4.2 Wegleitung

Wegleitung (engl. *routing*) dient dazu, IP-Datagramme, die nicht direkt im lokalen Netz zugestellt werden können, an die richtige Adresse zu schicken[3]. Tatsächlich können Sie argumentieren, dass Wegleitung die zentrale Eigenschaft ist, die TCP/IP von »Spielzeugprotokollen« wie NetBEUI und Appletalk unterscheidet und damit das Internet, so wie wir es kennen, erst möglich gemacht hat.

Wegleitung greift da, wo der Empfänger eines IP-Datagramms nicht im selben Netz zu finden ist wie der Absender. Feststellen kann die sendende Station das (natürlich) anhand der IP-Adresse der gewünschten Zielstation, indem sie den Teil der Zieladresse betrachtet, der von ihrer eigenen Netzmaske »abgedeckt« wird, und überprüft, ob er mit ihrer eigenen Netzwerkadresse übereinstimmt. Wenn das der Fall ist, ist der Empfänger »lokal« und kann direkt erreicht werden (Abschnitt 22.3.2 auf Seite 373).

Routing-Tabelle

Kann der Empfänger nicht direkt erreicht werden, konsultiert die Station (jedenfalls, wenn sie ein Linux-Rechner ist) eine Routing-Tabelle, die zumindest ein *default gateway* ausweisen sollte, also eine Station, die sich um die Weiterleitung nicht direkt zustellbarer Datagramme kümmert. (Diese Station muss in aller Regel selber direkt erreichbar sein.) Eine solche Station heißt »Router« und ist entweder selbst ein Computer oder ein spezielles für diese Funktion ausgelegtes Gerät.

Der Router geht grundsätzlich ganz analog vor wie eben beschrieben: Er verfügt über verschiedene Netzwerkschnittstellen, die jeweils eine Adresse und eine Netzmaske zugeordnet bekommen haben, und kann Datagramme unmittelbar an Stationen zustellen, die sich über die Netzmasken seiner Netzwerkschnittstellen als in einem »seiner« Netze befindlich identifizieren lassen. Für alles Weitere werden wiederum direkt erreichbare Stationen herangezogen, die als Router fungieren und so fort.

[3] Vorausgesehen wurde das bereits im Alten Testament, und zwar aus der Sicht des Datagramms: »Er führet mich auf rechter Straße um seines Namens willen.« (Ps 23:3) Denn im Internet gibt es für Sie kaum eine bessere Methode, das Ansehen Ihres Namens zu ramponieren, als eine kapitale Routing-Fehlkonfiguration …

Tabelle 22.3: Traditionelle IP-Netzklassen

Klasse	Netzanteil	Anzahl der Netze	Stationen pro Netz	Adressen
Class A	8 Bit	128 – 126 nutzbar	16.777.214 ($2^{24} - 2$)	`0.0.0.0` – `127.255.255.255`
Class B	16 Bit	16.384 (2^{14})	65.534 ($2^{16} - 2$)	`128.0.0.0` – `191.255.255.255`
Class C	24 Bit	2.097.152 (2^{21})	254 ($2^8 - 2$)	`192.0.0.0` – `223.255.255.255`
Class D	-	-	-	`224.0.0.0` – `239.255.255.255`
Class E	-	-	-	`240.0.0.0` – `254.255.255.255`

Im wirklichen Leben können Routingtabellen um einiges komplexer sein. Es ist zum Beispiel möglich, Datagramme, die an bestimmte Stationen oder Netze gerichtet sind, gezielt über Router zu leiten, die nicht das *default gateway* sind.

Eine wichtige Beobachtung ist, dass eine Station (PC oder Router) normalerweise nur über den direkt nächsten Schritt der Wegleitung (man sagt auch »Hop«) entscheidet, anstatt den kompletten Weg vom ursprünglichen Absender des Datagramms bis zum Empfänger vorzugeben. Das heißt, es liegt in der Hand jedes Routers zwischen Absender und Empfänger, denjenigen nächsten Hop auszuwählen, den er für am sinnvollsten hält. Gut konfigurierte Router gleichen sich mit ihren »Nachbarn« ab und können Informationen über die Netzauslastung und möglicherweise bekannte Blockagen anderswo im Netz in ihre Wegleitungsentscheidungen einfließen lassen. Für unsere Zwecke führt eine detaillierte Diskussion dieser Thematik allerdings zu weit.

Tatsächlich ist es auch möglich, dass ein Datagramm den kompletten Weg vorgibt, den es zu seinem Ziel durchlaufen möchte. Das heißt dann *source routing*, ist eigentlich verpönt und wird heutzutage von großen Teilen der Netzinfrastruktur völlig missachtet, da es einerseits mit der Idee der dynamischen Lastverteilung kollidiert und andererseits Vehikel für Sicherheitsprobleme sein kann.

22.4.3 IP-Netzklassen

Die Menge der IP-Adressen von `0.0.0.0` bis `255.255.255.255` wurde traditionell in mehrere **Netzklassen** eingeteilt, die als »Class A«, »Class B« und »Class C« bezeichnet werden. Netzklassen

Es gibt auch noch »Class D« (Multicast-Adressen) und »Class E« (für experimentelle Zwecke), diese sind jedoch für die Vergabe von IP-Adressen für Endgeräte von geringem Interesse.

Die Klassen A bis C unterscheiden sich durch ihre Netzmaske, unter dem Strich also in der Anzahl der pro Klasse möglichen Netze und der Anzahl der in einem solchen Netz möglichen Stationen. Während eine Class-A-Adresse einen Netzanteil von 8 Bit hat, hat eine Class-B-Adresse einen von 16 Bit und eine Class-C-Adresse einen von 24 Bit. Jeder der Netzklassen war ein fester Bereich der möglichen IP-Adressen zugeordnet (Tabelle 22.3)

Aufgrund der zunehmenden Verknappung von IP-Adressen wurde in den 1990er Jahren die Aufteilung der IP-Adressen in die drei Adressklassen aufgegeben. Inzwischen verwendet man klassenlose Wegleitung (*classless inter-domain routing*, CIDR) nach [RFC1519]. Während die Grenze zwischen Netzwerk- und Stationsadresse bei der »alten« Aufteilung nur an drei verschiedenen Positionen liegen konnte, ist es gemäß CIDR möglich, beliebige Netzmasken zu vergeben und so einerseits die Größe der einem Anwender (meist einem Provider) zur Verfügung gestellten Adressbereichs feiner zu steuern und andererseits die »Explosion« der Wegleitungstabellen zu vermeiden. Eine Installation mit sechzehn klassenlose Wegleitung

Tabelle 22.4: Beispiel für Subnetting

Bedeutung	IP-Adresse binär	dezimal
Netzmaske	11111111 11111111 11111111 11110000	255.255.255.240
Netzwerkadresse (1)	11001011 10110001 00001000 00000000	203.177.8.0
Stationsadressen (1)	11001011 10110001 00001000 00000001	203.177.8.1
⋮	⋮	⋮
	11001011 10110001 00001000 00001110	203.177.8.14
Broadcast-Adresse (1)	11001011 10110001 00001000 00001111	203.177.8.15
Netzwerkadresse (2)	11001011 10110001 00001000 00010000	203.177.8.16
Stationsadressen (2)	11001011 10110001 00001000 00010001	203.177.8.17
⋮	⋮	⋮
	11001011 10110001 00001000 00011110	203.177.8.30
Broadcast-Adresse (2)	11001011 10110001 00001000 00011111	203.177.8.31

aneinandergrenzenden »Class-C«-Netzen (Netzmaske »/24«) kann so aus der Sicht der Wegleitung als ein Netz mit der Netzmaske »/20« angesehen werden – eine grosse Vereinfachung, da die Tabellen in Routern wesentlich kompakter ausfallen können. Im Internet werden heute normalerweise keine Adressen direkt geroutet, deren Netzwerkanteil länger als 19 Bit ist; Sie müssen sich in der Regel eines Providers bedienen, der den ganzen Adressenblock verwaltet und die IP-Datagramme dann intern weiterleitet.

22.4.4 Subnetting

Oftmals ist die Einteilung in ein großes Netzwerk zu ungenau oder nicht sinnvoll. Deswegen teilen Betreiber ihre Netzwerke oft in mehrere kleinere Netzwerke auf. Das geschieht, indem dem festen Netzwerkanteil der IP-Adresse noch ein weiterer fester Anteil mitgegeben wird, der das unterteilte Netz, das »Subnet«, spezifiziert.

Subnetting

Subnetting könnte im Beispiel von oben möglicherweise so aussehen: Sie möchten statt eines »großen« Netzes mit 32 Adressen (für 30 Stationen) beispielsweise zwei »kleine« Netze mit je 16 Adressen (für jeweils bis zu 14 Stationen) betreiben, etwa um aus Sicherheitsgründen getrennte Ethernet-Stränge einsetzen zu können. Hierzu verlängern Sie die Netzmaske um 1 Bit; die Netzwerk-, Stations- und Broadcast-Adressen ergeben sich sinngemäß wie oben (Tabelle 22.4).

Ungleich große Subnetze

Es ist nicht notwendig, dass alle Subnetze gleich groß sind. Das Netz 203.177.8.0/24 ließe sich zum Beispiel ohne weiteres in ein Subnetz mit 126 Stationsadressen (etwa 203.177.8.0/25 mit den Stationsadressen 203.177.8.1 bis 203.177.8.126 und der Broadcast-Adresse 203.177.8.127) und zwei Subnetze mit je 62 Stationsadressen (etwa 203.177.8.128/26 und 203.177.8.192/26 mit den respektiven Stationsadressen 203.177.8.129 bis 203.177.8.190 und 203.177.8.193 bis 203.177.8.254 sowie den Broadcast-Adressen 203.177.8.191 und 203.177.8.255) aufteilen.

Kleinstmögliches Netz

Das kleinstmögliche IP-Netz hat einen 30 Bit langen Netzwerkanteil und einen 2 Bit langen Stationsanteil. Dies sind insgesamt vier Adressen, von denen eine Netzwerk- und eine Broadcast-Adresse abgehen, es bleiben also noch zwei Adressen für Stationen. Sie finden diese Konstellation mitunter bei Punkt-zu-Punkt-Verbindungen etwa über Modem oder ISDN.

22.4.5 Private IP-Adressen

Weltweite Vergabe von IP-Adressen

IP-Adressen sind weltweit eindeutig und müssen deswegen zentral vergeben werden. Sie können Ihre IP-Adresse also nicht beliebig wählen, sondern müssen sie beantragen – typischerweise bei Ihrem Provider, der seinerseits einen Block von

Tabelle 22.5: Private IP-Adressbereiche nach RFC 1918

Adressraum	von	bis
Class A	10.0.0.0	10.255.255.255
Class B	172.16.0.0	172.31.255.255
Class C	192.168.0.0	192.168.255.255

IP-Adressen von einer nationalen oder internationalen Organisation zugeordnet bekommen hat (Abschnitt 22.1.2). Die Anzahl der international möglichen Netzwerkadressen ist natürlich limitiert.

Anfang Februar 2011 teilte die IANA den fünf regionalen Registries die letzten fünf verfügbaren /8-Adressbereiche zu. Wahrscheinlich werden dem APNIC (*Asia Pacific Network Information Centre*) zuerst die IP-Adressen ausgehen, vermutlich Mitte 2011. Danach hilft dann nur noch Betteln oder IPv6.

Für Netzwerke, die nicht direkt an das Internet angeschlossen sind, sind besondere Adressbereiche, die privaten IP-Adressen gemäß [RFC1918], vorgesehen, die im Internet nicht geroutet werden (Tabelle 22.5). Diese Adressen können Sie ungeniert in Ihren lokalen Netzen verwenden – inklusive Subnetting und allen Finessen.

Private IP-Adressen

22.4.6 Masquerading und Portweiterleitung

IP-Adressen sind heute eine knappe Ressource, und das wird auch so bleiben, bis wir alle auf IPv6 (Abschnitt 22.5) umgestiegen sind. Es ist also höchst wahrscheinlich, dass Sie nur eine einzige »offizielle« (also Nicht-RFC-1918-)IP-Adresse zur Verfügung haben, um Ihr ganzes Netz ans Internet anzubinden – bei Heimnetzen oder solchen in kleinen Firmen ist das sogar die Regel. Die Abhilfe (euphemistisch für »lahme Krücke«) dagegen ist einerseits »Masquerading«, andererseits »Portweiterleitung«. Beiden Verfahren liegt zugrunde, dass als einziges Ihr Router mit einer öffentlichen Adresse am Internet teilnimmt. Alle anderen Stationen in Ihrem Netz haben Adressen nach [RFC1918]. Masquerading bedeutet, dass Ihr Router bei Datagrammen, die Stationen in Ihrem Netz nach »draußen« schicken, die Absender-IP-Adresse der betreffenden Station durch seine eigene ersetzt und die dazugehörigen Antwortdatagramme an den eigentlichen Absender weiterleitet. Sowohl die Stationen in Ihrem Netz als auch »das Internet« bekommen davon nichts mit – die einen denken, sie reden direkt mit dem Internet, während das andere nur die (offizielle) IP-Adresse Ihres Routers zu sehen bekommt. Umgekehrt ermöglicht Portweiterleitung, dass Rechner aus dem Internet Dienste wie zum Beispiel DNS, Mail oder HTTP über die jeweiligen Ports auf dem *Router* ansprechen, dieser die entsprechenden Datagramme aber an einen Rechner im internen Netz weiterleitet, der den tatsächlichen Dienst erbringt.

Masquerading

Portweiterleitung

Sie sollten der Versuchung widerstehen, Ihren Router gleichzeitig zum Web-, Mail- oder DNS-Server zu machen; die Gefahr, dass ein Angreifer über eines der großen dafür nötigen Serverprogramme Ihren Router kompromittiert und damit im schlimmsten Fall freien Zugriff auf Ihr ganzes lokales Netz bekommt, ist viel zu groß.

Portweiterleitung und Masquerading sind zwei Ausprägungen eines Konzepts, das allgemein als NAT (engl. *network address translation*, Netzadress-Umsetzung) bezeichnet wird. Im Speziellen sprechen wir bei Masquerading auch von *Source NAT*, da die Absenderadresse von ausgehenden Datagrammen modifiziert wird[4], während Portweiterleitung einen Fall von *Destination NAT* darstellt – hier wird die Zieladresse von an uns gerichteten Datagrammen verändert.

NAT

[4] Dass wir auch die Zieladresse der als Antwort eingehenden Datagramme umschreiben müssen, ignorieren wir hier der Bequemlichkeit halber.

Übungen

22.5 [!1] Können die folgenden IP-Adressen mit der aufgeführten Netzmaske als Stationsadressen in dem betreffenden IP-Netz verwendet werden? Wenn nein, warum nicht?

	IP-Adresse	Netzmaske
a)	`172.55.10.3`	`255.255.255.252`
b)	`138.44.33.12`	`255.255.240.0`
c)	`10.84.13.160`	`255.255.255.224`

22.6 [2] Welche Gründe könnten Sie dafür haben, einen Adressbereich, den Sie von einem Provider bekommen, in Subnetze aufzuteilen?

22.7 [!2] Das Netz mit der IP-Adresse `145.2.0.0` und der Netzmaske `255.255.0.0` wurde mit der Subnetzmaske `255.255.240.0` in folgende Subnetze aufgeteilt:

- `145.2.128.0`
- `145.2.64.0`
- `145.2.192.0`
- `145.2.32.0`
- `145.2.160.0`

Welche weiteren Subnetze sind noch möglich? In welchem Subnetz befindet sich die Station `145.2.195.13`?

22.5 IPv6

22.5.1 Überblick

IPv4 Die heute am weitesten verbreitete Fassung von IP ist die Version 4, kurz »IPv4« genannt. Durch das rasche Wachstum des Internets stößt diese Version inzwischen an ihre Grenzen – Hauptprobleme sind die zunehmende Adressenverknappung, der Wildwuchs bei der Adressvergabe und das daraus resultierende hochkomplexe Routing sowie die kaum vorhandene Unterstützung von Sicherheitsmechanismen und Methoden zur Dienstgütesteuerung. IPv6 soll hier Abhilfe schaffen.

IPv6

Eigenschaften Die wichtigsten Eigenschaften von IPv6 sind:

- Die Adresslänge wird von 32 auf 128 Bit vergrößert, wodurch man auf $3{,}4 \cdot 10^{38}$ Adressen kommt. Dies würde ausreichen, um jeder der heute lebenden 6,5 Milliarden Personen rund 50 Quadrillionen (eine Zahl mit 27 Nullen) IPv6-Adressen zuzuordnen. Das sollte für die vorhersehbare Zukunft genügen.
- IPv6-Rechner können sich automatisch Konfigurationsparameter von einem Router holen, wenn sie an ein Netz angeschlossen werden. Falls nötig, gibt es immer noch ein DHCPv6-Protokoll.
- Ein IP-Kopf enthält nur noch 7 Felder, damit Router die Pakete schneller verarbeiten können. Bei Bedarf können mehrere Köpfe für ein Datagramm verwendet werden.
- Erweiterte Unterstützung von Optionen und Erweiterungen, was auch dazu beiträgt, dass Router Datagramme schneller verarbeiten können.
- Bessere Übertragung von Audio- und Videodaten sowie bessere Unterstützung von Echtzeitanwendungen.

- Erhöhte Sicherheit durch abgesicherte Datenübertragung sowie Methoden zur Authentisierung und Integritätssicherung.
- Erweiterbarkeit, um die Zukunftsfähigkeit des Protokolls sicherzustellen. Es wurde nicht versucht, alle Möglichkeiten abzudecken, denn die Zukunft bringt Neuerungen, die heute nicht absehbar sind. Stattdessen ist das Protokoll offen für die rückwärtskompatible Integration weiterer Funktionen.

Während die Standardisierung von IPv6 schon seit einer ganzen Weile abgeschlossen ist, hapert es an der allgemeinen Umsetzung durchaus noch. Es sind vor allem die Dienstanbieter, die sich zieren. Linux unterstützt IPv6 schon jetzt, so dass die Umstellung einer Linux-Infrastruktur auf den neuen Standard kein Problem darstellen wird. Sie können auch IPv6-Pakete testhalber über IPv4 transportieren, indem Sie sie in IPv4-Pakete einpacken (»Tunneling«). Damit könnte eine Firma zum Beispiel ihre internen Netze auf IPv6 aufbauen lassen und sogar mehrere Standorte über ein »virtuelles« IPv6-Netz im traditionellen IPv4-Netz verbinden. Umsetzung

Wir sollten außerdem hervorheben, dass IPv6 nur einen gezielten Ersatz für das bisherige IPv4 darstellt. Die allermeisten Protokolle, die auf IP aufsetzen – beginnend mit TCP und UDP – bleiben unverändert. Lediglich auf der »Infrastrukturebene« werden einige Protokolle überflüssig oder durch IPv6-basierte Versionen ersetzt.

22.5.2 IPv6-Adressierung

IPv6 erlaubt 2^{128} verschiedene Adressen – eine unvorstellbar große Zahl. Im Grunde könnte jedes Sandkorn auf der Erde über etliche Adressen verfügen, aber das ist gar nicht Ziel der Sache: Der große Adressraum gestattet eine deutlich flexiblere Adressvergabe für die unterschiedlichsten Zwecke sowie viel einfachereres Routing.

IPv6-Adressen werden nicht wie IPv4-Adressen dezimal notiert, sondern hexadezimal (also zur Basis 16). Dabei werden immer vier Ziffern zusammengefasst und diese Gruppen durch Doppelpunkte getrennt. Eine IPv6-Adresse könnte also aussehen wie Schreibweise

```
fe80:0000:0000:0000:025a:b6ff:fe9c:406a
```

Führende Nullen in einer Gruppe dürfen wegfallen, und (maximal) eine Folge von Null-Blocks darf durch zwei Doppelpunkte ersetzt werden. Eine verkürzte Schreibweise für die Adresse aus dem vorigen Beispiel wäre also

```
fe80::25a:b6ff:fe9c:406a
```

Der IPv4-Loopback-Adresse `127.0.0.1` entspricht die IPv6-Adresse `::1` – eine Abkürzung von

```
0000:0000:0000:0000:0000:0000:0000:0001
```

»Broadcast-Adressen« à la `192.168.1.255` gibt es bei IPv6 nicht (hierzu später mehr).

IPv6-Adressen können in einen 64 Bit breiten Netz- und einen 64 Bit breiten Stationsanteil aufgeteilt werden. Das heißt, jedes IPv6-Subnetz verfügt über 2^{64} Adressen, also 2^{32}-mal so viele wie das komplette IPv4-Internet! Subnetting mit variablen Präfixlängen, so wie es in IPv4 gemacht wird (Abschnitt 22.4.4) ist bei IPv6 nicht wirklich vorgesehen. Es wird aber davon ausgegangen, dass Ihr Provider Ihnen zum Beispiel ein »`/56`«-Adresspräfix zur Verfügung stellt, so dass Sie über 256 Subnetze mit je 2^{64} Adressen verfügen können, was Ihren Stil nicht allzusehr einschränken sollte. (Netzwerkpräfixe können Sie angeben, indem Sie an die Adresse einen Schrägstrich und die Präfixlänge als Dezimalzahl anhängen – eine Adresse wie `fe80::/16` beschreibt also das Netz, wo Adressen mit `fe80` anfangen und dann beliebig weitergehen.) Subnetting

Grundsätzlich gibt es drei Arten von IPv6-Adressen: Arten von IPv6-Adressen

- »Unicast«-Adressen beziehen sich auf eine bestimmte Netzwerkschnittstelle (ein Rechner kann mehrere Netzwerkschnittstellen haben, die jeweils mit eigenen Adressen ausgestattet sind).
- »Anycast«-Adressen beziehen sich auf eine Gruppe von Netzwerkschnittstellen. Typischerweise gehören diese zu verschiedenen Stationen und die jeweils »nächste« antwortet tatsächlich. Sie können zum Beispiel alle Router in einem IPv6-Netz adressieren, indem Sie die Adresse verwenden, die sich ergibt, wenn Sie an das (64-Bit-)Adresspräfix des Netzes einen Stationsteil anhängen, der nur aus Nullen besteht.
- »Multicast«-Adressen werden benutzt, um dieselben Pakete an mehrere Schnittstellen zuzustellen. Wie erwähnt, verwendet IPv6 keinen Broadcast; Broadcast ist ein Sonderfall von Multicast. Die Adresse `ff02::1` spricht zum Beispiel alle Rechner im lokalen Netz an.

Sichtbarkeitsbereiche

Ferner kann man verschiedene Sichtbarkeitsbereiche (*scopes*) unterscheiden:

- Globale (*global*) Sichtbarkeit gilt für Adressen, die im gesamten (IPv6-)Internet geroutet werden.
- Verbindungslokale (*link-local*) Sichtbarkeit gilt für Adressen, die nicht geroutet werden und nur im selben Netz verwendbar sind. Solche Adressen werden meist für interne IPv6-Verwaltungszwecke benutzt. Verbindungslokale Adressen sind immer im Netz `fe80::/64`; die übrigen 64 Bit ergeben sich im einfachsten Fall aus der MAC-Adresse der Schnittstelle.
- Standortlokale (*site-local*) Sichtbarkeit gilt für Adressen, die nur innerhalb eines »Standorts« geroutet werden. Niemand weiß genau, was das heißen soll, und standortlokale Adressen sind inzwischen (wieder) verpönt. Standortlokale Adressen haben das Präfix `fec0::/10`.
- Eindeutig lokale (*unique-local*) Adressen ähneln standortlokalen Adressen und erfüllen mehr oder weniger den Zweck, den die RFC-1918-Adressen (`192.168.`*x.y* & Co.) bei IPv4 hatten – wobei IPv6 es einfach macht, »richtige«, also global sichtbare Adressen zu vergeben, so dass Sie nicht auf eindeutig lokale Adressen zurückgreifen müssen, nur damit Ihre Stationen überhaupt Adressen bekommen können. Es gibt also keinen wichtigen Grund, eindeutig lokale Adressen zu benutzen, außer vielleicht als Rückzugsposition, falls irgendetwas mit Ihrem »echten« Präfix nicht stimmt. Eindeutig lokale Adressen haben das Präfix `fd00::/8`; die nächsten 40 Bit für ein `/48`-Netz dürfen Sie sich selbst aussuchen (aber nehmen Sie nicht `fd00::/48`).

mehrere Adressen

Eine sehr wichtige Beobachtung ist, dass in IPv6 jede Netzwerkschnittstelle mehrere Adressen haben kann. Sie bekommt automatisch eine verbindungslokale Adresse, aber kann ohne Weiteres auch noch mehrere eindeutig lokale oder global sichtbare Adressen haben. Alle diese Adressen sind gleichwertig.

`ipv6calc`

Ein nützliches Kommando für den geplagten IPv6-Administrator ist `ipv6calc`, das den Umgang mit IPv6-Adressen erleichtert. Zum Beispiel kann es Informationen über eine Adresse ausgeben:

```
$ ipv6calc --showinfo fe80::224:feff:fee4:1aa1
No input type specified, try autodetection... found type: ipv6addr
No output type specified, try autodetection... found type: ipv6addr
Address type: unicast, link-local
Error getting registry string for IPv6 address:▷
 ◁ reserved(RFC4291#2.5.6)
Interface identifier: 0224:feff:fee4:1aa1
EUI-48/MAC address: 00:24:fe:e4:1a:a1
MAC is a global unique one
MAC is an unicast one
OUI is: AVM GmbH
```

Bei der Adresse in unserem Beispiel handelt es sich also um eine verbindungslokale Unicast-Adresse, deren Stationsteil auf ein Gerät hindeutet, das von der AVM GmbH hergestellt wurde (tatsächlich eine FRITZ!Box).

`ipv6calc` erlaubt auch die Umrechnung von Adressen von einem Format in ein anderes. Sie können damit zum Beispiel die Methode nachstellen, mit der aus einer MAC-Adresse der Stationsteil einer IPv6-Adresse (auch „EUI-64“ genannt) erzeugt wird:

```
$ ipv6calc --in mac --out eui64 00:24:fe:e4:1a:a1
No action type specified, try autodetection... found type: geneui64
0224:feff:fee4:1aa1
```

Kommandos in diesem Kapitel

arp Erlaubt Zugriff auf den ARP-Cache (Abbildung von IP- auf MAC-Adressen) `arp`(8) 373

inetd Internet-Superserver, überwacht Ports und startet ggf. Dienste `inetd`(8) 377

ipv6calc Hilfsprogramm für Berechnungen mit IPv6-Adressen `ipv6calc`(8) 386

xinetd Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste `xinetd`(8) 377

Zusammenfassung

- Das Internet wurzelt in den ARPAnet-Anfängen der 1960er Jahre, wurde in den frühen 1980er Jahren auf seine heutige technische Basis gestellt und nahm in den 1980er und 1990er Jahren einen ungeahnten Aufschwung.
- Das ISO/OSI-Referenzmodell dient zur Begriffsbildung über die Struktur von Rechnerkommunikation.
- TCP/IP ist die heute populärste Protokollfamilie für den Datentransfer in Rechnernetzen.
- ICMP dient zur Netzverwaltung und Problemweitermeldung.
- TCP bietet auf der Basis von IP einen verbindungsorientierten und zuverlässigen Transportdienst.
- UDP ist wie IP verbindungslos und unzuverlässig, aber viel einfacher und schneller als TCP.
- TCP und UDP verwenden Portnummern, um zwischen verschiedenen Verbindungen auf demselben Rechner zu unterscheiden.
- Verschiedene TCP/IP-Dienste haben feste Portnummern zugeordnet. Diese Zuordnung ist der Datei `/etc/services` zu entnehmen.
- IP-Adressen identifizieren Stationen weltweit. Sie sind 32 Bit lang und bestehen aus einem Netzwerk- und einem Stationsteil, zwischen denen über die Netzmaske unterschieden wird.
- Früher wurden die verfügbaren IP-Adressen in Klassen eingeteilt. Heute verwendet man klassenlose Wegleitung mit Netzmasken variabler Länge.
- IP-Netze können weiter in Subnetze unterteilt werden, indem man die Netzmaske anpasst.
- Für die Verwendung in lokalen Netzen sind einige IP-Adressbereiche reserviert, die von Providern nicht geroutet werden.
- IPv6 hebt diverse Einschränkungen des heute üblichen IPv4 auf, befindet sich aber noch nicht in weitem Gebrauch.

Literaturverzeichnis

IPv6-HOWTO Peter Bieringer. »Linux IPv6 HOWTO«, Oktober 2005. http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/

RFC0768 J. Postel. »User Datagram Protocol«, August 1980. http://www.ietf.org/rfc/rfc0768.txt

RFC0791 Information Sciences Institute. »Internet Protocol«, September 1981. http://www.ietf.org/rfc/rfc0791.txt

RFC0792 J. Postel. »Internet Control Message Protocol«, September 1981. http://www.ietf.org/rfc/rfc0792.txt

RFC0793 Information Sciences Institute. »Transmission Control Protocol«, September 1981. http://www.ietf.org/rfc/rfc0793.txt

RFC0826 David C. Plummer. »An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Addresses for Transmission on Ethernet Hardware«, November 1982. http://www.ietf.org/rfc/rfc0826.txt

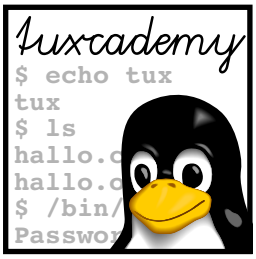
RFC1519 V. Fuller, T. Li, J. Yu, et al. »Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy«, September 1993. http://www.ietf.org/rfc/rfc1519.txt

RFC1918 Y. Rekhter, B. Moskowitz, D. Karrenberg, et al. »Address Allocation for Private Internets«, Februar 1996. http://www.ietf.org/rfc/rfc1918.txt

RFC4291 R. Hinden, S. Deering. »IP Version 6 Addressing Architecture«, Februar 2006. http://www.ietf.org/rfc/rfc4291.txt

Ste94 W. Richard Stevens. *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley Professional Computing Series. Boston etc.: Addison-Wesley, 1994.

Tan02 Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, 2002, 3. Auflage. Unbedingt auf englisch lesen, die deutsche Übersetzung ist eine Katastrophe.



23

Linux-Netzkonfiguration

Inhalt

23.1 Netzschnittstellen 390
23.1.1 Hardware und Treiber 390
23.1.2 Netzwerkkarten konfigurieren mit `ifconfig` 391
23.1.3 Wegleitung konfigurieren mit `route` 392
23.1.4 Netzkonfiguration mit `ip` 395
23.2 Dauerhafte Netzkonfiguration 396
23.3 DHCP 398
23.4 IPv6-Konfiguration 400
23.5 Namensauflösung und DNS 401

Lernziele

- Die Netzkonfigurationsmechanismen der wichtigsten Distributionen kennen
- Netzschnittstellen konfigurieren können
- Statische Routen einrichten können
- Linux als DHCP- und DNS-Client konfigurieren können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 22)

23.1 Netzschnittstellen

23.1.1 Hardware und Treiber

Je nach verwendeter Technik und Zugangsverfahren sprechen Linux-Rechner das Netz über Modems, ISDN-Karten, Ethernet- oder WLAN-Adapter und ähnliches an. Die folgenden Abschnitte beschäftigen esich hauptsächlich mit der Einrichtung von Ethernetkarten.

Eine Netzwerkkarte wird unter Linux wie andere Hardware auch vom Kernel angesteuert – heute normalerweise über modulare Treiber, die bei Bedarf dynamisch geladen werden. Anders als zum Beispiel Festplattenpartitionen oder Drucker erscheinen Netzwerkkarten aber nicht als Gerätedateien in `/dev`, sondern werden über Schnittstellen (engl. *interfaces*) angesprochen. Diese Schnittstellen sind »virtuell« in dem Sinne, dass der Kernel sie nach dem Einbinden des passenden Treibers direkt bereitstellt und eine Netzwerkkarte durchaus über mehr als eine (fast) unabhängige Schnittstelle angesprochen werden kann. Die Schnittstellen haben Namen; ein typischer Name für eine Ethernet-Karte wäre zum Beispiel `eth0`.

Schnittstellen

Netzwerkkarten werden heutzutage beim Start des Systems vom Kernel erkannt, der anhand der PCI-ID das richtige Treibermodul identifizieren kann. Es obliegt der `udev`-Infrastruktur, der Netzwerkkarte einen Namen zu geben und den Treiber tatsächlich zu laden.

Ein Fallstrick, den moderne Linux-Distributionen hier aufbauen, ist, dass der Schnittstellenname einer Netzwerkkarte an deren MAC-Adresse gekoppelt wird. (Jede Netzwerkkarte hat eine weltweit eindeutige MAC-Adresse, die vom Hersteller gesetzt wird.) Wenn Sie in einem Rechner also die Netzwerkkarte austauschen, ohne die Informationen zurückzusetzen, die `udev` sich über die Netzwerkkarten merkt, die es schon mal gesehen hat, dann stehen die Chancen gut, dass Ihre neue Karte den Namen `eth1` bekommt und die Konfiguration, die von der Existenz von `eth0` ausgeht, nicht beachtet wird.

Ein typischer Platz, wo solche Informationen landen können, ist das Verzeichnis `/etc/udev/rules.d`. In einer Datei wie `70-persistent-net.rules` können sich Zeilen finden wie

```
SUBSYSTEM=="net", DRIVERS=="?*",▷
 ◁ ATTRS{address}=="00:13:77:01:e5:4a", NAME="eth0"
```

die der Karte mit der MAC-Adresse `00:13:77:01:e5:4a` den Namen `eth0` zuordnet. Sie können die MAC-Adresse mit der Hand korrigieren oder die Zeile komplett löschen und darauf warten, dass `udev` beim nächsten Systemstart die Einträge an die veränderte Realität anpasst.

Machen Sie sich keinen Kopf, wenn Sie Linux in einer virtuellen Maschine laufen lassen und die Datei `70-persistent-net.rules` nicht finden können. Für die meisten »virtuellen« Netzwerkschnittstellen wird sie nämlich gar nicht angelegt.

Früher (in der Zeit vor `udev`) oblag es den Installationsprozeduren der Distributionen, die richtigen Treiber für Netzwerkkarten zu finden und dem System bekannt zu machen. Typischerweise ging das über `/etc/modules.conf`, wo Einträge wie

```
alias eth0 3c59x
```

untergebracht wurden – dies war ein Signal an den Kernel, beim ersten Zugriff auf die Schnittstelle `eth0` das Treibermodul `3c59x.o` zu laden. Aber das ist vorbei …


Der Linux-Kernel ist natürlich nicht notwendigerweise modular ausgelegt, auch wenn die Standardkernels der meisten Distributionen ohne Module nicht auskommen könnten. Wenn Sie Ihren eigenen Kernel übersetzen (siehe hierzu etwa *Linux-Systemanpassungen*), dann können Sie die Treiber für Ihre Netzwerkkarten auch fest einbauen.


Für besondere Anforderungen, typischerweise bei Rechnern mit erhöhtem Sicherheitsbedürfnis wie paketfilternden Routern oder Servern, die dem Internet ausgesetzt sind, können Sie die Infrastruktur für ladbare Module auch komplett aus dem Kernel verbannen. Das macht es Crackern schwerer (aber leider nicht unmöglich), sich unbemerkt auf dem System einzunisten.

23.1.2 Netzwerkkarten konfigurieren mit `ifconfig`

Bevor Sie eine Schnittstelle zum Zugriff auf das Netz verwenden können, müssen Sie ihr eine IP-Adresse, eine Netzmaske und so weiter zuweisen. Manuell geht das traditionellerweise mit dem Kommando `ifconfig`:

```
# ifconfig eth0 192.168.0.75 up
# ifconfig eth0
eth0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:73
     inet addr:192.168.0.75 Bcast:192.168.0.255 Mask:255.255.255.0
     inet6 addr: fe80::2a0:24ff:fe56:e373/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:6 errors:0 dropped:0 overruns:0 carrier:6
     collisions:0 txqueuelen:100
     RX bytes:0 (0.0 b)  TX bytes:460 (460.0 b)
     Interrupt:5 Base address:0xd800
```

Nach der Zuweisung einer IP-Adresse können Sie durch Aufruf des gleichen Kommandos ohne Angabe einer IP-Adresse den Status der Schnittstelle auslesen. Hier werden nicht nur die aktuelle IP-Adresse, sondern auch der Hardwaretyp, die MAC- (oder »Hardware-«)Adresse, die Broadcast-Adresse, die Netzmaske, die IPv6-Adresse und viele weitere Daten angezeigt. Am Beispiel lässt sich erkennen, dass Werte wie Netzmaske und Broadcast-Adresse auch ohne explizite Zuweisung vom Kernel auf Standardwerte (hier gemäß dem ersten Oktett der vergebenen IP-Adresse die für ein Class-C-Netz) gesetzt werden. Sollten die gewünschten Werte vom Standard abweichen, müssen Sie sie explizit angeben:

```
# ifconfig eth0 192.168.0.75 netmask 255.255.255.192 \
>      broadcast 192.168.0.64
# ifconfig eth0
eth0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:73
     inet addr:192.168.0.75 Bcast:192.168.0.64 Mask:255.255.255.192
     inet6 addr: fe80::2a0:24ff:fe56:e373/64 Scope:Link
◁◁◁◁◁
```


Mit den Parametern `up` und `down` können Sie mit `ifconfig` einzelne Interfaces gezielt hoch- bzw. herunterfahren.


Das Loopback-Interface hat nach Konvention die IP-Adresse `127.0.0.1` und wird automatisch konfiguriert. Sollte das aus irgendwelchen Gründen einmal nicht klappen oder die Konfiguration verlorengehen, dann können Sie das über

Loopback-Interface

```
# ifconfig lo 127.0.0.1 up
```

nachholen.

Aliasnamen

Zu Testzwecken oder für besondere Anforderungen kann es sinnvoll sein, einer Schnittstelle einen Aliasnamen mit einer abweichenden IP-Adresse, Netzmaske usw. zu geben. Das ist mit `ifconfig` kein Problem:

```
# ifconfig eth0:0 192.168.0.111
# ifconfig eth0:0
eth0:0 Link encap:Ethernet  HWaddr 00:A0:24:56:E3:72
     inet addr:192.168.0.111 Bcast:192.168.0.255 Mask:255.255.255.0
     UP BROADCAST MULTICAST  MTU:1500  Metric:1
     Interrupt:5 Base address:0xd800
```

Der Aliasname wird aus dem Interfacenamen gebildet, indem getrennt durch einen Doppelpunkt eine Namenserweiterung angehängt wird. Wie diese Namenserweiterung aussieht, ist gleichgültig (es spräche nichts gegen `eth0:Mr.X`), nach Konvention numeriert man die Aliasnamen aber fortlaufend durch: `eth0:0`, `eth0:1`, …

Übungen

23.1 [1] Welches Kernel-Modul gehört zu Ihrer Netzwerkkarte? Ist es geladen?

23.2 [!1] Kontrollieren Sie, ob Ihre Netzwerkkarte läuft und welche IP-Adresse ihr zugeordnet ist.

23.3 [!2] Weisen Sie Ihrer Netzwerkkarte eine neue IP-Adresse (ggf. nach Angaben Ihres Trainers) zu. Kontrollieren Sie, ob Sie die anderen Rechner im Netz noch erreichen können.

23.1.3 Wegleitung konfigurieren mit `route`

Jeder Rechner in einem TCP/IP-Netzwerk benötigt Wegleitung, denn selbst die einfachste Station besitzt mindestens zwei Netzschnittstellen – das Loopback-Interface und die Schnittstelle zum restlichen Netz, also eine Ethernet- oder WLAN-Karte oder eine Verbindung zum Internet. Die Routen für das Loopback-Interface und für die Netze, in denen Netzwerkkarten sich direkt befinden, werden bei aktuellen Linux-Kernels bei der Initialisierung der Netzwerkkarten automatisch gesetzt. Andere Routen – insbesondere die »Default-Route«, die angibt, wo Datagramme hingeschickt werden, für die es keine bessere Antwort gibt – müssen explizit konfiguriert werden.

Grundsätzlich unterscheiden wir zwischen *statischer* und *dynamischer* Wegleitung. Bei ersterer werden die Routen manuell gesetzt und danach nicht oder nur selten geändert. Bei letzterer unterhält das System sich mit anderen Routern in seiner Umgebung und passt seine Routen den aktuellen Gegebenheiten im Netz an. Dynamische Wegleitung erfordert die Installation und Konfiguration von »Routing-Daemons« wie `gated` oder `routed` und wird hier nicht weiter behandelt. Wir beschäftigen uns im Rest dieses Abschnitts ausschließlich mit statischer Wegleitung.

Routing-Tabelle

Der Kernel unterhält eine Routing-Tabelle, die die aktuelle Konfiguration der Wegleitung zusammenfasst. Sie enthält Regeln (die Routen), die beschreiben, wohin welche Datagramme geschickt werden müssen. Maßgeblich dafür ist deren Zieladresse. Sie können die Routing-Tabelle mit dem Kommando `route` abrufen:

```
# ifconfig eth0 192.168.0.75
# route
Kernel IP routing table
Destination  Gateway Genmask       Flags Metric Ref Use Iface
192.168.0.0  *       255.255.255.0 U     0      0   0   eth0
```

Die Spalten der Tabelle haben folgende Bedeutung:

- Die erste Spalte enthält die Zieladresse. Als Adresse kommen Netz- und Stationsadressen sowie der Eintrag (`default`) für die sogenannte Default-Route in Frage. Die Default-Route legt das Ziel für alle Datagramme fest, für die keine der übrigen Routen gilt. Default-Route
- Die zweite Spalte definiert als Ziel für die Datagramme einen Router, an den die Pakete weitergegeben werden. Gültige Einträge an dieser Stelle sind Stationsadressen sowie der Eintrag »*« wenn die Pakete nicht an einen anderen Rechner gehen sollen.
- Die dritte Spalte enthält die zur Zieladresse passende Netzmaske. Handelt es sich bei der Zieladresse um eine Station, dann steht hier die Netzmaske `255.255.255.255`. Die Default-Route hat die Netzmaske `0.0.0.0`.
- Die vierte Spalte enthält Flags, die die Route näher beschreiben. Folgende Werte können u. a. vorkommen:

  `U` die Route ist aktiv (up)

  `G` die Route ist eine »Gateway-Route«, das heisst, als Ziel ist ein Router (und kein direkt angeschlossenes Netz wie mit "*") angegeben.

  `H` die Route ist eine »Host-Route«, das heisst, die Zieladdresse bezeichnet einen einzelnen Rechner. `G` und `H` schliessen sich natürlich nicht aus und tauchen manchmal zusammen auf.
- Die fünfte und sechste Spalte enthalten Angaben, die bei dynamischem Routing eine Rolle spielen: Die »Metrik« in der fünften Spalte gibt die Anzahl der »Hops« zum Ziel an; sie wird vom Linux-Kern nicht ausgewertet, sondern ist vor allem für Programme wie den `gated` interessant. Der Wert in der sechsten Spalte wird in Linux nicht verwendet.
- Die siebte Spalte gibt an, wie oft die Route schon verwendet wurde.
- Die achte Spalte schließlich enthält optional die Schnittstelle, über die die Datagramme weitergeleitet werden sollen. Das kommt insbesondere bei Routern vor, die mehrere Interfaces besitzen, etwa Ethernet-Schnittstellen in verschiedenen Netzsegmenten oder eine Ethernet-Schnittstelle und eine Schnittstelle zum ISDN.

Am Beispiel wird deutlich, dass der Kernel beim Setzen der IP-Adresse mit `ifconfig` nicht nur eigenständig Netzmaske und Broadcast-Adresse setzt, sondern auch mindestens eine Route – diejenige nämlich, die alle Datagramme, deren Zieladressen im direkt an die Schnittstelle angeschlossenen Netz liegen, auf dieses Netz leitet.

Ein komplexeres Beispiel für eine Routing-Tabelle könnte so aussehen:

```
# route
Kernel IP Routentabelle
Ziel        Router      Genmask         Flags Metric Ref Use Iface
192.168.0.0 *           255.255.255.0   U     0      0   0   eth0
192.168.2.0 *           255.255.255.0   U     0      0   0   eth1
10.10.3.0   192.168.0.1 255.255.255.0   UG    0      0   0   eth0
112.22.3.4  *           255.255.255.255 UH    0      0   0   ppp0
default     112.22.3.4  0.0.0.0         UG    0      0   0   ppp0
```

Beim Beispielrechner handelt es sich offenbar um einen Router mit drei Schnittstellen. Die ersten drei Routen sind Netzwerkrouten, die Datagramme laufen entsprechend ihres Zielnetzes entweder über `eth0`, `eth1` oder den Router `192.168.0.1` (der über die erste Route zu erreichen ist). Die vierte Route ist eine »Host-Route«, die eine Punkt-zu-Punkt-Verbindung zum Providerrechner über das Modem `ppp0`

aufbaut. Die fünfte Route ist die entsprechende Default-Route, die alle Datagramme, die nicht in die lokalen Netze `192.168.0.0/24`, `192.168.2.0/24` oder `10.10.3.0/24` gerichtet sind, über das Modem in die weite Welt weiterleitet.

Das Kommando `route` dient nicht nur zum Abrufen, sondern auch zur Manipulation der Routing-Tabelle. Für das oben gezeigte Beispiel (drei lokale Ethernet-Segmente und die PPP-Verbindung) würde die Routing-Tabelle etwa folgendermassen aufgebaut:

```
# route add -net 192.168.0.0 netmask 255.255.255.0 dev eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 dev eth1
# route add -net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
# route add -host 112.22.3.4 dev ppp0
# route add default dev ppp0
```

Die ersten beiden Zeilen im Beispiel sind eigentlich nicht nötig, da die entsprechenden Routen automatisch in Kraft gesetzt werden, wenn die Schnittstellen ihre Adressen bekommen.

Allgemeiner gesagt hat `route` zum Setzen und Entfernen von Routen folgende Syntax:

```
route add [-net|-host] ⟨Ziel⟩ [netmask ⟨Netzmaske⟩]▷
 ◁      [gw ⟨Gateway⟩] [[dev] ⟨Interface⟩]
route del [-net|-host] ⟨Ziel⟩ [netmask ⟨Netzmaske⟩]▷
 ◁      [gw ⟨Gateway⟩] [[dev] ⟨Interface⟩]
```

Zum Hinzufügen einer Route müssen Sie den entsprechenden Parameter (`add`) setzen; danach geben Sie an, ob es sich um eine Stations- oder Netzroute handelt (`-host` oder `-net`). Danach wird das Ziel definiert. Handelt es sich um eine Netzroute, muss immer eine Netzmaske (`netmask` ⟨*Netzmaske*⟩) angegeben werden; Sie können die Netzmaske auch im CIDR-Stil an die Zieladresse anhängen. Für jede Route muss entweder ein Router (⟨*Gateway*⟩) oder eine Schnittstelle als »nächste Station« angegeben werden.

Routen löschen

Löschen könnten Sie die Routen zum Beispiel so:

```
# route del -net 192.168.0.0 netmask 255.255.255.0
# route del -net 192.168.2.0 netmask 255.255.255.0
# route del -net 10.0.3.0 netmask 255.255.255.0
# route del -host 112.22.3.4
# route del default
```

Zum Löschen einer Route müssen Sie die gleichen Angaben machen wie zum Hinzufügen einer Route. Lediglich die Angabe des Gateways bzw. des Interfaces können Sie weglassen. Bei doppelt vorkommenden Zielen, etwa dasselbe Zielnetz über zwei verschiedene Schnittstellen, wird zuerst die jüngste (zuletzt gesetzte) Route gelöscht.

IP-Forwarding

Soll ein Rechner wie im Beispiel als Gateway zwischen zwei Netzen dienen, so sollte der Kernel IP-Pakete, die nicht für den Rechner selbst bestimmt sind, entsprechend der Routing-Tabelle weitergeben. Dieses sogenannte **IP-Forwarding** ist standardmäßig ausgeschaltet. Die Einstellung und Anzeige des IP-Forwarding findet über die (Pseudo!-)Datei `/proc/sys/net/ipv4/ip_forward` statt. In ihr »steht« nur ein Zeichen – entweder eine Null (abgeschaltet) oder eine Eins (aktiviert). »Geschrieben« wird die Datei üblicherweise mit `echo`:

```
# cat /proc/sys/net/ipv4/ip_forward
0
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
# cat /proc/sys/net/ipv4/ip_forward
1
```

Achtung: Diese Einstellung geht, wie die bisher besprochenen Einstellungen per Kommando, verloren, wenn der Rechner neu gestartet wird. (Die Distributionen haben Mittel und Wege, diese Einstellung permanent abzulegen; bei Debian GNU/Linux etwa über einen Eintrag der Form »ip_forward=yes« in der Datei /etc/network/options, bei den Novell/SUSE-Distributionen durch »IP_FORWARD="yes"« in /etc/sysconfig/sysctl. Bei den RedHat-Distributionen können Sie eine Zeile der Form

```
net.ipv4.ip_forward = 1
```

in die Datei /etc/sysctl.conf eintragen.)

23.1.4 Netzkonfiguration mit ip

Mit dem Kommando ip können sowohl die Schnittstelle als auch Routen konfiguriert werden. Das Kommando soll die eben beschriebenen Kommandos über kurz oder lang ablösen. Die Syntax lautet allgemein wie folgt:

```
ip [⟨Option⟩] ⟨Objekt⟩ [⟨Kommando⟩ [⟨Parameter⟩]]
```

Als ⟨*Objekt*⟩ kommen unter anderem link (Parameter der Schnittstelle), addr (IP-Adresse und andere Adressen der Schnittstelle) und route (Auslesen, Setzen und Löschen von Routen) in Frage. Für jedes Objekt stehen spezifische Kommandos zur Verfügung.

Wird kein Kommando angegeben, werden die momentanen Einstellungen entsprechend des Kommandos list bzw. show angezeigt. Weitere typische Kommandos sind set für das Objekt link sowie add und del für die Objekte addr und route.

Die meisten Kommandos erfordern noch weitere Parameter, schließlich müssen Sie, möchten Sie mit »ip addr add« eine IP-Adresse zuweisen, diese IP-Adresse auch angeben.

Die entsprechende Syntax können Sie dem Befehl ip mit dem Kommando help entlocken. So zeigt »ip help« alle möglichen Objekte an und »ip link help« alle zum Objekt link gehörenden Parameter inklusive Syntax. Leider ist die Syntax nicht immer ganz einfach zu durchschauen.

Wenn Sie sich mit Cisco-Routern auskennen, werden Sie gewisse Ähnlichkeiten zum Cisco-ip-Kommando bemerkt haben. Diese Ähnlichkeiten sind beabsichtigt.

Als Beispiel: Möchten Sie einer Netzwerkkarte eine IP-Adresse zuweisen, können Sie dafür folgendes Kommando verwenden:

```
# ip addr add local 192.168.2.1/24 dev eth0 brd +
```

Anders als bei ifconfig *müssen* hier Netzmaske und Broadcastadresse mit angegeben werden, letztere auch indirekt mit brd +. Der Parameter local wird als Parameter zur Angabe einer IP-Adresse verwendet, da es sich für »ip addr add« hierbei aber um den Default-Parameter handelt, kann er auch weggelassen werden. Die Default-Parameter können Sie der Handbuchseite ip(8) entnehmen.

Achtung: Im Unterschied zu ifconfig ist die Schnittstelle nach der Zuweisung einer IP-Adresse noch nicht aktiv. Die Aktivierung erfolgt gesondert:

```
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo-fast qlen 100
        link/ether 00:a0:24:56:e3:72 brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0
```

```
# ip link set up dev eth0
# ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo-fast qlen 100
        link/ether 00:a0:24:56:e3:72 brd ff:ff:ff:ff:ff:ff
        inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0
        inet6 fe80::2a0:24ff:fe56:e372/64 scope link
```

Auch mit `ip` können Sie Aliasnamen für Schnittstellen vergeben:

```
# ip addr add 192.168.0.222/24 dev eth0 brd + label eth0:0
```

Es empfiehlt sich, die Syntax von `ip` kennenzulernen, nicht nur, weil dieses Programm der zukünftige Standard ist, sondern weil es zum Teil auch anschaulicher zu verwenden ist als die Alternativen: Das Setzen und Löschen von Routen ist beispielsweise einfacher als mit `route`:

```
# ip route add 192.168.2.1 via 192.168.0.254
# ip route del 192.168.2.1
```

23.2 Dauerhafte Netzkonfiguration

Eins ist sicher: Wenn Sie einmal die richtige Netzkonfiguration für Ihr System herausbekommen haben, werden Sie diese nicht immer wieder von neuem einstellen wollen. Leider vergisst der Linux-Kernel sie aber beim Herunterfahren.

Die verschiedenen Linux-Distributionen haben dieses Problem auf unterschiedliche Weise gelöst:

Bei Debian GNU/Linux und den davon abgeleiteten Distributionen steht die Netzkonfiguration in der Datei `/etc/network/interfaces`. Diese Datei ist mehr oder weniger selbsterklärend:

```
# cat /etc/network/interfaces
auto lo eth0

iface lo inet loopback

iface eth0 inet static                                    oder »... inet dhcp«
  address 192.168.0.2
  netmask 255.255.255.0
  network 192.168.0.0
  broadcast 192.168.0.255
  up route add -net 10.10.3.0/24 gw 192.168.0.1
  down route del -net 10.10.3.0/24 gw 192.168.0.1
```

In der Datei gibt es einen Eintrag für jede Schnittstelle. Die Schnittstellen können mit den Kommandos `ifup` und `ifdown` einzeln oder (mit der Option `-a`) kollektiv aktiviert bzw. deaktiviert werden; beim Systemstart kümmert sich das Skript `/etc/init.d/networking` um die Initialisierung der Schnittstellen. (Alternativ macht das auch `udev`, nur müssen die betreffenden Schnittstellen dann in einer Zeile wie »`allow-hotplug eth0`« aufgeführt werden. Interessant ist das für Netzwerkkarten, die nicht immer zur Verfügung stehen, etwa USB-basierte Ethernet- oder UMTS-Adapter.) – Zeilen mit `up` am Anfang enthalten Kommandos, die beim Start der Schnittstelle ausgeführt werden (in der Reihenfolge, wie sie in der Datei stehen), Zeilen mit `down` am Anfang entsprechend Kommandos für den Stopp. Mehr Beispiele für die fremdartigen und wundervollen Dinge, die mit dem Debian-Netzkonfigurationsmechanismus möglich sind, finden Sie in `interfaces`(5) und der Datei `/usr/share/doc/ifupdown/examples/network-interfaces.gz`.

Der YaST als zentrales Konfigurationswerkzeug der Novell/SUSE-Distributionen enthält selbstverständlich auch Module zur Netzkonfiguration (Netzwerkgeräte/Netzwerkkarte). Einstellungen, die mit dem YaST vorgenommen werden, legt dieser häufig in Form von Variablen in Dateien unterhalb von `/etc/sysconfig` ab, wo Init-Skripte oder das Programm `SuSEconfig` sie auslesen. Die Netzkonfiguration im besonderen wird (und das können Sie auch manuell) im Verzeichnis `/etc/sysconfig/network` abgelegt. Hier befindet sich für jede Schnittstelle eine Datei namens `ifcfg-`⟨*Schnittstelle*⟩ (also zum Beispiel `ifcfg-eth0`), die die Einstellungen für die betreffende Schnittstelle enthält. Das kann ungefähr so aussehen:

```
BOOTPROTO='static'                                  oder dhcp (unter anderem)
BROADCAST='192.168.0.255'
ETHTOOL_OPTIONS=''
IPADDR='192.168.0.2'
MTU=''
NAME='79c970 [PCnet32 LANCE]'                       Name im YaST
                                                    (VMware läßt grüßen)
NETMASK='255.255.255.0'                             Oder PREFIXLEN=24
NETWORK='192.168.0.0'
REMOTE_IPADDR=''                                    Gegenstelle bei PPP
STARTMODE='auto'                                    oder manual, hotplug, ...
USERCONTROL='no'
```

(Eine ausführliche Erklärung steht in `ifcfg`(5).) Allgemeine Einstellungen für die Netzkonfiguration stehen in `/etc/sysconfig/network/config`. – Auch die SUSE-Distributionen unterstützen Kommandos namens `ifup` und `ifdown`, die allerdings subtil anders funktionieren als die von Debian GNU/Linux. Zumindest die grundlegenden Aufrufe wie »`ifup eth0`« sind gleich, aber schon »`ifup -a`« geht nicht – um alle Schnittstellen zu starten oder anzuhalten, müssen Sie »`rcnetwork start`« oder »`rcnetwork stop`« sagen. (Zum Trost funktioniert auch »`rcnetwork start eth0`«.) `rcnetwork` ist SUSE-typisch natürlich nichts anderes als ein symbolisches Link auf das Init-Skript `/etc/init.d/network`.

Routen können Sie bei den Novell/SUSE-Distributionen über die Datei `/etc/sysconfig/network/routes` konfigurieren. Der Inhalt der Datei (hier passend zum oben verwendeten Beispiel) ähnelt der Darstellung des Befehls `route`:

```
# cat /etc/sysconfig/network/routes
10.10.3.0       192.168.0.1     255.255.255.0   eth0
112.22.3.4      0.0.0.0         255.255.255.255 ppp0
default         112.22.3.4      -               -
```

Soll kein Gateway verwendet werden, lautet der Eintrag »`0.0.0.0`«, nicht gesetzte Netzmasken oder Schnittstellennamen werden durch ein »`-`« dargestellt. Auch die Routen werden durch Aufruf von »`rcnetwork restart`« gesetzt. Zu den letzten beiden Routen im Beispiel ist zu sagen, dass Punkt-zu-Punkt-Routen für Wählverbindungen in der Regel dynamisch von den entsprechenden Daemons (etwa `pppd`) gesetzt werden. – Wenn Sie Routen für einzelne Schnittstellen definieren möchten, können Sie die entsprechenden Zeilen statt in die `routes`-Datei auch in eine Datei namens `ifroute-`⟨*Schnittstelle*⟩ (also zum Beispiel `ifroute-eth0`) tun. Die vierte Spalte (die mit dem Schnittstellennamen) wird dabei durch den Namen der Schnittstelle ersetzt, falls Sie sie in der Datei leer lassen.

Bei Fedora und den anderen Red-Hat-Distributionen existiert ähnlich wie bei SUSE ein Verzeichnis `/etc/sysconfig`, in dem sich Dateien befinden, in denen diverse Variablen gesetzt werden. Wie bei SUSE existieren Dateien in der Art von `ifcfg-eth0` für die Konfiguration jeder Schnittstelle, nur dass

sie sich in einem Verzeichnis namens `/etc/sysconfig/network-scripts` befinden. Die SUSE-Dateien sind aber nicht 1 : 1 übertragbar, da sie sich von den Red-Hat-Dateien im internen Aufbau unterscheiden. Bei Red Hat könnten Sie unsere Beispielkonfiguration für `eth0` etwa wie folgt realisieren: In `/etc/sysconfig/network-scripts/ifcfg-eth0` steht

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
NETWORK=192.168.0.0
NETMASK=255.255.255.0
IPADDR=192.168.0.2
USERCTL=no
```

Die `ifup`- und `ifdown`-Kommandos gibt es auch bei Fedora, aber auch hier können Sie immer nur eine Schnittstelle auf einmal starten oder anhalten.

Statische Routen können Sie bei Red Hat in eine Datei in `/etc/sysconfig/network-scripts` tun, die `route-`⟨*Schnittstelle*⟩ heißt (also zum Beispiel `route-eth0`). In diesem Fall ist das Format wie folgt:

```
ADDRESS0=10.10.3.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.1
```

(zusätzliche Routen verwenden entsprechend `ADDRESS1`, `NETMASK1`, …, `ADDRESS2` und so weiter). Es gibt auch noch ein älteres Dateiformat, in dem einfach jede Zeile der Datei an »`ip route add`« angehängt wird. Dabei bieten sich also Zeilen an wie

```
10.10.3.0/24 via 192.168.0.1
```

Zu allem Überfluss können Sie auch noch in `/etc/sysconfig/static-routes` statische Routen angeben, ohne sich auf einzelne Schnittstellen beziehen zu müssen. Die Zeilen in dieser Datei werden nur beachtet, wenn sie mit dem Schlüsselwort `any` anfangen; der Rest der Zeile wird an »`route add -`« angehängt (Konsistenz? Wer braucht Konsistenz?), so dass aus einer Zeile wie

```
any net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
```

das Kommando

```
route add -net 10.10.3.0 netmask 255.255.255.0 gw 192.168.0.1
```

resultiert.

23.3 DHCP

DHCP, das »Dynamic Host Configuration Protocol«, dient dazu, Ihnen als Administrator die Mühe abzunehmen, die passenden Netzparameter auf jeder einzelnen Station im Netz konfigurieren zu müssen. Statt dessen holt ein Linux-Rechner sich die Netzparameter – neben der IP-Adresse mit Zubehör typischerweise die Adresse eines Default-Routers und eines oder mehrerer DNS-Server – von einem entfernten DHCP-Server, wenn die Netzwerkkarte gestartet wird.

Voraussetzung dafür, dass das funktioniert, ist natürlich ein vorhandener DHCP-Server. Die Installation und den Betrieb eines DHCP-Servers zu erklären würde den Umfang dieser Unterlage leider sprengen. Sollten Sie jedoch einen der gängigen DSL-Router für Ihren Internet-Zugang benutzen oder in der Firma auf eine kompetente Netzwerk-Abteilung zurückgreifen können, dann ist das nich wirklich Ihr Problem, da die nötige Funktionalität dann in der Regel vorgekocht zur Verfügung steht oder sich leicht aktivieren läßt.

Um DHCP zur Konfiguration zu verwenden, müssen Sie die Konfiguration der gängigen Linux-Distributionen nur geringfügig anpassen:

Setzen Sie bei Debian GNU/Linux oder Ubuntu einfach in `/etc/network/interfaces` statt

```
iface eth0 inet static
```

und den darauffolgenden Zeilen mit den Adressen- und Routeninformationen die Zeile

```
iface eth0 inet dhcp
```

ein. Adresse, Netzmaske und Default-Route werden dann vom DHCP-Server bezogen. Sie können natürlich weiterhin mit `up` und `down` Kommandos ausführen, wenn die Verbindung steht oder bevor sie abgebaut wird.

Bei den Novell/SUSE-Distributionen setzen Sie in der Datei mit der Konfiguration für die betreffende Schnittstelle (`ifcfg-eth0` oder so) statt

```
BOOTPROTO='static'
```

den Parameter

```
BOOTPROTO='dhcp'
```

Die Felder `BROADCAST`, `IPADDR`, `NETMASK` und `NETWORK` lassen Sie einfach leer.

Bei Fedora und den anderen Red-Hat-Distributionen setzen Sie zur Verwendung von DHCP in der Konfigurationsdatei für die Schnittstelle statt

```
BOOTPROTO=none
```

den Parameter

```
BOOTPROTO=dhcp
```

Die Adressenparameter können Sie dann einfach weglassen.

Allgemein unterstützen die Netzkonfigurationsmethoden der Distributionen diverse andere Optionen, unter anderem VLAN (auf demselben Kabel werden mehrere »virtuelle« Netze übertragen, die einander nicht sehen), Verschlüsselung oder Bonding (mehrere Netzwerkkarten arbeiten parallel, für mehr Kapazität und/oder Ausfallsicherheit). Wichtig ist auch der Fall, dass ein mobiler Rechner flexibel an mehreren Netzen teilnehmen kann, etwa im Büro und daheim. Die gebotenen Optionen unterscheiden sich stark von Distribution zu Distribution und können hier nicht im Detail besprochen werden.

23.4 IPv6-Konfiguration

SLAAC

Um Ihren Rechner in ein IPv6-Netz zu integrieren, müssen Sie im Idealfall gar nichts machen: Der Mechanismus der »zustandslosen Adressen-Autokonfiguration« (*stateless address autoconfiguration*, SLAAC) macht es möglich, dass alles automatisch vonstatten geht. Bei IPv6 spielt SLAAC in etwa die Rolle, die DHCP bei IPv4 spielt, jedenfalls für einfache Anwendungen.

Vorgehensweise

Wenn eine neue IPv6-Schnittstelle aktiviert wird, erzeugt die Station zuerst die passende verbindungslokale Adresse. Dabei wird das Präfix `fe80::/64` angenommen und die Stationsadresse aus der MAC-Adresse der Schnittstelle abgeleitet[1]. Anschließend verschickt die Station eine verbindungslokale »Router-Aufforderung« (*router solicitation*, RS) über die Schnittstelle an die Multicast-Adresse `ff02::2`, die alle Router im Subnetz anspricht. Diese bringt den oder die Router auf dem physikalischen Netz der Schnittstelle dazu, über »Router-Ankündigungen« (*router advertisements*, RA) mitzuteilen, welche Präfixe sie routen können. Die Station konstruiert auf deren Basis zusätzliche (beispielsweise global sichtbare) Adressen für die Schnittstelle. – RS und RA sind Bestandteile des »Neighbor Discovery Protocol« (NDP), das wiederum Teil von ICMPv6, dem IPv6-Pendant von ICMP, ist. RAs und damit die davon abgeleiteten IPv6-Adressen bleiben nur für eine gewisse Zeit gültig, wenn sie nicht erneuert werden. Router versenden RAs auch unaufgefordert in periodischen Abständen; die RS dient nur dazu, dass Sie für eine neue Schnittstelle nicht die nächste unaufgeforderte RA abwarten müssen, sondern die nötigen Informationen sofort bekommen können.

Vorteile

Der Vorteil dieses Ansatzes ist, dass er ohne die explizite Konfiguration eines DHCP-Servers auskommt. Auch Redundanz ist leicht zu erreichen, indem man mehrere Router im Subnetz konfiguriert. Außerdem müssen die Router sich nicht wie bei DHCP merken, welche Station gerade welche IP-Adresse zugeteilt bekommen hat (deshalb »zustandslos«). Das Ganze heißt aber nicht, dass Sie bei IPv6 ganz auf DHCP verzichten können (es gibt DHCPv6), da es wichtige Netzwerkinformationen gibt, die SLAAC Ihnen nicht liefert (Stichwort: DNS-Server – wobei es hierfür einen neuen, noch nicht weit unterstützten Standard gibt).

Adressen abfragen

Sie können die Adressen abfragen, die das System einer Schnittstelle zugewiesen hat:

```
# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500▷
 ◁ qdisc pfifo_fast state UP qlen 1000
  link/ether 70:5a:b6:9c:40:6a brd ff:ff:ff:ff:ff:ff
  inet 192.168.178.130/24 brd 192.168.178.255 scope global eth0
  inet6 2001:db8:56ee:0:725a:b6ff:fe9c:406a/64 scope global dynamic
     valid_lft 6696sec preferred_lft 3096sec
  inet6 fe80::725a:b6ff:fe9c:406a/64 scope link
     valid_lft forever preferred_lft forever
```

Sie sehen hier sowohl die verbindungslokale Adresse (»`scope link`«, beginnend mit `fe80::`) als auch eine global sichtbare Adresse (»`scope global dynamic`«, beginnend mit `2001:`), die die Schnittstelle über SLAAC erhalten hat. Wenn Sie genau hinschauen, können Sie auch die MAC-Adresse (in der `link/ether`-Zeile) mit den Stationsteilen der IPv6-Adressen korrelieren.

Privatsphäre

Die von Ihrer MAC-Adresse abgeleiteten Stationsteile Ihrer global sichtbaren IPv6-Adressen sind übrigens ein potentielles Problem, was Ihre Privatsphäre angeht: Wenn Sie immer mit derselben Adresse im Netz unterwegs sind, ist es leicht, Ihre Aktionen (aufgerufene Web-Seiten und ähnliches) dieser Adresse zuzuordnen. Auch wenn Sie, wie es so schön heißt, nichts zu verbergen haben, kann Ihnen niemand gewisse Bauchschmerzen verdenken, die Ihnen das schon aus Prinzip

[1] Das Verfahren dafür ist wie folgt: Betrachten Sie die MAC-Adresse *mn*:*op*:*qr*:*st*:*uv*:*wx*. Das (von links gezählt) 3. Bit von *n*, das bei MAC-Adressen immer 0 ist, wird auf 1 gesetzt (wir nennen das Resultat *n*'), und die Stationsadresse ist dann *mn*'*op*:*qr*`ff:fe`*st*:*uvwx*. Aus der MAC-Adresse `70:5a:b6:9c:40:6a` wird also die Stationsadresse `725a:b6ff:fe9c:406a`.

machen könnte. Abhilfe schaffen die »Privatsphäre-Erweiterungen« (*privacy extensions*), die eine zufällige anderweitig unbenutzte Stationsadresse setzen und in periodischen Abständen neu auswürfeln. Die Privatsphäre-Erweiterungen können Sie für eine Schnittstelle – hier `eth0` – mit `sysctl` aktivieren:

```
# sysctl -w net.ipv6.conf.eth0.use_tempaddr=2
# ip link set dev eth0 down
# ip link set dev eth0 up
```

Um die Einstellung permanent zu machen, können Sie sie in `/etc/sysctl.conf` eintragen.

Manuelle Konfiguration

Schließlich ist es natürlich auch möglich, IP-Adressen manuell zu vergeben. Das können Sie entweder mit `ifconfig` erledigen

```
# ifconfig eth0 inet6 add 2001:db8:abcd::1/64
```

oder mit `ip`:

```
# ip addr add 2001:db8:abcd::1/64 dev eth0
```

Wie Sie diese Konfiguration permanent machen können, hängt von Ihrer Distribution ab; die Techniken dafür entsprechen weitgehend den in Abschnitt 23.2 diskutierten.

23.5 Namensauflösung und DNS

Das DNS oder »Domain Name System« ist eine der Grundlagen für die Skalierbarkeit des Internet. Seine Aufgabe besteht darin, Rechnern textuelle Namen zuzuordnen und bei Bedarf die zugehörigen IP-Adressen herauszufinden (oder umgekehrt). Dies realisiert es im wesentlichen über eine weltweit verteilte »Datenbank« aus DNS-Servern.

Inzwischen kümmert sich das DNS auch um zahlreiche andere Aufgaben, vom Aufspüren der für eine Domain zuständigen Mail-Server bis zur Hilfe bei der Vermeidung von Spam.

Resolver

Programme auf einem Linux-Rechner reden in der Regel nicht direkt mit dem DNS, sondern bedienen sich dafür eines »Resolvers«. Dieser ist üblicherweise Teil der C-Bibliothek. Die zentrale Konfigurationsdatei für den Resolver heißt `/etc/resolv.conf`. Hier werden zum Beispiel die DNS-Server konfiguriert, die der Resolver konsultieren soll. Dazu existieren fünf Hauptdirektiven:

`domain` ⟨*Name*⟩ (lokale Domain) Anhand dieses Eintrags versucht der Resolver, unvollständige Rechnernamen (typisch solche, die keinen Punkt enthalten) um einen Domainanteil zu ergänzen.

Was ein unvollständiger Name ist, wird von der Option `ndots` (siehe Tabelle 23.1) bestimmt.

`search` ⟨*Domain*$_1$⟩ ⟨*Domain*$_2$⟩ **…** (Suchliste) Alternativ zu einem einzigen Eintrag mittels `domain` kann mit `search` auch eine Liste mit mehreren Ergänzungen für unvollständige Rechnernamen angegeben werden. Die Einträge in der Liste werden durch Leerzeichen getrennt. Zunächst wird versucht, den unveränderten Rechnernamen aufzulösen. Wenn dies scheitert, werden die Listeneinträge der Reihe nach angehängt und diese Namen ausprobiert. `domain` und `search` schließen sich gegenseitig aus; tauchen beide in der Konfiguration auf, gilt der textuell letzte Eintrag in der Datei.

Tabelle 23.1: Optionen innerhalb `/etc/resolv.conf`

Option	Wirkung
`debug`	Die regulären Betriebsmeldungen werden auf `stdout` ausgegeben (meist nicht unterstützt).
`ndots` ⟨*n*⟩	Die minimale Anzahl von Punkten im Namen, bei welcher der Resolver direkt nachsieht, ohne auf die Suchliste zuzugreifen.
`attempts` ⟨*n*⟩	Die Anzahl der Anfragen an einen Server, bis der Resolver aufgibt. Maximalwert ist 5.
`timeout` ⟨*n*⟩	Der Anfangs-Timeout für Abfrageversuche in Sekunden. Maximalwert ist 30.
`rotate`	Nicht nur der erste, sondern alle Server werden abwechselnd befragt.
`no-check-names`	Deaktiviert die standardmäßige Überprüfung, ob zurückgelieferte Hostnamen nur gültige Zeichen enthalten.

```
nameserver 192.168.10.1
nameserver 192.168.0.99
search     foo.example.com bar.example.com example.com
```

Bild 23.1: Beispiel für `/etc/resolv.conf`

`nameserver` ⟨*IP-Adresse*⟩ (Lokaler DNS-Server) Der lokale Resolver wird den hier eingetragenen DNS-Server befragen. Es sind bis zu drei `nameserver`-Direktiven erlaubt, die bei Bedarf nacheinander abgefragt werden.

`sortlist` ⟨*IP-Adresse*⟩[/⟨*Netzmaske*⟩] (Sortierung) Falls zu einem Rechnernamen mehrere Adressen zurückgeliefert werden, so wird die hier eingetragene bevorzugt. Bis zu 10 Einträge sind in der Sortierliste möglich.

`options` ⟨*Option*⟩ (Optionen) Hiermit lassen sich besondere Resolver-Einstellungen vornehmen, die in der Tabelle 23.1 mit den Vorgabewerten aufgelistet sind. In der Praxis werden diese selten bis nie verändert.

Eine typische `/etc/resolv.conf`-Datei sehen Sie in Bild 23.1.

Die Alternative zum DNS ist die »lokale« Auflösung von Rechnernamen und IP-Adressen über die Datei `/etc/hosts`. Als ausschließliche Methode zur Namensauflösung ist sie höchstens in kleinen Netzen interessant, die nicht ans Internet angebunden sind, aber erwähnen sollten wir sie trotzdem – wenn Sie es nur mit ein paar Rechnern zu tun haben, dann ist es möglicherweise simpler, nur die DNS-Client-Seite zu konfigurieren und Ihren eigenen Rechnern per `/etc/hosts` Namen und Adressen zu geben. Sie müssen dann nur darauf achten, dass die Datei auf allen beteiligten Rechnern gleich ist.

Für kleine Netze empfiehlt sich das Programm `dnsmasq`, das den Inhalt einer `/etc/hosts`-Datei lokal über DNS verfügbar macht und alle anderen DNS-Anfragen ans »echte« DNS weiterleitet. Nebenbei fungiert es auch noch als DHCP-Server.

Bei der Datei `/etc/hosts` handelt es sich um gewöhnlichen ASCII-Text, in dem neben Kommentarzeilen, die mit »#« eingeleitet werden, zeilenweise Einträge vorgenommen werden können. Diese enthalten spaltenweise mindestens die IP-Adresse und den vollständigen Namen (FQDN) des Rechners. Daneben ist es erlaubt, noch Kurznamen für den Rechner anzugeben. Als Trennzeichen der einzelnen Spalten dienen Leer- und/oder Tabulatorzeichen, also »Freiplatz« (*white space*). Bild 23.2 zeigt den Inhalt einer exemplarischen `/etc/hosts`-Datei.

```
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#
# special IPv6 addresses

127.0.0.1       localhost
192.168.0.99    linux.example.com  linux
```

Bild 23.2: Die Datei `/etc/hosts` (SUSE)

In der Frühzeit des Internet – etwa bis in die frühen 1980er Jahre – wurde im wesentlichen eine große `/etc/hosts`-Datei zur Namensauflösung herangezogen. Auch Domains waren noch nicht erfunden. Damals hatte das Internet noch weniger Stationen (Tausende statt Abermillionen), aber die Wartung und Verteilung aktueller Versionen dieser Datei entwickelte sich zu einem zunehmenden Problem. Daher DNS.

Über die genaue Methodik der Namensauflösung durch die C-Bibliothek entscheidet tatsächlich eine Datei namens `/etc/nsswitch.conf`. Darin ist beispielsweise festgelegt, welche Dienste zur Namensauflösung in welcher Reihenfolge verwendet werden. Daneben finden sich Einträge zur Auflösung von Benutzernamen, Gruppen usw., die uns an dieser Stelle nicht weiter interessieren. Eine genaue Beschreibung von Syntax und Funktionsweise können Sie in `nsswitch.conf`(5) einsehen.

Dienste zur Namensauflösung

Der für die Auflösung von Rechnernamen interessante Teil von `/etc/nsswitch.conf` kann etwa so aussehen:

```
hosts: files dns
```

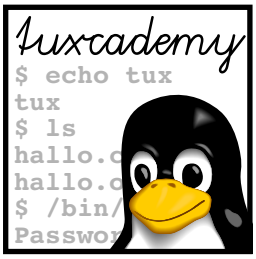
Hier wird also zunächst versucht, Rechnernamen mit Hilfe der lokalen Dateien (namentlich `/etc/hosts`) aufzulösen. Erst wenn dies scheitert, kommt das DNS zum Zug.

Kommandos in diesem Kapitel

dnsmasq	Ein einfacher DHCP- und cachender DNS-Server für kleine Installationen	`dnsmasq`(8)	402
ifconfig	Konfiguriert Netzwerk-Schnittstellen	`ifconfig`(8)	391
ifdown	Schaltet eine Netzwerk-Schnittstelle aus (Debian)	`ifdown`(8)	396
ifup	Schaltet eine Netzwerk-Schnittstelle ein (Debian)	`ifup`(8)	396
ip	Verwaltet Netzwerkschnittstellen und Routing	`ip`(8)	395
route	Verwaltet die statische Routing-Tabelle im Linux-Kern	`route`(8)	392

Zusammenfassung

- Treiber für Netzwerkkarten lädt der Linux-Kernel heute bei Bedarf über die `udev`-Infrastruktur.
- Das Kommando `ifconfig` dient zur Konfiguration von Interface-Parametern auf niedriger Ebene. Sie können damit auch das Loopback-Interface konfigurieren und Aliasnamen für Interfaces vergeben.
- Routen geben an, wie IP-Datagramme an ihren Empfänger geleitet werden.
- Zur Konfiguration von Routen dient das Kommando `route`.
- Das Kommando `ip` ist ein komfortabler Ersatz für `ifconfig` und `route`.
- Die verschiedenen Linux-Distributionen bieten unterschiedliche Methoden zur dauerhaften Netzkonfiguration an.
- Mit DHCP können Linux-Rechner Netzparameter dynamisch von einem zentralen Server beziehen.
- Gängige Mechanismen zur Namensauflösung beruhen auf dem DNS oder auf lokalen Konfigurationsdateien.
- Die Reihenfolge der Namensauflösung wird in der Datei `/etc/nsswitch.conf` eingerichtet.



24

Fehlersuche und Fehlerbehebung im Netz

Inhalt

24.1 Einführung 406
24.2 Lokale Probleme 406
24.3 Erreichbarkeit von Stationen prüfen mit `ping` 407
24.4 Wegleitung testen mit `traceroute` und `tracepath` 409
24.5 Dienste überprüfen mit `netstat` und `nmap` 412
24.6 DNS testen mit `host` und `dig` 415
24.7 Andere nützliche Diagnosewerkzeuge 418
24.7.1 `telnet` und `netcat` 418
24.7.2 `tcpdump` 420
24.7.3 `wireshark` 420

Lernziele

- Strategien zur Fehlersuche im Netz kennen
- Werkzeuge wie `ping`, `traceroute` und `netstat` zur Problemdiagnose einsetzen können
- Einfache Fehler in der Netzkonfiguration beheben können

Vorkenntnisse

- Kenntnisse über Linux-Systemadministration
- Kenntnisse über TCP/IP-Grundlagen (Kapitel 22)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 23)

24.1 Einführung

Systemadministratoren lieben es: Gerade haben Sie sich gemütlich mit einer schönen Tasse Kaffee oder Tee vor Ihrem Rechner niedergelassen und gedenken in aller Ruhe die neuesten Nachrichten auf LWN.net zu lesen, da steht eine lästige Person in der Tür: »Ich komme nicht ins Netz!« Mit dem Idyll ist es also erst mal wieder vorbei. Aber was tun?

Die Vernetzung von Rechnern ist ein komplexes Thema, und so sollten Sie sich nicht wundern, wenn alles Mögliche schief gehen kann. In diesem Abschnitt zeigen wir Ihnen die wichtigsten Werkzeuge und Strategien, mit denen Sie Probleme finden und ausbügeln können.

24.2 Lokale Probleme

Als erstes sollten Sie sich überzeugen, dass die Netzwerkkarte vorhanden ist und erkannt wird. (Zuallererst hilft vielleicht auch ein diskreter Blick hinter den Rechner, ob das Kabel noch im richtigen Port steckt oder nicht vielleicht die Damen und Herren von der Putzkolonne ein bisschen »kreative Rekonfiguration« gespielt haben.)

Schauen Sie nach, was »`ifconfig -a`« ausgibt. Mit diesem Parameter abgesetzt liefert das Programm Ihnen eine Übersicht über *alle* Netz-Schnittstellen des Rechners, auch die aktuell nicht aktiv konfigurierten, und zumindest `lo` und `eth0` (bei einem über Ethernet vernetzten Rechner) sollten zu sehen sein. Ist das nicht der Fall, dann haben Sie Ihr erstes Problem schon gefunden: Möglicherweise stimmt mit dem Treiber oder der Erkennung der Karte etwas nicht.

Alternativ tut es natürlich auch das Kommando »`ip link list`«, das ebenfalls alle vorhandenen Netz-Schnittstellen auflistet, unabhängig vom Konfigurationszustand.

Wenn Sie statt `eth0` nur etwas sehen wie `eth1`, dann kann es sein, dass die Netzwerkkarte im Rechner ausgewechselt wurde und `udev` der neuen Netzwerkkarte mit ihrer ebenso neuen MAC-Adresse auch einen neuen Schnittstellennamen verpasst hat. Bei einigermaßen fest in den Rechner eingebauten Netzwerkkarten sollte das eigentlich nicht passieren (oder wenn, dann nur, weil Sie als Administrator es eigenhändig gemacht haben), aber vielleicht haben Ihre Kollegen heimlich ihre PC(MCIA)-Netzwerkkarten oder USB-UMTS-Sticks vertauscht. Die Abhilfe besteht darin, in der Datei `/etc/udev/rules.d/70-persistent-net.rules` (oder so ähnlich) die Zeile mit dem alten Gerät zu löschen und in der Zeile mit dem neuen Gerät den Schnittstellennamen zu korrigieren. Starten Sie anschließend `udev` neu.

Wenn in der Ausgabe von `ifconfig` oder `ip` überhaupt nichts auftaucht, was Ihrer Netzwerkkarte ähnlich sieht, dann prüfen Sie mit `lsmod`, ob das passende Treibermodul geladen wurde. Wenn Sie nicht wissen, welches das passende Treibermodul überhaupt *ist*, können Sie in der Ausgabe von »`lspci -k`« nach dem Eintrag für Ihre Netzwerkkarte suchen. Dieser könnte ungefähr so aussehen:

```
02.00.0 Ethernet controller: Broadcom Corporation NetXtreme▷
 ◁ BCM5751 Gigabit Ethernet PCI Express (rev 01)
        Kernel driver in use: tg3
        Kernel modules: tg3
```

In diesem Fall sollten Sie sich davon überzeugen, dass das Modul `tg3` geladen ist.

Übungen

24.1 [!1] Welche Netzwerkkarten stehen in Ihrem System zur Verfügung? Sind sie alle konfiguriert? Welche Treiber werden benutzt?

24.3 Erreichbarkeit von Stationen prüfen mit ping

Wenn die Ausgabe von ifconfig die Schnittstelle zeigt und auch die Parameter, die dort zu sehen sind, vernünftig scheinen (schauen Sie vor allem nach der IP-Adresse, der Netzmaske – sehr wichtig – und der Broadcast-Adresse), dann ist es Zeit für ein paar Erreichbarkeitstests. Das einfachste Werkzeug dafür ist ein Programm namens ping, das eine IP-Adresse (oder einen DNS-Namen) übernimmt und versucht, der betreffenden Station ein ICMP-ECHO-REQUEST-Datagramm zu schicken. Diese Station sollte darauf mit einem ICMP-ECHO-REPLY-Datagramm antworten, das ping bemerkt und Ihnen meldet.

Als erstes sollten Sie testen, ob der Rechner mit sich selbst reden kann:

```
# ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.039 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.040 ms
                                                    Abbruch mit Strg+c ...
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.032/0.037/0.040/0.006 ms
```

Diese Ausgabe zeigt Ihnen, dass der »andere Rechner« (in diesem Fall nur die Loopback-Schnittstelle 127.0.0.1) zuverlässig erreicht werden konnte (es sind keine Pakete verlorengegangen).

Was soll »56(84) bytes of data« heißen? Ganz einfach: Ein IP-Datagrammkopf ohne Optionen ist 20 Bytes lang. Dazu kommt noch der Kopf eines ICMP-ECHO-REQUEST-Datagramms im Werte von 8 Bytes. Das erklärt schon mal die Differenz zwischen 56 und 84. Die magische Zahl 56 kommt daher, dass ping normalerweise dafür sorgt, dass im IP-Datagramm genau 64 Bytes Daten übertragen werden, nämlich der 8-Byte-ICMP-Kopf und 56 Bytes »Füllstoff«. Wenn »genug« Füllstoff vorhanden ist, namentlich mindestens die Größe einer struct timeval in C (acht Bytes oder so), dann verwendet ping den Anfang des Füllstoffs für einen Zeitstempel, um die Paketlaufzeit zu berechnen.

Im nächsten Schritt sollten Sie die Schnittstelle Ihrer Netzwerkkarte »anpingen«. Die Ausgabe sollte hier im wesentlichen so aussehen wie beim Ping auf die Loopback-Schnittstelle.

Wenn Sie hier angekommen sind, ohne dass Fehlermeldungen auftreten, legt das den Schluss nahe, dass die elementaren Netzfunktionen des Rechners funktionieren. Die verbleibenden Möglichkeiten für Probleme liegen entweder anderswo im Netz oder weiter oben im Protokollstapel Ihres Rechners.

Der nächste Ping geht auf das Default-Gateway (oder einen anderen Rechner im lokalen Netz). Wenn das gar nicht funktioniert, dann könnte zum Beispiel die Netzmaske falsch gesetzt sein (auch auf dem anderen Rechner!?). Ebenfalls möglich sind Hardwareprobleme, etwa ein heftiger Knick im Kabel oder ein kaputter Stecker – so etwas würde auch eine Verbindung erklären, die mal funktioniert und mal nicht.

Tabelle 24.1: Wichtige Optionen von `ping`

Option	Bedeutung
`-a`	Hörbare Pings
`-b` ⟨*Netzadresse*⟩	Broadcast-Ping
`-c` ⟨*Anzahl*⟩	Anzahl der zu sendenden Datagramme (`ping` beendet sich danach von selbst)
`-f`	»Flut-Ping«: Für jedes versandte `ECHO-REQUEST`-Datagramm wird ein Punkt ausgegeben, für jedes empfangene `ECHO-REPLY` ein Backspace-Zeichen. Das Ergebnis ist eine Reihe von Punkten, aus der Sie ersehen können, wieviele Datagramme bei der Übertragung fallen gelassen werden. Wenn nicht gleichzeitig die Option `-i` angegeben wurde, verschickt `ping` mindestens 100 Datagramme pro Sekunde (mehr, wenn das Netz mehr verarbeiten kann). Das darf allerdings nur `root`; normale Benutzer sind auf ein minimales Intervall von 0,2 Sekunden beschränkt.
`-i` ⟨*Zeit*⟩	Wartet ⟨*Zeit*⟩ Sekunden zwischen dem Verschicken zweier Datagramme. Der Standardwert ist eine Sekunde, außer bei Flut-Ping als `root`.
`-I` ⟨*Absender*⟩	Setzt die Absendeadresse für die Datagramme. ⟨*Absender*⟩ darf eine IP-Adresse sein oder der Name einer Schnittstelle (dann wird deren IP-Adresse verwendet).
`-n`	Anzeige ohne DNS-Namensauflösung
`-s` ⟨*Größe*⟩	Bestimmt die Größe des »Füllstoffs« in Byte; der Standardwert ist 56. Manchmal gibt es Probleme mit sehr großen Datagrammen, die fragmentiert werden müssen, und mit dieser Option kann `ping` so etwas diagnostizieren helfen. (Ganz früher konnte man Rechner mit riesengroßen `ping`-Datagrammen zum Absturz bringen – der berüchtigte *ping of death*.)

Die gängigen rechteckigen Stecker am Ethernet-Kabel werden mit einer Plastezunge in der Buchse fixiert. Diese Zunge bricht leicht ab, und das kann dazu führen, dass der Kontakt nicht 100% hergestellt wird.

»Freiliegende« Kabel sind empfindlich gegenüber scharfkantigen Gegenständen und mögen es auch nicht, wenn Sie (oder Ihre Kollegen) mit dem Bürostuhl über sie hinwegrollern. Einen Verdacht gegen ein Kabel können Sie durch einen prophylaktischen Austausch gegen ein als funktionierend bekanntes Exemplar oder einen Test mit einem Ethernet-Kabelprüfgerät erhärten oder widerlegen. Normalerweise gehören Kabel natürlich in einen Kabelkanal, auf die abgehängte Decke oder unter den hochgestellten Boden.

Jetzt können Sie damit fortfahren, Rechner außerhalb Ihres lokalen Netzes anzupingen. Wenn das klappt, ist das ein gutes Zeichen; wenn Sie überhaupt keine Antworten bekommen, dann haben Sie es entweder mit einem Problem bei der Wegleitung zu tun oder mit einem Firewall, der ICMP-Verkehr à la `ping` zumindest teilweise filtert (was er nicht sollte, aber manche Leute schütten das Kind halt mit dem Badewasser aus).

`ping` unterstützt eine Menge von Optionen, die die Testmöglichkeiten erweitern oder die Arbeitsweise des Programms ändern. Die wichtigsten für Diagnosezwecke sind wahrscheinlich `-f` (Flut-Ping), um intermittierende Netzwerkprobleme schnell zu überprüfen, und `-s`, um eine Größe für die Datagramme angeben zu können.

`-a` kann nützlich sein, wenn Sie unter dem Tisch herumkriechen, um ein wackliges Kabel zu finden.

`ping6` Das entsprechende Kommando zum Test von IPv6 heißt `ping6` und wird im Wesentlichen genauso aufgerufen wie `ping`. Sie müssen nur darauf achten, dass Sie die Schnittstelle angeben, die Sie meinen. Achten Sie auf das „`%eth0`“ am Ende der IPv6-Adresse:

```
$ ping6 fe80::224:feff:fee4:1aa1%eth0
PING fe80::224:feff:fee4:1aa1%eth0(fe80::224:feff:fee4:1aa1)▷
 ◁ 56 data bytes
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=1 ttl=64 time=3.65 ms
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=2 ttl=64 time=4.30 ms
◁◁◁◁◁
```

Gerade bei verbindungslokalen Adressen ist es grundsätzlich möglich, dass mehrere Schnittstellen dieselbe Adresse haben, und deswegen müssen Mehrdeutigkeiten vermieden werden. Ansonsten entsprechen die Optionen von ping6 weitestgehend denen von ping.

Übungen

24.2 [!2] Vergleichen Sie die Laufzeiten für Datagramme zwischen einem ping auf 127.0.0.1 und einem ping auf eine entfernte Station (anderer Rechner im LAN oder Standard-Gateway/DSL-Router/…).

24.3 [2] Wie lange braucht Ihr System, um im Flut-Ping-Modus eine Million Datagramme an sich selbst zu schicken?

24.4 [2] (Falls Ihr lokales Netz IPv6 unterstützt.) Prüfen Sie mit ping6 die Konnektivität zu allfälligen IPv6-Routern in Ihrem LAN (Multicast-Adresse ff02::2). Was für Antworten bekommen Sie zurück?

24.4 Wegleitung testen mit traceroute und tracepath

Wenn Sie eine Station außerhalb Ihres lokalen Netzes nicht mit ping erreichen können, könnte ein Problem mit der Wegleitung dafür verantwortlich sein. Programme wie traceroute und tracepath helfen dabei, solchen Problemen auf den Grund zu gehen.

Der typische Fall ist, dass Sie zwar alle Stationen im lokalen Netz mit ping erreichen können, aber keine jenseits davon. Die naheliegenden Verdächtigen sind dabei zum einen Ihre Default-Route, zum anderen der Rechner, auf den die Default-Route zeigt. Vergewissern Sie sich, dass die Ausgabe von route (oder »ip route list«) die richtige Default-Route liefert. Wenn ein ping auf den Default-Router funktioniert, ein ping auf einen Rechner jenseits des Default-Routers dagegen nicht, dann ist möglicherweise mit dem Default-Router etwas nicht in Ordnung. Prüfen Sie, ob ein anderer Rechner über den Default-Router hinaus Stationen erreichen kann und ob Ihr Rechner vom Default-Router aus zu erreichen ist. (Denken Sie immer auch daran, dass auf dem Default-Router möglicherweise ein Paketfilter läuft, der ICMP blockiert.)

Eine andere Sorte von Problem kann sich ergeben, wenn Sie mit dem Router, der Sie mit dem Internet verbindet, nicht direkt, sondern nur über einen anderen Router Kontakt haben. In diesem Fall kann es passieren, dass Sie zwar ping-Datagramme an den Internet-Router schicken können, dessen Antworten Sie aber nicht erreichen, weil er keine Route hat, die Verkehr in »Ihr« Netz über den dazwischenliegenden Router leitet.

traceroute ist im Prinzip eine erweiterte Form von ping. Hier wird eine entfernte Station nicht einfach auf Lebenszeichen abgefragt, sondern der Weg angezeigt, den die Datagramme im Netz nehmen. Es wird verfolgt, über welche Router ein Datagramm läuft, und wie die Verbindung zu den benutzten Routern ist.

Das ganze beruht im Gegensatz zu `ping` nicht auf ICMP, sondern (traditionell) auf UDP. `traceroute` schickt drei UDP-Datagramme an willkürliche Ports auf der Zielstation (man hofft, dass nicht auf allen drei Ports irgendein Server lauscht). Die ersten drei Datagramme haben eine TTL von 1, die nächsten drei eine TTL von 2 und so weiter. Der erste Router auf dem Weg zum Ziel verringert die TTL jeweils um 1. Für die erste Runde von Datagrammen, die von Anfang an nur die TTL 1 hatten, ist damit Schicht – sie werden verworfen und der Absender bekommt eine ICMP-`TIME-EXCEEDED`-Nachricht, die natürlich (als IP-Datagramm) auch die IP-Adresse des Routers enthält. Die zweiten drei Datagramme werden vom zweiten Router verworfen und so fort. Auf diese Wiese können Sie den genauen Weg der Datagramme zur Zielstation verfolgen. Die Zielstation selbst schickt natürlich kein `TIME-EXCEEDED`, sondern ein `PORT-UNREACHABLE`, so dass `traceroute` erkennen kann, dass es am Ziel angekommen ist.

Aussehen kann das ungefähr so:

```
$ traceroute www.linupfront.de
traceroute to www.linupfront.de (31.24.175.68), 30 hops max,▷
 ◁ 60 byte packets
 1  fritz.box (192.168.178.1)  5.959 ms  5.952 ms  5.944 ms
 2  217.0.119.34 (217.0.119.34)  28.889 ms  30.625 ms  32.575 ms
 3  87.186.202.242 (87.186.202.242)  35.163 ms  36.961 ms  38.551 ms
 4  217.239.48.134 (217.239.48.134)  41.413 ms  43.002 ms  44.908 ms
 5  xe-11-0-1.fra29.ip4.gtt.net (141.136.101.233)  46.769 ms  ▷
 ◁ 49.231 ms  51.282 ms
 6  xe-8-1-2.fra21.ip4.gtt.net (141.136.110.101)  53.412 ms ▷
 ◁ xe-0-2-3.fra21.ip4.gtt.net (89.149.129.37)  49.198 ms ▷
 ◁ xe-8-1-2.fra21.ip4.gtt.net (141.136.110.101)  52.314 ms
 7  21cloud-gw.ip4.gtt.net (77.67.76.90)  52.547 ms  30.822 ms▷
 ◁  30.018 ms
 8  s0a.linupfront.de (31.24.175.68)  38.127 ms  38.406 ms  38.402 ms
```

Die Ausgabe besteht aus mehreren durchnummerierten Zeilen. Eine Zeile entspricht jeweils einer gesendeten Gruppe von drei Datagrammen. Angezeigt werden immer die Station, von der die `TIME-EXCEEDED`-Nachrichten kamen, sowie die Laufzeit der drei Datagramme.

Sternchen in der Ausgabe bedeuten, dass für eins der Datagramme innerhalb von (normalerweise) 5 Sekunden keine Antwort gekommen ist. Sowas kommt vor.

Vielleicht wundern Sie sich, dass die Ausgabe mit `s0a.linupfront.de` endet, wo wir doch `www.linupfront.de` erreichen wollten. Das ist aber kein Problem; die Web-Präsenz `www.linupfront.de` liegt – zusammen mit ein paar anderen nützlichen Diensten – auf dem Rechner, den wir `s0a.linupfront.de` nennen, und das ist halt die Antwort, die das DNS liefert, wenn man es nach dem Namen für die Adresse `31.24.175.68` fragt.

Aus dem Umstand, dass IP-Netze Paketvermittlung verwenden, folgt theoretisch natürlich, dass die Ausgabe von `traceroute` nur eine Momentaufnahme darstellt. Wenn Sie es gleich wieder probieren, können die neuen Datagramme prinzipiell schon eine ganz andere Route zur Zielstation nehmen. Allerdings kommt das in der Praxis nicht so häufig vor.

Die traditionelle Technik mit UDP-Datagrammen funktioniert heute nicht mehr immer, da es übereifrige Firewalls gibt, die Datagramme an »unwahrscheinliche« UDP-Ports verwerfen. Mit der Option `-I` können Sie `traceroute` dazu bringen, statt UDP ICMP zu benutzen (es arbeitet dann im wesentlichen wie `ping`). Sollten Sie an einen besonders übereifrigen Firewall geraten, der auch ICMP filtert, dann können Sie mit `-T` (kurz für »`-M tcp`«) eine TCP-basierte Technik verwenden. Diese

versucht, den Port 80 auf der Zielstation anzusprechen, und bietet sich besonders an, wenn es sich bei der Zielstation um einen Web-Server handelt. (Sie können sich mit der Option `-p` einen anderen Port wünschen.)

Die »TCP-basierte Technik« baut keine komplette TCP-Verbindung zur Zielstation auf und bleibt darum für Anwendungsprogramme dort unsichtbar. `traceroute` bietet auch noch ein paar andere Methoden an.

Sie können `traceroute` mit IPv6 verwenden, indem Sie die Option `-6` angeben. Eine bequeme Abkürzung dafür ist `traceroute6`. Alles andere bleibt beim Alten. `traceroute6`

Das Programm `tracepath` macht im Grunde das gleiche wie `traceroute`, verzichtet allerdings auf fast alle trickreichen Optionen und kann auch von normalen Benutzern (ohne `root`-Rechte) aufgerufen werden. Außerdem bestimmt es die »Pfad-MTU« (hierzu gleich mehr). Hier ist eine beispielhafte Ausgabe von `tracepath`: `tracepath`

```
$ tracepath www.linupfront.de
 1?: [LOCALHOST]                          pmtu 1500
 1:  fritz.box                             13.808ms
 1:  fritz.box                              5.767ms
 2:  p5B0FFBB4.dip0.t-ipconnect.de         11.485ms pmtu 1492
 2:  217.0.119.34                          48.297ms
 3:  87.186.202.242                        46.817ms asymm  4
 4:  217.239.48.134                        48.607ms asymm  5
 5:  xe-11-0-1.fra29.ip4.gtt.net           47.635ms
 6:  xe-7-1-0.fra21.ip4.gtt.net            49.070ms asymm  5
 7:  21cloud-gw.ip4.gtt.net                48.792ms asymm  6
 8:  s0a.linupfront.de                     57.063ms reached
     Resume: pmtu 1492 hops 8 back 7
```

Genau wie `traceroute` gibt `tracepath` die Adressen der Router auf dem Weg zur Zielstation aus. Der Rest der Zeile zeigt die Laufzeit der Datagramme sowie Zusatzinformationen; »`asymm 5`« zum Beispiel bedeutet, dass die Antwort des Routers 5 Hops brauchte statt den 4 der Anfrage, aber diese Information ist nicht immer verläßlich.

Das bringt uns zum Problem der »Pfad-MTU«, das sich etwa wie folgt erklären läßt: IP erlaubt fundamental Datagramme von bis zu 65535 Bytes Länge, aber nicht jedes Zugangsverfahren kann solche Datagramme auf einen Sitz übertragen. Bei Ethernet zum Beispiel sind maximal Frames von 1518 Byte möglich, von denen noch 14 Byte für den Frame-Kopf und 4 Byte für eine Prüfsumme am Ende des Frames abgehen. Das heißt, in ein Ethernet-Frame passen höchstens 1500 Bytes an Nutzdaten, und wenn die IP-Schicht darüber ein größeres Datagramm übertragen will, muss das »fragmentiert«, also auf mehrere Frames aufgeteilt werden. Man sagt, dass die *Maximum Transmission Unit* oder MTU für Ethernet 1500 ist. Pfad-MTU

Natürlich kann die IP-Implementierung der sendenden Station nicht vorausahnen, welche Übertragungsverfahren auf dem Weg zur Zielstation im Spiel sind und ob eine Fragmentierung nötig ist (und wenn ja, wie groß die Fragmente sein dürfen). Das ergibt sich erst, wenn wirklich Daten geschickt werden. *Eigentlich* sollten Router das ja transparent behandeln – kommt am einen Ende ein Datagramm herein, das zu groß ist, um im ganzen am anderen Ende wieder hinausgeschickt zu werden, könnte der Router es fragmentieren –, aber die Hersteller von Routern drücken sich gerne um diese ressourcenintensive Tätigkeit. Statt dessen werden Datagramme typischerweise mit dem gesetzten »Don't-Fragment«-Bit im Kopf auf die Reise geschickt, das es anderen Routern verbietet, sie weiter zu zerstückeln. Kommt so ein Datagramm dann an einen Punkt, wo es zu groß für den nächsten Hop ist, schickt der betreffende Router per ICMP die Meldung »Ziel unerreichbar; Fragmentierung nötig, aber verboten; MTU wäre *n*«. In diesem Fall kann die sendende Station es nochmal mit kleineren Fragmenten versuchen. Dieses Verfahren heißt »Pfad-MTU-Bestimmung« (engl. *path MTU discovery*).

Das Ganze kann immer noch glorreich schief gehen, nämlich wenn ein übereifriger Firewall auf dem Weg den ICMP-Verkehr blockiert. In diesem Fall kommen die Fehlermeldungen über die zu große MTU nie beim Absender der Datagramme an und der weiß nicht, wie ihm geschieht. In der Praxis führt das zum Beispiel dazu, dass Webseiten nicht korrekt angezeigt werden und/oder Verbindungen »hängenbleiben«. Das Problem tritt vor allem da auf, wo ADSL à la »Deutsche Telekom« im Spiel ist, denn dort wird ein Protokoll namens »PPP over Ethernet« (PPPoE) verwendet, bei dem von den 1500 Bytes der üblichen Ethernet-MTU noch 8 Bytes für Verwaltungsinformationen abgehen. Die Probleme verschwinden normalerweise, wenn Sie die MTU für die betroffene Schnittstelle manuell auf 1492 setzen. Die Gegenstelle orientiert sich dann an diesem Wert.

In Debian GNU/Linux (und Ubuntu) können Sie die MTU für eine statisch konfigurierte Schnittstelle setzen, indem Sie eine `mtu`-Klausel in die Definition der Schnittstelle in `/etc/network/interfaces` aufnehmen:

```
iface eth0 inet static
    ◁◁◁◁
    mtu 1492
    ◁◁◁◁
```

Beim nächsten Start der Schnittstelle sollte dieser Wert dann wirksam werden.

Wenn Ihre Schnittstelle über DHCP konfiguriert wird und der DHCP-Server eine falsche MTU liefert (soll vorkommen), dann können Sie in der Datei `/etc/dhcp/dhclient.conf` im `request`-Eintrag die Klausel `interface-mtu` löschen. Dann nimmt Linux bei der nächsten DHCP-Aushandlung den Standardwert 1500 an. Einen anderen Wert können Sie explizit über

```
iface eth0 inet dhcp
    ◁◁◁◁
    post-up /sbin/ifconfig eth0 mtu 1492
    ◁◁◁◁
```

einstellen. Alternativ geht natürlich auch

```
iface eth0 inet dhcp
    ◁◁◁◁
    post-up /sbin/ip link set dev eth0 mtu 1492
    ◁◁◁◁
```

Bei den SUSE-Distributionen können Sie die MTU in der `ifcfg-`-Datei der betreffenden Netzwerkschnittstelle setzen (es gibt eine `MTU=`-Zeile). Alternativ geht das auch mit dem »`/etc/sysconfig`-Editor« des YaST, unter »Hardware/Network«. Sie müssen die Netzwerkschnittstelle anschließend neu starten (mit `ifdown`/`ifup`) oder den Rechner rebooten.

Die Red-Hat-Distributionen erlauben wie die SUSE eine MTU-Einstellung in der `ifcfg-`-Datei der betreffenden Schnittstelle. Auch hier ist ein Neustart der Schnittstelle nötig, damit die Einstellung wirksam wird.

Wenn Sie IPv6 verwenden: `tracepath6` verhält sich zu `tracepath` wie `traceroute6` zu `traceroute`.

24.5 Dienste überprüfen mit `netstat` und `nmap`

Wenn Sie einen Dienst betreiben wollen, Client-Rechner aber keinen Kontakt mit ihm bekommen, sondern mit Fehlermeldungen wie

```
Unable to connect to remote host: Connection refused
```

abgewiesen werden, sollten Sie sicherstellen, dass der Dienst tatsächlich ordnungsgemäß auf Verbindungen »lauscht«. Das können Sie zum Beispiel mit dem Programm netstat:

```
$ netstat -tul
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address       Foreign Address  State
tcp        0      0 red.example.com:www *:*              LISTEN
tcp        0      0 red.example.com:ftp *:*              LISTEN
tcp        0      0 red.example.com:ssh *:*              LISTEN
```

Die Option -l bringt netstat dazu, nur »lauschende« Programme aufzuführen. Mit den Optionen -t und -u können Sie die Ausgabe auf TCP- bzw. UDP-basierte Dienste beschränken.

In der Ausgabe bedeuten die Spalten das Folgende:

Proto Das Protokoll (tcp, udp, raw, …), das von dem Socket benutzt wird.

Recv-Q Die Anzahl der empfangenen, aber (noch) nicht vom Anwenderprogramm bei diesem Socket abgeholten Bytes.

Send-Q Die Anzahl der versendeten Bytes, die von der Gegenstelle noch nicht bestätigt wurden.

Local Address Lokale Adresse und Portnummer des Sockets. Ein Stern (»*«) an dieser Stelle steht bei »lauschenden« Sockets dafür, dass sie auf allen verfügbaren Adressen lauschen, also zum Beispiel 127.0.0.1 *und* der IP-Adresse der Ethernet-Karte.

Foreign Address Die Adresse und Portnummer der Gegenstelle auf dem entfernten Rechner.

State Zustand des Sockets. raw-Sockets haben keine Zustände und udp-Sockets normalerweise auch nicht, in welchem Fall dieses Feld leer bleibt. Ansonsten, insbesondere für tcp-Sockets, sind u. a. folgende Einträge für den Zustand möglich:

- **ESTABLISHED** Eine Verbindung ist etabliert.
- **SYN_SENT** Das Socket versucht, eine Verbindung aufzubauen, und hat das erste Paket des Drei-Wege-Handshakes verschickt, aber noch keine Antwort bekommen.
- **SYN_RECV** Das Socket (ein »lauschendes«) hat eine Verbindungsanfrage bekommen und bestätigt.
- **FIN_WAIT1** Das Socket ist geschlossen, die Verbindung wird gerade abgebaut.
- **FIN_WAIT2** Die Verbindung ist abgebrochen und das Socket wartet auf die Beendigung durch die Gegenstelle.
- **TIME_WAIT** Nach Abbau der Verbindung wartet das Socket darauf, noch im Netzwerk verbliebene Pakete zu bearbeiten.
- **CLOSE** Das Socket wird nicht verwendet.
- **CLOSE_WAIT** Die Gegenstelle hat sich abgemeldet und wartet darauf, dass das Socket geschlossen wird.
- **LISTEN** Das Socket »lauscht« auf eingehende Verbindungen. Solche Sockets werden nur angezeigt, wenn Sie die Optionen -l oder -a angegeben haben.


Ohne `-t` und `-u` liefert `netstat` außer Informationen über TCP- und UDP-Dienste noch Daten über aktive Unix-Domain-Sockets. Diese sind allerdings größtenteils uninteressant.


Wenn Sie die Option `-l` weglassen, bekommen Sie statt dessen eine Liste der aktiven Netzwerkverbindungen (sowohl diejenigen, bei denen Ihr Rechner als Server fungiert, als auch diejenigen, bei denen er als Client auftritt).

Wenn Ihr Dienst in der Ausgabe von »`netstat -tul`« nicht auftaucht, ist das ein Indiz dafür, dass das betreffende Programm nicht läuft. Steht der Dienst in der Liste, könnte es einerseits sein, dass Clients durch eine Firewall-Konfiguration abgewiesen werden, noch bevor sie ihn überhaupt erreichen. Andererseits wäre es möglich, dass der betreffende Port durch ein anderes Programm blockiert wird, das aus irgendwelchen Gründen nicht richtig funktioniert. In diesem Fall können Sie sich mit »`netstat -tulp`« auch noch die Prozess-ID und den Namen des Programms anzeigen lassen, das den Port bedient. Dazu müssen Sie allerdings `root`-Privilegien haben.

`netstat` setzt voraus, dass Sie zumindest Shell-Zugang, wenn nicht gar `root`-Rechte auf dem Rechner haben, wo Sie das Kommando ausführen wollen. Wie sieht es aber damit aus, »von außen« zu testen, welche Ports auf einem Rechner zugänglich sind? Auch dafür gibt es Lösungen. Das Programm `nmap` ist ein **Portscanner**, der über das Netz prüft, welche TCP- oder UDP-Ports auf einem Rechner offen sind, welche gesperrt und welche unbenutzt. Natürlich kann der »Rechner« auch eine Firewall-Infrastruktur sein, so dass `nmap` Ihnen dabei helfen kann, Lücken in Ihrer Sicherheitsstrategie aufzudecken.

Portscanner


`nmap` ist nicht automatisch Bestandteil einer Linux-Installation. Sie müssen es wahrscheinlich manuell nachinstallieren.


Das Scannen von Rechnern, die sich nicht in Ihrem unmittelbaren juristischen Einflußgebiet befinden, kann strafbar sein! (Tatsächlich könnte man Ihnen vermutlich schon aus dem *Besitz* von »Hacker-Werkzeugen« wie `nmap` einen Strick drehen, wenn Sie Pech haben und/oder sich ungeschickt anstellen.) Beschränken Sie sich also auf Rechner, wo definitiv klar ist, dass Sie `nmap` anwenden dürfen. Lassen Sie es sich zur Sicherheit schriftlich von Ihrem Auftraggeber oder hinreichend hohen Vorgesetzten erlauben.

Im einfachsten Fall übergeben Sie `nmap` den Namen oder die Adresse des zu untersuchenden Rechners (seien Sie vorbereitet auf eine gewisse Wartezeit):

```
# nmap blue.example.com

Starting Nmap 4.68 ( http://nmap.org ) at 2009-02-04 00:09 CET
Interesting ports on blue.example.com (172.16.79.2):
Not shown: 1710 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
443/tcp  open  https
MAC Address: 00:50:56:FE:05:04 (VMWare)

Nmap done: 1 IP address (1 host up) scanned in 9.751 seconds
```

`nmap` betrachtet Ports als *open*, wenn sich dort ein Dienst meldet. Ports, für die der untersuchte Rechner eine Fehlermeldung ausgibt, gelten als *closed*, während Ports, auf denen überhaupt keine Reaktion festzustellen ist (etwa weil die Anfragepakete vom untersuchten Rechner oder einem Firewall einfach weggeworfen und nicht einmal mit einer Fehlermeldung beantwortet werden), als *filtered* bezeichnet werden.


Wenn Sie nichts anderes angeben, analysiert nmap die TCP-Ports des Zielrechners mit einem »SYN Scan«. Dabei schickt das Programm an jeden der betreffenden Ports ein TCP-Segment, das das SYN-Flag gesetzt hat (so als ob es eine neue Verbindung aufbauen wollte). Antwortet der Zielrechner mit einem TCP-Segment, das die SYN- und ACK-Flags gesetzt hat, so nimmt nmap an, dass der Port benutzt wird. Es unternimmt aber weiter nichts (insbesondere bestätigt es dieses Segment nicht), so dass die »halboffene« Verbindung nach Ablauf der vorgeschriebenen Fristen vom Zielrechner verworfen wird. Antwortet der Zielrechner statt dessen mit einem Segment, das das RST-Flag gesetzt hat, ist der Port *closed*. Kommt nach einigen Versuchen immer noch keine Antwort oder nur ICMP-Unerreichbarkeitsmeldungen, dann wird der Port auf *filtered* gesetzt. – Für SYN Scans sind root-Rechte erforderlich.


Andere Techniken, die nmap anbietet, sind der »TCP Connect Scan« (der keine besonderen Privilegien benötigt, aber plump und von der Gegenseite leicht zu erkennen ist), der »UDP Scan« und verschiedene andere Varianten von TCP-basierten Scans, etwa um die Regelsätze von Firewalls auszuspähen. Konsultieren Sie die Dokumentation in nmap(1).


nmap kann nicht nur bestimmen, welche Ports auf einem Rechner aktiv sind, sondern in vielen Fällen sogar sagen, welche Software sich dahinter verbirgt. Dazu müssen Sie die Option -A angeben und *richtig* Geduld mitbringen. nmap verläßt sich dafür auf eine mitgelieferte Datenbank von »Signaturen« verschiedener Programme.


Die Möglichkeiten von nmap übersteigen bei weitem das, was wir in dieser Schulungsunterlage erklären können. Lesen Sie die Dokumentation (in nmap(1)) und denken Sie beim Testen immer an die oben gemachte juristische Einschränkung.

24.6 DNS testen mit host und dig

Wenn Verbindungen zu namentlich angesprochenen Rechnern ungebührlich lange dauern oder nach einer Wartezeit gar nicht zustande kommen, es über die IP-Adresse aber üblich flott geht, dann ist vermutlich das DNS schuld. Ebenso kann es sein, dass Ihr Rechner lange braucht, um eine Verbindung aufzubauen, weil die Gegenstelle versucht, einen zu *Ihrer* IP-Adresse passenden Namen zu finden und das aus irgendwelchen Gründen schief geht. Um das DNS zu testen, können Sie zum Beispiel die Programme host und dig verwenden.


»Und was ist mit nslookup?« hören wir Sie sagen. Sorry, aber nslookup ist seit einer Weile verpönt und wird nur noch aus Freundlichkeit unterstützt.

host ist ein sehr einfaches Programm, das im simpelsten Fall einen DNS-Namen entgegennimmt und die dazugehörige(n) IP-Adresse(n) ausgibt:

```
$ host www.linupfront.de
www.linupfront.de is an alias for s0a.linupfront.de.
s0a.linupfront.de has address 31.24.175.68
```

Es funktioniert auch umgekehrt:

```
$ host 193.99.144.85
85.144.99.193.in-addr.arpa domain name pointer www.heise.de
```

(Fragen Sie nicht.)

Sie können die Ausgabe mehrerer DNS-Server miteinander vergleichen, indem Sie die IP-Adresse (oder den Namen, aber die Adresse ist sicherer) eines DNS-Servers bei der Anfrage mit angeben:

```
$ host www.linupfront.de 127.0.0.1
Using domain server:
Name: 127.0.0.1
Address: 127.0.0.1#53
Aliases:

www.linupfront.de is an alias for s0a.linupfront.de.
s0a.linupfront.de has address 31.24.175.68
```

Auf diese Weise können Sie prüfen, ob ein DNS-Server die richtigen Antworten liefert.

Sie können bestimmte Arten von DNS-Datensätzen anfordern, indem Sie die Option `-t` verwenden, etwa wie

```
$ host -t mx linupfront.de                          MX-Datensatz gewünscht
linupfront.de mail is handled by 10 s0a.linupfront.de
```

Mit `-l` bekommen Sie eine Liste der wichtigsten Namen in einer Domain – jedenfalls wenn Sie dürfen. Zusammen mit der Option `-a` können Sie eine Liste *aller* Namen bekommen.

Das Programm `dig` tut im wesentlichen dasselbe wie `host`, aber erlaubt eine detailliertere Diagnose von Problemen. Es liefert eine ausführlichere Ausgabe als `host`:

```
$ dig www.linupfront.de

; <<>> DiG 9.9.5-10-Debian <<>> www.linupfront.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1443
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.linupfront.de.             IN      A

;; ANSWER SECTION:
www.linupfront.de.      3600    IN      CNAME   s0a.linupfront.de.
s0a.linupfront.de.      3600    IN      A       31.24.175.68

;; Query time: 51 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul 22 18:00:34 CEST 2015
;; MSG SIZE  rcvd: 69
```

Zur Rückwärtsauflösung von IP-Adressen in Namen müssen Sie die Option `-x` angeben:

```
$ dig -x 31.24.175.68

; <<>> DiG 9.9.5-10-Debian <<>> -x 31.24.175.68
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 63823
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
```

```
;68.175.24.31.in-addr.arpa.     IN      PTR

;; ANSWER SECTION:
68.175.24.31.in-addr.arpa. 86400 IN     PTR     s0a.linupfront.de.

;; Query time: 50 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul 22 18:01:31 CEST 2015
;; MSG SIZE  rcvd: 74
```

Um einen bestimmten DNS-Server zu befragen, geben Sie dessen Adresse hinter einem `@` an:

```
$ dig www.linupfront.de @192.168.20.254
```

Einen DNS-Datensatztyp können Sie einfach hinter dem gesuchten Namen angeben:

```
$ dig linupfront.de mx

; <<>> DiG 9.9.5-10-Debian <<>> linupfront.de mx
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15641
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;linupfront.de.                 IN      MX

;; ANSWER SECTION:
linupfront.de.          3600    IN      MX     10 s0a.linupfront.de.

;; Query time: 49 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Wed Jul 22 17:59:36 CEST 2015
;; MSG SIZE  rcvd: 51
```

Prinzipiell können Sie auch das Kommando `getent` verwenden, um die Namensauflösung zu testen: `getent`

```
$ getent hosts www.linupfront.de
31.24.175.68    s0a.linupfront.de www.linupfront.de
```

Der Unterschied zwischen `host` und `dig` auf der einen und `getent` auf der anderen Seite ist, dass die ersteren das DNS direkt befragen. Das letztere Kommando befragt dagegen die C-Bibliothek. Das bedeutet einerseits, dass die Abfragereihenfolge eingehalten wird, die in `/etc/nsswitch.conf` vorgegeben wurde. Andererseits bekommen Sie die Antwort in der Form, wie sie auch in `/etc/hosts` auftauchen würde.

In `/etc/nsswitch.conf` steht für gewöhnlich etwas wie

```
hosts: files dns
```

Das heißt, es wird erst in `/etc/hosts` nachgeschaut und dann im DNS. Der Vorteil ist, dass Sie auf diese Weise genau das sehen, was Anwendungsprogramme sehen, die die C-Bibliothek verwenden. Es könnte zum Beispiel sein, dass aus irgendwelchen Gründen für einen Namen eine Definition in `/etc/hosts` existiert, die dann Vorrang gegenüber dem DNS hat (weil nach einem `/etc/hosts`-Treffer das DNS nicht mehr konsultiert wird).

Von anderen Anwendungen von `getent` sind Sie es vielleicht gewöhnt, dass ein einfaches

```
$ getent passwd
```

Ihnen eine Liste aller dem System bekannten Benutzer im Format von `/etc/passwd` ausspuckt, selbst wenn die Benutzer gar nicht alle in der lokalen Kennwortdatei stehen. Für Benutzer kann das klappen, aber muss nicht (wenn Sie in einem großen Unternehmen arbeiten, haben die Administratoren Ihrer Benutzerdatenbank dem möglicherweise einen Riegel vorgeschoben). Für das DNS führt ein

```
$ getent hosts
```

aber *definitiv* nicht dazu, dass alle weltweit im DNS definierten Namen aufgelistet werden. (Ist wahrscheinlich auch besser so.)

DNS ist ein sehr vielschichtiges Thema, bei dem auch vieles schief gehen kann. Allerdings erfordert die detaillierte Diagnose von DNS-Problemen einiges an Vorkenntnissen. Das Thema DNS behandelt im Detail die Linup-Front-Schulungsunterlage *Das Domain Name System*.

24.7 Andere nützliche Diagnosewerkzeuge

24.7.1 `telnet` und `netcat`

Das `telnet`-Kommando wird verwendet, um sich über das TELNET-Protokoll interaktiv auf einem anderen Rechner anzumelden oder – allgemein – mit einem TCP-Port Kontakt aufzunehmen. Als Dienst für den Fernzugriff sollte TELNET nicht mehr verwendet werden, da keine starke Authentisierung verwendet wird und die Datenübertragung unverschlüsselt stattfindet. Die Secure Shell (`ssh`, Kapitel 25) ist eine sinnvolle Alternative.

Das Client-Programm `telnet` ist aber hervorragend zum Testen vieler anderer Dienste geeignet. Mit »`telnet` ⟨*Adresse*⟩ ⟨*Dienst*⟩« kann eine Verbindung zu jedem beliebigen Port aufgebeut werden (»⟨*Dienst*⟩« ist entweder eine Portnummer oder ein Dienstname aus `/etc/services`). So baut »`telnet 192.168.0.100 80`« eine Verbindung zu einem Web-Server auf. In diesem Fall wäre es sogar möglich, mit den entsprechenden HTTP-Befehlen Dokumente vom Server aufzurufen. Ein anderes Beispiel:

```
$ telnet 192.168.0.1 22
Trying 192.168.0.1...
Connected to 192.168.0.1.
Escape character is ']^'.
SSH-2.0-OpenSSH_6.7p1 Debian-6
```

Im Beispiel wird eine Verbindung zum SSH-Port auf einem anderen Rechner aufgebaut, der entfernte `sshd` meldet sich mit Protokoll- und Programm-Version.

Mit dem *escape character* können Sie eine »Auszeit« von der TCP-Verbindung nehmen und `telnet`-Kommandos eingeben. Die interessantesten Kommandos sind wahrscheinlich `close` (bricht die Verbindung ab), `status` (zeigt den Verbindungsstatus an) und `!` (kann verwendet werden, um auf dem lokalen Rechner bei laufender Verbindung Kommandos auszuführen):

```
$ telnet 192.168.0.1 22
Trying 192.168.0.1...
Connected to 192.168.0.1.
```

```
Escape character is ']^'.
SSH-2.0-OpenSSH_6.7p1 Debian-6
Strg+Esc
telnet> status
Connected to 192.168.0.1.
Operating in obsolete linemode
Local character echo
Escape character is ']^'.
_
```

Möglicherweise ist bei Ihrem `telnet` das »!«-Kommando herauskonfiguriert. Dann können Sie immer noch mit dem Kommando `z` das `telnet`-Programm in den Hintergrund schicken (denken Sie »Jobkontrolle der Shell«) und später mit `fg` wieder reaktivieren.

Eine Alternative zum TELNET-Client `telnet` ist das Programm `netcat`. Im einfachsten Fall benimmt `netcat` sich wie `telnet` (ist aber nicht ganz so geschwätzig):

```
$ netcat 192.168.0.1 22
SSH-2.0-OpenSSH_6.7p1 Debian-6
```

Oft heißt das Kommando statt (oder zusätzlich zu) `netcat` auch `nc`. Der Rest bleibt allerdings derselbe.

Von `netcat` sind zwei Versionen im Umlauf, eine »traditionelle« (von einem Menschen namens »Hobbit«) und eine aus dem OpenBSD-System. Letztere kann deutlich mehr (etwa IPv6 oder Unix-Domain-Sockets). Wir setzen für den Rest des Abschnitts das OpenBSD-`netcat` voraus.

Bei Debian GNU/Linux ist das Standard-`netcat` die traditionelle Version (aus dem Paket `netcat-traditional`). Wenn Sie die aufgebohrte Version haben möchten, müssen Sie das Paket `netcat-openbsd` installieren. Das OpenBSD-`netcat` installiert sich *nur* unter dem Namen `nc`; die traditionelle Version bleibt unter dem Namen `netcat` erhalten, solange Sie das Paket nicht deinstallieren.

Neben der Client-Seite einer TCP-Verbindung implementiert `netcat` auf Wunsch auch die Server-Seite (natürlich ohne direkt viel Nützliches zu tun). Zum Beispiel: Server-Seite
Mit

```
$ nc -l 4711
```

lauscht `netcat` auf dem Port 4711 auf Verbindungen. In einem anderen Fenster können Sie dann mit

```
$ nc localhost 4711
```

Verbindung zu Ihrem »Server« aufnehmen. Was Sie auf der Client-Seite eintippen, erscheint beim Server und umgekehrt. Dateitransfer »für Arme« geht so: Tippen Dateitransfer »für Arme«
Sie auf dem Zielrechner

```
$ nc -l 4711 >myfile
```

und auf dem Ausgangsrechner

```
$ nc red.example.com 4711 <myfile
```

24.7.2 tcpdump

Netzwerk-Sniffer

Bei `tcpdump` handelt es sich um einen Netzwerk-Sniffer, der die Analyse der über ein Netzwerkinterface laufenden Netzwerkpakete ermöglicht. Die Netzwerkkarte wird dabei in den sog. **Promiscuous Mode** versetzt, in dem alle Pakete, also nicht nur wie üblich die für das lokale Interface bestimmten, gelesen werden. Das Kommando kann deshalb nur als Benutzer `root` verwendet werden.

Promiscuous Mode

Ein Beispiel für die Verwendung:

```
# tcpdump -ni eth0
tcpdump: listening on eth0
14:26:37.292993 arp who-has 192.168.0.100 tell 192.168.0.1
14:26:37.293281 arp reply 192.168.0.100 is-at 00:A0:24:56:E3:75
14:26:37.293311 192.168.0.1.35993 > 192.168.0.100.21: S 140265170:
140265170(0) ...
14:26:37.293617 192.168.0.100.21 > 192.168.0.1.35993: S 135130228:
135130228(0) ack 140265171 ...
14:26:37.293722 192.168.0.1.35993 > 192.168.0.100.21: . ack 1 ...
                                                    Abbruch des Programms
5 packets received by filter
0 packets dropped by kernel
```

Im Beispiel wurde ein Verbindungsaufbau zu einem FTP-Server »belauscht«. Die Parameter »`-ni eth0`« schalten die Namensauflösung und die Auflösung der Portnummern ab und involvieren nur das Interface `eth0`. Zu jedem Paket werden die genaue Uhrzeit, Quell- und Zielrechner, gesetzte Flags im TCP-Header (S: `SYN`-Bit), die Folgenummer der Daten, ein eventuell gesetztes `ACK`-Bit, die erwartete Folgenummer des nächsten Pakets und diverse weitere Informationen angezeigt.

Das erste gezeigte Paket enthält keinen Zielrechner, es handelt es sich um eine ARP-Anfrage: Der Rechner `192.168.0.100` soll seine MAC-Adresse bekannt geben – was er im zweiten Paket dann auch tut. Die nächsten Pakete zeigen einen typischen TCP-Drei-Wege-Handshake.

24.7.3 wireshark

Bei `wireshark` handelt es sich bei wie bei `tcpdump` um einen Netzwerksniffer. Allerdings besitzt `wireshark` einen ungleich größeren Funktionsumfang. Es handelt sich um ein grafisches Programm, das die genaue Analyse aller Netzwerkpakete ermöglicht. Die Anzeige besteht aus drei Fenstern: Im oberen werden die eingelaufenen Pakete angezeigt, im unteren wird der Paketinhalt hexadezimal aufgeschlüsselt, und im mittleren Fenster können die Kopfinformationen (und Nutzdaten) jeder Protokollschicht bis aufs letzte Bit und ungemein komfortabel analysiert werden.

Ähnlich wie `nmap` ist `wireshark` kein Standard-Unix-Werkzeug und muss oft nachinstalliert werden. Sowohl `tcpdump` als auch `wireshark` sollten mit Bedacht angewandt werden, da auch im lokalen Netz schnell gegen geltendes Recht verstoßen werden kann. Schließlich werden eventuell Daten angezeigt, die einen nichts angehen.

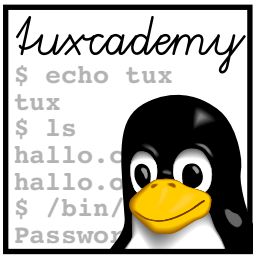
Das Programm `wireshark` hieß bis vor einigen Jahren `ethereal` und ist auf älteren Rechnern möglicherweise noch unter diesem Namen zu finden.

Kommandos in diesem Kapitel

dig Sucht Informationen im DNS (sehr komfortabel) dig(1) 416
getent Ruft Einträge aus administrativen Datenbanken ab getent(1) 417
host Sucht Informationen im DNS host(1) 415
netcat Sehr allgemeines Netzwerk-Client-Programm nc(8) 419
nmap Netzwerk-Portscanner, analysiert offene Ports auf Rechnern nmap(1) 414
ping Prüft grundlegende Netzwerk-Konnektivität mit ICMP ping(8) 407
ping6 Prüft grundlegende Netzwerk-Konnektivität (für IPv6) ping(8) 408
tcpdump Netzwerk-Sniffer, protokolliert und analysiert Netzwerkverkehr tcpdump(1) 419
telnet Erlaubt Verbindungen zu beliebigen TCP-Diensten, insbesondere TELNET (Fernzugriff) telnet(1) 418
tracepath Prüft die Wegleitung zu einer anderen Station, mit Pfad-MTU tracepath(8) 411
tracepath6 Entspricht tracepath, aber für IPv6 tracepath(8) 412
traceroute Prüft die Wegleitung zu einer anderen Station traceroute(8) 409
wireshark Paket-Sniffer, liest und analysiert Netzwerkverkehr (Ex-ethereal) wireshark(1) 420

Zusammenfassung

- Mit Programmen wie netstat, telnet, nmap, tcpdump oder wireshark stehen leistungsfähige Werkzeuge zur Diagnose von Fehlern bei Netzwerkdiensten zur Verfügung.



25

Die Secure Shell

Inhalt

25.1 Einführung 424
25.2 Anmelden auf entfernten Rechnern mit ssh 424
25.3 Andere nützliche Anwendungen: scp und sftp 428
25.4 Client-Authentisierung über Schlüsselpaare 428
25.5 Portweiterleitung über SSH 431
25.5.1 X11-Weiterleitung 431
25.5.2 Beliebige TCP-Ports weiterleiten 432

Lernziele

- Die Secure Shell (SSH) anwenden und konfigurieren können

Vorkenntnisse

- TCP/IP-Kenntnisse (Kapitel 22)
- Kenntnisse über Linux-Netzkonfiguration (Kapitel 23)
- Kenntnisse über Linux-Systemkonfiguration
- Grundkenntnisse über Kryptografie sind hilfreich

25.1 Einführung

SSH (*Secure Shell*) ist ein Netzwerkprotokoll der TCP/IP-Familie. Es ermöglicht die Datenübertragung im Netz mit sicherer Authentisierung und Verschlüsselung. Zu seinen Anwendungen gehören interaktive Anmeldevorgänge, Übertragung von Dateien und die gesicherte Weiterleitung anderer Protokolle (engl. *tunneling*).

Verschlüsselung ist wichtig dafür, dass Unbefugte, die den Netzverkehr belauschen, keine Kenntnis von den übertragenen Inhalten nehmen können. Authentisierung stellt einerseits sicher, dass Sie als Benutzer mit dem richtigen Server kommunizieren, und andererseits, dass der Server Ihnen Zugriff auf das richtige Benutzerkonto einräumt.

OpenSSH

OpenSSH, das mit den meisten Linux-Distributionen ausgeliefert wird, stellt eine frei verfügbare Implementierung dieses Protokolls dar. Diese Implementierung enthält einige SSH-Clients sowie einen SSH-Server (`sshd`).

Angriffe

Richtig angewendet kann SSH die folgenden Angriffe verhindern:

- »DNS Spoofing«, also ge- oder verfälschte DNS-Einträge
- »IP Spoofing«, bei dem ein Angreifer von einem Rechner aus Pakete schickt, die so tun, als ob sie von einem anderen (vertrauenswürdigen) Rechner kämen.
- IP Source Routing, bei dem ein Rechner vortäuschen kann, dass Pakete von einem anderen (vertrauenswürdigen) Rechner kommen.
- Ausspähen von Klartextkennwörtern und Nutzdaten durch Stationen entlang des Übertragungswegs.
- Manipulation von übertragenen Daten durch Stationen entlang des Übertragungswegs.
- Angriffe auf den X11-Server durch ausgespähte X-Authentisierungsdaten und vorgetäuschte Verbindungen zum X11-Server.

SSH hilft nicht gegen andere Angriffe auf Ihren Rechner oder den SSH-Server. Insbesondere kann ein Angreifer, der auf einem der beiden Rechner auf andere Weise Administratorrechte erlangt hat, auch SSH kompromittieren.

Einsatzmöglichkeiten

SSH ersetzt die unsicheren Protokolle TELNET, RLOGIN und RSH für interaktive Anmeldevorgänge. Zusätzlich bietet es die Möglichkeit, Dateien von einem entfernten Rechner zu kopieren und ist so ein sicherer Ersatz für RCP und viele Anwendungen von FTP.

Protokollversionen

Das SSH-Protokoll existiert in zwei Versionen, 1 und 2. Viele Clients unterstützen beide Versionen und die meisten Server können Verbindungen mit beiden Versionen entgegennehmen. Machen Sie trotzdem einen Bogen um die Version 1, die diverse Sicherheitslücken aufweist.

25.2 Anmelden auf entfernten Rechnern mit `ssh`

Um sich über SSH auf einem entfernten Rechner anzumelden, müssen Sie das Kommando `ssh` aufrufen, etwa so:

```
$ ssh blue.example.com
hugo@blue.example.com's password: geHe1m
Last login: Mon Feb  2 10:05:25 2009 from 192.168.33.1
Debian GNU/Linux (etch/i686) blue.example.com
hugo@blue:~$ _
```

Dabei geht ssh davon aus, dass Ihr Benutzerkonto auf dem entfernten Rechner genauso heißt wie auf dem lokalen Rechner. Wenn das nicht so ist, können Sie Ihren entfernten Benutzernamen wie folgt angeben:

```
$ ssh hschulz@blue.example.com
```

Hinter den Kulissen passiert beim Verbindungsaufbau ungefähr das Folgende:

- Client und Server senden einander Informationen über ihre Rechnerschlüssel, unterstützte kryptografische Verfahren und ähnliches. Der Client prüft, ob der öffentliche Schlüssel noch so ist wie früher (dazu gleich mehr) und handelt mit dem Server ein gemeinsames Geheimnis aus, das dann als (symmetrischer) Schlüssel für die Verschlüsselung der Verbindung dient. Im selben Schritt prüft der Client die Authentizität des Servers und bricht die Verbindung ab, falls daran Zweifel bestehen. Die (unappetitlichen) Details stehen in [RFC4253].
- Der Server prüft die Authentizität des Clients über eines von mehreren Verfahren (hier im Beispiel fragt er nach einem Kennwort). Das Kennwort wird bereits über die verschlüsselte Verbindung übertragen und kann im Gegensatz zu Protokollen wie FTP oder TELNET von »Mithörern« nicht abgefangen werden.

Der erste Schritt ist dabei durchaus entscheidend. Das folgende Beispiel zeigt, was passiert, wenn Sie zum ersten Mal mit dem entfernten Rechner Kontakt aufnehmen:

```
$ ssh blue.example.com
The authenticity of host 'blue.example.com (192.168.33.2)' can't be▷
 ◁ established.
RSA key fingerprint is 81:24:bf:3b:29:b8:f9:f3:46:57:18:1b:e8:40:5a▷
 ◁ :09.
Are you sure you want to continue connecting (yes/no)? _
```

Der Rechner blue.example.com ist hier noch unbekannt, und ssh bittet Sie darum, seinen Rechnerschlüssel zu verifizieren. *Das ist ernst gemeint.* Wenn Sie diesen Überprüfungsschritt überspringen, verlieren Sie die Garantie dafür, dass niemand Ihre Verbindung belauscht.

Die Gefahr ist hier, dass jemand Ihren Verbindungswunsch abfängt und so tut, als wäre er blue.example.com. Hinter den Kulissen kann er selbst eine Verbindung zu blue.example.com aufbauen, über die er alles schickt, was Sie ihm naiverweise schicken, und Ihnen die Antworten von blue weiterleiten. Sie sehen keinen Unterschied, aber der Angreifer kann alles lesen, was Sie übertragen. Man nennt das einen *Man-in-the-middle*-Angriff.

Zur Überprüfung kontaktieren Sie den Administrator des entfernten Rechners (zum Beispiel per Telefon) und bitten ihn, Ihnen den »Fingerabdruck« (engl. *fingerprint*) des öffentlichen Rechnerschlüssels vorzulesen. Dieser kann mit »ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key« ermittelt werden und muss mit dem »RSA key fingerprint« aus dem SSH-Anmeldedialog übereinstimmen.

Die SSH-Schlüsselpaare eines Rechners finden Sie in den Dateien ssh_host_*x*_key bzw. ssh_host_*x*_key.pub im Verzeichnis /etc/ssh. *x* steht dabei für ein bestimmtes kryptografisches Verfahren, das Clients verwenden können, um die Authentizitität des Servers zu überprüfen.

SSH-Schlüsselpaare

Mögliche Werte für *x* sind (Juli 2015):

`rsa` Das RSA-Verfahren. Ist (nach dem Stand der Forschung) sicher, solange Sie Schlüssel verwenden, die länger als 1024 Bit sind. (2048 Bit klingen gut. Benutzen Sie 4096 Bit, wenn Sie Edward Snowden sind oder anderweitig davon ausgehen, dass Organisationen wie die NSA Sie gezielt – und nicht nur nebenbei zufällig – auf dem Kieker haben.)

`dsa` Das DSA-Verfahren. Erlaubt nur 1024-Bit-Schlüssel und sollte heutzutage vermieden werden, auch weil es anfällig gegen Schwächen bei der Zufallszahlen-Erzeugung ist.

`ecdsa` Das DSA-Verfahren auf der Basis elliptischer Kurven. Hier haben Sie die Auswahl zwischen 256, 384 und 521 Bits[1]. (Elliptische Kurven brauchen nicht so viele Bits, darum sind die niedrigen Zahlen unproblematisch.)

`ed25519` Ein schnelles und (nach bisheriger Kenntnis) sehr sicheres Verfahren, das auf Daniel J. Bernstein zurückgeht. In der Secure Shell allerdings noch ziemlich neu.

Mit 2048-Bit-RSA machen Sie für die nächsten paar Jahre mit ziemlicher Sicherheit nichts falsch. Wenn Sie sicher sind, dass Ihre Clients und Server Ed25519 unterstützen, ist das eine empfehlenswerte Alternative.

Ein »Schlüsselpaar«, nur um das mal gesagt zu haben, ist übrigens ein Paar von zwei zusammengehörenden Schlüsseln (!), einem privaten und einem öffentlichen Schlüssel. Der öffentliche Schlüssel darf überall herumerzählt werden, solange der private Schlüssel vertraulich bleibt. Was mit dem öffentlichen Schlüssel verschlüsselt wurde, kann *nur* mit dem privaten Schlüssel aus demselben Paar wieder entschlüsselt werden und umgekehrt.

Ist der öffentliche Schlüssel des entfernten Rechners authentisch, dann beantworten Sie die Frage mit »`yes`«. `ssh` speichert den öffentlichen Schlüssel dann in der Datei `~/.ssh/known_hosts` ab, um ihn bei späteren Verbindungsaufbauten als Vergleichsgrundlage heranzuziehen.

Sollten Sie beim Aufbauen einer `ssh`-Verbindung jemals eine Meldung sehen wie

```
$ ssh blue.example.com
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle▷
 ◁ attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
38:fa:2e:d3:c7:c1:0f:26:2e:59:e8:16:a4:0a:0b:94.
Please contact your system administrator.
Add correct host key in /home/hugo/.ssh/known_hosts to get rid of▷
 ◁ this message.
Offending key in /home/hugo/.ssh/known_hosts:4
RSA host key for blue.example.com has changed and you have requested▷
 ◁ strict checking.
Host key verification failed.
```

dann besteht der Verdacht, dass Sie gerade einem *Man-in-the-middle*-Angriff ausgesetzt sind – der öffentliche Rechnerschlüssel, den der Server präsentiert, stimmt nicht mit dem überein, der für den Server in der `known_hosts`-Datei abgelegt ist. Sie sollten sich an den Systemadministrator wenden und nachfragen, was Sache ist – vielleicht musste der Rechnerschlüssel aus anderen Gründen geändert werden.

[1] Ja, wirklich 521, das ist kein Tippfehler für 512. ($2^{521}-1$ ist eine Mersenne-Primzahl, und das macht die Implementierung schneller. 521 Bits sind allerdings ziemlicher Overkill.)

Dieses Verhalten können Sie ändern, indem Sie eine entsprechende Einstellung in der Datei ~/.ssh/config machen:

`StrictHostKeyChecking ask`	*Voreinstellung*
`StrictHostKeyChecking no`	*Alles immer akzeptieren*
`StrictHostKeyChecking yes`	*Nichts Neues akzeptieren*

Bei »`StrictHostKeyChecking yes`« können Sie nur Verbindungen zu Rechnern aufbauen, die schon in Ihrer `known_hosts`-Datei stehen. Alle anderen werden abgewiesen.

Nachdem Sie über `ssh` eine Verbindung aufgebaut haben, können Sie an dem entfernten Rechner so arbeiten, als säßen Sie davor. Sie können die Verbindung mit `exit` oder Strg+d beenden.

Wenn Sie nichts anderes sagen, gilt bei interaktiven `ssh`-Sitzungen eine Tilde (»~«), wenn ihr in der Eingabe unmittelbar ein Zeilentrenner vorausgeht, als Sonderzeichen, mit dem Sie die `ssh` steuern können. Insbesondere bricht die Kombination »~.« die Verbindung ab, was nützlich sein kann, wenn sich am »anderen Ende« ein Programm aufgehängt hat. Sie können auch noch andere interessante Dinge tun – schauen Sie sich den Abschnitt »ESCAPE CHARACTERS« in `ssh`(1) an.

Übrigens sind Sie mit `ssh` nicht auf interaktive Sitzungen beschränkt, sondern können auf dem entfernten Rechner auch einzelne Kommandos ausführen:

```
$ ssh blue.example.com hostname
hugo@blue.example.com's password: geHelm
blue.example.com
$ _
```

Dabei müssen Sie natürlich beachten, dass die Shell auf *Ihrem* Rechner die Kommandozeile bearbeitet und zum Beispiel etwaige Suchmuster zu ersetzen versucht, bevor das Kommando an den entfernten Rechner übertragen wird. Verwenden Sie gegebenenfalls Anführungszeichen oder Rückstriche.

Übungen

25.1 [!1] Verwenden Sie das `ssh`-Kommando, um sich auf einem anderen Rechner anzumelden (Ihr Trainer erklärt Ihnen gegebenfalls, auf welchem). Was passiert? Melden Sie sich ab und melden Sie sich nochmals auf demselben Rechner an. Was ist anders?

25.2 [2] Entfernen Sie den Eintrag für den entfernten Rechner aus Übung 25.1 aus der Datei `~/.ssh/known_hosts` und stellen Sie in der Datei `~/.ssh/ssh_config` den Parameter `StrictHostKeyChecking` auf `yes`. Versuchen Sie dann nochmals, sich auf dem entfernten Rechner anzumelden. Was passiert? Was passiert, wenn der Parameter `StrictHostKeyChecking` auf `no` steht?

25.3 [2] Darf die Datei `~/.ssh/known_hosts` nur für den Benutzer lesbar sein und wenn ja, warum? (Wenn nein, warum nicht?)

25.4 [!2] Führen Sie auf dem entfernten Rechner mit einem einzigen `ssh`-Aufruf die Kommandos `hostname` und `date` aus.

25.3 Andere nützliche Anwendungen: `scp` und `sftp`

Mit `scp` können Sie Dateien zwischen zwei Rechnern über eine SSH-Verbindung kopieren:

```
$ scp blue.example.com:hello.c .
hugo@blue.example.com's password: geHe1m
hello.c   100% |**********************| 33 KB   00:01
```

Die Syntax ist dabei an das `cp`-Kommando angelehnt: Genau wie dort können Sie zwei Dateinamen (Quelle und Ziel) oder eine Reihe von Dateinamen und ein Verzeichnis angeben. Mit der Option `-r` kopiert `scp` Verzeichnisinhalte rekursiv.

Sie können sogar Dateien zwischen zwei verschiedenen entfernten Rechnern kopieren:

```
$ scp hugo@blue.example.com:hello.c \
>    hschulz@pink.example.com:hello-new.c
```

Das Kommando `sftp` ist locker an gängige FTP-Clients angelehnt, aber verwendet eine SSH-Verbindung. Mit FTP hat es ansonsten überhaupt nichts zu tun – insbesondere können Sie es nicht verwenden, um mit einem FTP-Server zu kommunizieren.

Nachdem Sie mit einem Kommando wie

```
$ sftp hugo@blue.example.com
```

eine Verbindung aufgebaut haben, können Sie mit Kommandos wie `get`, `put` oder `mget` Dateien zwischen Ihrem lokalen Rechner und dem entfernten Rechner übertragen, mit `ls` den Inhalt eines Verzeichnisses auf dem entfernten Rechner anschauen und mit `cd` dort in andere Verzeichnisse wechseln. Am Anfang einer Sitzung stehen Sie auf dem entfernten Rechner in Ihrem Heimatverzeichnis.

25.4 Client-Authentisierung über Schlüsselpaare

Im Normalfall authentisiert der SSH-Server Sie als Benutzer über ein Kennwort, das für Ihr Benutzerkonto auf dem Server hinterlegt ist (üblicherweise in `/etc/passwd` oder `/etc/shadow`). Da die Kennwortabfrage erst erfolgt, wenn die verschlüsselte Verbindung bereits steht, ist das grundsätzlich sicher vor unerwünschten Mithörern. Allerdings könnte es Ihnen ein Dorn im Auge sein, dass das Kennwort selbst auf dem Server liegt – auch wenn es verschlüsselt ist, könnte die Kennwortdatei Crackern in die Hände fallen, die dann »John the Ripper« darauf loslassen. Es wäre besser, wenn auf dem entfernten Rechner überhaupt nichts Geheimes von Ihnen gespeichert wäre.

Dies können Sie erreichen, indem Sie statt der einfachen kennwortbasierten Client-Authentisierung die Client-Authentisierung über Schlüsselpaare (engl. *public-key authentication*) verwenden. Kurz gesagt erzeugen Sie dafür ein Paar aus einem öffentlichen und einem privaten Schlüssel und hinterlegen den öffentlichen Schlüssel auf dem SSH-Server. Der öffentliche Schlüssel muss nicht besonders geschützt werden (immerhin ist er öffentlich); auf den privaten Schlüssel müssen Sie gut aufpassen, aber er verläßt nie Ihren eigenen Rechner (den Sie ja nie aus den Augen lassen, nicht wahr?).

Sie können den privaten Schlüssel auch auf einem USB-Stick hinterlegen, wenn Ihnen das sicherer vorkommt.

Der Server kann Sie als rechtmäßigen Inhaber des privaten Schlüssels authentisieren, der zu dem hinterlegten öffentlichen Schlüssel passt, indem er eine Zufallszahl auswürfelt, diese mit dem hinterlegten öffentlichen Schlüssel verschlüsselt

und Ihnen schickt. Sie entschlüsseln (oder genauer gesagt Ihre `ssh` entschlüsselt) die verschlüsselte Zufallszahl dann mit dem privaten Schlüssel. Das Ergebnis geht zurück an den Server; er vergleicht es mit seiner ursprünglichen Zufallszahl, und wenn die beiden übereinstimmen, glaubt er Ihnen, dass Sie Sie sind.

Das ganze geht natürlich über die verschlüsselte Verbindung und ist darum sicher vor unerwünschten Zuhörern oder Fieslingen, die an den Daten herumbasteln wollen.

Um die Client-Authentisierung über Schlüsselpaare verwenden zu können, müssen Sie zuerst ein Schlüsselpaar generieren. Das geht mit dem Kommando `ssh-keygen`:

```
$ ssh-keygen -t rsa -b 2048                                        oder ed25519
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hugo/.ssh/id_rsa): ↩
Created directory '/home/hugo/.ssh'.
Enter passphrase (empty for no passphrase): geheim
Enter same passphrase again: geheim
Your identification has been saved in /home/hugo/.ssh/id_rsa.
Your public key has been saved in /home/hugo/.ssh/id_rsa.pub.
The key fingerprint is:
39:ab:15:f4:2f:c4:e6:21:26:c4:43:d7:27:22:a6:c4 hugo@blue
The key's randomart image is:
+---[RSA 2048]----+
|   .  . ..       |
|    Eoo.. o .    |
|   . o+... o     |
|    .. o +       |
|      . S *      |
|       o 0 o     |
|        o o .    |
|       o   .     |
|      .          |
+-----------------+
```

Das Kommando fragt zunächst nach dem gewünschten Speicherort für das Schlüsselpaar. Die Vorgabe ist dabei vernünftig und Sie sollten sie einfach bestätigen.

Als nächstes möchte `ssh-keygen` eine *passphrase* wissen. Diese dient zur Verschlüsselung des *privaten* Schlüssels, um zu verhindern, dass jemand, dem Ihr privater Schlüssel in die Hände fällt, sich gegenüber dem SSH-Server als Sie ausgeben kann.

Sie können (oder sollten) hier durchaus einen längeren Satz angeben. Ein kürzeres Kennwort aus einer bunten Mischung von Buchstaben, Ziffern und Sonderzeichen ist wahrscheinlich auch in Ordnung. Es gelten die üblichen Regeln für solche Geheimnisse.

Schlüssel ohne *passphrase* müssen Sie benutzen, wenn Sie unbeaufsichtigte SSH-Verbindungen benötigen, etwa für Shellskripte und `cron`-Jobs. In diesem Fall drücken Sie bei den Fragen nach der *passphrase* einfach auf ↩.

Es ist möglich, einen öffentlichen Schlüssel auf dem Server fest mit einem bestimmten Kommando zu verbinden. Client-Aufrufe, die diesen öffentlichen Schlüssel verwenden, führen dann nicht zu einer Shell-Sitzung, sondern zu einem sofortigen Start des angegebenen Kommandos. Das Sicherheitsrisiko von unverschlüsselten privaten Schlüsseln zum Gebrauch durch Skripte kann damit signifikant vermindert werden.

Das Ergebnis von `ssh-keygen` sind die beiden Dateien `id_rsa` und `id_rsa.pub` im Verzeichnis `~/.ssh`. Dabei enthält erstere den privaten und letztere den öffentlichen Schlüssel.

Wenn Sie bei der Schlüsselgenerierung »`-t ed25519`« angegeben haben, heißen die Dateien natürlich `id_ed25519` und `id_ed25519.pub`.

Das Kommando `ssh-keygen` gibt Ihnen außerdem den Fingerabdruck des öffentlichen Schlüssels und ein *randomart image* aus. Dabei handelt es sich um eine grafische Darstellung des öffentlichen Schlüssels, eine Art grafischen Fingerabdruck. Dieser soll Sie theoretisch in die Lage versetzen, mit einem Blick sehen zu können, ob ein öffentlicher Schlüssel sich geändert hat oder nicht. Das Konzept ist, vorsichtig gesagt, umstritten.

Es hält Sie natürlich niemand davon ab, `ssh-keygen` mehrmals aufzurufen, um mehrere Schlüsselpaare mit verschiedenen Verschlüsselungsverfahren zu erzeugen. (Oder mehrere Schlüsselpaare mit demselben Verschlüsselungsverfahren zur Verwendung mit verschiedenen Servern. Dabei müssen Sie natürlich darauf achten, dass Sie verschiedene Dateinamen benutzen.)

Im nächsten Schritt müssen Sie den öffentlichen Schlüssel, also die `id_rsa.pub`-Datei, in der Datei `~/.ssh/authorized_keys` in Ihrem Benutzerkonto auf dem entfernten Rechner ablegen. Am bequemsten geht das mit dem Kommando `ssh-copy-id`:

```
$ ssh-copy-id hugo@blue.example.com
hugo@blue.example.com's password: geHe1m                    Ein letztes Mal
Now try logging into the machine, with "ssh 'hugo@blue.example.com'",▷
 ◁ and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

$ _
```

Natürlich können Sie das auch »zu Fuß« mit `scp` und/oder `ssh` erledigen. Achten Sie dabei nur darauf, etwaige schon vorhandene Schlüssel in `~/.ssh/authorized_keys` nicht zu überschreiben, auf die Sie vielleicht noch Wert legen.

Wenn Sie in der Datei `/etc/ssh/sshd_config` auf dem Server die Einträge `PasswordAuthentication` auf `no` und `PubkeyAuthentication` auf `yes` setzen, dann können Benutzer sich *nur* noch über die Schlüsselpaar-Methode authentisieren. Das ist grundsätzlich eine gute Idee, da Cracker sich einen Spaß daraus machen, mit automatischen Programmen SSH-Server nach naheliegenden Kennwörtern abzuklopfen.

Die Authentisierung über Schlüsselpaare ist zumindest bei Verwendung einer *passphrase* nicht komfortabler als die Kennwortauthentisierung, dafür aber um einiges sicherer. Falls Sie sich häufiger hintereinander auf dem gleichen Rechner als der gleiche Benutzer anmelden wollen, ist das ständige Wiedereingaben der *passphrase* allerdings lästig. Für solche Fälle wurde der `ssh-agent` entwickelt.

`ssh-agent` Der `ssh-agent` merkt sich die *passphrase* und übergibt sie bei jedem Aufruf eines SSH-Client-Programmes an dieses. Das Programm wird mit »`ssh-agent bash`«
`ssh-add` gestartet. Dabei öffnet sich eine neue `bash`, in der Sie die Passphrase mit `ssh-add` bekannt machen müssen:

```
$ ssh-add
Enter passphrase for /home/hugo/.ssh/id_rsa: Habe nun, ach!
Identity added: /home/hugo/.ssh/id_rsa (/home/hugo/.ssh/id_rsa)
```

Jedem aus der neuen Shell heraus gestarteten `ssh-`, `scp-` oder `sftp`-Programm wird vom SSH-Agent die *passphrase* übergeben. Der Agent »vergisst« die *passphrase* wieder, wenn Sie mit einem `exit` die Shell verlassen oder ihn mit »`ssh-add -D`« anweisen, die gespeicherten Identitäten wieder zu löschen.

Bei Debian GNU/Linux werden die Loginshell bzw. die grafische Umgebung auf Wunsch schon mit dem `ssh-agent` gestartet, so dass Sie gleich mit `ssh-add` Ihre *passphrase* eingeben können.

Der Fairness halber sollten wir erwähnen, dass die Verwendung des `ssh-agent` zwar die Bequemlichkeit erhöht, aber auf Kosten der Sicherheit geht. Wenn Sie Ihren Rechner unbeaufsichtigt lassen (oder Ihr in den Schlummermodus versetztes Notebook Ihnen abhanden kommt), kann eine unberechtigte Person möglicherweise die SSH-Programme benutzen, ohne nach einer *passphrase* gefragt zu werden. Dasselbe gilt für Programme, die irgendwie Zugang zu Ihrer Sitzung bekommen, also Viren, Würmer und ähnliches Viechzeug …

Übungen

25.5 [!2] Erzeugen Sie sich mit `ssh-keygen` ein RSA-Schlüsselpaar für die SSH-Protokollversion 2. (Denken Sie dran, mindestens 2048 Bit!) Installieren Sie den öffentlichen Schlüssel auf dem entfernten Rechner und überzeugen Sie sich, dass Sie bei der Anmeldung nicht mehr nach dem Kennwort für den entfernten Rechner gefragt werden. Was müssen Sie statt dessen eingeben?

25.6 [!1] Bestimmen Sie den »Fingerabdruck« Ihres öffentlichen Schlüssels.

25.7 [2] Unter welchen Umständen könnten Sie auf die Verwendung einer *passphrase* für den privaten Schlüssel verzichten wollen?

25.5 Portweiterleitung über SSH

25.5.1 X11-Weiterleitung

Ausführen von grafischen Programmen

Die X11-Weiterleitung ermöglicht das Ausführen von grafischen Programmen (X-Clients) auf einem entfernten Rechner, wobei die Grafikausgabe und Bedienung auf dem lokalen Rechner erfolgt. Sie müssen sich dafür lediglich per `ssh` auf dem entfernten Rechner anmelden und dabei die Option `-X` (großes X!) verwenden. Voraussetzung ist, dass auf der Server-Seite die X11-Weiterleitung (Parameter `X11Forwarding` in `/etc/ssh/sshd_config`) aktiviert ist.

Nach einer Anmeldung mit »`ssh -X` [⟨*Benutzername*⟩`@`]⟨*Rechner*⟩« befinden Sie sich auf dem Server und können beliebige X-Programme ausführen, die auf dem lokalen X-Server angezeigt und bedient werden können. Das beruht auf mehreren Faktoren:

- Beim Anmelden mit `-X` wird automatisch die `DISPLAY`-Variable gesetzt – sie zeigt auf einen »Proxy«-X-Server, den der `sshd` zur Verfügung stellt. Dadurch werden X-Clients auf dem entfernten Rechner an diesen Server verwiesen.
- Alle Äußerungen eines auf dem entfernten Rechner gestarteten X-Clients an den dortigen Proxy-X-Server werden an den (echten) X-Server auf dem SSH-Client weitergeleitet.
- Die weitergeleiteten X-Pakete des X-Clients werden von der SSH außerdem verschlüsselt, so dass die Verbindung von außen nicht zu belauschen ist (Tunneling).


Sie können das X11-Forwarding auch pauschal in Kraft setzen, um sich die `-X` sparen zu können. Dazu müssen Sie die Zeile »`ForwardX11 yes`« in die Datei `~/.ssh/config` (oder systemweit `/etc/ssh/ssh_config`) aufnehmen.

Das X11-Forwarding ist dem herkömmlichen Umleiten der X-Pakete per `DISPLAY`-Variable auf jeden Fall vorzuziehen, nicht nur wegen der Sicherheit, sondern, weil sie schlicht komfortabler ist. Sie bezahlen dafür mit einem gewissen Extraaufwand für die Verschlüsselung, was bei heutigen Rechnern aber in der Regel nicht ins Gewicht fällt.


Ganz ohne Sicherheitsrisiken ist auch X11-Forwarding nicht: Benutzer, die die Dateizugriffsrechte auf dem entfernten Rechner unterlaufen können (etwa weil sie dort `root` sind), können auf das lokale X-Display zugreifen, indem sie sich den Inhalt Ihrer `.Xauthority`-Datei auf dem entfernten Rechner anschauen. Aus diesem Grund sollten Sie X11-Forwarding wahrscheinlich nicht pauschal einschalten. Dasselbe Risiko besteht natürlich bei »herkömmlichem« X11-basierter Ausgabeweiterleitung über `DISPLAY`.

25.5.2 Beliebige TCP-Ports weiterleiten

Portweiterleitung

SSH kann nicht nur das X-Protokoll, sondern so gut wie jedes andere TCP-basierte Protokoll weiterleiten und tunneln. Dazu stehen die Optionen `-R` und `-L` zur Verfügung. Das folgende Kommando leitet Verbindungen an den lokalen TCP-Port 10110 zunächst über eine SSH-Verbindung auf den Rechner `blue.example.com` um. Von da geht es (unverschlüsselt) weiter auf den TCP-Port 110 (POP3) auf dem Rechner `mail.example.com`:

```
$ ssh -L 10110:mail.example.com:110 hugo@blue.example.com
```

Der Nutzen dieses Szenarios ist etwa wie folgt: Stellen Sie sich vor, Ihr Firewall sperrt POP3, aber läßt SSH durch. Über die Portweiterleitung kommen Sie via SSH ins interne Netz und können vom Rechner `blue.example.com` aus rein im internen Netz mit dem Mailserver reden. In Ihrem Mailprogramm geben Sie dann `localhost` und den lokalen TCP-Port 10110 als »POP3-Server« an.


Theoretisch könnten Sie auch den lokalen TCP-Port 110 weiterleiten, aber dazu müssen Sie `root` sein.


Der Name des Rechners für die Weiterleitung (hier `mail.example.com`) wird aus der Sicht des SSH-Servers (hier `blue.example.com`) aufgelöst. Das heißt, eine Weiterleitung der Form

```
$ ssh -L 10110:localhost:110 hugo@blue.example.com
```

verbindet Sie mit dem Port 110 auf `blue.example.com`, nicht etwa auf Ihrem Rechner.


Eine Weiterleitung der Form

```
-L 10110:mail.example.com:110
```

öffnet den Port 10110 auf *allen* IP-Adressen Ihres Rechners. Damit wird die Weiterleitung prinzipiell auch für andere Stationen zugänglich, die über das Netz mit diesem Port Kontakt aufnehmen können. Um das zu verhindern, können Sie ausnutzen, dass `ssh` es Ihnen erlaubt, eine lokale Adresse für den weitergeleiteten Port anzugeben: Mit

```
-L localhost:10110:mail.example.com:110
```

gilt die Weiterleitung nur für die lokale Schnittstelle.

Wenn Sie `ssh` wie gezeigt aufrufen, bekommen Sie außer der Portweiterleitung auch eine interaktive Sitzung. Wenn Sie das nicht möchten – etwa weil die Weiterleitung in einem `cron`-Job stattfindet –, können Sie die Option `-N` angeben, die `ssh` auf die Weiterleitung beschränkt und keine interaktive Sitzung aufbaut.

Eine andere (möglicherweise bessere) Technik für das automatische Weiterleiten von Diensten verwendet einen `ssh`-Aufruf der Form

```
$ ssh -f -L 10110:mail.example.com:110 blue sleep 10
$ getmail_fetch -p10110 localhost hugomail MaIl123 Maildir/
```

Die Option `-f` sorgt dafür, dass der `ssh`-Prozess unmittelbar vor der Ausführung des »`sleep 10`«-Kommandos in den Hintergrund geht. Das heißt, ein Kommando, das Sie unmittelbar nach dem `ssh`-Aufruf ausführen (hier `getmail_fetch`, das Mail über POP3 abruft), hat 10 Sekunden Zeit, eine Verbindung über den lokalen Port 10110 aufzubauen. Der `ssh`-Prozess beendet sich entweder nach 10 Sekunden oder aber wenn die (letzte) Verbindung über den lokalen Port 10110 wieder abgebaut wird, was auch immer später passiert.

Die Portweiterleitung funktioniert auch umgekehrt: Mit

```
$ ssh -R 10631:localhost:631 hugo@blue.example.com
```

wird der TCP-Port 10631 *auf dem SSH-Server* geöffnet und Verbindungen, die Programme dort mit diesem Port aufnehmen, über die SSH-Verbindung auf Ihren lokalen Rechner geleitet. Ihr lokaler Rechner übernimmt dann die unverschlüsselte Weiterleitung an das angegebene Ziel, hier den TCP-Port 631 auf Ihrem lokalen Rechner selbst. (Diese Form der Weiterleitung ist ungleich weniger wichtig als die über `-L`.)

Die `-R`-Weiterleitung bindet den entfernten Port normalerweise an die `localhost`-Schnittstelle auf dem SSH-Server. Sie können sich grundsätzlich wie oben gezeigt eine andere Schnittstelle wünschen (»`*`« steht für »alle«), aber ob das funktioniert, hängt von der Konfiguration des SSH-Servers ab.

Portweiterleitungen können Sie auch nachträglich einrichten. Dazu müssen Sie die Tastenkombination »`~C`« verwenden (es muss ein großes C sein), die eine »Kommandozeile« zur Verfügung stellt:

```
⋖⋖⋖⋖                                   Hier läuft gerade eine SSH-Sitzung
remote$ ↩
remote$ ~ C
ssh> -L 10025:localhost:25
Forwarding port.
⋖⋖⋖⋖                                   SSH-Sitzung geht weiter
```

Auf der »Kommandozeile« können Sie (unter anderem) `-L`- und `-R`-Optionen nachtragen, so als ob Sie sie gleich beim `ssh`-Aufruf angegeben hätten. Mit `-KR` gefolgt von der Portnummer können Sie eine `-R`-Weiterleitung auch wieder rückgängig machen (`-KL` gibt es leider nicht). Mit der Tastenkombination »`~#`« bekommen Sie einen Überblick über die gerade aktiven Verbindungen:

```
⋖⋖⋖⋖
remote$ ~#
The following connections are open:
  #2 client-session (t4 r0 i0/0 o0/0 fd 6/7 cfd -1)
  #3 direct-tcpip: listening port 10025 for localhost port 25,▷
 ◁ connect from 127.0.0.1 port 57250 ▷
 ◁ (t4 r1 i0/0 o0/0 fd 9/9 cfd -1)
⋖⋖⋖⋖
```

Wie Sie den vorstehenden Abschnitten zweifellos entnommen haben, bietet `ssh` diverse Möglichkeiten an, Schindluder zu treiben, die einem IT-Sicherheitsbeauftragten Tränen in die Augen steigen lassen dürften. Bitte werten Sie dieses Kapitel als Vorstellung einiger Fähigkeiten der `ssh`, nicht als Empfehlung dafür, möglichst viele davon tatsächlich zu verwenden (jedenfalls ohne guten Grund). Als Betreiber eines SSH-Servers sollten Sie insbesondere dessen Dokumentation (etwa `sshd_config`(5)) studieren, um zu lernen, wie Sie die Verwendung der gefährlicheren Optionen unterdrücken können. Für eine ausführliche Beschreibung der SSH-Serverkonfiguration ist in dieser Schulungsunterlage leider kein Platz.

Übungen

25.8 [!1] Wie können Sie die `ssh` verwenden, um von einem normalen Benutzerzugang aus auf dem lokalen Rechner bequem X-Clients mit `root`-Rechten zu starten?

25.9 [3] Verwenden Sie die `ssh`, um den Port 4711 (oder einen anderen geeigneten lokalen Port) auf Ihrem Rechner auf den `echo`-Port (Port 7) des entfernten Rechners umzuleiten. Überzeugen Sie sich mit einem Paketsniffer (`tcpdump` oder `wireshark`), dass eine Verbindung zum lokalen Port 4711, etwa mit »`telnet localhost 4711`« tatsächlich eine verschlüsselte Datenübertragung zum entfernten Rechner bewirkt und erst auf diesem wieder im Klartext weitergeführt wird.

Kommandos in diesem Kapitel

`scp`	Sicheres Dateikopierprogramm auf SSH-Basis	`scp`(1)	427
`sftp`	Sicheres FTP-artiges Programm auf SSH-Basis	`sftp`(1)	428
`ssh`	„Secure Shell“, erlaubt sichere interaktive Sitzungen auf anderen Rechnern	`ssh`(1)	424
`ssh-add`	Akkreditiert private Schlüssel beim `ssh-agent`	`ssh-add`(1)	430
`ssh-agent`	Verwaltet private Schlüssel und Kennwörter für die SSH	`ssh-agent`(1)	430
`ssh-copy-id`	Kopiert öffentliche SSH-Schlüssel auf andere Rechner	`ssh-copy-id`(1)	430
`ssh-keygen`	Generiert und verwaltet Schlüssel für die SSH	`ssh-keygen`(1)	429
`sshd`	Server für das SSH-Protokoll (sicherer interaktiver Fernzugriff)	`sshd`(8)	424

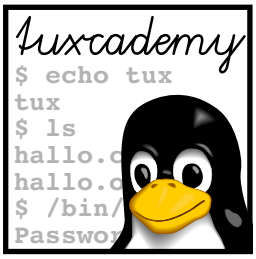
Zusammenfassung

- Die Secure Shell erlaubt das komfortable und sichere Anmelden auf entfernten Rechnern (und ersetzt so TELNET, RSH und RLOGIN) sowie die gesicherte Übertragung von Dateien ähnlich RCP und FTP.
- Mit OpenSSH steht eine leistungsfähige Implementierung der Secure Shell frei zur Verfügung.
- Der Benutzer hat die Wahl zwischen kennwortbasierter Authentisierung und Authentisierung über ein asymmetrisches Schlüsselpaar. Letztere ist sicherer, aber aufwendiger zu konfigurieren.
- Die Secure Shell kann die X11-Grafikdarstellung und -interaktion sowie beliebige TCP-Verbindungen über die verschlüsselte Strecke weiterleiten.

Literaturverzeichnis

BS01 Daniel J. Barrett, Richard Silverman. *SSH, The Secure Shell: The Definitive Guide*. Sebastopol, CA: O'Reilly & Associates, 2001. ISBN 0-596-00011-1. http://www.oreilly.com/catalog/sshtdg/

RFC4253 T. Ylonen, C. Lonvick. »The Secure Shell (SSH) Transport Layer Protocol«, Januar 2006. http://www.ietf.org/rfc/rfc4253.txt



26

Paketverwaltung mit Debian-Werkzeugen

Inhalt

26.1 Überblick 438
26.2 Das Fundament: dpkg 438
26.2.1 Debian-Pakete 438
26.2.2 Paketinstallation 439
26.2.3 Pakete löschen 440
26.2.4 Debian-Pakete und ihr Quellcode 441
26.2.5 Informationen über Pakete 442
26.2.6 Verifikation von Paketen 444
26.3 Debian-Paketverwaltung der nächsten Generation 445
26.3.1 APT 445
26.3.2 Paketinstallation mit apt-get 446
26.3.3 Informationen über Pakete 447
26.3.4 aptitude 449
26.4 Integrität von Debian-Paketen 451
26.5 Die debconf-Infrastruktur 452
26.6 alien: Pakete aus fremden Welten 453

Lernziele

- Grundzüge der Debian-Paketwerkzeuge kennen
- dpkg zur Paketverwaltung verwenden können
- apt-get, apt-cache und aptitude einsetzen können
- Prinzipien der Integritätssicherung für Debian-Pakete kennen
- RPM-Pakete mit alien in Debian-Pakete umwandeln können

Vorkenntnisse

- Kenntnisse der Linux-Systemadministration
- Erfahrung mit Debian GNU/Linux oder einer von Debian GNU/Linux abgeleiteten Distribution ist hilfreich

26.1 Überblick

Die Softwarepakete von Debian GNU/Linux und den davon abstammenden Distributionen wie Ubuntu, Knoppix, Xandros oder Sidux werden mit dem Werkzeug `dpkg` verwaltet. Es dient zur Installation von Softwarepaketen, zur Verwaltung von Abhängigkeiten, zur Katalogisierung der installierten Software, zum kontrollierten Aktualisieren von Softwarepaketen und zur Deinstallation nicht mehr benötigter Pakete. Die komfortable Auswahl von Softwarepaketen erlauben Programme wie `aptitude`, die als Oberfläche für `dpkg` fungieren. Für die Konfiguration von Paketen bei der Installation sorgt die `debconf`-Infrastruktur.

Die Paketverwaltungssysteme von Debian und Red Hat wurden etwa zeitgleich entwickelt und haben verschiedene Stärken und Schwächen. Wie auch sonst so oft in der Freie-Software-Szene haben die Religionskriege rund um `dpkg` und `rpm` aber nicht dazu geführt, dass einer der beiden Wettbewerber das Rennen gemacht hätte. Mit der zunehmenden Beliebtheit von Debian-Ablegern – allen voran Ubuntu – bleibt das auch für die Zukunft unwahrscheinlich.

Der LSB-Standard für eine vorhersehbare Linux-Grundinfrastruktur, auf die Drittanbieter ihre Software portieren können, schreibt eine restringierte Version von RPM als Paketformat vor. Das bedeutet aber nicht, dass eine LSB-konforme Linux-Distribution komplett auf RPM aufbauen muss, sondern lediglich, dass sie in der Lage sein muss, Softwarepakete von Drittanbietern zu installieren, die der LSB-Geschmacksrichtung von RPM entsprechen.

Für Debian GNU/Linux ist das natürlich ein Klacks. Dass Debian GNU/Linux nicht offiziell als »LSB-konform« geführt wird, liegt daran, dass der LSB-Standard von einem Industriekonsortium finanziert wird, in dem Debian als nichtkommerzielles Projekt nicht vertreten ist. In der Beschreibung des `lsb`-Pakets für Debian GNU/Linux heißt es:

> *The intent of this package is to provide a best current practice way of installing and running LSB packages on Debian GNU/Linux. Its presence does not imply that Debian fully complies with the Linux Standard Base, and should not be construed as a statement that Debian is LSB-compliant.*

Obwohl der Titel von »Debian-Werkzeugen« spricht, gilt alles in diesem Kapitel sinngemäß auch für Ubuntu, da Ubuntu die wesentlichen Teile seiner Infrastruktur von Debian GNU/Linux übernimmt. Auf allfällige Unterschiede machen wir Sie gesondert aufmerksam.

26.2 Das Fundament: `dpkg`

26.2.1 Debian-Pakete

Pakete
Paketnamen

In der Debian-Infrastruktur ist die Software auf dem System in Pakete eingeteilt. Pakete haben Namen, die auf die enthaltene Software und deren Versionsstand hinweisen. Die Datei

```
hello_2.8-2_amd64.deb
```

zum Beispiel enthält das Programm `hello` in der Version 2.8, dabei handelt es sich um die 2. Auflage dieses Pakets in der Distribution (bei einem künftigen Paket für die Version 2.9 würde wieder bei 1 begonnen). Pakete wie `apt`, die speziell für Debian entwickelt wurden, haben keine »Auflagennummer«. Das `amd64` zeigt an,

dass das Paket architekturspezifische Teile für Intel- und AMD-x86-Prozessoren (und Kompatible) im 64-Bit-Modus enthält – 32-Bit-Pakete benutzen `i386` und Pakete, in denen nur Dokumentation oder architekturunabhängige Skripte stehen, verwenden stattdessen `all`.

Ein Debian-Paket ist ein mit dem Archivprogramm `ar` erstelltes Archiv, das üblicherweise drei Bestandteile enthält: Paketstruktur

```
$ ar t hello_2.8-2_amd64.deb
debian-binary
control.tar.gz
data.tar.gz
```

Die Datei `debian-binary` enthält die Versionsnummer des Paketformats (aktuell `2.0`). In `control.tar.gz` befinden sich Debian-spezifische Skripte und Steuerdateien und `data.tar.gz` enthält die eigentlichen Paketdaten. Bei der Installation wird zunächst `control.tar.gz` entpackt, damit ein eventuell vorhandenes `preinst`-Skript vor dem Auspacken der eigentlichen Paketdaten ausgeführt werden kann. Anschließend wird `data.tar.gz` entpackt und bei Bedarf das Paket konfiguriert, indem das `postinst`-Skript aus `control.tar.gz` ausgeführt wird. Installationsvorgang

Übungen

26.1 [2] Holen Sie sich ein beliebiges Debian-Paket (etwa `hello`) und nehmen Sie es mit Hilfe von `ar` und `tar` auseinander. Finden Sie die Installationsskripte? Welche Informationen stehen in `control.tar.gz`, welche in `data.tar.gz`?

26.2.2 Paketinstallation

Ein lokal vorliegendes Debian-Paket können Sie einfach mit dem Kommando

```
# dpkg --install hello_2.8-2_amd64.deb
```

installieren, wobei `--install` auch zu `-i` abgekürzt werden kann. Mit `--unpack` und `--configure` (`-a`) können der Entpack- und der Konfigurationsvorgang auch separat ausgeführt werden.

Im wirklichen Leben sind die kurzen Optionsnamen wie `-i` bequem. Wenn Sie die LPI-101-Prüfung ablegen wollen, sollten Sie aber unbedingt auch die langen Optionsnamen lernen, da diese lästigerweise in den Prüfungsfragen vorkommen. Im Falle von `-i` und `--install` können Sie sich die Korrespondenz wahrscheinlich noch herleiten; bei `-a` und `--configure` ist das schon etwas weniger offensichtlich.

Optionen für `dpkg` können Sie auf der Kommandozeile angeben oder in der Datei `/etc/dpkg/dpkg.cfg` hinterlegen. In dieser Datei müssen die Minuszeichen am Anfang der Optionsnamen weggelassen werden. dpkg.cfg

Wenn mit »`dpkg --install`« ein Paket installiert wird, das bereits in einer älteren Version vorliegt, wird die ältere Version deinstalliert, bevor die neue konfiguriert wird. Wenn bei der Installation ein Fehler auftritt, kann die alte Paketversion in vielen Fällen wiederhergestellt werden. Upgrade

Es gibt verschiedene Gründe, die eine erfolgreiche Paketinstallation verhindern können, zum Beispiel: Probleme beim Installieren

- Das Paket benötigt ein oder mehrere andere Pakete, die nicht entweder schon installiert sind oder im selben Installationsvorgang mitinstalliert werden sollen. Die entsprechende Prüfung kann mit der Option `--force-depends` außer Kraft gesetzt werden – allerdings kann das System dann übel durcheinandergeraten.

- Eine frühere Version des Pakets ist vorhanden und auf `hold` gesetzt (etwa mit `aptitude`). Damit können keine neueren Versionen dieses Pakets installiert werden.
- Das Paket versucht eine Datei zu entpacken, die bereits im System vorliegt und einem Paket anderen Namens gehört, es sei denn, das aktuelle Paket ist explizit als das andere Paket »überlagernd« gekennzeichnet oder die Option `--force-overwrite` wurde angegeben.

Ausschluss

Manche Pakete schließen einander aus (siehe die Möglichkeiten für Paketabhängigkeiten auf Seite 443). Zum Beispiel kann immer nur ein Mailtransportprogramm installiert sein; wenn Sie also zum Beispiel Postfix installieren wollen, muss Exim (das Debian-Standardprogramm) gleichzeitig entfernt werden. `dpkg` kümmert sich darum, wenn bestimmte Bedingungen erfüllt sind.

Virtuelle Pakete

Manchmal hängen Pakete nicht von einem konkreten anderen Paket ab, sondern von einem »virtuellen« Paket, das eine Funktionalität beschreibt, die prinzipiell von mehreren Paketen erbracht werden kann, etwa `mail-transport-agent`, das von Paketen wie `postfix`, `exim` oder `sendmail` zur Verfügung gestellt wird. In diesem Fall ist es möglich, trotz Abhängigkeiten zum Beispiel Exim durch Postfix zu ersetzen, weil zu jeder Zeit ein Paket installiert ist, das die »virtuelle« Funktionalität bietet.

Übungen

26.2 [1] Laden Sie ein Debian-Paket – etwa `hello` – von `ftp.de.debian.org` (oder einem beliebigen anderen Debian-Mirror) herunter und installieren Sie es mit `dpkg`. (Wer etwas anderes benutzt, etwa `apt-get` – siehe nächster Abschnitt –, der schummelt!) Sie finden Debian-Pakete auf dem Server halbwegs bequem über ihren Namen, indem Sie in `pool/main` in einem Unterverzeichnis nachschauen, das dem ersten Buchstaben des Namens entspricht, und darin einem Verzeichnis, das so heißt wie der Paketname[1], in unserem Beispiel also `pool/main/h/hello`. Ausnahme: Da sehr viele Paketnamen mit `lib` anfangen, steht ein Paket wie `libblafasel` in `pool/main/libb`.

26.3 [2] Finden Sie die aktuelle Liste von virtuellen Paketen in Debian GNU/Linux. Wo wird diese abgelegt? Wann war die letzte Änderung?

26.2.3 Pakete löschen

Mit dem Kommando

```
# dpkg --remove hello
```

(kurz »`dpkg -r`«) wird ein Paket entfernt, aber die Konfigurationsdateien bleiben erhalten. Dies vereinfacht eine spätere Wiederinstallation des Pakets. Das Kommando

```
# dpkg --purge hello
```

(oder »`dpkg -P`«) entfernt das Paket mitsamt den Konfigurationsdateien.

Als »Konfigurationsdateien« eines Debian-Pakets zählen alle Dateien im Paket, deren Namen in der Datei `conffiles` in `control.tar.gz` aufgezählt sind. (Schauen Sie in `/var/lib/dpkg/info/`⟨*package name*⟩`.conffiles`.)

Probleme beim Entfernen

Auch das Entfernen eines Pakets muss nicht funktionieren. Mögliche Hinderungsgründe sind:

[1] Eigentlich der Name des Quellcode-Pakets, der abweichen kann. Machen Sie es sich nicht zu kompliziert.

- Das Paket wird von einem oder mehreren anderen Paketen benötigt, die nicht gleichzeitig mit entfernt werden.
- Das Paket ist als *essential* (unverzichtbar für die Systemfunktionalität) gekennzeichnet. Die Shell zum Beispiel kann nicht ohne Weiteres entfernt werden, da sonst diverse wichtige Skripte nicht mehr ausgeführt werden können.

Auch hier lassen die Prüfungen sich über geeignete `--force-`…-Optionen außer Kraft setzen (auf eigene Gefahr).

Übungen

26.4 [1] Entfernen Sie das Paket aus Übung 26.2 wieder. Sorgen Sie dafür, dass auch allfällige Konfigurationsdateien mit verschwinden.

26.2.4 Debian-Pakete und ihr Quellcode

Im Umgang mit Quellcode ist es ein Grundprinzip des Projekts, sauber zwischen dem originalen Quellcode und den Debian-spezifischen Änderungen zu trennen. Entsprechend werden alle Änderungen in einem eigenen Archiv untergebracht. Zu diesen gehören neben den Debian-eigenen Steuerdateien auch mehr oder weniger umfangreiche Reparaturen oder Anpassungen an der Software selbst. Ferner gibt es zu jeder Paketversion eine Quellcode-Steuerdatei (Endung `.dsc`), die Prüfsummen des Originalarchivs und der Änderungsdatei enthält und die von dem für das Paket verantwortlichen Debian-Entwickler digital signiert wird:

Quellcode-Steuerdatei

```
$ ls hello*
-rw-r--r-- 1 anselm anselm   6540 Jun  7 13:18 hello_2.8-2.debian.tar.gz
-rw-r--r-- 1 anselm anselm   1287 Jun  7 13:18 hello_2.8-2.dsc
-rw-r--r-- 1 anselm anselm 697483 Mai 27 23:47 hello_2.8.orig.tar.gz
```

Hier sehen Sie auch, dass das originale Quellcodearchiv sich für die ganze Version 2.3 des Programms nicht ändert (es hat keine Debian-Auflagennummer). Jede neue Version des Debian-Pakets für die Version 2.8 von `hello` hat allerdings eine neue `.dsc`- und eine neue `.debian.tar.gz`-Datei. Letztere enthält alle Änderungen relativ zum Originalarchiv (nicht etwa zum Paket `hello_2.8-2`).

Früher verwendete das Debian-Projekt eine einfachere Struktur, bei der die Debian-spezifischen Änderungen komplett in einer mit `diff` erzeugten Datei standen – in unserem Beispiel hypothetischerweise `hello_2.8-2.diff.gz`. Dieser Ansatz wird nach wie vor unterstützt, und Sie finden diese Struktur möglicherweise noch bei älteren Paketen, die nicht auf die neue Struktur hin umgebaut wurden. Die neue Struktur hat den Vorteil, dass sich verschiedene Änderungen – etwa die Einführung der Debian-spezifischen Steuerdateien und allfällige Reparaturen am Originalpaket – sauber voneinander trennen lassen, was die Wartung des Pakets im Debian-Projekt stark vereinfacht.

Das Kommando `dpkg-source` dient dazu, aus dem Originalarchiv (vom Debian-Server) und den Debian-spezifischen Änderungen den Quellcode des Pakets so zu rekonstruieren, dass Sie Ihre eigene Fassung des Debian-Pakets übersetzen können. Dazu rufen Sie es mit dem Namen der Quellcode-Steuerdatei des Pakets auf:

`dpkg-source`

```
$ dpkg-source -x hello_2.8-2.dsc
```

Das Originalarchiv und die `.debian.tar.gz`- oder `.diff.gz`-Datei müssen dabei im selben Verzeichnis wie die Quellcode-Steuerdatei stehen; `dpkg-source` legt dort auch den ausgepackten Quellcode ab.

Debian-Pakete erstellen

`dpkg-source` dient auch zum Erzeugen von Quellcodearchiven und Debian-Änderungsdateien im Rahmen der Erstellung von Debian-Paketen. Dieses Thema ist jedoch kein Bestandteil der LPIC-1-Zertifizierung.

Übungen

26.5 [1] Holen Sie sich den Quellcode des Pakets, das Sie in Übung 26.2 installiert haben, und packen Sie ihn aus. Werfen Sie einen Blick in das Unterverzeichnis `debian` in dem resultierenden Verzeichnis.

26.2.5 Informationen über Pakete

Paketliste

Eine Liste der installierten Pakete erhalten Sie mit »`dpkg --list`« (kurz `-l`):

```
$ dpkg --list
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/H
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,
||/ Name           Version        Description
+++-==============-==============-========================
ii  a2ps           4.13b+cvs.2003 GNU a2ps - 'Anything to Po
ii  aalib1         1.4p5-19       ascii art library
ii  abcm2ps        4.0.7-1        Translates ABC music descr
ii  abcmidi        20030521-1     A converter from ABC to MI
◁◁◁◁◁
```

Shell-Suchmuster

(rechts aus Platzgründen abgeschnitten) Die Liste kann über ein Shell-Suchmuster eingeschränkt werden:

```
$ dpkg -l lib*-tcl
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/H
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,
||/ Name           Version        Description
+++-==============-==============-========================
pn  libdb3-tcl     <none>         (no description available)
un  libdb4.0-tcl   <none>         (no description available)
un  libdb4.1-tcl   <none>         (no description available)
un  libmetakit-tcl <none>         (no description available)
ii  libsqlite-tcl  2.8.9-1        SQLite TCL bindings
rc  libsqlite0-tcl 2.6.1-2        SQLite TCL bindings
```

Die Pakete mit Version »`<none>`« sind zwar Bestandteil der Distribution, aber auf dem vorliegenden System nicht installiert (Zustand `un`) oder entfernt worden (Zustand `pn`). Von mit `rc` markierten Paketen liegen nur noch Konfigurationsdateien vor.

Paketstatus

Den Status eines einzelnen Pakets liefert die Option `--status` (`-s`):

```
$ dpkg --status hello
Package: hello
Status: install ok installed
Priority: optional
Section: devel
Installed-Size: 553
Maintainer: Santiago Vila <sanvila@debian.org>
Architecture: amd64
Version: 2.8-2
Depends: libc6 (>= 2.4), dpkg (>= 1.15.4) | install-info
Description: The classic greeting, and a good example
```

```
 The GNU hello program produces a familiar, friendly greeting.  It
 allows non-programmers to use a classic computer science tool which
 would otherwise be unavailable to them.
 .
 Seriously, though: this is an example of how to do a Debian package.
 It is the Debian version of the GNU Project's `hello world' program
 (which is itself an example for the GNU Project).
Homepage: http://www.gnu.org/software/hello/
```

In der Ausgabe finden sich neben dem Paketnamen (`Package:`) Angaben über den Status und die Priorität des Pakets (von `required` über `important`, `standard` und `optional` bis `extra`) und den ungefähren Themenbereich (`Section:`). Der `Maintainer:` ist die Person, die sich im Debian-Projekt um das Paket kümmert.

Pakete der Priorität `required` sind nötig für einen korrekten Systembetrieb (in der Regel weil `dpkg` von diesen Paketen abhängt). Die Priorität `important` umfasst Pakete, mit deren Anwesenheit man bei einem Unix-artigen System rechnen würde[2]. `standard` fügt diejenigen Pakete hinzu, die für ein überschaubares, aber nicht zu geiziges System, das im Textmodus läuft, sinnvoll sind – diese Priorität beschreibt das, was Sie bekommen, wenn Sie Debian GNU/Linux installieren, ohne sich gleich zusätzliche Software auszusuchen. Die Priorität `optional` gilt für alles, was Sie vielleicht installieren wollen, wenn Sie nicht zu genau hinschauen und keine speziellen Wünsche habe. Dazu gehören Sachen wie die X11-Oberfläche und eine ganze Reihe von Anwendungen (etwa TEX). Innerhalb von `optional` sollte es keine Konflikte geben. In `extra` schließlich landen alle Pakete, die Konflikte mit Paketen anderer Prioritäten haben oder die nur für spezielle Anwendungen nützlich sind.

Prioritäten

Pakete dürfen nicht von Paketen niedrigerer Priorität abhängen. Damit das immer klappt, sind die Prioritäten von ein paar Paketen künstlich angepasst worden.

Wichtig sind die Angaben über Paketabhängigkeiten, von denen es diverse Arten gibt:

Paketabhängigkeiten

`Depends` Die angegebenen Pakete müssen konfiguriert sein, damit das Paket konfiguriert werden kann. Eventuell können (wie im obigen Beispiel) sogar bestimmte Paketversionen nötig sein.

`Pre-Depends` Die angegebenen Pakete müssen fertig installiert sein, bevor überhaupt mit der Installation des Pakets begonnen werden kann. Diese Form von Abhängigkeit wird verwendet, wenn zum Beispiel die Installationsskripte des Pakets zwingend Software aus dem anderen Paket benötigen.

`Recommends` Eine nicht absolute, aber sehr naheliegende Abhängigkeit. Die erwähnten Pakete würde man fast immer gemeinsam mit dem Paket installieren und nur in sehr ungewöhnlichen Umständen darauf verzichten.

`Suggests` Die genannten Pakete sind im Zusammenhang mit dem aktuellen Paket nützlich, aber nicht erforderlich.

`Enhances` Wie `Suggests`, aber andersherum – dieses Paket ist nützlich für das oder die genannten Pakete.

`Conflicts` Das Paket kann nicht gleichzeitig mit den genannten Paketen installiert sein.

Wenn ein Paket lokal gar nicht installiert ist, gibt »`dpkg --status`« nur eine Fehlermeldung aus:

[2] Die Definition ist etwas wie »Ein Paket ist `important`, wenn, falls es fehlt, ein erfahrener Unix-Benutzer den Kopf schütteln und sich fragen würde, was eigentlich hier los ist.«

```
# dpkg -s xyzzy
Package `xyzzy' is not installed and no info is available.
Use dpkg --info (= dpkg-deb --info) to examine archive files,
and dpkg --contents (= dpkg-deb --contents) to list their contents.
```

Liste der Dateien

Die Option `--listfiles` (`-L`) liefert eine Liste der Dateien im Paket:

```
$ dpkg --listfiles hello
/.
/usr
/usr/share
/usr/share/doc
/usr/share/doc/hello
/usr/share/doc/hello/changelog.Debian.gz
/usr/share/doc/hello/copyright
/usr/share/doc/hello/NEWS
/usr/share/doc/hello/changelog.gz
/usr/share/info
/usr/share/info/hello.info.gz
/usr/share/man
/usr/share/man/man1
/usr/share/man/man1/hello.1.gz
/usr/share/locale
◁◁◁◁◁
```

Paketsuche

Schließlich können Sie mit der Option `--search` (kurz `-s`) herausfinden, zu welchem Paket (wenn überhaupt) eine Datei gehört. Suchmuster sind dabei erlaubt:

```
$ dpkg -S bin/m*fs
dosfstools: /sbin/mkdosfs
cramfsprogs: /usr/sbin/mkcramfs
util-linux: /sbin/mkfs.cramfs
smbfs: /sbin/mount.smbfs
◁◁◁◁◁
```

Die Suche kann allerdings einige Zeit dauern.

Wenn Sie wissen wollen, in welchem Paket eine Datei enthalten ist, die sich *nicht* auf Ihrem System befindet – etwa weil Sie als nächstes dann das betreffende Paket installieren möchten –, können Sie die Suchfunktion auf `http://www.debian.org/distrib/packages#search_contents` verwenden. Diese erlaubt Ihnen die gezielte oder auch übergreifende Suche in bestimmten Debian-Distributionen und Rechnerarchitekturen sowie die Suche nach genau passenden Dateinamen wie auch Dateinamen, die einen bestimmten Suchbegriff enthalten.

Übungen

26.6 [3] Wieviele Pakete sind auf Ihrem System installiert, deren Namen mit `lib` anfangen? Wieviele dieser Pakete haben die Priorität `required`?

26.2.6 Verifikation von Paketen

Integrität eines installierten Pakets

Die Integrität eines installierten Pakets kann mit dem Kommando `debsums` (aus dem gleichnamigen Paket) überprüft werden:

```
$ debsums hello
/usr/share/doc/hello/changelog.Debian.gz    OK
```

```
/usr/share/doc/hello/copyright              OK
/usr/share/doc/hello/NEWS                   OK
/usr/share/doc/hello/changelog.gz           OK
/usr/share/info/hello.info.gz               OK
◁◁◁◁◁
```

Dabei werden die MD5-Prüfsummen der einzelnen Dateien mit dem Inhalt der entsprechenden Datei in `/var/lib/dpkg/info` (hier `hello.md5sums`) verglichen. Stimmt die Prüfsumme der tatsächlichen Datei nicht mit dem Vorgabewert überein, erscheint statt `OK` die Meldung `FAILED`.

Mit `debsums` können »versehentliche« Änderungen der Dateien eines Pakets aufgedeckt werden, aber einen Schutz vor Eindringlingen, die mutwillig Dateien ändern, bietet der Mechanismus nicht. Schließlich kann ein Cracker auch in der `.md5sums`-Datei des Pakets die korrekte Prüfsumme eintragen. Ebenso wenig hilft die Methode gegen »trojanische« Pakete, die hinter einer unverfänglichen Fassade bösartigen Code verstecken. Auf das Thema »Integrität von Paketen« kommen wir in Abschnitt 26.4 zurück.

Schutz vor Eindringlingen

Übungen

26.7 [!2] Manipulieren Sie eine Datei in einem installierten Debian-Paket. (Suchen Sie sich ein nicht so wichtiges aus, etwa das aus Übung 26.2.) Sie könnten zum Beispiel – mit `root`-Rechten – ein paar Zeilen an die `README.Debian`-Datei anhängen. Prüfen Sie die Integrität der Dateien des Pakets mit `debsums`.

26.3 Debian-Paketverwaltung der nächsten Generation

26.3.1 APT

`dpkg` ist ein leistungsfähiges Werkzeug, aber in seinen Möglichkeiten doch etwas eingeschränkt. Ärgerlich ist zum Beispiel der Umstand, dass es verletzte Abhängigkeiten zwischen Paketen zwar bemerkt, aber anschließend nur das Handtuch wirft, anstatt konstruktiv zur Behebung des Problems beizutragen. Ferner wünscht man sich die Möglichkeit, über das Installieren lokal vorliegender Pakete hinaus komfortabel zum Beispiel auf FTP- oder Web-Server zugreifen zu können, die Pakete anbieten.

Das Programm `dselect`, das in der Debian-Frühzeit als interaktive Oberfläche zur Paketauswahl diente, ist inzwischen offiziell verpönt – seine Unbequemlichkeit war sprichwörtlich, auch wenn das Lesen des Handbuchs in der Regel half.

Schon relativ früh in der Geschichte des Debian-Projekts (nach heutigen Maßstäben) begann die Gemeinde daher mit der Entwicklung von APT, dem *advanced packaging tool*. Dieses Projekt ist in seiner Bedeutung wie auch ultimaten Zwecklosigkeit vergleichbar mit der Queste der Ritter der Tafelrunde nach dem Heiligen Gral, aber wie die Gralssuche führte auch die APT-Entwicklung zu zahlreichen edlen Taten am Rande. Zwar wurden wenige Drachen erlegt und Burgfräulein befreit, aber es entstanden sehr wichtige und leistungsfähige »Teillösungen« des Problems, deren Komfort und Leistungsumfang ihresgleichen suchen (einige eigentlich RPM-basierte Distributionen sind deswegen dazu übergegangen, diese für ihre Zwecke zu ad-»apt«-ieren).

26.3.2 Paketinstallation mit `apt-get`

Das erste dieser Werkzeuge ist `apt-get`, das eine Art intelligenten Überbau für `dpkg` darstellt. Es bietet keine interaktive Oberfläche zur Paketauswahl an, sondern konnte zunächst als *back-end* für `dselect` benutzt werden, um in `dselect` ausgewählte Pakete zu installieren. Heutzutage ist es vor allem auf der Kommandozeile nützlich. Zu den wichtigsten Eigenschaften von `apt-get` gehören die folgenden:

Menge von Installationsquellen

- `apt-get` kann eine Menge von Installationsquellen gleichzeitig verwalten. Beispielsweise ist es möglich, eine »stabile« Debian-Distribution auf CD parallel zu einem HTTP-basierten Server mit Sicherheits-Updates zu verwenden. Pakete werden normalerweise von CD installiert; nur wenn der HTTP-Server eine aktuellere Version des Pakets anbietet, wird das Paket über das Netz geholt. Bestimmte Pakete können aus bestimmten Quellen angefordert werden, zum Beispiel können Sie weitestgehend eine stabile Debian-Distribution benutzen, aber ein paar Pakete der neueren *unstable*-Distribution entnehmen.

Alles automatisch aktualisieren

- Die ganze Distribution kann automatisch aktualisiert werden (mit »`apt-get dist-upgrade`«), auch wenn Pakete umbenannt oder gelöscht wurden.

Vielzahl von Hilfsprogrammen

- Eine Vielzahl von Hilfsprogrammen erlaubt es etwa, cachende Proxy-Server für Debian-Pakete aufzubauen (`apt-proxy`), Pakete auf Rechnern zu installieren, die nicht direkt am Internet sind (`apt-zip`), oder vor der Installation eines Pakets eine Liste der für das Paket gemeldeten Fehler anzuzeigen (`apt-listbugs`). Mit `apt-build` können Sie Pakete speziell für Ihre Systeme optimiert übersetzen und eine lokale Paketquelle mit solchen Paketen zusammenstellen.

Paketquellen

Paketquellen für `apt-get` werden in `/etc/apt/sources.list` vereinbart:

```
deb http://ftp.de.debian.org/debian/ stable main
deb http://security.debian.org/ stable/updates main
deb-src http://ftp.de.debian.org/debian/stable main
```

Binärpakete werden von `http://ftp.de.debian.org/` geholt, genau wie der dazugehörige Quellcode. Außerdem wird der Server `security.debian.org` mit eingebunden, wo das Debian-Projekt aktualisierte Paketversionen zur Verfügung stellt, in denen Sicherheitsprobleme repariert sind.

Arbeitsweise

Die allgemeine Arbeitsweise mit `apt-get` ist wie folgt: Zunächst aktualisieren Sie die lokalen Informationen über verfügbare Pakete:

```
# apt-get update
```

Hiermit werden sämtliche Paketquellen konsultiert und die Resultate in eine gemeinsame Paketliste aufgenommen. Pakete installieren Sie mit »`apt-get install`«:

```
# apt-get install hello
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden NEUEN Pakete werden installiert:
  hello
0 aktualisiert, 1 neu installiert, 0 zu entfernen und 529 nicht▷
 ◁ aktualisiert.
Es müssen noch 0 B von 68,7 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 566 kB Plattenplatz zusätzlich benutzt.
Vormals nicht ausgewähltes Paket hello wird gewählt.
(Lese Datenbank ... 381459 Dateien und Verzeichnisse sind derzeit▷
 ◁ installiert.)
```

```
Entpacken von hello (aus .../archives/hello_2.8-2_amd64.deb) ...
Trigger für install-info werden verarbeitet ...
Trigger für man-db werden verarbeitet ...
hello (2.8-2) wird eingerichtet ...
```

Dabei werden alle in `Depends:`-Abhängigkeiten erwähnten Pakete ebenfalls installiert oder auf den erforderlichen Versionsstand aktualisiert, genau wie jegliche Pakete, von denen diese Pakete abhängen und so weiter.

Sie können auch mehrere Pakete gleichzeitig installieren:

```
# apt-get install hello python
```

Oder einige Pakete installieren und andere gleichzeitig deinstallieren: Das Kommando

```
# apt-get install hello- python python-django+
```

würde das Paket `hello` entfernen und die Pakete `python` und `python-django` (mit ihren Abhängigkeiten) installieren (das »+« ist nicht erforderlich, aber erlaubt). Mit »`apt-get remove`« können Sie Pakete direkt entfernen.

Einfache Aktualisierung

Das Kommando »`apt-get upgrade`« installiert die neuesten verfügbaren Versionen aller im System vorhandenen Pakete. Dabei werden aber keine installierten Pakete entfernt und keine neuen Pakete installiert; Pakete, die sich ohne solche Aktionen nicht aktualisieren lassen (weil sich Abhängigkeiten geändert haben), bleiben auf ihrem älteren Stand.

»Intelligente« Aktualisierung

Mit dem Kommando »`apt-get dist-upgrade`« wird ein »intelligentes« Konfliktauflösungsschema aktiviert, das versucht, geänderte Abhängigkeiten durch das gezielte Entfernen und Installieren von Paketen zu erfüllen. Dabei werden gemäß der Priorität wichtigere Pakete gegenüber weniger wichtigen bevorzugt.

Quellcode

Den Quellcode zu einem Paket holen Sie sich mit dem Kommando »`apt-get source`«:

```
# apt-get source hello
```

Dies funktioniert auch, wenn das Binärpaket eines von mehreren ist, die aus demselben (anders heißenden) Quellcodepaket erstellt werden.

Die `apt`-Programme werden in der Datei `/etc/apt/apt.conf` konfiguriert. Hier können Sie Optionen für `apt-get`, `apt-cache` und andere Kommandos aus dem `apt`-Umfeld unterbringen.

Übungen

26.8 [!1] Verwenden Sie `apt-get`, um das Paket `hello` zu installieren und anschließend wieder zu entfernen.

26.9 [1] Laden Sie mit `apt-get` den Quellcode für das Paket `hello` herunter.

26.3.3 Informationen über Pakete

`apt-cache`

Ein nützliches Programm ist `apt-cache`, das die Paketquellen von `apt-get` durchsucht:

```
$ apt-cache search hello                          hello im Namen oder Beschreibung
gpe-othello - Brettspiel Othello für GPE
grhino - Othello/Reversi Brettspiel
gtkboard - viele Brettspiele in einem Programm
hello - Der klassische Gruß und ein gutes Beispiel
$ apt-cache show hello                            Informationen über hello
```

```
Package: hello
Version: 2.8-2
Installed-Size: 553
Maintainer: Santiago Vila <sanvila@debian.org>
Architecture: amd64
◁◁◁◁◁
```

Die Ausgabe von »`apt-cache show`« entspricht im wesentlichen der von »`dpkg --status`«, bis darauf, dass es für alle Pakete in einer Paketquelle funktioniert, egal ob sie lokal installiert sind, während `dpkg` sich nur mit lokal installierten Paketen befasst.

Es gibt auch noch ein paar andere interessante Subkommandos von `apt-cache`: `depends` zeigt alle Abhängigkeiten eines Pakets an sowie die Namen der Pakete, die diese Abhängigkeit erfüllen können:

```
$ apt-cache depends hello
hello
  Hängt ab von: libc6
 |Hängt ab von: dpkg
  Hängt ab von: install-info
```

Der vertikale Balken in der zweiten Zeile der Abhängigkeiten deutet an, dass die Abhängigkeit in dieser Zeile oder die in der darauffolgenden erfüllt sein müssen. Im Beispiel muss also das Paket `dpkg` oder das Paket `install-info` installiert sein.

`rdepends` liefert umgekehrt die Namen aller Pakete, die von dem genannten Paket abhängen:

```
$ apt-cache rdepends python
python
Reverse Depends:
  libboost-python1.46.1
  mercurial-nested
  mercurial-nested
  python-apt
  python-apt
◁◁◁◁◁
```

Wenn ein Paket in der Liste mehrmals vorkommt, dann wahrscheinlich, weil das ursprüngliche Paket in seiner Abhängigkeitenliste mehrmals vorkommt, typischerweise mit Versionsnummern. Das Paket `python-apt` enthält zum Beispiel unter anderem

```
… python (>= 2.6.6-7~), python (<< 2.8), …
```

um zu signalisieren, dass es nur mit bestimmten Versionen des Debian-Python-Pakets funktioniert.

`stats` gibt einen Überblick über den Inhalt des Paket-Caches aus:

```
$ apt-cache stats
Total package names: 33365 (1335k)                    Alle Pakete im Cache
  Normal packages: 25672                              Pakete, die es wirklich gibt
  Pure virtual packages: 757                          Platzhalter für Funktionalität
  Single virtual packages: 1885                       Nur eine Implementierung
  Mixed virtual packages: 267                         Mehrere Implementierungen
  Missing: 4784                  Pakete in Abhängigkeiten, die es nicht (mehr?) gibt
Total distinct versions: 28955 (1506k)                Paketversionen im Cache
```

```
Total distinct descriptions: 28955 (695k)
Total dependencies: 182689 (5115k)
Total ver/file relations: 31273 (500k)
Total Desc/File relations: 28955 (463k)
Total Provides mappings: 5747 (115k)
Total globbed strings: 100 (1148)
Total dependency version space: 756k
Total slack space: 73.5k
Total space accounted for: 8646k
```

Anzahl der paarweisen Beziehungen

Übungen

26.10 [2] Wie können Sie *alle* Pakete bestimmen, die installiert sein müssen, damit ein bestimmtes Paket funktioniert? (Vergleichen Sie die Ausgabe von »`apt-cache depends x11-apps`« und »`apt-cache depends libxt6`«.)

26.3.4 `aptitude`

Das Programm `aptitude` dient zur Paketauswahl und -verwaltung und ist bei Debian GNU/Linux an die Stelle des alten `dselect` getreten. Es stellt auf der Konsole oder in einem Terminal(emulator) eine interaktive Oberfläche mit Menüs, Dialogen und ähnlichem zur Verfügung, aber bietet auch Kommandooptionen, die lose mit denen von `apt-get` kompatibel sind. Seit Debian 4.0 (vulgo »etch«) ist `aptitude` das empfohlene Programm für Paketinstallation und -aktualisierung.

Neuere Versionen von `aptitude` enthalten auch eine GTK+-basierte Oberfläche, die Sie zusätzlich installieren können.

Verbesserungen

Verglichen mit `apt-get` und `dselect` bietet `aptitude` diverse Verbesserungen an. Unter anderem:

- Es muss nicht notwendigerweise als `root` aufgerufen werden, sondern fragt selbstständig das `root`-Kennwort ab, bevor Aktionen durchgeführt werden, die Administratorrechte verlangen.
- Es kann sich merken, welche Pakete aufgrund von Abhängigkeiten installiert wurden, und diese automatisch entfernen, wenn alle Pakete, die von ihnen abhängen, entfernt worden sind. (Inzwischen hat `apt-get` das auch gelernt; siehe `apt-get`(8), Kommando `autoremove`.)
- Mit `aptitude` haben Sie interaktiven Zugriff auf alle Versionen eines Pakets, die in den verschiedenen Paketquellen zur Verfügung stehen, nicht nur die aktuelle.

interaktive Oberfläche

Mit `aptitude` kommen Sie in die interaktive Oberfläche (Bild 26.1). Am oberen Rand der Konsole (oder des Terminals, je nachdem) sehen Sie eine »Menüleiste«, darunter eine Zeile mit einem kurzen Hilfetext und eine Zeile mit der Versionsnummer des Programms. Der Rest des Bildschirms ist in zwei Teile getrennt: Die obere Hälfte zeigt eine Übersicht über die verschiedenen Arten von Paketen (aktualisierbar, neu, installiert und so weiter), die untere Hälfte ist für erläuternde Texte gedacht.

Paketliste

Mit den Tasten ↑ und ↓ können Sie in der Paketliste navigieren. Zeilen, die mit `---` anfangen, sind Überschriften eines »Teilbaums« der Paketliste, und ↩ dient zum »Aufklappen« der nächsten Ebene eines solchen Teilbaums. (Den kompletten Teilbaum können Sie mit] aufklappen.) Mit / bekommen Sie ein Fenster, in dem Sie einen Suchbegriff (oder regulären Ausdruck) für einen Paketnamen eingeben können. Beim Blättern durch die Paketlisten werden im unteren Teil des Bildschirms Erläuterungen der betreffenden Pakete angezeigt, die Sie mit den Tasten a und z nach oben oder unten rollen können. Die Taste i erlaubt den Wechsel vom Erklärungstext zu einer Darstellung der Abhängigkeiten.

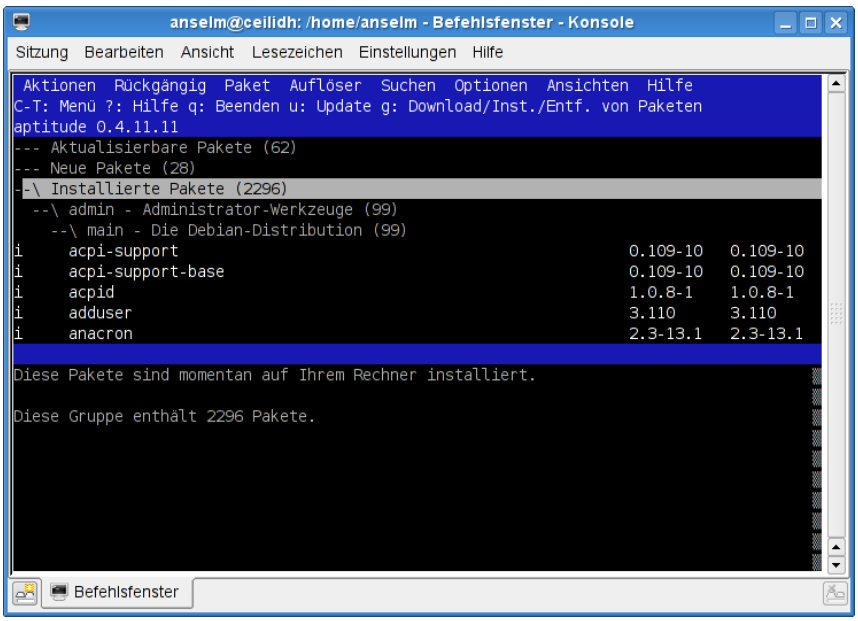


Bild 26.1: Das Programm `aptitude`

Wenn Sie den Cursorbalken auf der Zeile für ein Paket stehen haben, können Sie es mit + zur Installation oder Aktualisierung auswählen oder (falls es installiert ist) mit - zum Entfernen vormerken. Wenn Sie es restlos entfernen wollen (à la »`dpkg --purge`«), verwenden Sie _. = setzt den Zustand eines Pakets auf *hold*, das heißt, es wird nicht mehr automatisch aktualisiert.

Mit u können Sie die Paketlisten aktualisieren (analog zu »`apt-get update`«) und anschließend im Teilbaum »Aktualisierbare Pakete« nachschauen, welche Pakete `aptitude` aktualisieren würde. Mit U können Sie alle diese Pakete tatsächlich für die Aktualisierung vormerken. Im Teilbaum »Neue Pakete« sehen Sie diejenigen Pakete, die seit der letzten Aktualisierung neu dazugekommen sind; mit f können Sie diese Liste leeren und die Pakete in die »normalen« Listen einsortieren. Die Tastenkombination Strg+t öffnet die Menüleiste, in der Sie ebenfalls mit den Pfeiltasten navigieren und mit ↩ eine Funktion auswählen können.

Das Kommando g startet die Installation, Aktualisierung oder das Entfernen von Paketen. Zunächst sehen Sie eine Übersicht der vorgeschlagenen Aktionen, die Sie noch mit den üblichen Kommandos modifizieren können. Ein weiteres g stößt die eigentliche Arbeit an: Zunächst werden alle benötigten neuen Pakete geholt, dann verwendet `aptitude` wie üblich `dpkg` zur Installation und Entfernung der gewünschten Pakete.

Entgegen einer verbreiteten Annahme ist `aptitude` keine Oberfläche für `apt-get`, sondern erledigt das, was sonst `apt-get` tut, selbst.

Lösungsstrategien

Treten Konflikte auf, so bietet `aptitude` Lösungsstrategien in Form von Vorschlägen für Paketinstallationen, -aktualisierungen oder -entfernungen an, unter denen Sie sich die passende aussuchen können.

In der Standardkonfiguration installiert `aptitude` automatisch auch diejenigen Pakete, die ein Paket als `Recommended:` bezeichnet. Das ist nicht immer erwünscht und kann im Menü »Optionen« ausgeschaltet werden.

Bei Ubuntu können Sie `aptitude` installieren und benutzen, aber es ist nicht das empfohlene Programm. Aus diesem Grund verträgt es sich nicht zu 100% mit den von Ubuntu vorgeschlagenen grafischen Werkzeugen zur Paketverwaltung – Sie sollten also entweder konsequent alles machen, wie Ubuntu es Ihnen empfiehlt, oder aber konsequent `aptitude` verwenden.

26.4 Integrität von Debian-Paketen

Das Programm `debsums` dient dazu, die Integrität der Dateien in einem einzelnen Paket zu überprüfen (Abschnitt 26.2.6). Das ist nett, aber stellt nicht sicher, dass nicht ein Angreifer sowohl die Dateien im Paket als auch die `.md5sums`-Datei mit den Prüfsummen manipuliert hat. Die Frage ist also, wie das Debian-Projekt die Integrität kompletter Pakete (und darauf aufbauend die Integrität der Distribution) sicherstellt. Und das geht wie folgt:

Integrität kompletter Pakete

- Jedes Debian-Paket wird von einem Debian-Entwickler kryptografisch signiert. Das heißt, der Empfänger eines Pakets kann mit Hilfe des öffentlichen Schlüssels des Entwicklers verifizieren, dass er das Paket tatsächlich so erhalten hat, wie es von diesem freigegeben wurde.

  Der Debian-Entwickler, der das Paket signiert hat, muss es nicht unbedingt auch selber zusammengestellt haben. Grundsätzlich darf jeder Debian-Entwickler jedes Paket in Debian signieren und zur Veröffentlichung freigeben (ein *non-maintainer upload*), und das wird zur zeitnahen Korrektur kritischer Sicherheitslücken oder zur Fortführung »verwaister« Pakete auch gemacht. Ferner gibt es zahlreiche Leute, die an Debian GNU/Linux mitarbeiten und formell keine Debian-Entwickler sind (oder deren Entwicklerstatus noch in Bearbeitung ist), aber sich dennoch um Pakete kümmern. Diese Leute können selber keine Pakete zur Veröffentlichung freigeben, sondern müssen das über einen »Sponsor« tun, der Debian-Entwickler sein muss. Der Sponsor übernimmt dabei die Verantwortung dafür, dass das Paket vernünftig ist.

  Sie sollten den Sicherheitsgewinn durch digitale Signaturen nicht überbewerten: Die Signatur des Entwicklers garantiert nicht, dass sich im Paket kein schädlicher Code versteckt, sondern nur, dass der Entwickler das Paket signiert hat. Theoretisch ist es möglich, dass ein Cracker das Aufnahmeverfahren als Debian-Entwickler durchläuft und ganz offiziell Pakete in die Distribution tun kann – deren Steuerskripte von den meisten Anwendern unkritisch mit `root`-Rechten ausgeführt werden. Die meisten anderen Linux-Distributionen haben die gleichen Schwächen.

- Die Debian-Infrastruktur nimmt nur solche Pakete zur Veröffentlichung an, die von einem Debian-Entwickler signiert worden sind.

- Auf dem Debian-Server (und allen anderen Server, die Debian GNU/Linux von ihm übernehmen) steht für jede aktuelle Debian-Distribution eine Datei (oder mehrere) namens `Packages.gz`. Diese Datei enthält die MD5-Prüfsummen aller Pakete in der Distribution, so wie sie auf dem Server stehen; da der Server nur Pakete von akkreditierten Entwicklern annimmt, sind diese also authentisch.

- Zu jeder aktuellen Debian-Distribution auf dem Server gehört ferner eine Datei namens `Release`, die die MD5-Prüfsumme der beteiligten `Packages.gz`-Datei(en) enthält. Diese Datei wird kryptografisch signiert (die Signatur steht in einer Datei namens `Release.gpg`).

Mit dieser Kette von Prüfsummen und Signaturen kann die Integrität von Paketen in der Distribution überprüft werden:

- Ein neues Paket wird heruntergeladen und die MD5-Prüfsumme des heruntergeladenen Pakets bestimmt.

- Es wird geprüft, ob die Signatur der `Release`-Datei korrekt ist, und wenn ja, wird die MD5-Prüfsumme von `Packages.gz` aus dieser Datei gelesen.

- Anhand dieser Prüfsumme wird die Integrität der tatsächlich vorliegenden Datei `Packages.gz` verifiziert.
- Die MD5-Prüfsumme des Pakets, die in `Packages.gz` angegeben ist, muss mit der des tatsächlich heruntergeladenen Pakets übereinstimmen.

Stimmt die MD5-Prüfsumme des tatsächlich heruntergeladenen Pakets nicht mit dem »Sollwert« aus `Packages.gz` überein, dann wird der Administrator auf diesen Umstand aufmerksam gemacht und das Paket (zunächst) nicht installiert.

Es ist möglich, ein Debian-System so zu konfigurieren, dass es *nur* Pakete installiert, die sich auf diesem Weg überprüfen lassen. (Im Normalfall bleibt es bei Warnungen, die sich aber notfalls manuell übersteuern lassen.) Damit könnten Sie zum Beispiel in einem Unternehmen eine Infrastruktur aufbauen, in der nur solche Pakete installiert werden können, die aus einer »Teildistribution« von als sicher und sinnvoll erachteten Paketen kommen. Dies können sowohl Pakete aus Debian GNU/Linux als auch lokal selbst erstellte Pakete sein.

Die APT-Infrastruktur vertraut nur Paketquellen, für die ein öffentlicher GnuPG-Schlüssel in der Datei `/etc/apt/trusted.gpg` hinterlegt ist. Das Programm `apt-key` dient zur Wartung dieser Datei.

Die aktuellen öffentlichen Schlüssel für die Debian-Paketquellen stehen im Paket `debian-archive-keyring` und können durch Aktualisierungen dieses Pakets erneuert werden. (Debian rotiert die Schlüssel im jährlichen Turnus).

Näheres über den Umgang mit signierten Paketen in Debian finden Sie in [F+07, Kapitel 7]. GnuPG erklären wir in der Linup-Front-Schulungsunterlage *Linux-Administration II*.

26.5 Die debconf-Infrastruktur

Bei der Installation von Softwarepaketen treten mitunter Fragen auf. Wenn Sie zum Beispiel ein Mailserverprogramm installieren, ist es zum Erzeugen einer passenden Konfigurationsdatei wichtig zu wissen, ob der betreffende Rechner direkt im Internet ist, in einem lokalen Netz mit einem designierten zentralen Mailserver steht oder über einen Wählzugang angebunden ist. Ebenso ist es nötig, zu wissen, welche Domain der Rechner für seine Nachrichten verwenden soll und so weiter.

Der debconf-Mechanismus ist dafür gedacht, die entsprechenden Informationen bei der Installation zu erheben und zur späteren Verwendung abzuspeichern. Er ist also im wesentlichen eine Datenbank für systemweite Konfigurationseinstellungen, auf die zum Beispiel die Installationsskripten eines Pakets zugreifen können. Zur Manipulation der Datenbank unterstützt debconf modulare Bedienungsoberflächen, die von sehr simplen Texteingaben über textorientierte Dialoge und verschiedene grafische Arbeitsumgebungen wie KDE und GNOME alle Ansprüche abdecken. Es gibt auch Schnittstellen zu gängigen Programmiersprachen wie Python.

Sie können die anfängliche debconf-basierte Konfiguration eines Softwarepakets jederzeit wiederholen, indem Sie ein Kommando wie

```
# dpkg-reconfigure mein-paket
```

geben. `dpkg-reconfigure` wiederholt dann die bei der ursprünglichen Installation des Pakets gestellten Fragen und verwendet dabei die voreingestellte Bedienungsoberfläche.

Mit der Option `--frontend` (oder `-f`) können Sie sich für dieses Mal eine andere Bedienungsoberfläche wünschen. Die möglichen Namen können Sie in `debconf`(7) nachschlagen, sofern Sie das Paket `debconf-doc` installiert haben. Standard ist `dialog`.

Um die Bedienungsoberfläche zeitweilig zu ändern, wenn Sie `dpkg-reconfigure` nicht direkt aufrufen, können Sie die Umgebungsvariable `DEBCONF_FRONTEND` verwenden:

```
# DEBCONF_FRONTEND=noninteractive aptitude upgrade
```

Mit `dpkg-reconfigure` haben Sie außerdem die Möglichkeit, zu bestimmen, wie detailliert Sie nach Ihren Wünschen gefragt werden wollen. Dazu können Sie die Option `--priority` (oder `-p`) verwenden, gefolgt von einer Priorität. Die möglichen Prioritäten sind (in absteigender Folge):

`critical` Fragen, die Sie auf jeden Fall beantworten müssen, weil sonst etwas ganz Schlimmes passiert.

`high` Fragen ohne vernünftige Voreinstellung – Ihre Meinung zählt.

`medium` Fragen mit vernünftiger Voreinstellung.

`low` Triviale Fragen mit meist funktionierender Voreinstellung.

Wenn Sie etwas wie

```
# dpkg-reconfigure --priority=medium mein-paket
```

sagen, bekommen Sie alle Fragen der Prioritäten `critical`, `high` und `medium` gestellt; die Fragen der Priorität `low` werden übersprungen.

Für zeitweilige Änderungen beim indirekten Aufruf von debconf gibt es auch die Umgebungsvariable `DEBCONF_PRIORITY`.

Die debconf-Infrastruktur ist ziemlich komplex, aber nützlich. So ist es zum Beispiel möglich, die Antworten zum Beispiel in einer LDAP-Datenbank zu hinterlegen, auf die dann alle Rechner in einem Netz zugreifen können. Sie können also eine große Anzahl von Rechnern ohne manuelle Intervention installieren. Dies im Detail zu erklären würde hier allerdings viel zu weit führen.

Übungen

26.11 [1] Wie können Sie die voreingestellte Bedienungsoberfläche von debconf dauerhaft ändern?

26.6 `alien`: Pakete aus fremden Welten

Viele Softwarepakete stehen nur in dem verbreiteten RPM-Format zur Verfügung. Gerade kommerzielle Pakete werden eher für die Red-Hat- oder SUSE-Distributionen angeboten, obwohl grundsätzlich nichts dagegen spricht, die Software auch unter Debian GNU/Linux auszuprobieren (dem ernsthaften Einsatz steht möglicherweise entgegen, dass der Softwarehersteller dafür keine Unterstützung leistet). Sie können RPM-Pakete nicht direkt auf einem Debian-System installieren, aber das Programm `alien` erlaubt es, die Paketformate diverser Linux-Distributionen – neben RPM auch die Stampede- und Slackware-Formate (nicht dass man die dringend bräuchte) – ins Debian-Paketformat umzuwandeln (und umgekehrt).

Wichtig: Mit `alien` können Sie zwar Pakete von einem Paketformat in ein anderes umwandeln, aber es ist in keiner Weise garantiert, dass Sie mit dem Paket hinterher auch etwas anfangen können. Zum einen können die Programme in dem Paket von Bibliotheken abhängen, die in der Zieldistribution nicht (oder nicht in der richtigen Version) zur Verfügung stehen – da `alien` sich nicht um Abhängigkeiten kümmert, müssen Sie allfällige Probleme dieser Art selber »zu Fuß« beheben.

Zum anderen ist es gut möglich, dass das Paket sich bei der Ursprungsdistribution auf eine Weise ins System einklinkt, die bei der Zieldistribution nicht oder nicht in derselben Weise nachzuvollziehen ist.

Grundsätzlich gilt: Je weiter »unten« ein Paket im System angesiedelt ist, desto geringer ist die Wahrscheinlichkeit, dass `alien` das tut, was Sie wollen. Bei Paketen, die aus ein paar ausführbaren Programmen ohne absonderliche Bibliotheksabhängigkeiten, den dazugehörigen Handbuchseiten und ein paar Beispieldateien bestehen, stehen die Chancen gut, dass `alien` nach Wunsch funktioniert. Mit Systemdiensten, die zum Beispiel in den Systemstartvorgang integriert werden müssen, kann es schon anders aussehen. Und Sie sollten nicht mal auf die Idee kommen, etwa die `libc` ersetzen zu wollen …

`alien` wird insbesondere gebraucht, um LSB-konforme Softwarepakete auf einem Debian-GNU/Linux-System zu installieren – die LSB setzt ja RPM als Paketformat für die Softwareverteilung voraus.

Nach dieser Vorrede zeigen wir Ihnen noch schnell, wie Sie mit `alien` ein RPM-Paket in ein Debian-Paket umwandeln können:

```
# alien --to-deb paket.rpm
```

(Wobei `--to-deb` den Standardfall darstellt und auch weggelassen werden darf.) Den umgekehrten Weg gehen Sie mit

```
# alien --to-rpm paket.deb
```

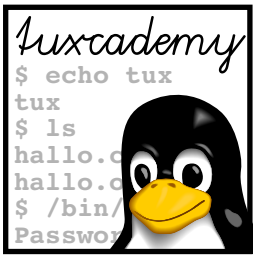
Zum Auseinandernehmen und Zusammensetzen von RPM-Paketen muss das Programm `rpm` installiert sein (das es für Debian GNU/Linux als Paket gibt); zum Zusammensetzen von `deb`-Paketen brauchen Sie ein paar einschlägige Debian-Pakete, die in `alien`(1p) aufgelistet sind. (Auseinandernehmen können Sie `deb`-Pakete, wie in Abschnitt 26.2.1 angedeutet, auf fast allen Linux-Systemen mit »Bordmitteln« wie `tar`, `gzip` und `ar`.)

Kommandos in diesem Kapitel

alien Konvertiert verschiedene Software-Paketformate `alien`(1) 453
apt-get Komfortable Oberfläche für Debian-GNU/Linux-Paketverwaltung `apt-get`(8) 445
aptitude Komfortables Werkzeug zur Paketinstallation und -wartung (Debian) `aptitude`(8) 449
dpkg Verwaltungswerkzeug für Debian-GNU/Linux-Pakete `dpkg`(8) 438
dpkg-reconfigure Rekonfiguriert ein bereits installiertes Debian-Paket `dpkg-reconfigure`(8) 452

Literaturverzeichnis

F+07 Javier Fernández-Sanguino Peña, et al. »Securing Debian Manual«, 2007. `http://www.debian.org/doc/manuals/securing-debian-howto/`



27

Paketverwaltung mit RPM & Co.

Inhalt

27.1 Einleitung 456
27.2 Paketverwaltung mit rpm 457
27.2.1 Installation und Update 457
27.2.2 Deinstallation von Paketen 457
27.2.3 Datenbank- und Paketanfragen 458
27.2.4 Verifikation von Paketen 460
27.2.5 Das Programm rpm2cpio. 461
27.3 YUM 461
27.3.1 Überblick. 461
27.3.2 Paketquellen 461
27.3.3 Pakete installieren und entfernen mit YUM 462
27.3.4 Informationen über Pakete 464
27.3.5 Pakete nur herunterladen 466

Lernziele

- Grundzüge von RPM und verwandten Werkzeugen kennen
- rpm zur Paketverwaltung verwenden können
- YUM einsetzen können

Vorkenntnisse

- Kenntnisse der Linux-Systemadministration
- Erfahrung mit einer RPM-basierten Distribution ist hilfreich

27.1 Einleitung

Der *RPM Package Manager* (oder kurz RPM) ist ein Werkzeug zur Verwaltung von Softwarepaketen. Er erlaubt die einfache Installation und Deinstallation von Software, wobei dafür gesorgt wird, dass sich unterschiedliche Pakete nicht in die Quere kommen bzw. dass Abhängigkeiten zwischen den Paketen berücksichtigt werden. Außerdem erlaubt es RPM Ihnen, detaillierte Informationen über Pakete zu erheben sowie die Integrität von Paketen sicherzustellen.

Der Kern von RPM ist eine Datenbank. Bei ihr melden sich Softwarepakete beim Installieren und Deinstallieren an bzw. ab. Dazu müssen die Softwarepakete in einer standardisierten Form vorliegen, eben als RPM-Pakete.

Das RPM-Paketformat wird von vielen Distributionen (unter anderem denen von Red Hat, Novell/SUSE, TurboLinux und Mandriva) verwendet. Ein beliebiges RPM-Paket kann allerdings normalerweise *nicht* bedenkenlos auf jeder RPM-basierten Distribution installiert werden: Da das RPM-Paket die Software in schon übersetzter Form enthält, muss das Paket zur benutzten Prozessor-Architektur passen; da Dateisystem-Struktur, die Form der Dienststeuerung (etwa Init-Skripte) und die paketinterne Beschreibung von Paketabhängigkeiten sich von Distribution zu Distribution, mitunter auch zwischen verschiedenen Versionen einer Distribution unterscheiden, kann das Quer-Installieren zu Problemen führen.

RPM ist ursprünglich eine Erfindung von Red Hat und hieß deswegen auch zuerst *Red Hat Package Manager*. Seitdem diverse andere Distributionen sich auch dieses Programms bedienen, wurde er jedoch in *RPM Package Manager* umbenannt.

Im Moment besteht eine gewisse Kontroverse darüber, wer für die Weiterentwicklung dieses kritischen Stücks Infrastruktur zuständig ist. Nach einer langen Pause, in der sich niemand wirklich beflissen zeigte, eine standardisierte Version herauszugeben, haben 2006/2007 einige Fedora-Entwickler versucht, die Weiterentwicklung von RPM als offiziell distributionsabhängiges Produkt anzugehen (die Federführung liegt inzwischen bei Panu Matilainen von Red Hat, während Entwickler von einigen anderen RPM-benutzenden Distributionen zuarbeiten). Unabhängig davon beschäftigt sich auch Jeff Johnson, der letzte offizielle RPM-Entwickler bei Red Hat (der inzwischen nicht mehr bei Red Hat arbeitet), mit der Weiterentwicklung von RPM und bezeichnet seinen Code als »the official code base« – bis darauf, dass sich von den Linux-Distributionen niemand so wirklich dafür zu interessieren scheint.

Dateinamen

Ein RPM-Paket hat einen zusammengesetzten Dateinamen, beispielsweise

```
openssh-3.5p1-107.i586.rpm
```

der normalerweise aus dem Paketnamen (`openssh-3.5p1-107`), der Architektur (`i586`) und der Endung `.rpm` besteht. Der Paketname dient zur internen Identifizierung des Pakets, wenn es installiert ist. Er setzt sich zusammen aus dem Namen der Software (`openssh`), der Version der Software, so wie die *Entwickler* sie ihr gegeben haben (`3.5p1`), gefolgt von einer Release-Nummer (`107`), die ihr der Paket-Bauer, also der *Distributor*, gegeben hat.

Basis-Modus

Der *RPM Package Manager* wird über das Kommando `rpm` aufgerufen, gefolgt von einem Basis-Modus. Die wichtigsten Modi werden im Folgenden besprochen – bis auf die Modi für das Einrichten der RPM-Datenbank und das Bauen und Signieren von RPM-Paketen, die für LPIC-1 keine Rolle spielen.

Optionen

Es gibt eine Reihe von globalen Optionen sowie ergänzende, modusspezifische Optionen. Weil einige Modi und ergänzende Optionen gleich heißen, muss der Modus (anders als bei `tar`) zwingend als Erstes kommen.

Globale Optionen sind u. a. `-v` und `-vv`, die den Wortreichtum der Ausgabe erhöhen (engl. *verbose*, »wortreich«).

Die Konfiguration von RPM ist im Verzeichnis `/usr/lib/rpm` abgelegt; lokale oder individuelle Anpassungen erfolgen in `/etc/rpmrc` bzw. `~/.rpmrc`, dürften aber für den Normalbetrieb nicht notwendig sein.

Konfiguration

27.2 Paketverwaltung mit `rpm`

27.2.1 Installation und Update

Ein RPM-Paket wird durch den Modus `-i` gefolgt vom Pfad zur Paketdatei installiert, also beispielsweise

```
# rpm -i /tmp/openssh-3.5p1-107.i586.rpm
```

Dabei können Sie den Pfad auch als HTTP- oder FTP-URL angeben und so Paketdateien installieren, die auf entfernten Servern liegen. Die Angabe mehrerer Pakete auf einmal ist erlaubt, wie in »`rpm -i /tmp/*.rpm`«.

Daneben gibt es noch die beiden verwandten Modi `-U` (engl. *upgrade*) und `-F` (engl. *freshen*). Ersterer entfernt zusätzlich evtl. vorhandene ältere Versionen eines zu installierenden Pakets, während letzterer das Paket nur installiert, wenn es eine ältere Version gibt (die dann entfernt wird).

Alle drei Modi erlauben eine Reihe von Optionen, die zwingend *hinter* dem Modus aufgeführt werden müssen. Neben `-h` (engl. *hashmark* = »#«; für einen Fortschrittsbalken) gibt es `--test`, was eine Installation verhindert und nur auf mögliche Konflikte testet.

Bei einem Konflikt wird das entsprechende Paket nicht installiert. Konflikte entstehen, wenn

- ein bereits installiertes Paket erneut installiert werden soll,
- ein Paket installiert werden soll, obwohl es in einer anderen Version (Modus `-i`) oder einer aktuelleren Version (Modus `-U`) bereits installiert ist,
- die Installation eine Datei überschreiben würde, die einem anderen Paket gehört,
- ein Paket ein anderes benötigt, das nicht ebenfalls installiert wird oder schon installiert ist.

Sollte die Installation deshalb fehlschlagen, so können Sie sie durch Optionen erzwingen. Beispielsweise wird durch `--nodeps` die Abhängigkeitsprüfung (engl. *dependencies*) deaktiviert.

Durch weitere Optionen lässt sich darüber hinaus die Installation selbst (und nicht nur die Sicherheitsüberprüfung) beeinflussen. Beispielsweise können Sie Pakete bei der Installation in andere Verzeichnisse verschieben. Nur so ist es etwa möglich, sowohl Apache 1.3 als auch Apache 2.0 parallel zu installieren, da normalerweise beide `/sbin/httpd` für sich reklamieren würden: Einer von beiden muss nach `/usr/local` ausweichen.

27.2.2 Deinstallation von Paketen

Die Deinstallation erfolgt durch den Modus `-e` (engl. *erase*, »löschen«), also beispielsweise

```
# rpm -e openssh-3.5p1-107
```

Dabei ist zu beachten, dass Sie hier den *internen Paketnamen* angeben und nicht den Pfad zum Paket, denn RPM merkt sich diesen nicht. (Im nächsten Abschnitt werden wir sehen, wie Sie den Paketnamen in Erfahrung bringen können.) Sie können den Paketnamen auch gekürzt angeben, falls er eindeutig ist: Gibt es kein weiteres Paket `openssh`, so dürften Sie es daher auch mit

```
# rpm -e openssh
```

entfernen. Auch hier achtet RPM darauf, dass Sie keine Pakete entfernen, von deren Anwesenheit andere Pakete abhängen.

Die Optionen `--test` und `--nodeps` haben die gleiche Bedeutung wie beim Installieren; auch sie müssen nach dem Modus erscheinen.

Konfigurationsdateien

Beim Deinstallieren werden alle installierten Dateien dieses Pakets entfernt, es sei denn, es handelt sich um Konfigurationsdateien, die Sie verändert haben. Diese werden nicht entfernt, sondern nur umbenannt: RPM hängt die Endung `.rpmsave` an. (Was als Konfigurationsdatei gilt, legt das jeweilige RPM-Paket selbst fest.)

27.2.3 Datenbank- und Paketanfragen

Seine volle Nützlichkeit entwickelt der *RPM Package Manager*, wenn Sie ihn nicht nur als Installationswerkzeug betrachtet, sondern auch als Informationsquelle. Der Modus dafür ist `-q` (für engl. *query*, »abfragen«); wobei Sie genauer angeben können, welche Informationen Sie erhalten möchten und über welches Paket.

Spezifikation des Pakets Ohne weitere Optionen erwartet `rpm` die Angabe eines internen Paketnamens, der auch gekürzt sein kann, und es antwortet mit dem vollen Paketnamen:

```
$ rpm -q openssh
openssh-3.5p1-107
```

Damit lässt sich schnell feststellen, wie aktuell Ihr System ist. Sie können auch (durch `-f`) das Paket finden, dem eine Datei gehört:

```
$ rpm -qf /usr/bin/ssh
openssh-3.5p1-107
```

Damit lassen sich unbekannte Dateien zumindest einem Paket zuordnen. Als dritte Möglichkeit können Sie mit `-a` *alle* installierten Pakete befragen. So erstellt

```
$ rpm -qa
```

eine Liste aller installierten Pakete, die sich natürlich auch weiterverarbeiten lässt, wie im folgenden Beispiel:[1]

```
$ rpm -qa | grep cups
cups-client-1.1.18-82
cups-libs-1.1.18-82
kdelibs3-cups-3.1.1-13
cups-drivers-1.1.18-34
cups-drivers-stp-1.1.18-34
cups-1.1.18-82
```

Schließlich erlaubt Ihnen RPM, ein nicht installiertes Paket zu befragen. Verwenden Sie `-p` gefolgt vom Pfad des Pakets:

```
$ rpm -qp /tmp/openssh-3.5p1-107.i586.rpm
openssh-3.5p1-107
```

Das sieht nicht sehr spektakulär aus, war der interne Paketname doch schon Teil des Dateinamens. Aber erstens könnte der Dateiname verändert worden sein und mit dem eigentlichen Paketnamen nichts zu tun haben und zweitens gibt es ja noch andere Anfragen, die Sie stellen können.

[1] Die Benennung und Aufteilung von Paketen ist Sache des Paket-Bauers; je nach Distribution und Version können sich Abweichungen ergeben.

Spezifikation der Anfrage Interessiert Sie nicht nur der Paketname, so können Sie Ihre Anfrage erweitern, um zusätzliche Informationen auszugeben. Dabei ist jede Erweiterung mit jeder Spezifikation des Pakets kombinierbar. Durch

```
$ rpm -qi openssh
```

erhalten Sie Informationen (`-i`) zum Paket; während `-l` eine Liste aller zu diesem Paket gehörenden Dateien ergibt, zusammen mit `-v` ergibt sich das Äquivalent von `ls -l`:

```
$ rpm -qlf /bin/bash
/bin/bash
/bin/sh
<<<<<
$ rpm -qlvf /bin/bash
-rwxr-xr-x  root  root   491992 Mar 14  2003 /bin/bash
lrwxrwxrwx  root  root        4 Mar 14  2003 /bin/sh -> bash
<<<<<
```

Zu beachten ist hier, dass die zu einem Paket gehörenden Dateien nur die in der RPM-Datenbank vermerkten sind. Und das sind nur die, die ein Paket bei der Installation mitbringt. Nicht darunter fallen Dateien, die während der Installation erzeugt werden (durch paketinterne Installations-Skripte) oder die im laufenden Betrieb erst entstanden sind (Protokolldateien usw.).

Wir hatten gesehen, dass RPM Konfigurationsdateien gesondert behandelt (bei der Deinstallation). Die zweite Klasse von besonderen Dateien sind Dateien der Dokumentation; diese können Sie von der Installation ausnehmen. Die Optionen `-c` und `-d` des Anfrage-Modus verhalten sich wie `-l`, nur dass sie sich auf Konfigurations- bzw. Dokumentationsdateien beschränken.

Fortgeschrittene Anfragen Die folgenden Details sind für LPIC-1 nicht erforderlich, sie erleichtern aber das Verständnis für die hinter RPM steckenden Konzepte und die Datenbankstruktur.

Abhängigkeiten

Die Abhängigkeiten zwischen Paketen können vielfältiger Natur sein. So kann es sein, dass ein Paket einfach nur eine Shell benötigt, d. h. `/bin/sh`. Durch

```
$ rpm -qf /bin/sh
bash-2.05b-105
```

ist schnell herausgefunden, dass diese Datei durch das Bash-Paket bereitgestellt wird (analog für nicht installierte Pakete).

Anders sieht die Sache z. B. beim Programm SAINT aus, einem Sicherheits-Analyse-Werkzeug, das zwingend einen Web-Browser voraussetzt. Jede Festlegung auf einen konkreten Web-Browser wäre unangemessen einschränkend. Deswegen bietet RPM die Möglichkeit, dass ein Paket eine *capability* anbieten oder voraussetzen kann, was man in diesem Zusammenhang mit »abstrakte Dienstleistung« übersetzen könnte. Im unserem Beispiel fordert SAINT daher die abstrakte Dienstleistung »Web-Browser« ein. Welche Dateien und Dienstleistungen ein Paket benötigt, lässt sich durch die Option `--requires` erfahren:[2]

```
$ rpm -q --requires saint
web_browser
/bin/rm
/bin/sh
/usr/bin/perl
<<<<<
```

[2] Auch hier gilt: Die Zuordnung und Benennung der Dienstleistungen ist Sache der Paket-Bauers, sie ist damit je nach Distribution und Version verschieden.

Welche Pakete diese Dienstleistung hingegen zur Verfügung stellen, verrät die Option `--whatprovides`:

```
$ rpm -q --whatprovides web_browser
w3m-0.3.2.2-53
mozilla-1.2.1-65
lynx-2.8.4-391
```

Für SAINT brauchen Sie also nur eines dieser Pakete.

In analoger Weise können mit `--provides` und `--whatrequires` das Angebot eines Pakets (nur Dienstleistungen; Dateien über `-l`) und die Abnehmer einer Dienstleistung erfragt werden.

27.2.4 Verifikation von Paketen

Überprüfung vor der Installation Zwei Dinge können einem Paket zustoßen, die gegen eine Installation sprechen: Beim Herunterladen des Pakets wurde es beschädigt, das Paket ist also fehlerhaft. Oder das Paket ist nicht das, was es vorgibt zu sein, es wurde also verfälscht. (Beispielsweise weil ein übelwollender Zeitgenosse Pakete mit Trojanischen Pferden als Original ausgibt.)

Gegen beide Szenarien sind Sie dank RPM gewappnet; durch

```
$ rpm --checksig /tmp/openssh-3.5p1-107.i586.rpm
/tmp/openssh-3.5p1-107.i586.rpm: md5 gpg OK
```

wird eine MD5-Prüfsumme des Pakets mit dem im Paket abgelegten Wert verglichen, was garantiert, dass das Paket richtig angekommen ist. Ferner wird die Signatur im Paket, die mit dem privaten PGP- oder GPG-Schlüssel des Paket-Bauers erzeugt wurde, mit dem öffentlichen Schlüssel des Paket-Bauers verglichen. Dies garantiert, dass das richtige Paket angekommen ist.

Sollte zwar die MD5-Prüfsumme, aber nicht die Signatur korrekt sein, so sieht die Ausgabe entsprechend anders aus:

```
$ rpm --checksig /tmp/openssh-3.5p1-107.i586.rpm
/tmp/openssh-3.5p1-107.i586.rpm: md5 GPG NOT OK
```

Natürlich muss für die Signaturprüfung der öffentliche Schlüssel Ihres Distributors auf dem System vorhanden sein.

Überprüfung nach der Installation RPM erlaubt es, gewisse Werte der RPM-Datenbank mit dem Dateisystem zu vergleichen. Dies geschieht durch den Modus `-V` (engl. *verify*, »überprüfen«); statt einem oder mehreren internen Paketnamen verträgt dieser Modus auch alle Spezifikationen, die der Anfrage-Modus erlaubt.

```
# rpm -V openssh
.......T c /etc/init.d/sshd
S.5....T c /etc/pam.d/sshd
S.5....T c /etc/ssh/ssh_config
SM5....T c /etc/ssh/sshd_config
.M......   /usr/bin/ssh
```

Die Ausgabe listet alle Dateien auf, für die mindestens ein Soll-Wert der Datenbank vom Ist-Wert des Dateisystems abweicht: Ein ».« kennzeichnet Übereinstimmung, während ein Buchstabe eine Abweichung darstellt. Die durchgeführten Tests sind: Zugriffsrechte und Dateityp (`M`), Eigentümer und Gruppe (`U`, `G`); bei symbolischen Links der Pfad der Referenzdatei (`L`); bei Gerätedateien Haupt- und Nebengerätenummer (`D`); bei regulären Dateien Größe (`S`), Änderungszeit (`T`) und

Inhalt (5). Da sich Konfigurationsdateien sehr wahrscheinlich nicht mehr im Originalzustand befinden, sind diese durch ein c gekennzeichnet.

Wenngleich die Überprüfung installierter Pakete durch RPM ein *Intrusion Detection System* nicht ersetzen kann (warum sollte ein Eindringling nicht auch die RPM-Datenbank manipulieren?), so kann sie doch sehr nützlich sein zur Schadenseingrenzung, beispielsweise nach einem Dateisystemcrash.

27.2.5 Das Programm `rpm2cpio`

RPM-Pakete sind im wesentlichen `cpio`-Archive mit einem davorgehängten »Kopf«. Diese Tatsache können Sie zum Beispiel ausnutzen, um einzelne Dateien aus einem RPM-Paket zu extrahieren, ohne das Paket dafür installieren zu müssen. Dazu wandeln Sie das RPM-Paket mit dem Programm `rpm2cpio` zu einem `cpio`-Archiv um, das Sie anschließend an `cpio` verfüttern. Da `rpm2cpio` als Filter arbeitet, können Sie die beiden Programme einfach mit einer Pipe verbinden:

```
$ rpm2cpio hello-2.4-1.fc10.i386.rpm \
> | cpio -idv ./usr/share/man/man1/hello.1.gz
./usr/share/man/man1/hello.1.gz
387 blocks
$ zcat usr/share/man/man1/hello.1.gz | head
.\" DO NOT MODIFY THIS FILE!  It was generated by help2man 1.35.
.TH HELLO "1" "December 2008" "hello 2.4" "User Commands"
.SH NAME
hello \- friendly greeting program
◁◁◁◁◁
```

Übungen

27.1 [2] Benutzen Sie `rpm2cpio` und `cpio`, um eine Liste der Dateien in einem RPM-Paket anzuzeigen.

27.3 YUM

27.3.1 Überblick

Das Programm `rpm` ist nützlich, aber hat seine Grenzen. Als grundlegendes Werkzeug kann es zwar als Dateien oder URLs vorliegende Pakete installieren, aber hilft zum Beispiel nicht beim Finden passender, möglicherweise installierbarer Pakete. Viele RPM-basierte Distributionen verwenden hierfür YUM (kurz für »Yellow Dog Updater, Modified«, nach der Distribution, für die das Programm zuerst entwickelt wurde), das den Zugriff auf Paketquellen (engl. *repositories*) erlaubt, die im Internet oder auch auf CD-ROM zur Verfügung stehen.

Paketquellen

YUM lebt in RPM-basierten Distributionen ungefähr in der »ökologischen Nische«, die bei Debian GNU/Linux und seinen Ablegern von `apt-get` eingenommen wird.

YUM wird normalerweise über die Kommandozeile gesteuert, aber das Kommando »`yum shell`« startet eine »YUM-Shell«, in der Sie mehrere YUM-Kommandos interaktiv eingeben können.

27.3.2 Paketquellen

YUM führt das Konzept von *Paketquellen* ein. Eine Paketquelle ist eine Menge von RPM-Paketen, die über das Netz zugänglich ist und die Installation von Paketen über YUM ermöglicht. Das Kommando »`yum repolist`« gibt eine Liste der konfigurierten Paketquellen aus:

```
$ yum repolist
Geladene Plugins: refresh-packagekit
Repo-ID        Repo-Name:                  Status
fedora         Fedora 10 - i386            aktiviert: 11.416
updates        Fedora 10 - i386 - Updates  aktiviert:  2.902
repolist: 14.318
```

»`yum repolist disabled`« liefert eine Liste der bekannten, aber deaktivierten Paketquellen:

```
$ yum repolist disabled
Geladene Plugins: refresh-packagekit
Repo-ID           Repo-Name:                      Status
fedora-debuginfo  Fedora 10 - i386 - Debug        deaktiviert
fedora-source     Fedora 10 - Source              deaktiviert
rawhide           Fedora - Rawhide - Development deaktiviert
◁◁◁◁◁
```

Um eine Paketquelle zu aktivieren, müssen Sie die Option `--enablerepo=` gefolgt von der »Repo-ID« aus der Liste angeben. Das geht nur in Verbindung mit einem »echten« `yum`-Kommando; `repolist` ist relativ unverfänglich:

```
$ yum --enablerepo=rawhide repolist
Geladene Plugins: refresh-packagekit
rawhide                                   | 3.4 kB   00:00
rawhide/primary_db                        | 7.2 MB   00:14
Repo-ID        Repo-Name:                  Status
fedora         Fedora 10 - i386            aktiviert: 11.416
rawhide        Fedora - Rawhide - Develop  aktiviert  12.243
updates        Fedora 10 - i386 - Updates  aktiviert:  2.902
repolist: 26.561
```

Deaktivieren können Sie eine Paketquelle mit der Option `--disablerepo`.

Paketquellen werden YUM bequemerweise durch Konfigurationsdateien im Verzeichnis `/etc/yum.repos.d` bekannt gemacht. (Sie könnten sie auch direkt in die Datei `/etc/yum.conf` eintragen, aber das ist weniger wartungsfreundlich.)

YUM hält sich selbst aktuell, was den Inhalt von Paketquellen angeht. Ein Äquivalent zu »`apt-get update`« bei den Debian-Paketwerkzeugen gibt es also nicht.

27.3.3 Pakete installieren und entfernen mit YUM

Um mit YUM ein neues Paket zu installieren, müssen Sie nur dessen Namen wissen. YUM prüft, ob die aktivierten Paketquellen ein Paket dieses Namens enthalten, löst gegebenenfalls Abhängigkeiten des Pakets auf, lädt das Paket und möglicherweise die Pakete, von denen es abhängt, herunter, und installiert sie:

```
# yum install hello
Geladene Plugins: refresh-packagekit
Einrichten des Installationsprozess
Analysiere Installationsargumente des Pakets
Löse Abhängigkeiten auf
--> Führe Transaktionsprüfung aus
---> Paket hello.i386 0:2.4-1.fc10 markiert, um aktualisiert▷
 ◁ zu werden
```

```
--> Abhängigkeitsauflösung beendet

Abhängigkeiten aufgelöst

================================================================
 Paket         Arch      Version         Repository     Grösse
================================================================
Installieren:
 hello         i386      2.4-1.fc10      updates         68 k

Transaktionszusammenfassung
================================================================
Installieren          1 Paket(e)
Aktualisieren         0 Paket(e)
Entfernen             0 Paket(e)

Gesamte Downloadgrösse: 68 k
Ist dies in Ordnung? [j/N] :j
Lade Pakete herunter:
hello-2.4-1.fc10.i386.rpm                     |  68 kB  00:00
Führe rpm_check_debug durch
Führe Verarbeitungsstest durch
Verarbeitungstest beendet
Verarbeitungstest erfolgreich
Führe Verarbeitung durch
  Installieren   : hello                                   1/1

Installiert:
  hello.i386 0:2.4-1.fc10

Komplett!
```

YUM akzeptiert nicht nur einfache Paketnamen, sondern auch Paketnamen mit Architekturangaben, Versions- und Releasenummern. Schauen Sie in `yum`(8) nach, um die erlaubten Formate zu finden.

Das Entfernen von Paketen ist genauso einfach:

```
# yum remove hello
```

Allerdings werden damit auch Pakete gelöscht, von denen dieses Paket abhängt – jedenfalls solange sie nicht noch von einem anderen installierten Paket benötigt werden.

Statt »`yum remove`« können Sie auch »`yum erase`« sagen – die beiden sind gleichbedeutend.

Mit »`yum update`« können Sie Pakete aktualisieren:

```
# yum update hello
```

prüft, ob eine aktuellere Version des Pakets vorliegt und installiert diese gegebenenfalls. Dabei sorgt YUM dafür, dass alle Abhängigkeiten aufgelöst werden. »`yum update`« ohne Paketnamen versucht alle installierten Pakete zu aktualisieren.

Wenn die Option `--obsoletes` angegeben ist (der Standardfall), dann versucht `yum`, den Fall zu behandeln, dass ein Paket durch ein anderes (das anders heißt) ersetzt wurde. Das erleichtert Komplett-Upgrades der ganzen Distribution.

»yum upgrade« ist dasselbe wie »yum update --obsoletes« – aber spart Tipparbeit für den Fall, dass Sie die obsoletes-Option in der Konfiguration ausgeschaltet haben.

YUM kennt das Konzept von »Paketgruppen«, also Paketen, die gemeinsam nützlich für eine bestimmte Aufgabe sind. Die verfügbaren Paketgruppen können Sie mit »yum grouplist« anzeigen:

```
$ yum grouplist
Geladene Plugins: refresh-packagekit
Einrichten des Gruppenprozess
Installierte Gruppen:
   Audio und Video
   Basis
   Büro/Produktivität
   Drucker-Unterstützung
◁◁◁◁◁
```

Wenn Sie wissen wollen, was sich hinter einer Gruppe verbirgt, verwenden Sie »yum groupinfo«:

```
$ yum groupinfo Drucker-Unterstützung
Geladene Plugins: refresh-packagekit
Einrichten des Gruppenprozess

Gruppe: Drucker-Unterstützung
 Beschreibung: Installieren Sie diese Programme, um es dem System
 zu ermöglichen zu drucken oder als Drucker-Server zu fungieren.
 Obligatorische Pakete:
   cups
   ghostscript
 Standard-Pakete:
   a2ps
   bluez-cups
   enscript
◁◁◁◁◁
```

Eine Gruppe gilt als »installiert«, wenn alle ihre »obligatorischen« Pakete installiert sind. Außer diesen gibt es »Standard-Pakete« und »optionale Pakete«.

Mit »yum groupinstall« können Sie die Pakete einer Gruppe installieren. Die Konfigurationsoption group_package_types entscheidet dabei darüber, welche Arten von Paketen tatsächlich installiert werden – normalerweise die »obligatorischen« und die »Standard-Pakete«.

»yum groupremove« entfernt alle Pakete einer Gruppe, *ohne* dabei auf die Paketart zu achten (group_package_types wird ignoriert). Beachten Sie auch, dass Pakete zu mehr als einer Gruppe gleichzeitig gehören können, so dass sie möglicherweise in Gruppe X fehlen, nachdem sie mit Gruppe Y entfernt wurden.

27.3.4 Informationen über Pakete

Um herauszufinden, welche Pakete es gibt, steht das Kommando »yum list« zur Verfügung:

```
$ yum list gcc
Geladene Plugins: refresh-packagekit
```

```
Installierte Pakete
gcc.i386              4.3.2-7              installed
```

Sie können auch ein Suchmuster angeben (am besten in Anführungszeichen, damit die Shell sich nicht dazwischen drängelt):

```
$ yum list "gcc*"
Geladene Plugins: refresh-packagekit
Installierte Pakete
gcc.i386              4.3.2-7              installed
gcc-c++.i386          4.3.2-7              installed
Verfügbare Pakete
gcc-gfortran.i386     4.3.2-7              fedora
gcc-gnat.i386         4.3.2-7              fedora
<<<<<
```

Die »installierten Pakete« sind auf dem lokalen System vorhanden, während die »verfügbaren Pakete« von Paketquellen geholt werden können. Ganz rechts auf der Zeile ist die Paketquelle angegeben, in der das Paket zu finden ist.

Um die Suche auf die lokal installierten oder die nicht installierten, aber verfügbaren Pakete zu beschränken, können Sie »`yum list installed`« oder »`yum list available`« benutzen:

```
$ yum list installed "gcc*"
Geladene Plugins: refresh-packagekit
Installierte Pakete
gcc.i386              4.3.2-7              installed
gcc-c++.i386          4.3.2-7              installed
$ yum list available "gcc*"
Geladene Plugins: refresh-packagekit
Verfügbare Pakete
gcc-gfortran.i386     4.3.2-7              fedora
gcc-gnat.i386         4.3.2-7              fedora
<<<<<
```

»`yum list updates`« listet die Pakete auf, die installiert sind und für die es Aktualisierungen gibt, während »`yum list recent`« diejenigen Pakete auflistet, die »kürzlich« in einer Paketquelle angekommen sind. »`yum list extras`« nennt Ihnen diejenigen Pakete, die lokal installiert, aber in keiner Paketquelle zu finden sind.

Um Informationen über ein Paket zu bekommen, steht Ihnen »`yum info`« zur Verfügung:

```
$ yum info hello
Geladene Plugins: refresh-packagekit
Installierte Pakete
Name       : hello
Architektur : i386
Version    : 2.4
Ausgabe    : 1.fc10
Grösse     : 186 k
Repo       : installed
Zusammenfassung    : Prints a Familiar, Friendly Greeting
URL        : http://www.gnu.org/software/hello/
Lizenz     : GPLv3+ and GFDL and BSD and Public Domain
Beschreibung:Hello prints a friendly greeting. It also serves as a
           : sample GNU package, showing practices that may be
           : useful for GNU projects.
```

Der Vorteil gegenüber »rpm -qi« besteht darin, dass »yum info« auch für Pakete funktioniert, die lokal nicht installiert sind, aber in einer Paketquelle vorliegen.

»yum info« können Sie ansonsten verwenden wie »yum list« – »yum info installed« zeigt Ihnen zum Beispiel Detailinformationen über *alle* installierten Pakete an.

Nach allen Paketen suchen, in deren Namen oder Beschreibung eine bestimmte Zeichenkette vorkommt, können Sie mit »yum search«:

```
$ yum search mysql
Geladene Plugins: refresh-packagekit
============================ Matched: mysql ========================
dovecot-mysql.i386 : MySQL backend for dovecot
koffice-kexi-driver-mysql.i386 : Mysql-driver for kexi
libgda-mysql.i386 : MySQL provider for libgda
<<<<<
```

Die resultierende Liste ist leider unsortiert und etwas unübersichtlich. yum hebt in seiner Ausgabe die gefundenen Begriffe durch Fettschrift hervor.

Die Abhängigkeiten eines Pakets können Sie mit »yum deplist« untersuchen:

```
$ yum deplist gcc
Geladene Plugins: refresh-packagekit
Suche Abhängigkeiten:
Paket: gcc.i386 4.3.2-7
  Abhängigkeit: binutils >= 2.17.50.0.17-3
    provider: binutils.i386 2.18.50.0.9-7.fc10
  Abhängigkeit: libc.so.6(GLIBC_2.3)
    provider: glibc.i386 2.9-2
<<<<<
```

27.3.5 Pakete nur herunterladen

Wenn Sie ein Paket nur aus einer Paketquelle herunterladen, aber nicht gleich installieren wollen, können Sie das Programm yumdownloader verwenden. Ein Kommando wie

```
$ yumdownloader --destdir /tmp hello
```

durchsucht die Paketquellen genau so, wie YUM das machen würde, nach dem Paket hello und lädt die korrespondierende Datei ins Verzeichnis /tmp herunter.

Mit der Option --resolve werden auch die Abhängigkeiten aufgelöst und allfällige fehlende andere Pakete mit heruntergeladen – allerdings nur die, die auf dem lokalen System nicht installiert sind.

Mit der Option --urls wird überhaupt nichts heruntergeladen, sondern yumdownloader gibt nur die URLs der Pakete aus, die es herunterladen *würde*.

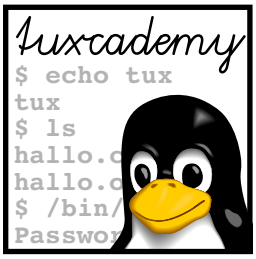
Mit der Option --source lädt yumdownloader keine Binär-, sondern Quellcode-RPMs herunter.

Kommandos in diesem Kapitel

cpio Dateiarchiv-Verwaltungsprogramm cpio(1) 461
rpm Dient zur Paketverwaltung in vielen Linux-Systemen (Red Hat, SUSE, …) rpm(8) 456
rpm2cpio Wandelt RPM-Pakete in cpio-Archive um rpm2cpio(1) 461
yum Komfortables Werkzeug zur Verwaltung von RPM-Paketen yum(8) 461
yumdownloader Lädt Pakete aus YUM-Paketquellen (ohne Installation) yumdownloader(8) 466

Zusammenfassung

- RPM ist ein System zur Verwaltung von Linux-Softwarepaketen, das bei verschiedenen Distributionen wie denen von Red Hat und Novell/SUSE eingesetzt wird.
- YUM ist eine Oberfläche für rpm, die den Zugriff auf Paketquellen im Netz ermöglicht.



A

Musterlösungen

Dieser Anhang enthält Musterlösungen für ausgewählte Aufgaben.

1.2 Auf `ftp.kernel.org` steht noch ein `linux-0.01.tar.gz`.

1.3

1. Falsch. GPL-Programme dürfen zu beliebigen Preisen verkauft werden, solange der Käufer den Quellcode bekommt (usw.) und die GPL-Rechte zugesprochen bekommt.

2. Falsch. Firmen sind herzlich eingeladen, eigene Produkte auf der Basis von GPL-Software zu entwickeln, nur fallen diese Produkte auch wieder unter die GPL. Natürlich ist die Firma nicht gezwungen, ihr Produkt an die Allgemeinheit zu verschenken – es genügt, wenn der Quellcode den tatsächlichen direkten Kunden zugänglich gemacht wird, die auch die ausführbaren Programme gekauft haben, aber die dürfen damit alles machen, was die GPL ihnen erlaubt.

3. Richtig.

4. Falsch. Sie dürfen ein Programm beliebig *benutzen*, ohne dass Sie die GPL akzeptiert haben. Die GPL regelt erst die *Weitergabe* der Software, und Sie können von der GPL vorher Kenntnis nehmen. (Interaktive Programme sollen Sie ja auf die GPL hinweisen.) Grundsätzlich gilt natürlich die Feststellung, dass nur solche Bestimmungen verbindlich sind, die der Empfänger der Software *vor* dem Erwerb (Kauf, …) kennen konnte; da die GPL dem Empfänger aber zusätzliche Rechte gibt, die er sonst überhaupt nicht hätte – etwa das Recht der originalgetreuen oder modifizierten Weitergabe –, ist das kein Problem; man kann die GPL völlig links liegenlassen und darf trotzdem alles mit der Software machen, was das Urheberrecht zuläßt. Anders sieht das mit den EULAs proprietärer Programme aus; diese versuchen ein Vertragsverhältnis herzustellen, in dem der Käufer ausdrücklich auf Rechte *verzichtet*, die er laut Urheberrecht hätte (etwa das Recht, die Programmstruktur zu analysieren), und sowas geht natürlich, wenn überhaupt, nur vor dem Kauf.

1.5 Für diese Aufgabe können wir in einer gedruckten Schulungsunterlage natürlich keine korrekte Lösung angeben. Schauen Sie sich um – auf `ftp.kernel.org` oder in der wöchentlichen Ausgabe von `http://lwn.net/`.

2.2 Auf dem Textbildschirm gilt, dass in beiden Fällen die Fehlermeldung »`Login incorrect`« erscheint, aber erst nach der Kennwortabfrage. Der Grund dafür ist, dass es sonst einfach wäre, gültige Benutzernamen zu raten (die, bei denen nicht gleich eine Fehlermeldung erscheint). So, wie das System jetzt ist, kann ein »Cracker« nicht entscheiden, ob schon der Benutzername ungültig war oder nur das Kennwort falsch ist, was das Einbrechen ins System deutlich erschwert.

2.5 Der Anstand verbietet es uns, hier ein entsprechendes Programm abzudrucken. Einfach wird es allerdings mit Hilfe der (verpönten) C-Funktion `getpass`(3).

3.2 In der Loginshell lautet die Ausgabe »`-bash`«, in der zusätzlich gestarteten »Subshell« dagegen »`bash`«. Das Minuszeichen am Anfang sagt der Shell, dass sie sich wie eine Loginshell benehmen soll und nicht wie eine »normale« Shell, was Auswirkungen auf die Initialisierung hat.

3.3 `alias` ist ein internes Kommando (geht nicht anders). `rm` ist ein externes Kommando. `echo` und `test` sind bei der Bash intern, stehen aber auch als externe Kommandos zur Verfügung, weil andere Shells sie nicht intern implementieren. Bei der Bash sind sie hauptsächlich aus Effizienzgründen intern.

4.2 Probieren Sie »`apropos process`« oder »`man -k process`«.

4.5 Das Format und die Werkzeuge für Info-Dateien wurden Mitte der 1980er Jahre entwickelt. HTML war da noch überhaupt nicht erfunden.

5.1 Theoretisch können Sie natürlich jedes Programm im System starten und schauen, ob es sich wie ein Texteditor benimmt … aber das würde wohl doch etwas länger dauern, als diese Aufgabe wert ist. Sie können zum Beispiel mit etwas wie »`apropos edit`« anfangen und schauen, welche der aufgezählten Handbuchseiten mit einem Texteditor korrespondieren (und nicht mit einem Editor für Grafiken, Icons, X-Ressourcen oder anderes). Die Texteditoren aus Arbeitsumgebungen wie KDE oder GNOME haben oft keine Handbuchseiten, sondern werden innerhalb der Arbeitsumgebung dokumentiert. Hier kann es nützlich sein, in den Menüs der Arbeitsumgebung nach einem Untermenü wie »Editoren« zu suchen. Die dritte Möglichkeit besteht darin, sich mit einem Kommando des Paketverwaltungssystems – etwa »`rpm -qa`« oder »`dpkg -l`« – eine Liste der installierten Softwarepakete anzeigen zu lassen und darin nach Texteditoren zu suchen.

5.2 Das Programm heißt `vimtutor`.

6.1 Das aktuelle Verzeichnis ist in Linux ein Prozessattribut, das heißt, jeder Prozess hat sein eigenes aktuelles Verzeichnis (bei DOS ist das aktuelle Verzeichnis eine Eigenschaft des Laufwerks, aber das ist in einem Mehrbenutzersystem natürlich nicht tragbar). Darum muss `cd` in die Shell eingebaut sein. Wenn es ein externes Kommando wäre, würde es in einem neuen Prozess ausgeführt, würde *dessen* aktuelles Verzeichnis ändern und sich dann prompt beenden, während das aktuelle Verzeichnis der Shell unverändert bliebe.

6.4 Bekommt `ls` einen Dateinamen übergeben, gibt es Informationen nur über diese Datei aus. Bei einem Verzeichnisnamen liefert es Informationen über alle Dateien in dem betreffenden Verzeichnis.

6.5 Hierfür hat `ls` die Option `-d`.

6.6 Das ganze könnte etwa so aussehen:

```
$ mkdir -p grd1-test/dir1 grd1-test/dir2 grd1-test/dir3
$ cd grd1-test/dir1
$ vi hallo
$ cd
$ vi grd1-test/dir2/huhu
$ ls grd1-test/dir1/hallo grd1-test/dir2/huhu
grd1-test/dir1/hallo
grd1-test/dir2/huhu
$ rmdir grd1-test/dir3
$ rmdir grd1-test/dir2
rmdir: grd1-test/dir2: Directory not empty
```

Damit Sie mit `rmdir` ein Verzeichnis entfernen können, muss dieses (bis auf die zwangsläufig vorhandenen Einträge ».« und »..«) leer sein.

6.7 Auf die Suchmuster passen jeweils:

(a) `prog.c`, `prog1.c`, `prog2.c`, `progabc.c`

(b) `prog1.c`, `prog2.c`

(c) `p1.txt`, `p2.txt`, `p21.txt`, `p22.txt`

(d) `p1.txt`, `p21.txt`, `p22.txt`, `p22.dat`

(e) alle Dateien

(f) alle Dateien außer `prog` (hat keinen Punkt im Namen)

6.8 »`ls`« ohne Argumente listet den Inhalt des aktuellen Verzeichnisses auf. Verzeichnisse im aktuellen Verzeichnis werden nur namentlich aufgezählt. »`ls`« mit Argumenten dagegen (also auch »`ls *`« – `ls` bekommt das Suchmuster ja nicht zu sehen) listet Informationen über die angegebenen Argumente auf. Für Verzeichnisse heißt das, dass auch der *Inhalt* der Verzeichnisse aufgelistet wird.

6.9 Die Datei »`-l`« (in der Ausgabe des ersten Kommandos zu sehen) wird von `ls` als Option verstanden. Darum taucht sie in der Ausgabe des zweiten Kommandos auch nicht mehr auf, da bei `ls` mit Pfadnamenargumenten nur die als Argument angegebenen Dateien ausgegeben werden.

6.10 Wenn der Stern auf Dateinamen mit Punkt am Anfang passte, dann würde zum Beispiel das rekursive Lösch-Kommando »`rm -r *`« auch den Namen »..« bearbeiten. Damit würden nicht nur Unterverzeichnisse des aktuellen Verzeichnisses gelöscht, sondern auch das übergeordnete Verzeichnis und so weiter.

6.11 Hier sind die Kommandos:

```
$ cd
$ cp /etc/services myservices
$ mv myservices src.dat
$ cp src.dat /tmp
$ rm src.dat /tmp/src.dat
```

6.12 Wenn Sie ein Verzeichnis umbenennen, sind alle darin enthaltenen Dateien und Unterverzeichnisse automatisch unter dem neuen Verzeichnisnamen zu finden. Eine `-R`-Option ist daher vollkommen überflüssig.

6.13 Der naive Ansatz – etwas wie »`rm -file`« – schlägt fehl, da `rm` den Dateinamen als Folge von Optionen missversteht. Dasselbe gilt für Kommandos wie »`rm "-file"`« oder »`rm '-file'`«. Die folgenden Möglichkeiten funktionieren besser:

1. Mit »`rm ./-file`« ist das Minuszeichen nicht mehr am Anfang des Parameters und leitet so keine Option ein.
2. Mit »`rm -- -file`« sagen Sie `rm`, dass nach dem »`--`« definitiv keine Optionen mehr kommen, sondern nur noch Dateinamen. Das funktioniert auch bei vielen anderen Programmen.

6.14 Im Rahmen der Ersetzung von »*« wird auch die Datei »`-i`« erfasst. Da die Dateinamen bei der Ersetzung in ASCII-Reihenfolge in die Kommandozeile getan werden, sieht `rm` eine Parameterliste wie

```
-i a.txt b.jpg c.dat                                    oder was auch immer
```

und hält das »`-i`« für die *Option* `-i`, die es dazu bringt, Dateien nur nach Rückfrage zu entfernen. Wir hoffen mal, dass das reicht, damit Sie nochmal in Ruhe nachdenken.

6.15 Wenn Sie die Datei über das eine Link im Editor aufrufen und ändern, sollte anschließend auch unter dem anderen Link der veränderte Inhalt zu sehen sein. Es gibt allerdings »clevere« Editoren, die Ihre Datei beim Speichern nicht überschreiben, sondern neu abspeichern und anschließend umbenennen. In diesem Fall haben Sie hinterher wieder zwei verschiedene Dateien.

6.16 Wenn die Zieldatei eines symbolischen Links nicht (mehr) existiert, dann gehen Zugriffe auf das Link ins Leere und führen zu einer Fehlermeldung.

6.17 Auf sich selbst. Man kann das Wurzelverzeichnis daran erkennen.

6.18 Das Verzeichnis `/home` liegt auf diesem Rechner auf einer eigenen Partition und hat im Dateisystem auf jener Partition die Inodenummer 2, während das Verzeichnis `/` auf seinem eigenen Dateisystem die Inodenummer 2 hat. Da Inodenummern nur dateisystemweit eindeutig sind, kann in der Ausgabe von »`ls -i`« durchaus bei verschiedenen Dateien dieselbe Zahl stehen; das ist kein Grund zur Besorgnis.

6.19 Im Wurzelverzeichnis (und nur in diesem) haben »`.`« und »`..`« dieselbe Inode-Nummer. Programme, die ausgehend von einem bestimmten Verzeichnis die »`..`«-Links verfolgen, um das Wurzelverzeichnis zu finden, können so feststellen, dass sie dort angekommen sind.

6.20 Harte Links sind ununterscheidbare, also gleichberechtigte Namen für eine Datei (oder hypothetischerweise ein Verzeichnis). Jedes Verzeichnis hat aber ein »Link« namens »`..`«, das auf das Verzeichnis »darüber« verweist. Dieses Link kann es pro Verzeichnis nur einmal geben, was sich mit der Idee mehrerer gleichberechtigter Namen nicht verträgt. Ein anderes Argument dagegen ist, dass es für jeden Namen einen eindeutigen Pfad geben muss, der in endlich vielen Schritten zum Wurzelverzeichnis (`/`) führt. Wären Links auf Verzeichnisse erlaubt, dann könnte eine Befehlsfolge wie

```
$ mkdir -p a/b
$ cd a/b
$ ln .. c
```

zu einer Schleife führen.

6.21 Der Referenzzähler für das Unterverzeichnis hat den Wert 2 (ein Verweis entsteht durch den Namen des Unterverzeichnisses in ~, einer durch das .-Link im Unterverzeichnis selbst). Hätte das Unterverzeichnis weitere Unterverzeichnisse, dann würden die ..-Links dieser Verzeichnisse den Referenzzähler in die Höhe treiben.

6.22 Die Kette der symbolischen Links wird verfolgt, bis Sie bei etwas ankommen, was kein symbolisches Link ist. Die maximale Länge solcher Ketten ist allerdings in der Regel beschränkt (siehe Übung 6.23).

6.23 Die Untersuchung dieser Fragestellung wird einfacher, wenn man Shell-Schleifen benutzen kann (siehe Abschnitt 8.6.) Etwas wie

```
$ touch d
$ ln -s d L1
$ i=1
$ while ls -lH L$i >/dev/null
> do
>   ln -s L$i L$((i+1))
>   i=$((i+1))
> done
```

legt eine »Kette« von symbolischen Links an, wobei jedes Link auf das vorige zeigt. Dies wird fortgesetzt, bis das »`ls -lH`«-Kommando fehlschlägt. An der Fehlermeldung können Sie dann sehen, welche Länge gerade noch erlaubt ist. (Auf dem Rechner des Autors ist das Ergebnis »40«, was im wirklichen Leben keine allzu störende Einschränkung darstellen dürfte.)

6.24 Harte Links brauchen fast keinen Platz, da es sich dabei nur um zusätzliche Verzeichniseinträge handelt. Symbolische Links sind eigene Dateien und kosten darum zumindest schon mal ein Inode (jede Datei braucht ein eigenes Inode). Dazu kommt der Speicherplatz für den Namen der Zieldatei. Prinzipiell wird Plattenplatz an Dateien in Einheiten der Blockgröße des Dateisystems vergeben (also 1 KiB und mehr, meist 4 KiB), aber in den `ext-` Dateisystemen gibt es eine spezielle Ausnahme für »kurze« symbolische Links (kleiner als ca. 60 Bytes), die im Inode selber gespeichert werden und keinen Datenblock brauchen. Andere Dateisysteme wie das Reiser-Dateisystem gehen von sich aus sehr effizient mit kurzen Dateien um, so dass auch dort der Platzbedarf für symbolische Links vernachlässigbar sein dürfte.

6.25 Ein mögliches Kommando wäre »`find / -size +1024k -print`«.

6.26 Der grundlegende Ansatz ist etwas wie

```
find . -maxdepth 1 ⟨Tests⟩ -ok rm '{}' \;
```

Die ⟨*Tests*⟩ sollten dabei die Datei so eindeutig wie möglich bestimmen. Das »`-maxdepth 1`« sorgt dafür, dass Unterverzeichnisse nicht durchsucht werden. Im naheliegendsten Fall verschaffen Sie sich mit »`ls -i`« die Inodenummer der Datei (zum Beispiel 4711) und löschen Sie dann mit

```
find . -maxdepth 1 -inum 4711 -exec rm -f '{}' \;
```

6.27 Schreiben Sie in die Datei `.bash_logout` in Ihrem Heimatverzeichnis etwas wie

```
find /tmp -user $LOGNAME -type f -exec rm '{}' \;
```

oder – effizienter –

```
find /tmp -user $LOGNAME -type f -print0 \
    | xargs -0 -r rm -f
```

(in der Umgebungsvariablen `LOGNAME` steht der aktuelle Benutzername).

6.28 Verwenden Sie ein Kommando wie »`locate '*/README'`«. Natürlich würde »`find / -name README`« es auch tun, aber das braucht *viel* länger.

6.29 Direkt nach dem Anlegen steht die neue Datei nicht in der Datenbank und wird entsprechend auch nicht gefunden (Sie müssen erst `updatedb` laufen lassen). Die Datenbank bekommt auch nichts davon mit, dass Sie die Datei gelöscht haben, bis Sie wiederum `updatedb` aufrufen. – Am geschicktesten rufen Sie `updatedb` übrigens nicht direkt auf, sondern über das Shellskript, das auch Ihre Distribution benutzt (etwa `/etc/cron.daily/find` bei Debian GNU/Linux). Auf diese Weise stellen Sie sicher, dass `updatedb` dieselben Parameter verwendet wie sonst auch immer.

6.30 `slocate` sollte nur diejenigen Dateinamen ausgeben, auf die der aufrufende Benutzer auch zugreifen darf. Die Datei `/etc/shadow`, in der die verschlüsselten Kennwörter der Benutzer stehen, ist dem Systemverwalter vorbehalten (siehe auch *Linux-Administration I*).

7.1 Eine (wahrscheinliche) Möglichkeit ist, dass das Programm `ls` nach etwa dem folgenden Schema abläuft:

```
Lies die Verzeichnisdaten in Liste l ein;
if (Option -U nicht angegeben) {
   Sortiere die Elemente von l;
}
Gib l auf die Standardausgabe aus;
```

Es würde also nach wie vor erst alles gelesen, dann sortiert (oder eben nicht sortiert) und dann ausgegeben.

Die andere Erklärung ist, dass zum Zeitpunkt des Lesens des `inhalt`-Eintrags wirklich noch nichts in die Datei geschrieben wurde; aus Effizienzgründen puffern die meisten Programme, wenn sie in Dateien schreiben sollen, ihre Ausgabe intern zwischen und führen tatsächliche Betriebssystemaufrufe zum Schreiben erst aus, wenn wirklich substanzielle Datenmengen (typischerweise 8192 Bytes) zusammengekommen sind. Dies läßt sich bei Programmen beobachten, die relativ langsam sehr viel Ausgabe erzeugen; hier wächst eine Ausgabedatei in Schritten von 8192 Bytes.

7.2 Wenn `ls` auf den Bildschirm (oder allgemein ein »bildschirmartiges« Gerät) schreibt, dann formatiert es die Ausgabe anders als wenn es in eine »richtige« Datei schreibt: Es versucht, mehrere Dateinamen in einer Zeile anzuzeigen, wenn die Länge der Dateinamen es zuläßt, und kann je nach Einstellung die Dateinamen auch entsprechend ihres Typs färben. Bei der Ausgabeumlenkung in eine »richtige« Datei werden einfach nur die Namen ohne Formatierung zeilenweise ausgegeben.

Auf den ersten Blick scheint das der Behauptung zu widersprechen, dass Programme gar nicht mitbekommen, dass ihre Ausgabe nicht auf den Bildschirm, sondern anderswohin geht. Im Normalfall ist die Behauptung richtig, aber wenn ein Programm sich ernsthaft dafür interessiert, ob sein Ausgabeziel ein bildschirmartiges Gerät (vulgo »Terminal«) ist, kann es das System danach fragen. Im Falle von `ls` ist die Überlegung dahinter einfach die, dass die Terminalausgabe normalerweise von menschlichen Benutzern angeschaut wird und man diesen möglichst

viel Information bieten will. Umgeleitete Ausgabe dagegen wird von anderen Programmen verarbeitet und sollte einfach sein; daher die Beschränkung auf einen Dateinamen pro Zeile und der Verzicht auf Farben, die ja über Terminalsteuerzeichen angesprochen werden und die Ausgabe »verunreinigen« würden.

7.3 Die Shell arrangiert die Ausgabeumlenkung, bevor das Kommando aufgerufen wird. Das Kommando sieht also nur noch eine leere Eingabedatei, was in der Regel nicht zum erwarteten Ergebnis führt.

7.4 Die Datei wird von vorne gelesen und gleichzeitig hinten um alles Gelesene verlängert, sie wächst also, bis sie allen freien Plattenplatz einnimmt.

7.5 Dazu müssen Sie die Standard-Ausgabe in die Standard-Fehlerausgabe umlenken:

```
echo Fehler >&2
```

7.6 Prinzipiell spricht nichts gegen

```
… | tee bla | tee fasel | …
```

Kürzer ist allerdings

```
… | tee bla fasel | …
```

Siehe hierzu auch die Dokumentation zu `tee` (Handbuch- oder Info-Seite).

7.7 Leiten Sie die Dateiliste durch »`cat -A`«.

7.8 Eine Möglichkeit wäre »`head -n 13 | tail -n 1`«.

7.10 `tail` merkt das, gibt eine Warnung aus und macht am neuen Dateiende weiter.

7.11 Im `tail`-Fenster steht

```
Hallo
elt
```

Die erste Zeile kommt vom ersten `echo`; das zweite `echo` überschreibt den kompletten Inhalt der Datei, aber »`tail -f`« weiß, dass es die ersten sechs Zeichen der Datei (»`Hallo`« plus der Zeilentrenner) schon ausgegeben hat – es wartet nur darauf, dass die Datei länger wird, und liefert lediglich das, was neu dazu gekommen ist, nämlich »`elt`« (und ein Zeilentrenner).

7.12 Bei »`a`« werden unsichtbare Zeichen mit ihren symbolischen Namen wie »`cr`« oder »`lf`« benannt, bei »`c`« durch Rückstrichsequenzen wie »`\r`« oder »`\n`«. Das Leerzeichen erscheint unter »`a`« als »`sp`«.

7.13 Der gewünschte Wertebereich (0 ... 65535 = $2^{16}-1$) entspricht gerade dem in zwei Byte darstellbaren Zahlenbereich. Wir müssen also zwei Byte aus `/dev/random` lesen und als Dezimalzahl ausgeben:

```
$ r=`od -An -N2 -tu2 /dev/random`
$ echo $r
4711
```

Die Option `-N2` liest zwei Bytes, und `-tu2` formatiert sie als vorzeichenlose 2-Byte-Dezimalzahl. `-An` unterdrückt die Positionsangabe (siehe Tabelle 7.4).

7.14 Probieren Sie mal:

```
$ echo "ALEA IACTA EST" | tr A-Z D-ZA-C
DOHD LDFWD HVW
$ echo "DOHD LDFWD HVW" | tr A-Z X-ZA-W
ALEA IACTA EST
```

Ähnlich dem Caesarschen Verfahren ist das »ROT13«-Verfahren, das im USENET benutzt wird, beispielsweise um unanständige Witze zu veröffentlichen, ohne dass empfindliche Personen sie versehentlich zu lesen bekommen (für eine »echte« Verschlüsselung vertraulicher Inhalte ist das Verfahren natürlich ein bißchen schwach). ROT13 wird durch das Kommando »`tr A-Za-z N-ZA-Mn-za-m`« beschrieben; der Vorteil dieses Verfahrens ist, dass man den Originaltext wiederbekommt, wenn man es zweimal hintereinander anwendet. Die ROT13-Routine in einem Newsreader kann also zum Ver- und Entschlüsseln benutzt werden, was den Code vereinfacht.

7.15 Der einfache Weg ist natürlich »`tr AEIOU AAAAA`«. Mit weniger Tippaufwand reicht auch »`tr AEIOU A`«.

7.16 Eine Möglichkeit dafür ist

```
$  tr -cs '[:alpha:]' '\n'
```

Das »`-c [:alpha:] \n`« wandelt alle Nichtbuchstaben in Zeilenendezeichen um, die Option `-s` sorgt dafür, dass Folgen dieser Zeilenendezeichen entfernt und durch ein Zeilenende ersetzt werden. Es ist sinnvoll, die Parameter in Anführungszeichen zu setzen, um sicherzustellen, dass die eckigen Klammern nicht von der Shell bearbeitet werden. (Wenn Sie das für deutschsprachige Texte machen wollen, sollten Sie vorher die Umgebungsvariable `LANG` auf `de_DE` o. ä. setzen, damit Umlaute und »ß« als Buchstaben anerkannt werden.)

7.17 Damit das funktioniert, muss das Zeichen »`-`« am Ende von $\langle s_1 \rangle$ stehen – steht es am Anfang, sieht »`-az`« aus wie eine Kommandooption (die `tr` nicht versteht), steht es in der Mitte, so sieht »`a-z`« aus wie ein Bereich. Alternativ, aber tippaufwendiger können Sie das Zeichen »`-`« auch durch »`[=-=]`« umschreiben (siehe Tabelle 7.6), was an einer beliebigen Position stehen darf.

7.18 Mit etwas wie »`cat -T`« (siehe Dokumentation) oder »`od -tc`«.

7.19 Das Kommando hierfür ist »`nl -v 100 -i 2 frosch.txt`«.

7.20 Das Kommando `tac` ist unser Freund:

```
$  tac frosch.txt | cat -n | tac
```

(Statt »`cat -n`« hätte es auch »`nl -ba`« oder ähnliches getan.)

7.21 Beim ersten Kommando betrachtet `wc` alle Eingabedateien separat und gibt auch noch eine Gesamtsumme aus. Beim zweiten Kommando betrachtet `wc` seine Standardeingabe, wo die Tatsache, dass es ursprünglich um drei separate Dateien gibt, nicht mehr in Erscheinung tritt. Darum gibt es beim zweiten Kommando nur eine Ausgabezeile und nicht vier wie beim ersten.

7.24 Die Zeile mit dem Namen »von Traben« wird falsch einsortiert, weil in ihr das zweite Feld nicht wirklich der Vorname ist, sondern das Wort »Traben«. Wenn Sie sich die Beispiele genau anschauen, werden Sie feststellen, dass die Sortierung immer stimmt – nur halt für »Traben« statt »Gesine«. Dies ist ein eindeutiges Argument für die zweite Form der Beispieldatei, die mit den Doppelpunkten als Trenner.

7.25 Mit »`sort -k 1.4,1.8`« können Sie die Zeilen nach der Jahreszahl sortieren. Sind zwei Zeilen gemäß dem Sortierschlüssel gleich, macht `sort` einen »Notvergleich« über die ganze Zeile, der hier dazu führt, dass innerhalb der Jahre die Monate korrekt sortiert werden. Wenn Sie gerne sichergehen und ganz explizit sein möchten, können Sie natürlich auch »`sort -k 1.4,1.8 -k 1.1,1.2`« schreiben.

7.26 Mit der Lösung von Übung 7.16 ist das Ganze ziemlich einfach:

```
$ tr -cs '[:alpha:]' '\n' | sort -uf
```

Die Option `-u` von `sort` sorgt dafür, dass von einer Folge von gleichen Wörtern nur das erste ausgegeben wird. Durch `-f` werden Groß- und Kleinbuchstaben gleich behandelt (»`LC_COLLATE=de_DE`« würde das mit erledigen).

7.30 Benutzen Sie etwas wie

```
cut -d: -f 4 /etc/passwd | sort -u | wc -l
```

Das `cut`-Kommando isoliert die Gruppennummer in jeder Zeile der Benutzerdatenbank. »`sort -u`« (siehe auch Übung 7.26) liefert eine sortierte Liste aller Gruppennummern, in der jede Gruppennummer nur einmal vorkommt. »`wc -l`« schließlich zählt die Zeilen der Liste, das Ergebnis ist die Anzahl der verschiedenen primären Gruppen.

8.1 Zum Beispiel:

1. `%d-%m-%Y`
2. `%y-%j (KW%V)`
3. `%Hh%Mm%Ss`

8.2 Wir wissen das auch nicht genau, aber versuchen Sie testhalber mal etwas wie »`TZ=America/Los_Angeles date`«.

8.4 Wenn Sie eine Umgebungsvariable im Kindprozess ändern, bleibt der Wert der Variablen im Elterprozess davon unbeeinflusst. Es gibt Mittel und Wege, Informationen an den Elterprozess zurückfließen zu lassen, aber dies ist keiner davon.

8.5 Starten Sie eine neue Shell und entfernen Sie die Umgebungsvariable `PATH` aus deren Umgebung, ohne jedoch die Variable selbst zu löschen. Versuchen Sie, externe Programme zu starten. – Wenn `PATH` gar nicht existiert, startet die Shell keine externen Programme.

8.6 Eine systemunabhängige Musterlösung können wir hierfür leider nicht geben; probieren Sie es selber aus (mit `which`).

8.7 Sie sollten mit »`whereis`« zwei Dateien namens `/usr/share/man/man1/crontab.1.gz` und `/usr/share/man/man5/crontab.5.gz` finden. Die erstere enthält die Dokumentation für das eigentliche `crontab`-Kommando, die letztere die Dokumentation für das Dateiformat, das Sie mit `crontab` anlegen können. (Die Details sind für diese Aufgabe nicht wichtig; siehe *Linux-Administration I*.)

8.8 Die Bash verwendet Zeichenfolgen der Form »!⟨*Zeichen*⟩« zum Zugriff auf alte Kommandos (eine Alternative zu den Tastaturfunktionen wie Strg+r, die sich aus der C-Shell in die Bash hinübergerettet hat). Die Zeichenfolge »!"« hat aber keine Funktion, sondern gilt als Syntaxfehler.

8.9 Keiner.

8.10 Wenn der Dateiname als Parameter übergeben wird, fühlt `wc` sich bemüßigt, ihn mit der Zeilenzahl auszugeben. Wenn `wc` von der Standardeingabe liest, gibt es nur die reine Zeilenzahl aus.

8.11 Versuchen Sie etwas wie

```
#!/bin/bash
pattern=$1
shift
◁◁◁◁◁
for f
do
    grep $pattern "$f" && cp "$f" backup
done
```

Nach dem `shift` ist das Suchmuster nicht mehr der erste Parameter, und das wirkt sich auch auf »`for f`« aus.

8.12 Der `-f`-Dateitest bezieht sich, wenn er auf ein symbolisches Link angewendet wird, auf die Datei (oder Verzeichnis oder was auch immer), auf die das Link zeigt. Er ist also auch dann erfolgreich, wenn der betrachtete Name eigentlich nur ein symbolisches Link ist. (Warum hat `filetest2` dieses Problem *nicht*?)

8.14 Bei der ersten Kommandozeile müssen Sie insgesamt 10 Sekunden auf eine neue Eingabeaufforderung der Shell warten. Bei der zweiten beträgt die Wartezeit 5 Sekunden, danach wird das zweite `sleep` im Hintergrund gestartet. Bei der dritten Kommandozeile erscheint die neue Eingabeaufforderung sofort, und es werden zwei Hintergrundprozesse erzeugt.

9.2 Das können Sie mit etwas wie

```
ls /bin /sbin /usr/bin /usr/sbin | wc -l
```

bestimmen. Alternativ dazu können Sie einfach an der Eingabeaufforderung der Shell zweimal Tab drücken – die Shell antwortet dann mit etwas wie

```
Display all 2371 possibilities? (y or n)
```

und das ist – in Abhängigkeit von Ihrem `PATH` – Ihre Antwort. (Wenn Sie als normaler Benutzer angemeldet sind, dann sind die Programme in `/sbin` und `/usr/sbin` in der Regel nicht dabei.)

9.3 Benutzen Sie statt »`grep` ⟨*Muster*⟩ `*.txt`« das Kommando »`grep` ⟨*Muster*⟩ `*.txt /dev/null`«. Damit hat `grep` immer mindestens zwei Dateinamenparameter, wobei `/dev/null` die Ausgabe nicht weiter verfälscht. – Die unter Linux gebräuchliche GNU-Implementierung von `grep` unterstützt eine Option `-H`, die dasselbe bewirkt, aber das ist nicht portabel.

9.4 Bei `cp` auf eine existierende Zieldatei wird die Datei zum Schreiben geöffnet und auf die Länge 0 gesetzt, bevor die Quelldaten hineingeschrieben werden. Bei `/dev/null` führt das nur dazu, dass die Quelldaten verschwinden. Bei `mv` auf eine existierende Zieldatei wird die Zieldatei jedoch erst gelöscht – und das ist eine Verzeichnisoperation, die ungeachtet der besonderen Natur von `/dev/null` einfach den Namen `null` aus dem Verzeichnis `/dev` entfernt und eine neue Datei namens `null` mit dem Inhalt von `bla.txt` dort anlegt.

9.6 Es ist unklug, weil es zum einen nicht richtig funktioniert, zum zweiten die Daten sowieso nicht sichernswert sind, weil sie sich ständig ändern (man verschwendet also jede Menge Platz auf dem Sicherungsmedium und Zeit zum Kopieren), und zum dritten eine solche Sicherheitskopie überhaupt nicht wieder eingespielt werden kann. Unkontrollierte Schreibvorgänge zum Beispiel auf `/proc/kcore` führen mit an Sicherheit grenzender Wahrscheinlichkeit zum Systemabsturz.

10.1 Für einen normalen Benutzer gelten Zugriffsrechte, für den Administrator nicht. `root` darf alles! Mit dem `root`-Konto sollten nur Befehle ausgeführt werden, für die Sie wirklich `root`-Rechte benötigen, z. B. zur Partitionierung, beim Dateisystem zuweisen, beim Benutzer anlegen und für Veränderungen an systemweiten Konfigurationsdateien. Alle anderen Tätigkeiten sollten unter einem normalen Konto durchgeführt werden. Dazu gehören auch Tätigkeiten wie das Aufrufen von Admin-Befehlen zu Informationszwecken und das Auspacken von `tar`-Archiven.

10.2 Als `root` dürfen Sie alles, daher ist es sehr einfach, das System zu beschädigen, wenn Sie sich z. B. versehentlich vertippen.

10.3 Die Frage zielt auf einen Vergleich mit anderen Betriebssystemen ab. Hier finden Sie je nach System gar keine Zugriffsrechte (DOS, Windows 95/98) oder andersartige Verfahren zur Rechtevergabe (Windows NT/2000/XP oder Windows Vista). Entsprechend gibt es bei ersteren auch keinen Administrator, während letztere sogar das Anlegen mehrerer Administratorkonten zulassen.

10.4 Prinzipiell gibt es die Möglichkeiten, sich als `root` anzumelden oder per `su` eine Shell mit UID 0 zu erlangen. Letzteres ist die bessere Methode, u. a. weil hier der Wechsel inklusive vorheriger UID mitprotokolliert wird.

10.5 Die Eingabeaufforderung sieht oft anders aus. Außerdem hilft z. B. der Befehl `id` weiter.

10.6 Hier stehen Ihnen der Login und `su` zur Verfügung. Für häufige Wechsel bietet es sich an, sich auf zwei Konsolen anzumelden und auf einem mit `su` eine `root`-Shell zu bekommen. Alternativ können Sie auf der grafischen Oberfläche mehrere grafische Terminalfenster parallel öffnen oder, falls es auf Ihrem System existiert, `sudo` verwenden.

10.7 Den Eintrag finden Sie in der Regel in der Datei `/var/log/messages`. Alternative Plätze wären, je nach Distribution, zum Beispiel `/var/log/auth.log` oder `/var/log/syslog`.

10.10 Der offensichtliche Vorteil ist, dass Administration von überall möglich ist, notfalls per Internet-Handy am Strand, und ohne dass spezielle Hard- oder Software zur Verfügung stehen muss. Der offensichtliche Nachteil ist, dass Sie den Zugang zu dem Administrationswerkzeug sehr gründlich absichern müssen, damit nicht ungebetene Gäste Ihr System »verkonfigurieren« oder Schlimmeres

anstellen. Dies kann bedeuten, dass Sie das Administrationswerkzeug dem offensichtlichen Vorteil zum Trotz nur im lokalen Netz zur Verfügung stellen oder den Zugang über starke Kryptografie (etwa SSL mit Client-Zertifikaten) absichern sollten. – Wenn Sie gedenken, Webmin bei sich in der Firma einzuführen, sollten Sie die Möglichkeit etwaiger externer Zugänge *dringend* mit den relevanten Entscheidungsträgern und/oder dem betrieblichen Datenschutzbeauftragten abstimmen, um extrem großen Ärger zu vermeiden, den es sonst geben könnte, falls Probleme auftreten. Wir haben Sie gewarnt.

11.1 Durch deren numerische UID und GID.

11.2 Das funktioniert, ist aber nicht notwendigerweise eine gute Idee. Für das System ist das dann derselbe Benutzer, das heißt, alle Dateien und Prozesse mit der entsprechenden UID gehören beiden Benutzern.

11.3 Die UIDs von Pseudobenutzern werden von (Dienst-)Programmen genutzt, die dadurch exakt definierte Zugriffsrechte bekommen.

11.4 Wer in der Gruppe `disk` ist, hat Lese- und Schreibrecht auf die Platten auf Blockebene. Mit Kenntnissen über die Dateisystemstruktur ist es einfach, eine Kopie von `/bin/sh` zu einer SUID-`root`-Shell (Abschnitt 12.5) zu machen, indem man direkt die Verwaltungsinformationen auf der Platte ändert. Damit ist Mitgliedschaft in der Gruppe `disk` im Wesentlichen äquivalent zu `root`-Rechten; Sie sollten niemanden in die Gruppe `disk` aufnehmen, dem Sie nicht das `root`-Kennwort verraten würden.

11.5 Dort finden Sie in der Regel ein »x«. Das ist ein Verweis, dass das Kennwort, welches normalerweise dort stehen würde, in einer anderen Datei steht, nämlich in der `/etc/shadow`, die im Gegensatz zu der ersten Datei nur für `root` lesbar ist.

11.6 Es gibt im Grunde zwei Möglichkeiten:

1. Nichts. In diesem Fall sollte das System Sie abweisen, nachdem Sie auch noch Ihr Kennwort eingegeben haben, weil kein Benutzerkonto mit dem rein großgeschriebenen Benutzernamen korrespondiert.
2. Das System redet ab jetzt mit Ihnen ebenfalls rein in Großbuchstaben. In diesem Fall geht Ihr Linux-System davon aus, dass Sie vor einem absolut vorsintflutlichen Terminal (1970er Jahre oder so) sitzen, das keine Kleinbuchstaben anzeigen kann, und schaltet die Verarbeitung der Eingabe- und Ausgabedaten freundlicherweise so um, dass Großbuchstaben in der Eingabe als Kleinbuchstaben interpretiert und Kleinbuchstaben in der Ausgabe als Großbuchstaben angezeigt werden. Heute ist das von eingeschränktem praktischen Nährwert (es sei denn, Sie arbeiten in einem Computermuseum), und Sie sollten sich schleunigst wieder abmelden, bevor Ihnen der Kopf platzt. Weil dieses Benehmen so atavistisch ist, spielt auch nicht jede Linux-Distribution hierbei mit.

11.7 Benutzen Sie `getent`, `cut` und `sort`, um Listen der Benutzernamen für die Datenbanken zu generieren und `comm`, um die beiden Listen zu vergleichen.

11.8 Verwenden Sie den Befehl `passwd`, wenn Sie als Benutzer `hugo` eingeloggt sind oder »`passwd hugo`« als `root`. In der Datei sollte dann ein anderer Eintrag im zweiten Feld auftauchen, das Datum der letzten Kennwortänderung (Feld 3) sollte das aktuelle Datum tragen (in welcher Einheit?).

11.9 Sie geben ihm mit »`passwd trottel`« als `root` ein neues Kennwort, denn lesen können Sie sein altes auch mit `root`-Rechten nicht.

11.10 Verwenden Sie dazu den Befehl »`passwd -n 7 -x 14 -w 2 hugo`«. Testen können Sie die Einstellungen mit »`passwd -S`«.

11.11 Zum Anlegen des Benutzers verwenden Sie das Programm `useradd`, zum Verändern der UID »`usermod -u`«. Anstatt des Login-Namens sollte den Dateien nur noch eine UID als Besitzer zugeordnet sein, zu der ja kein Login-Name mehr bekannt ist …

11.12 Für jedes der drei Konten sollte je eine Zeile in `/etc/passwd` und in `/etc/shadow` vorhanden sein. Um mit den Konten arbeiten zu können, benötigen Sie nicht unbedingt ein Kennwort (Sie können als `root` das Programm `su` verwenden), wohl aber, um sich unter der neuen Kennung einzuloggen. Eine Datei können Sie auch ohne Heimatverzeichnis anlegen und zwar in `/tmp` (falls Sie es vergessen haben, ein Heimatverzeichnis wäre für einen Benutzer aber schon nicht schlecht).

11.13 Verwenden Sie zum Löschen den Befehl `userdel`. Um die Dateien zu löschen verwenden Sie das Kommando »`find / -uid` ⟨*UID*⟩ `-delete`« .

11.14 Verwenden Sie »`usermod -u`«, dann müssen Sie die Dateien des Benutzers der neuen UID zuordnen, zum Beispiel mit »`find / -uid` ⟨*UID*⟩ `-exec chown test1 {} ';'`« oder (besser) »`chown -R --from=`⟨*UID*⟩ `test1 /`«. ⟨*UID*⟩ ist dabei jeweils die (numerische) alte UID.

11.15 Sie können dazu unter anderem `/etc/passwd` mit `vipw` editieren, oder Sie verwenden `usermod`.

11.16 Gruppen bieten die Möglichkeit, differenziert Rechte an Gruppen (ach was?) von Benutzern zu vergeben. So können Sie zum Beispiel alle Mitarbeiter Ihrer Personalabteilung einer Gruppe zuordnen und dieser Gruppe ein eigenes Arbeitsverzeichnis auf einem File-Server zuteilen. Außerdem können Gruppen dazu dienen, die Rechte zum Zugriff auf bestimmte Peripheriegeräte zu steuern (etwa über die Gruppen `disk`, `audio` oder `video`).

11.17 Verwenden Sie das Kommando »`mkdir` ⟨*Verzeichnis*⟩« zum Erstellen und »`chgrp` ⟨*Gruppenname*⟩ ⟨*Verzeichnis*⟩« zum Verschenken des Verzeichnisses. Sinnvollerweise setzen Sie dann noch das SGID-Bit, damit im Verzeichnis angelegte Dateien sofort der Gruppe gehören.

11.18 Verwenden Sie dazu die folgenden Kommandos:

```
# groupadd test
# gpasswd -a test1 test
Adding user test1 to group test
# gpasswd -a test2 test
Adding user test2 to group test
# gpasswd test
Changing the password for group test
New Password:x9q.Rt/y
Re-enter new password:x9q.Rt/y
```

Zum Gruppenwechsel verwenden Sie das Kommand »`newgrp test`«. Nur wenn Sie nicht Mitglied der entsprechenden Gruppe sind, werden Sie nach dem Kennwort gefragt.

12.1 Eine neue Datei bekommt die Gruppe, die gerade Ihre primäre Gruppe ist. Sie können eine Datei keiner Gruppe zuordnen, in der Sie nicht selber Mitglied sind – es sei denn, Sie sind `root`.

12.3 077 beziehungsweise »`u=rwx,go=`«.

12.5 Hierbei handelt es sich um das SUID- bzw. SGID-Bit. Die Bits bewirken, dass ein Prozess die UID/GID der ausführbaren Datei und nicht des ausführenden Benutzers bekommt. Sie bekommen es mit »`ls -l`« angezeigt. Selbstverständlich können Sie alle Rechte von eigenen Dateien ändern. Sinnvoll ist zumindest das SUID-Bit aber nur bei echten ausführbaren Dateien, also nicht bei Shell-Skripten o.ä.

12.6 Eine der beiden folgenden (gleichwertigen) Möglichkeiten genügt:

```
$ umask 007
$ umask -S u=rwx,g=rwx
```

Sie werden sich vielleicht fragen, was die `x`-Bits in dieser *umask* zu suchen haben. Für Dateien sind sie in der Tat nicht relevant, da Dateien standardmäßig nicht ausführbar erzeugt werden. Allerdings könnte es sein, dass im Projektverzeichnis auch Unterverzeichnisse erwünscht sind, und dann ist es sinnvoll, diese gleich so mit Rechten zu versehen, dass man mit ihnen auch vernünftig arbeiten kann.

12.7 Das sogenannte *sticky bit* bewirkt in den betroffenen Verzeichnissen, dass nur der Besitzer einer Datei diese auch löschen/umbenennen kann. Sie finden es unter anderem beim Verzeichnis `/tmp`.

12.9 Mit der Bash funktioniert das nicht (jedenfalls nicht ohne Weiteres). Für andere Shells können wir hier nicht sprechen.

12.11 Nur mit `chattr` geht das nicht, da verschiedene Attribute zwar mit `lsattr` angezeigt werden, aber nicht mit `chattr` gesetzt werden können. Die Details stehen in `chattr`(1). – Dazu kommt außerdem, dass manche Attribute nur für »echte« Dateien und andere nur für Verzeichnisse erklärt sind; Sie werden zum Beispiel Schwierigkeiten bekommen, für dasselbe »Dateisystemobjekt« gleichzeitig die `D`- und `E`-Attribute sichtbar zu machen. (Das `E`-Attribut hat mit transparenter Komprimierung zu tun, die für Verzeichnisse nicht verwendet wird, während `D` nur für Verzeichnisse gilt – Schreibzugriffe auf solche Verzeichnisse werden synchron ausgeführt.)

13.1 Im Verzeichnis für einen Prozess unter `/proc` befindet sich eine Datei `environ`, die die Umgebungsvariablen des Prozesses enthält. Diese Datei können Sie mit `cat` ausgeben. Einzige Unschönheit ist, dass die Variablen in dieser Datei durch Nullbytes voneinander getrennt sind, was die Bildschirmdarstellung stört; bequemerweise benutzt man also zum Anzeigen etwas wie »`tr "\0" "\n" </proc/4711/environ`«.

13.2 Ulkigerweise ist die Grenze nirgendwo offensichtlich dokumentiert. In `/usr/include/linux/threads.h` wird beim Linux-2.6-Kern die Konstante `PID_MAX_LIMIT` mit dem Wert 32768 definiert; dies ist der niedrigste Wert, der standardmäßig *nicht* an Prozesse vergeben wird. Den tatsächlichen Wert können Sie in `/proc/sys/kernel/pid_max` abfragen (oder einstellen – das Maximum für 32-Bit-Plattformen ist tatsächlich 32768, während Sie auf 64-Bit-Systemen einen beliebigen Wert bis zu 2^{22}, also etwa 4 Millionen, wählen können).

Die vergebenen PIDs steigen zunächst monoton an. Wird die eben diskutierte Obergrenze erreicht, beginnt die Vergabe wieder bei niedrigen PIDs, wobei PIDs, die noch von Prozessen belegt sind, natürlich nicht an neue Prozesse vergeben werden. Viele niedrige PIDs werden während des Bootvorgangs an langlaufende Hintergrundprozesse (Daemons) vergeben; aus diesem Grund wird nach dem Erreichen der Obergrenze nicht ab der PID 1, sondern erst ab der PID 300 nach

freien PIDs gesucht. (Die Details finden sich in `kernel/pid_namespace.c` im Linux-Quellcode.)

13.4 Zombies entstehen, wie erwähnt, wenn der Elterprozess den Rückgabewert eines Kindprozesses nicht abfragt. Um einen Zombie zu erzeugen, müssen Sie also einen Kindprozess starten und den Elterprozess anschließend daran hindern, dessen Rückgabewert abzufragen, etwa indem Sie ihn durch ein Signal anhalten. Probieren Sie etwas wie

```
$ sh
$ echo $$                                                    In der Sub-Shell
12345
$ sleep 20
                                                        In einem anderen Fenster:
$ kill -STOP 12345
                                                                      Warten
$ ps u | grep sleep
hugo 12346 0.0  0.0  3612  456 pts/2  Z 18:19 0:00 sleep 20
```

13.5 Konsultieren Sie `ps`(1).

13.6 Probieren Sie

```
$ ps -o pid,ppid,state,cmd
```

13.7 Normalerweise `SIGCHLD` (Kindprozess wurde beendet – manchmal auch `SIGCLD` genannt), `SIGURG` (eilige Daten wurden auf einer Netzverbindung empfangen) und `SIGWINCH` (die Größe des Fensters für ein textorientiertes Programm wurde geändert). Diese drei Ereignisse sind so nichtig, dass man ihretwegen den Prozess nicht abbrechen sollte.

13.8 Etwas wie

```
$ pgrep -u hugo
```

sollte dafür ausreichen.

13.10 Verwenden Sie z. B. das Kommando `renice -10 PID`. Sie können allerdings nur als `root` negative *Nice*-Werte vergeben.

13.12 Das Ressourcenlimit funktioniert nur für einzelne Prozesse, die Kontingentierung gilt für den insgesamt von einem Benutzer (oder einer Gruppe) verbrauchten Plattenplatz.

14.2 `sda1`, `sda2`, `sda5`, `sda6` sowie `sdb1`, `sdb5`, `sdb6`, `sdb7`.

15.2 Verwenden Sie `tune2fs` mit den Optionen `-c`, `-u` und `-m`.

15.3 `mkreiserfs /dev/sdb5`

15.6 In `/etc/fstab` werden alle häufig verwendeten Dateisysteme mit ihren Mountpunkten eingetragen, in `/etc/mtab` sind alle momentan eingehängten Dateisysteme eingetragen.

16.1 Der Bootlader darf sich im MBR, in einem anderen Bootsektor oder in einer Datei auf der Festplatte befinden. In den beiden letzten Fällen brauchen Sie aber noch einen zusätzlichen Bootlader, der den Linux-Bootlader seinerseits laden kann. So oder so, Sie brauchen für ein Linux-System auf jeden Fall einen Bootlader, der einen Linux-Kernel booten kann, also zum Beispiel GRUB (es gibt andere).

16.2 Siehe Text der Unterlage.

16.3 Vergeben Sie ein Kennwort, das die nichtautorisierte Eingabe von Kernelparametern verhindert. Bei GRUB Legacy z. B. mit

```
password --md5 ⟨verschlüsseltes Kennwort⟩
```

Für die Kennwortabfrage für ein bestimmtes Betriebssystem hilft `lock` weiter. Siehe auch den Unterlagentext.

17.3 Mit `runlevel` sehen Sie den vorherigen Runlevel und den aktuellen Runlevel. Sollte der vorhergehende Runlevel mit »N« angezeigt werden, heißt das, es gab keinen (engl. *none*). Zum Wechseln sagen Sie »`telinit 2`«, dann wieder `runlevel`.

17.4 Ein möglicher Eintrag für die `inittab`-Datei ist zum Beispiel

```
aa:A:ondemand:/bin/date >/tmp/runlevel-a.txt
```

Dieser Eintrag sollte die aktuelle Uhrzeit in die angegebene Datei schreiben, wenn Sie ihn mit »`telinit A`« aktivieren. Vergessen Sie vorher nicht das »`telinit q`«, damit `init` die Konfiguration neu liest.

17.5 Rufen Sie dazu das `syslog`-Init-Skript mit dem Parameter `restart` oder `reload` auf.

17.6 Zum Beispiel mit »`chkconfig -l`«.

17.7 Es ist verlockend, einfach die Links aus dem entsprechenden Runlevel-Verzeichnis zu entfernen. Allerdings kann es je nach Distribution passieren, dass diese bei der nächsten automatischen Änderung wieder zum Vorschein kommen. Wenn Ihre Distribution also ein Werkzeug wie `chkconfig` oder `insserv` verwendet, dann sollten Sie besser dieses verwenden.

17.8 Sie sollten damit rechnen, dass der Rechner Sie nach dem `root`-Kennwort fragt.

17.10 Verwenden Sie das Kommando

```
# shutdown -h +15 "Das ist nur ein Test"
```

– alles, was Sie `shutdown` nach der Zeitangabe mit auf den Weg geben, wird als Broadcast-Nachricht verschickt. Sie können zum Abbrechen entweder die Tastenkombination Strg + C verwenden, wenn Sie `shutdown` als Vordergrund-Prozess gestartet haben, oder »`shutdown -c`« verwenden.

17.11 Der Dateiname wird als Meldung verschickt.

18.4 Die Unit-Datei muss nicht geändert werden, um Abhängigkeiten auszudrücken. Dies erleichtert die automatische Installation und vor allem Deinstallation von Units als Bestandteil von Softwarepaketen (etwa im Kontext eines distributionsspezifischen Paketverwaltungswerkzeugs) und ermöglicht die problemlose Aktualisierung von Unit-Dateien durch die Distribution.

18.10 Ein genaues Äquivalent gibt es nicht, weil systemd das Runlevel-Konzept nicht kennt. Sie können sich aber alle gerade aktiven Ziele anzeigen lassen:

```
# systemctl list-units -t target
```

18.11 Bei »`systemctl kill`« ist garantiert, dass das Signal nur an die Prozesse der betreffenden Unit geschickt wird. Die anderen beiden Kommandos schicken das Signal an alle Prozesse, die zufällig `example` heißen.

18.13 Gar nicht (»`systemctl mask`« liefert eine Fehlermeldung). Sie müssen den Dienst deaktivieren und dann die Unit-Datei löschen, verschieben oder umbenennen.

19.1 (a) Am 1. März um 17 Uhr; (b) Am 2. März um 14 Uhr; (c) Am 2. März um 16 Uhr; (d) Am 2. März um 1 Uhr.

19.2 Etwa mit »`at now + 3 minutes`«.

19.4 Eine Möglichkeit wäre »`atq | sort -bk 2`«.

19.6 Die Aufgabenlisten-Datei gehört Ihnen, aber Sie haben kein Zugriffsrecht auf das `crontabs`-Verzeichnis. Bei Debian GNU/Linux gilt zum Beispiel

```
$ ls -ld /var/spool/cron/crontabs
drwx-wx--T 2 root crontab 4096 Aug 31 01:03 /var/spool/cron/crontabs
```

Das heißt, `root` hat (wie üblich) vollen Zugriff (hätte er sowieso), und Mitglieder der Gruppe `crontab` dürfen zumindest auf Dateien zugreifen, von denen sie wissen, wie sie heißen (`ls` ist nicht erlaubt). Das Programm `crontab` ist Set-GID `crontab`:

```
$ ls -l $(which crontab)
-rwxr-sr-x 1 root crontab 27724 Sep 28 11:33 /usr/bin/crontab
```

und wird damit mit den Rechten der Gruppe `crontab` ausgeführt, egal welcher Benutzer es aufruft. (Der Set-GID-Mechanismus ist in der Schulungsunterlage *Linux-Administration I* genauer erklärt.)

19.7 Registrieren Sie die Aufgabe für den Monats-Dreizehnten und fragen Sie im aufgerufenen Skript ab (etwa mit »`date +%u`«), ob gerade Freitag ist.

19.8 Die Details sind distributionsabhängig.

19.9 Die passende Zeile für das minütliche Protokoll ist etwas wie

```
* * * * * /bin/date >>/tmp/date.log
```

Für den Zwei-Minuten-Abstand können Sie natürlich etwas schreiben wie

```
0,2,4,<◁◁◁◁,56,58 * * * * /bin/date >>/tmp/date.log
```

aber

```
*/2 * * * * /bin/date >>/tmp/date.log
```

ist bequemer.

19.10 Die Kommandos dafür sind »`crontab -l`« bzw. »`crontab -r`«.

19.11 Sie sollten `hugo` in `/etc/cron.deny` (bei SUSE-Distributionen `/var/spool/cron/deny`) eintragen bzw. ihn aus `/etc/cron.allow` löschen.

19.13 In `/etc/cron.daily` steht ein Skript namens `0anacron`, das als erster Job ausgeführt wird. Dieses Skript ruft »`anacron -u`« auf; mit dieser Option aktualisiert `anacron` die Zeitstempel, ohne tatsächlich Jobs auszuführen (denn das macht `cron` dann als nächstes). Wenn das System neu gestartet wird, vermeidet `anacron` so überflüssige Job-Ausführungen, jedenfalls sofern der Neustart erst erfolgt, nachdem `cron` sein Ding getan hat.

20.1 Solche Ereignisse werden standardmäßig von `syslogd` in die Datei `/var/log/messages` geschrieben. Am elegantesten lösen Sie die Aufgabe so:

```
# grep 'su: (to root)' /var/log/messages
```

20.2 Tragen Sie in `/etc/syslog.conf` an irgendeiner Stelle die Zeile:

```
*.*                -/var/log/test
```

ein. Danach signalisieren Sie dem `syslogd` mit »`kill -HUP`«, seine Konfigurationsdatei neu zu lesen. Wenn Sie dann in `/var/log` schauen, sollte die neue Datei samt ersten Inhalten (welchen?) bereits vorhanden sein.

20.3 Auf dem empfangenden Rechner muss der `syslogd` mit dem Parameter `-r` gestartet werden (siehe S. 336). Der sendende Rechner braucht eine Konfigurationszeile der Form

```
local0.*     @blue.example.com
```

(wenn der empfangende Rechner den Namen »`blue.example.com`« hat).

20.4 Die einzige sichere Methode besteht darin, das Protokoll dem Zugriff des Angreifers zu entziehen. Sie müssen die Nachrichten darum an einen anderen Rechner schicken. Wenn Sie nicht möchten, dass der Angreifer diesen Rechner auch noch kompromittiert, dann schließen Sie den protokollierenden Rechner über eine serielle Schnittstelle an den Rechner an, der das Protokoll speichert, und konfigurieren Sie `syslogd` so, dass er die Nachrichten an das entsprechende Gerät (`/dev/ttyS0` oder so) schickt. Auf dem Protokollrechner kann ein einfaches Programm die Nachrichten an der seriellen Schnittstelle entgegennehmen und speichern oder weiterverarbeiten. Alternativ können Sie natürlich auch einen (altmodischen) Matrix-Drucker mit Endlospapier verwenden.

20.5 Sie können unter anderem mit Informationen über die Speichermenge und -zuordnung, die vorhandenen Prozessoren, angeschlossene Platten und andere Massenspeicher (IDE und SCSI), USB-Geräte und Netzwerkkarten rechnen. Die Details hängen natürlich von Ihrem System und der Linux-Installation ab.

20.10 Versuchen Sie etwas wie

```
# Die Definition der Quelle setzen wir mal voraus
filter login_hugo {
    facility(authpriv)
      and (match("session opened") or match("session closed"))
      and match("user hugo");
};
```

```
destination d_root { usertty("root"); };
log { source(...);
    filter(login_hugo);
    destination(d_root);
};
```

20.12 Schauen Sie in `/etc/logrotate.conf` nach (und gegebenenfalls in `logrotate`(8)).

20.13 Erstellen Sie in `/etc/logrotate.d` eine Datei beliebigen Namens mit folgendem Inhalt:

```
/var/log/test {
    compress
    dateext
    rotate 10
    size 100
    create
}
```

21.1 Textdateien sind grundsätzlich den Standard-Unix-Werkzeugen (`grep` & Co.) zugänglich und darum ideologisch reiner. Man kann sie auch ohne spezialisierte Software anschauen. Außerdem ist das Prinzip sehr gut verstanden und es gibt jede Menge Werkzeuge, die sich um die Auswertung von traditionellen Protokolldateien kümmern. Als Nachteile kann man anführen, dass Textdateien umständlich zu durchsuchen sind und jede Form von gezielter Auswertung entweder extrem mühselig ist oder zusätzliche (nicht standardisierte) Software benötigt. Es gibt keine Form von kryptografischer Absicherung gegen die Manipulation von Protokolleinträgen, und der Umfang der Informationen, die ins Protokoll geschrieben werden können, ist beschränkt.

22.2 ISO/OSI-Schicht 2 beschreibt die Interaktion zwischen zwei direkt miteinander verbundenen Rechnern (z. B. über Ethernet). Schicht 3 beschreibt die Interaktion zwischen Rechnern, die nicht direkt miteinander verbunden sind, umfasst also Routing und medienunabhängige Adressierung (z. B. IP über Ethernet oder Token-Ring oder …).

22.3 Sie können sich Ideen in den Dateien `/etc/services` und (unter Umständen) `/etc/protocols` holen. Die Einordnung müssen Sie allerdings selbst vornehmen. *Tipp:* Praktisch alles, was in `/etc/services` erwähnt wird, kann der Anwendungsschicht zugeordnet werden.

22.4 Eine genaue Antwort läßt sich natürlich nicht geben, aber normalerweise sollte eine TTL von 30–40 mehr als ausreichen. (Der Standardwert ist 64.) Die minimale TTL können Sie ermitteln, indem Sie beginnend bei einer TTL von 1 sukzessive Pakete mit immer höherer TTL an das gewünschte Ziel schicken. Wenn Sie eine ICMP-Fehlermeldung eines Routers bekommen, dass das Paket verworfen wurde, ist die TTL noch nicht groß genug. (Das Programm `traceroute` automatisiert diesen Vorgang.) Diese »minimale« TTL ist natürlich keine Konstante, da IP keinen eindeutigen Weg für die Paketzustellung garantiert.

22.5

1. Die Adresse `172.55.10.3` kann nicht als Stationsadresse verwendet werden, denn sie ist die Broadcast-Adresse dieses Netzes.

2. Die Adresse `138.44.33.12` kann als Stationsadresse benutzt werden.

3. Die Adresse `10.84.13.160` ist die Netzwerkadresse des betreffenden Netzes und steht darum als Stationsadresse nicht zur Verfügung.

22.6 Beispielsweise um bestimmte Netzwerktopologien umzusetzen oder Teile des Adressbereichs an Rechner an unterschiedlichen Standorten zu vergeben (wenn der Provider mitmacht).

22.7 Es gibt insgesamt 16 Subnetze (was man auch daran sehen kann, dass die Subnetzmaske vier Einsen mehr hat als die ursprüngliche Netzmaske). Weitere Subnetze sind `145.2.0.0`, `145.2.16.0`, `145.2.48.0`, `145.2.80.0`, `145.2.96.0`, `145.2.112.0`, `145.2.144.0`, `145.2.176.0`, `145.2.208.0`, `145.2.224.0` und `145.2.240.0`. Die Station mit der IP-Adresse `145.2.195.13` liegt im Subnetz `145.2.192.0`.

23.1 `lsmod` zeigt alle geladenen Module an. »`rmmod` ⟨*Modulname*⟩« entlädt ein Modul, was aber fehlschlägt, wenn das Modul noch verwendet wird.

23.2 Das geht am einfachsten mit `ifconfig`.

23.3 Verwenden Sie »`ifconfig` ⟨*Interface*⟩ ⟨*IP-Adresse*⟩«. Um die Erreichbarkeit der anderen Rechner zu testen, verwenden Sie den Befehl `ping`.

24.2 Erstere sollte größenordnungsmäßig im Zehn-Mikrosekunden-Bereich liegen, letztere je nach Netzwerkinfrastruktur in der Gegend von Millisekunden.

24.3 Probieren Sie etwas wie

```
# ping -f -c 1000000 localhost
```

Die Laufzeit finden Sie am Ende der vorletzten Ausgabezeile von `ping`. (Auf dem System des Autors dauert es knapp 13 Sekunden.)

24.4 Zum Beispiel:

```
$ ping6 ff02::2%eth0
PING ff02::2%eth0(ff02::2) 56 data bytes
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=1 ttl=64 time=12.4 ms
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=2 ttl=64 time=5.27 ms
64 bytes from fe80::224:feff:fee4:1aa1: icmp_seq=3 ttl=64 time=4.53 ms
Strg+c
--- ff02::2%eth0 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 4.531/7.425/12.471/3.581 ms
```

25.1 Beim ersten Anmelden sollte `ssh` Sie um die Bestätigung des öffentlichen Rechner-Schlüssels des entfernten Rechners bitten. Beim zweiten Anmeldevorgang ist der öffentliche Rechner-Schlüssel in der Datei `~/.ssh/known_hosts` gespeichert und muss nicht mehr bestätigt werden.

25.2 Im ersten Fall wird die Sitzung abgewiesen, da kein öffentlicher Rechner-Schlüssel für den entfernten Rechner in `~/.ssh/known_hosts` steht. Im zweiten Fall wird die Sitzung ohne weitere Rückfrage aufgebaut.

25.3 Die Datei enthält ohnehin nur öffentliche Schlüssel und bedarf darum keiner besonderen Geheimhaltung.

25.6 Benutzen Sie etwas wie

```
$ ssh-keygen -l -f ~/.ssh/id_rsa.pub
```

(Macht es einen Unterschied, ob Sie `id_rsa.pub` oder `id_rsa` angeben?)

25.7 Nichtinteraktive Programme, die eine SSH-Verbindung nutzen wollen, können in der Regel keine *passphrase* eingeben. Für solche eingeschränkten Fälle ist es denkbar, einen privaten Schlüssel ohne *passphrase* zu verwenden. Dann sollten Sie aber von der Möglichkeit Gebrauch machen, den öffentlichen Schlüssel auf dem entfernten Rechner nur für bestimmte Kommandos nutzbar zu machen (nämlich die, die das nichtinteraktive Programm aufrufen muss). Details dazu stehen in `sshd`(8).

25.8 Versuchen Sie mal »`ssh -X root@localhost`«.

25.9 Eine mögliche Kommandozeile wäre

```
# ssh -L 4711:localhost:7 user@remote.example.com
```

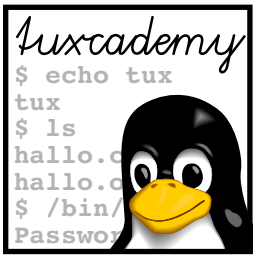
Beachten Sie hierbei, dass das `localhost` aus der Sicht des Rechners `remote` betrachtet wird. `ssh` läßt leider keine symbolischen Namen von *well-known ports* zu, so wie sie in `/etc/services` stehen.

26.11 Versuchen Sie mal

```
# dpkg-reconfigure debconf
```

27.1 Das geht am einfachsten mit etwas wie

```
$ rpm2cpio ⟨Paket⟩ | cpio -t
```



B

Beispieldateien

Als Beispiel verwenden wir an verschiedenen Stellen das Märchen vom Froschkönig, genauer gesagt »Der Froschkönig oder der eiserne Heinrich«, aus den »Deutschen Kinder- und Hausmärchen« der Gebrüder Grimm. Das Märchen drucken wir hier ganz und in der kompletten Form so wie in der Datei ab, um Vergleiche mit den Beispielen zuzulassen.

```
Der Froschkönig oder der eiserne Heinrich

In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
schön, daß die Sonne selber, die doch so vieles schon gesehen hat,
sich verwunderte, sooft sie ihr ins Gesicht schien.

Nahe bei dem Schlosse war ein großer, dunkler Wald, und mitten darin,
unter einer alten Linde, war ein Brunnen. Wenn nun der Tag recht heiß war,
ging die jüngste Prinzessin hinaus in den Wald und setzte sich an den Rand
des kühlen Brunnens. Und wenn sie Langeweile hatte, nahm sie eine
goldene Kugel, warf sie in die Höhe und fing sie wieder auf. Das war
ihr liebstes Spiel.

Nun trug es sich einmal zu, daß die goldene Kugel der Königstochter
nicht in die Händchen fiel, sondern auf die Erde schlug und gerade in
den Brunnen hineinrollte. Die Königstochter folgte ihr mit den Augen
nach, aber die Kugel verschwand, und der Brunnen war tief, so tief,
daß man keinen Grund sah.

Da fing die Prinzessin an zu weinen und weinte immer lauter und konnte
sich gar nicht trösten. Als sie so klagte, rief ihr plötzlich jemand
zu: »Was hast du nur, Königstochter? Du schreist ja, daß sich ein
Stein erbarmen möchte.«

Sie sah sich um, woher die Stimme käme, da erblickte sie einen Frosch,
der seinen dicken, häßlichen Kopf aus dem Wasser streckte. »Ach, du
bist's, alter Wasserpatscher«, sagte sie. »Ich weine über meine
goldene Kugel, die mir in den Brunnen hinabgefallen ist.«

»Sei still und weine nicht«, antwortete der Frosch, »ich kann wohl Rat
schaffen. Aber was gibst du mir, wenn ich dein Spielzeug wieder
heraufhole?«
```

»Was du haben willst, lieber Frosch«, sagte sie, »meine Kleider, meine Perlen und Edelsteine, auch noch die goldene Krone, die ich trage.«

Der Frosch antwortete: »Deine Kleider, deine Perlen und Edelsteine und deine goldene Krone, die mag ich nicht. Aber wenn du mich liebhaben willst und ich dein Geselle und Spielkamerad sein darf, wenn ich an deinem Tischlein neben dir sitzen, von deinem goldenen Tellerlein essen, aus deinem Becherlein trinken, in deinem Bettlein schlafen darf, dann will ich hinuntersteigen und dir die goldene Kugel heraufholen.«

»Ach, ja«, sagte sie, »ich verspreche dir alles, was du willst, wenn du mir nur die Kugel wiederbringst.« Sie dachte aber, der einfältige Frosch mag schwätzen, was er will, der sitzt doch im Wasser bei seinesgleichen und quakt und kann keines Menschen Geselle sein!

Als der Frosch das Versprechen der Königstochter erhalten hatte, tauchte er seinen Kopf unter, sank hinab, und über ein Weilchen kam er wieder heraufgerudert, hatte die Kugel im Maul und warf sie ins Gras. Die Königstochter war voll Freude, als sie ihr schönes Spielzeug wiedererblickte, hob es auf und sprang damit fort.

»Warte, warte!« rief der Frosch. »Nimm mich mit, ich kann nicht so laufen wie du!« Aber was half es ihm, daß er ihr sein Quak-quak so laut nachschrie, wie er nur konnte! Sie hörte nicht darauf, eilte nach Hause und hatte den Frosch bald vergessen.

Am andern Tag, als sie sich mit dem König und allen Hofleuten zur Tafel gesetzt hatte und eben von ihrem goldenen Tellerlein aß, da kam, plitsch platsch, plitsch platsch, etwas die Marmortreppe heraufgekrochen. Als es oben angelangt war, klopfte es an die Tür und rief. »Königstochter, jüngste, mach mir auf«

Sie lief und wollte sehen, wer draußen wäre. Als sie aber aufmachte, saß der Frosch vor der Tür. Da warf sie die Tür hastig zu, setzte sich wieder an den Tisch, und es war ihr ganz ängstlich zumute.

Der König sah wohl, daß ihr das Herz gewaltig klopfte, und sprach: »Mein Kind, was fürchtest du dich? Steht etwa ein Riese vor der Tür und will dich holen?«

»Ach, nein«, antwortete sie, »es ist kein Riese, sondern ein garstiger Frosch.«

»Was will der Frosch von dir?«

»Ach, lieber Vater, als ich gestern im Wald bei dem Brunnen saß und spielte, fiel meine goldene Kugel ins Wasser. Als ich deshalb weinte, hat sie mir der Frosch heraufgeholt. Und weil er es durchaus verlangte, versprach ich ihm, er sollte mein Spielgefährte werden. Ich dachte aber nimmermehr, daß er aus seinem Wasser käme. Nun ist er draußen und will zu mir herein.«

Da klopfte es zum zweiten Mal, und eine Stimme rief:

»Königstochter, jüngste,
Mach mir auf!
Weißt du nicht, was gestern

Du zu mir gesagt
Bei dem kühlen Brunnenwasser?
Königstochter, jüngste,
Mach mir auf!«

Da sagte der König: »Was du versprochen hast, das mußt du auch halten! Geh nur und mach ihm auf!«

Sie ging und öffnete die Tür. Da hüpfte der Frosch herein und hüpfte ihr immer nach bis zu ihrem Stuhl. Dort blieb er sitzen und rief: »Heb mich hinauf zu dir!« Sie zauderte, bis es endlich der König befahl. Als der Frosch auf dem Stuhl war, wollte er auf den Tisch, und als er da saß, sprach er: »Nun schieb mir dein goldenes Tellerlein näher, damit wir mitsammen essen können.« Der Frosch ließ sich's gut schmecken, ihr aber blieb fast jeder Bissen im Halse stecken.

Endlich sprach der Frosch: »Ich habe mich satt gegessen und bin müde. Nun trag mich in dein Kämmerlein und mach dein seidenes Bettlein zurecht!« Die Königstochter fing an zu weinen und fürchtete sich vor dem kalten Frosch, den sie sich nicht anzurühren getraute und der nun in ihrem schönen, reinen Bettlein schlafen sollte.

Der König aber wurde zornig und sprach: »Wer dir geholfen hat, als du in Not warst, den sollst du hernach nicht verachten!«

Da packte sie den Frosch mit zwei Fingern, trug ihn hinauf in ihr Kämmerlein und setzte ihn dort in eine Ecke. Als sie aber im Bette lag, kam er gekrochen und sprach: »Ich will schlafen so gut wie du. Heb mich hinauf, oder ich sag's deinem Vater!«

Da wurde sie bitterböse, holte ihn herauf und warf ihn gegen die Wand. »Nun wirst du Ruhe geben«, sagte sie, »du garstiger Frosch!« Als er aber herabfiel, war er kein Frosch mehr, sondern ein Königssohn mit schönen freundlichen Augen. Der war nun nach ihres Vaters Willen ihr lieber Geselle und Gemahl. Er erzählte ihr, er wäre von einer bösen Hexe verwünscht worden, und niemand hätte ihn aus dem Brunnen erlösen können als sie allein, und morgen wollten sie mitsammen in sein Reich gehen.

Und wirklich, am anderen Morgen kam ein Wagen herangefahren, mit acht weißen Pferden bespannt, die hatten weiße Straußfedern auf dem Kopf und gingen in goldenen Ketten. Hinten auf dem Wagen aber stand der Diener des jungen Königs, das war der treue Heinrich.

Der treue Heinrich hatte sich so gekränkt, als sein Herr in einen Frosch verwandelt worden war, daß er drei eiserne Bänder um sein Herz hatte legen lassen, damit es ihm nicht vor Weh und Traurigkeit zerspränge.

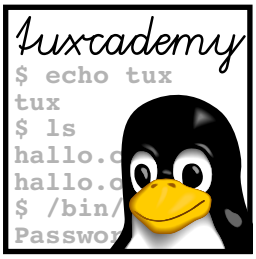
Der Wagen sollte nun den jungen König in sein Reich holen. Der treue Heinrich hob ihn und seine junge Gemahlin hinein, stellte sich wieder hinten hinauf und war voll Freude über die Erlösung seines Herrn. Als sie ein Stück des Weges gefahren waren, hörte der Königssohn, daß es hinter ihm krachte, als ob etwas zerbrochen wäre. Da drehte er sich um und rief:

»Heinrich, der Wagen bricht!«
»Nein, Herr, der Wagen nicht,

```
 Es ist ein Band von meinem Herzen,
 Das da lag in großen Schmerzen,
 Als Ihr in dem Brunnen saßt
 Und in einen Frosch verzaubert wart.«

Noch einmal und noch einmal krachte es auf dem Weg, und der Königssohn
meinte immer, der Wagen bräche. Doch es waren nur die Bänder, die vom
Herzen des treuen Heinrich absprangen, weil sein Herr nun erlöst und
glücklich war.
```

(Die Linup Front GmbH weist ausdrücklich darauf hin, dass die Autoren dieser Dokumentation jegliche Tierquälerei verurteilen.)



C

LPIC-1-Zertifizierung

C.1 Überblick

Das *Linux Professional Institute* (LPI) ist eine herstellerunabhängige, nicht profitorientierte Organisation, die sich der Förderung des professionellen Einsatzes von Linux widmet. Ein Aspekt der Arbeit des LPI ist die Erstellung und Durchführung weltweit anerkannter, distributionsunabhängiger Zertifizierungsprüfungen beispielsweise für Linux-Systemadministratoren.

Mit der „LPIC-1"-Zertifizierung des LPI können Sie nachweisen, dass Sie über grundlegende Linux-Kenntnisse verfügen, wie sie etwa für Systemadministratoren, Entwickler, Berater oder Mitarbeiter bei der Anwenderunterstützung sinnvoll sind. Die Zertifizierung richtet sich an Kandidaten mit etwa 1 bis 3 Jahren Erfahrung und besteht aus zwei Prüfungen, LPI-101 und LPI-102. Diese werden in Form von computerorientierten Multiple-Choice- und Kurzantworttests über die Prüfungszentren von Pearson VUE und Thomson Prometric angeboten oder können auf Veranstaltungen wie dem LinuxTag oder der CeBIT zu vergünstigten Preisen auf Papier abgelegt werden. Das LPI veröffentlicht auf seinen Web-Seiten unter `http://www.lpi.org/` die **Prüfungsziele**, die den Inhalt der Prüfungen umreißen.

Prüfungsziele

Die vorliegende Unterlage ist Teil eines Kurskonzepts der Linup Front GmbH zur Vorbereitung auf die Prüfung LPI-101 und deckt damit einen Teil der offiziellen Prüfungsziele ab. Details können Sie den folgenden Tabellen entnehmen. Eine wichtige Beobachtung in diesem Zusammenhang ist, dass die LPIC-1-Prüfungsziele nicht dazu geeignet oder vorgesehen sind, einen Einführungskurs in Linux didaktisch zu strukturieren. Aus diesem Grund verfolgt unser Kurskonzept keine strikte Ausrichtung auf die Prüfungen oder Prüfungsziele in der Form „Belegen Sie Kurs x und y, machen Sie Prüfung p, dann belegen Sie Kurs a und b und machen Sie Prüfung q". Ein solcher Ansatz verleitet viele Kurs-Interessenten zu der Annahme, sie könnten als absolute Linux-Einsteiger n Kurstage absolvieren (mit möglichst minimalem n) und wären anschließend fit für die LPIC-1-Prüfungen. Die Erfahrung lehrt, dass das in der Praxis nicht funktioniert, da die LPI-Prüfungen geschickt so angelegt sind, dass Intensivkurse und prüfungsorientiertes „Büffeln" nicht wirklich helfen.

Entsprechend ist unser Kurskonzept darauf ausgerichtet, Ihnen in didaktisch sinnvoller Form ein solides Linux-Basiswissen zu vermitteln und Sie als Teilnehmer in die Lage zu versetzen, selbständig mit dem System zu arbeiten. Die LPIC-1-Zertifizierung ist nicht primäres Ziel oder Selbstzweck, sondern natürliche Folge aus Ihren neuerworbenen Kenntnissen und Ihrer Erfahrung.

C.2 Prüfung LPI-101

Die folgende Tabelle zeigt die Prüfungsziele der Prüfung LPI-101 (Version 3.5) und die Unterlagen der „Linux kompakt“-Serie, die diese Prüfungsziele abdecken. Die Zahlen in den Spalten für die einzelnen Unterlagen verweisen auf die Kapitel, die das entsprechende Material enthalten.

Nr	Gew	Titel	LXK1
101.1	2	Hardware-Einstellungen ermitteln und konfigurieren	–
101.2	3	Das System starten	14
101.3	3	Runlevel wechseln und das System anhalten oder neu starten	15
102.1	2	Festplattenaufteilung planen	13
102.2	2	Einen Boot-Manager installieren	14
102.3	1	Shared Libraries verwalten	–
102.4	3	Debian-Paketverwaltung verwenden	–
102.5	3	RPM- und YUM-Paketverwaltung verwenden	–
103.1	4	Auf der Kommandozeile arbeiten	3–4
103.2	3	Textströme mit Filtern verarbeiten	7
103.3	4	Grundlegende Dateiverwaltung	6, 13.7
103.4	4	Ströme, Pipes und Umleitungen verwenden	7
103.5	4	Prozesse erzeugen, überwachen und beenden	3.8, 12
103.6	2	Prozess-Ausführungsprioritäten ändern	12
103.7	2	Textdateien mit regulären Ausdrücken durchsuchen	7
103.8	3	Grundlegendes Editieren von Dateien mit dem vi	5
104.1	2	Partitionen und Dateisysteme anlegen	13
104.2	2	Die Integrität von Dateisystemen sichern	13
104.3	3	Das Ein- und Aushängen von Dateisystemen steuern	13
104.4	1	Platten-Quotas verwalten	13.8
104.5	3	Dateizugriffsrechte und -eigentümerschaft verwalten	11
104.6	2	Harte und symbolische Links anlegen und ändern	6
104.7	2	Systemdateien finden und Dateien am richtigen Ort platzieren	6, 8

C.3 Prüfung LPI-102

Die folgende Tabelle zeigt die Prüfungsziele der Prüfung LPI-102 (Version 3.5) und die Unterlagen der „Linux kompakt“-Serie, die diese Prüfungsziele abdecken. Die Zahlen in den Spalten für die einzelnen Unterlagen verweisen auf die Kapitel, die das entsprechende Material enthalten.

Nr	Gew	Titel	LXK1
105.1	4	Die Shell-Umgebung anpassen und verwenden	–
105.2	4	Einfache Skripte anpassen oder schreiben	–
105.3	2	SQL-Datenverwaltung	–
106.1	2	X11 installieren und konfigurieren	–
106.2	2	Einen Display-Manager einrichten	–
106.3	1	Hilfen für Behinderte	–
107.1	5	Benutzer- und Gruppenkonten und dazugehörige Systemdateien verwalten	10
107.2	4	Systemadministrationsaufgaben durch Einplanen von Jobs automatisieren	16
107.3	3	Lokalisierung und Internationalisierung	–
108.1	3	Die Systemzeit verwalten	–
108.2	2	Systemprotokollierung	17
108.3	3	Grundlagen von Mail Transfer Agents (MTAs)	–
108.4	2	Drucker und Druckvorgänge verwalten	22
109.1	4	Grundlagen von Internet-Protokollen	18–19
109.2	4	Grundlegende Netz-Konfiguration	19–20
109.3	4	Grundlegende Netz-Fehlersuche	19–20
109.4	2	Clientseitiges DNS konfigurieren	19
110.1	3	Administrationsaufgaben für Sicherheit durchführen	10, 19–20
110.2	3	Einen Rechner absichern	10, 19
110.3	3	Daten durch Verschlüsselung schützen	21

C.4 LPI-Prüfungsziele in dieser Schulungsunterlage

101.2 Das System starten

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, das System durch den Startvorgang zu geleiten.

Wichtigste Wissensgebiete

- Zur Startzeit dem Bootlader gängige Kommandos und dem Systemkern Optionen übergeben
- Wissen über den Startvorgang vom BIOS zum Abschluss des Systemstarts demonstrieren
- Ereignisse beim Systemstart in den Protokolldateien nachschlagen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/var/log/messages`
- `dmesg`
- BIOS
- Bootlader
- Systemkern
- `init`

101.3 Runlevel wechseln und das System anhalten oder neu starten

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, den Runlevel des Systems zu verwalten. Dieses Prüfungsziel umfasst das Wechseln in den Einbenutzermodus, das Anhalten und den Neustart des Systems. Kandidaten sollten in der Lage sein, Benutzer vor einem Wechsel des Runlevels zu benachrichtigen und Prozesse korrekt anzuhalten. Dieses Prüfungsziel umfasst ferner das Einstellen des Standard-Runlevels sowie grundlegendes Wissen über die Eigenschaften möglicher neuer Init-Systeme.

Wichtigste Wissensgebiete

- Den Standard-Runlevel setzen
- Zwischen Runlevels wechseln, einschließlich dem Einbenutzermodus
- Systemhalt und Neustart von der Kommandozeile
- Benutzer vor einem Runlevel-Wechsel oder anderem größerem Ereignis benachrichtigen
- Prozesse korrekt beenden
- Wissen über grundlegende Eigenschaften von systemd und Upstart.

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/inittab`
- `init`
- `telinit`
- `shutdown`
- `/etc/init.d`

102.1 Festplattenaufteilung planen

Gewicht 2

Beschreibung Kandidaten sollten ein Platten-Partitionierungsschema für ein Linux-System entwerfen können.

Wichtigste Wissensgebiete

- Dateisysteme und Swap Space einzelnen Partition oder Platten zuordnen
- Die Partitionierung an den Einsatzzweck des Systems anpassen
- Sicherstellen, dass die `/boot`-Partition den Anforderungen der Hardware-Architektur für den Systemstart genügt
- Wissen über grundlegende Eigenschaften von LVM.

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/` (Wurzel- bzw. root-Dateisystem)
- `/var`-Dateisystem
- `/home`-Dateisystem
- Swap Space
- Mount Points
- Partitionen

102.2 Einen Boot-Manager installieren

Gewicht 2

Beschreibung Kandidaten sollten einen Boot-Manager auswählen, installieren und konfigurieren können.

Wichtigste Wissensgebiete

- Alternative und Notfall-Startmöglichkeiten vorsehen
- Einen Bootlader wie GRUB Legacy installieren und konfigurieren
- Einfache Konfigurationsänderungen an GRUB 2 vornehmen.
- Mit dem Bootlader interagieren

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/boot/grub/menu.lst`, `grub.cfg` und andere Variationen.
- `grub-install`
- MBR
- Superblock

103.1 Auf der Kommandozeile arbeiten

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, über die Kommandozeile mit Shells und Kommandos zu interagieren. Dieses Prüfungsziel setzt die Bash als Shell voraus.

Wichtigste Wissensgebiete

- Einzelne Shellkommandos und einzeilige Kommandofolgen verwenden, um einfache Aufgaben auf der Kommandozeile zu lösen
- Die Shellumgebung verwenden und anpassen, etwa um Umgebungsvariable zu definieren, zu verwenden und zu exportieren
- Die Kommando-Vorgeschichte verwenden und ändern
- Kommandos innerhalb und außerhalb des definierten Suchpfads aufrufen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `.`
- `bash`
- `echo`
- `env`
- `exec`
- `export`
- `pwd`
- `set`
- `unset`
- `man`
- `uname`
- `history`
- `.bash_history`

103.2 Textströme mit Filtern verarbeiten

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, Filter auf Textströme anzuwenden.

Wichtigste Wissensgebiete

- Textdateien und Ausgabeströme durch Text-Filter schicken, um die Ausgabe mit Standard-UNIX-Kommandos aus dem GNU-textutils-Paket zu verändern

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `cat`
- `cut`
- `expand`
- `fmt`
- `head`
- `od`
- `join`
- `nl`
- `paste`
- `pr`
- `sed`
- `sort`
- `split`
- `tail`
- `tr`
- `unexpand`
- `uniq`
- `wc`

103.3 Grundlegende Dateiverwaltung

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, die grundlegenden Linux-Kommandos zur Verwaltung von Dateien und Verzeichnissen zu verwenden.

Wichtigste Wissensgebiete

- Einzelne Dateien und Verzeichnisse kopieren, verschieben und entfernen
- Mehrere Dateien kopieren und Verzeichnisse rekursiv kopieren
- Dateien entfernen und Verzeichnisse rekursiv entfernen
- Einfache und fortgeschrittene Dateinamen-Suchmuster in Kommandos verwenden
- find verwenden, um Dateien auf der Basis ihres Typs, ihrer Größe oder ihrer Zeiten zu finden und zu bearbeiten
- `tar`, `cpio` und `dd` verwenden

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `cp`
- `find`
- `mkdir`
- `mv`
- `ls`
- `rm`
- `rmdir`
- `touch`
- `tar`
- `cpio`
- `dd`
- `file`
- `gzip`
- `gunzip`
- `bzip2`
- Dateisuchmuster

103.4 Ströme, Pipes und Umleitungen verwenden

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, Ströme umzuleiten und zu verbinden, um Textdaten effizient zu verarbeiten. Zu diesen Aufgaben gehören das Umleiten der Standardeingabe, Standardausgabe und Standardfehlerausgabe, das Weiterleiten der Ausgabe eines Kommandos an die Eingabe eines anderen Kommandos, die Verwendung der Ausgabe eines Kommandos als Argumente für ein anderes Kommando und das Senden der Ausgabe sowohl an die Standardausgabe als auch eine Datei.

Wichtigste Wissensgebiete

- Umleiten der Standardeingabe, Standardausgabe und Standardfehlerausgabe
- Weiterleiten der Ausgabe eines Kommandos an die Eingabe eines anderen Kommandos (Pipe)
- Verwenden der Ausgabe eines Kommandos als Argumente für ein anderes Kommando
- Senden der Ausgabe sowohl an die Standardausgabe als auch eine Datei

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `tee`

103.5 Prozesse erzeugen, überwachen und beenden

Gewicht 4

Beschreibung Kandidaten sollten einfache Prozessverwaltung beherrschen.

Wichtigste Wissensgebiete

- Jobs im Vordergrund und Hintergrund ablaufen lassen
- Einem Programm signalisieren, dass es nach dem Abmelden weiterlaufen soll
- Aktive Prozesse beobachten
- Prozesse zur Ausgabe auswählen und sortieren
- Signale an Prozesse schicken

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `&`
- `bg`
- `fg`
- `jobs`
- `kill`
- `nohup`
- `ps`
- `top`
- `free`
- `uptime`
- `killall`

103.6 Prozess-Ausführungsprioritäten ändern

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, die Ausführungsprioritäten von Prozessen zu verwalten.

Wichtigste Wissensgebiete

- Die Standardpriorität eines neu erzeugten Jobs kennen
- Ein Programm mit einer höheren oder niedrigeren Priorität als im Normalfall laufen lassen
- Die Priorität eines laufenden Prozesses ändern

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `nice`
- `ps`
- `renice`
- `top`

103.7 Textdateien mit regulären Ausdrücken durchsuchen

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, Dateien und Textdaten mit regulären Ausdrücken zu manipulieren. Dieses Prüfungsziel umfasst etwa die Erstellung einfacher regulärer Ausdrücke, die mehrere Beschreibungselemente enthalten. Es umfasst ebenfalls den Einsatz von Werkzeugen, die reguläre Ausdrücke zum Durchsuchen eines Dateisystems oder von Dateiinhalten verwenden.

Wichtigste Wissensgebiete

- Einfache reguläre Ausdrücke mit mehreren Beschreibungselementen aufstellen
- Werkzeuge verwenden, die mit regulären Ausdrücken Dateisysteme oder Dateiinhalte durchsuchen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `grep`
- `egrep`
- `fgrep`
- `sed`
- `regex`(7)

103.8 Grundlegendes Editieren von Dateien mit dem vi

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, Textdateien mit dem `vi` zu editieren. Dieses Prüfungsziel umfasst `vi`-Navigation, grundlegende `vi`-Modi, Einfügen, Ändern, Löschen, Kopieren und Finden von Text.

Wichtigste Wissensgebiete

- Mit `vi` in einem Dokument navigieren
- Grundlegende `vi`-Modi verwenden
- Text einfügen, ändern, löschen, kopieren und finden

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `vi`
- `/`, `?`
- `h`, `j`, `k`, `l`
- `l`, `o`, `a`
- `c`, `d`, `p`, `y`, `dd`, `yy`
- `ZZ`, `:w!`, `:q!`, `:e!`

104.1 Partitionen und Dateisysteme anlegen

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, Plattenpartitionen zu konfigurieren und dann Dateisysteme auf Medien wie Festplatten anzulegen. Dies umfasst auch den Umgang mit Swap-Partitionen.

Wichtigste Wissensgebiete

- Verschiedene mkfs-Kommandos verwenden, um Partitionen zu installieren und verschiedene Dateisysteme anzulegen wie
  - ext2/ext3/ext4
  - xfs
  - reiserfs v3
  - vfat

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `fdisk`
- `mkfs`
- `mkswap`

104.2 Die Integrität von Dateisystemen sichern

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, ein Standarddateisystem und die zusätzlichen Daten eines Journaling-Dateisystems zu verwalten.

Wichtigste Wissensgebiete

- Die Integrität von Dateisystemen überprüfen
- Freien Platz und verfügbare Inodes überwachen
- Einfache Probleme von Dateisystemen reparieren

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `du`
- `df`
- `fsck`
- `e2fsck`
- `mke2fs`
- `debugfs`
- `dumpe2fs`
- `tune2fs`
- xfs-Werkzeuge (etwa `xfs_metadump` und `xfs_info`)

104.3 Das Ein- und Aushängen von Dateisystemen steuern

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, das Einhängen eines Dateisystems zu konfigurieren.

Wichtigste Wissensgebiete

- Dateisysteme manuell ein- und aushängen
- Das Einhängen von Dateisystemen beim Systemstart konfigurieren
- Von Benutzern einhängbare Wechseldateisysteme konfigurieren

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/fstab`
- `/media`
- `mount`
- `umount`

104.4 Platten-Quotas verwalten

Gewicht 1

Beschreibung Kandidaten sollten in der Lage sein, Platten-Quotas für Benutzer zu verwalten.

Wichtigste Wissensgebiete

- Platten-Quotas für ein Dateisystem in Kraft setzen
- Benutzer-Quota-Berichte anpassen, prüfen und erzeugen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `quota`
- `edquota`
- `repquota`
- `quotaon`

104.5 Dateizugriffsrechte und -eigentümerschaft verwalten

Gewicht 3

Beschreibung Kandidaten sollten in der Lage sein, Dateizugriffe durch angemessenen Einsatz von Rechten und Eigentümerschaft zu steuern.

Wichtigste Wissensgebiete

- Zugriffsrechte für reguläre und besondere Dateien sowie Verzeichnisse verwalten
- Zugriffsmodi wie SUID, SGID und das Sticky Bit verwenden, um die Sicherheit aufrechtzuerhalten
- Wissen, wie man die umask ändert
- Das Gruppen-Feld verwenden, um Gruppenmitgliedern Dateizugriff einzuräumen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `chmod`
- `umask`
- `chown`
- `chgrp`

104.6 Harte und symbolische Links anlegen und ändern

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, harte und symbolische Links auf eine Datei anzulegen und zu verwalten.

Wichtigste Wissensgebiete

- Links anlegen
- Harte und/oder symbolische Links identifizieren
- Dateien kopieren vs. verlinken
- Links verwenden, um Systemadministrationsaufgaben zu unterstützen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `ln`

104.7 Systemdateien finden und Dateien am richtigen Ort platzieren

Gewicht 2

Beschreibung Kandidaten sollten mit dem Filesystem Hierarchy Standard (FHS) vertraut sein und typische Dateiorte und Verzeichnisklassifizierungen kennen.

Wichtigste Wissensgebiete

- Die korrekten Orte von Dateien unter dem FHS kennen
- Dateien und Kommandos auf einem Linux-System finden
- Den Ort und den Zweck wichtiger Dateien und Verzeichnisse gemäß dem FHS kennen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `find`
- `locate`
- `updatedb`
- `whereis`
- `which`
- `type`
- `/etc/updatedb.conf`

107.1 Benutzer- und Gruppenkonten und dazugehörige Systemdateien verwalten

Gewicht 5
Beschreibung Kandidaten sollten in der Lage sein, Benutzerkonten hinzuzufügen, zu entfernen, zu suspendieren und zu verändern.
Wichtigste Wissensgebiete

- Benutzer und Gruppen hinzufügen, ändern und entfernen
- Benutzer- und Gruppeninformationen in `password/group`-Datenbanken verwalten
- Konten für spezielle Zwecke und beschränkte Konten anlegen und verwalten

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/passwd`
- `/etc/shadow`
- `/etc/group`
- `/etc/skel`
- `chage`
- `groupadd`
- `groupdel`
- `groupmod`
- `passwd`
- `useradd`
- `userdel`
- `usermod`

107.2 Systemadministrationsaufgaben durch Einplanen von Jobs automatisieren

Gewicht 4
Beschreibung Kandidaten sollten in der Lage sein, `cron` oder `anacron` zu verwenden, um Jobs in regelmäßigen Abständen auszuführen, und `at`, um Jobs zu einem bestimmten Zeitpunkt auszuführen.
Wichtigste Wissensgebiete

- Cron- und At-Jobs verwalten
- Benutzerzugang zu den Diensten `cron` und `at` konfigurieren

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/cron.{d,daily,hourly,monthly,weekly}`
- `/etc/at.deny`
- `/etc/at.allow`
- `/etc/crontab`
- `/etc/cron.allow`
- `/etc/cron.deny`
- `/var/spool/cron/*`
- `crontab`
- `at`
- `atq`
- `atrm`

108.2 Systemprotokollierung

Gewicht 2
Beschreibung Kandidaten sollten in der Lage sein, den Syslog-Daemon zu konfigurieren. Dieses Lernziel umfasst auch die Konfiguration des Syslog-Daemons für den Versand von Ausgabe an einen zentralen Protokollserver oder das Annehmen von Ausgabe als zentraler Protokollserver.
Wichtigste Wissensgebiete

- Syslog-Konfigurationsdateien
- `syslog`

- Standard-Facilities, -Prioritäten und -Aktionen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `syslog.conf`
- `syslogd`
- `klogd`
- `logger`

108.4 Drucker und Druckvorgänge verwalten

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, Druckerwarteschlangen und Druckaufträge von Benutzern mit CUPS und der LPD-Kompatibilitätsschnittstelle zu verwalten.

Wichtigste Wissensgebiete

- Grundlegende CUPS-Konfiguration (für lokale und entfernte Drucker)
- Benutzer-Druckerwarteschlangen verwalten
- Allgemeine Druckprobleme lösen
- Druckaufträge zu konfigurierten Druckerwarteschlangen hinzufügen und daraus löschen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- CUPS-Konfigurationsdateien, -Werkzeuge und -Hilfsprogramme
- `/etc/cups`
- LPD-Kompatibilitätsschnittstelle (`lpr`, `lprm`, `lpq`)

109.1 Grundlagen von Internet-Protokollen

Gewicht 4

Beschreibung Kandidaten sollten ein angemessenes Verständnis der Grundlagen von TCP/IP-Netzen demonstrieren.

Wichtigste Wissensgebiete

- Verständnis von Netzmasken demonstrieren
- Wissen über die Unterschiede zwischen privaten und öffentlichen IP-Adressen als »dotted quads«
- Eine Default-Route einstellen
- Wissen über gängige TCP- und UDP-Ports (20, 21, 22, 23, 25, 53, 80, 110, 119, 139, 143, 161, 443, 465, 993, 995)
- Wissen über die Unterschiede und wesentlichen Eigenschaften von UDP, TCP und ICMP
- Wissen über die wesentlichen Unterschiede zwischen IPv4 und IPv6
- Wissen über die grundlegenden Eigenschaften von IPv6

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/services`
- `ftp`
- `telnet`
- `host`
- `ping`
- `dig`
- `traceroute`
- `tracepath`

109.2 Grundlegende Netz-Konfiguration

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, Konfigurationseinstellungen auf Client-Rechnern anzuschauen, zu verändern und zu überprüfen.

Wichtigste Wissensgebiete

- Netzschnittstellen manuell und automatisch konfigurieren
- Grundlegende TCP/IP-Rechnerkonfiguration

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/hostname`
- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`
- `ifconfig`
- `ifup`
- `ifdown`
- `route`
- `ping`

109.3 Grundlegende Netz-Fehlersuche

Gewicht 4

Beschreibung Kandidaten sollten in der Lage sein, Netzprobleme auf Client-Rechnern zu lösen.

Wichtigste Wissensgebiete

- Netzschnittstellen und Routingtabellen manuell und automatisch konfigurieren; dies umfasst das Hinzufügen, Starten, Stoppen, neu Starten, Löschen oder Umkonfigurieren von Netzschnittstellen
- Die Routingtabelle ändern, anschauen oder konfigurieren und eine falsch gesetzte Default-Route manuell richtig stellen.
- Probleme mit der Netzkonfiguration finden und lösen.

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `ifconfig`
- `ifup`
- `ifdown`
- `route`
- `host`
- `hostname`
- `dig`
- `netstat`
- `ping`
- `traceroute`

109.4 Clientseitiges DNS konfigurieren

Gewicht 2

Beschreibung Kandidaten sollten in der Lage sein, DNS auf einem Client-Rechner einzurichten.

Wichtigste Wissensgebiete

- Den Gebrauch von DNS auf dem lokalen System demonstrieren
- Die Reihenfolge der Namensauflösung ändern

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/hosts`
- `/etc/resolv.conf`
- `/etc/nsswitch.conf`

110.1 Administrationsaufgaben für Sicherheit durchführen

Gewicht 3

Beschreibung Kandidaten sollten wissen, wie sie die Systemkonfiguration prüfen, um die Sicherheit des Rechners in Übereinstimmung mit örtlichen Sicherheitsrichtlinien zu gewährleisten.

Wichtigste Wissensgebiete

- Ein System nach Dateien mit gesetztem SUID/SGID-Bit durchsuchen
- Benutzerkennwörter und die Informationen über das Alter von Kennwörtern setzen oder ändern

- Mit nmap und netstat offene Ports auf einem System finden können
- Grenzen für Benutzeranmeldungen, Prozesse und Speicherverbrauch setzen
- Grundlegende Konfiguration und Gebrauch von sudo

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `find`
- `passwd`
- `lsof`
- `nmap`
- `chage`
- `netstat`
- `sudo`
- `/etc/sudoers`
- `su`
- `usermod`
- `ulimit`

110.2 Einen Rechner absichern

Gewicht 3

Beschreibung Kandidaten sollten wissen, wie sie grundlegende Rechnersicherheit konfigurieren können.

Wichtigste Wissensgebiete

- Kenntnisse über Shadow-Kennwörter und wie sie funktionieren
- Nicht verwendete Netzdienste abschalten
- Die Rolle der TCP-Wrapper verstehen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/etc/nologin`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/xinetd.d/*`
- `/etc/xinetd.conf`
- `/etc/inet.d/*`
- `/etc/inetd.conf`
- `/etc/inittab`
- `/etc/init.d/*`
- `/etc/hosts.allow`
- `/etc/hosts.deny`

110.3 Daten durch Verschlüsselung schützen

Gewicht 3

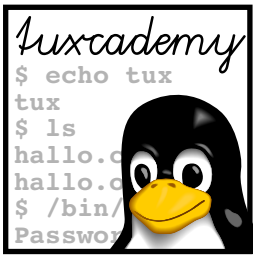
Beschreibung Der Kandidat sollte in der Lage sein, Public-Key-Techniken zum Schutz von Daten und Kommunikation einzusetzen.

Wichtigste Wissensgebiete

- Einen OpenSSH-2-Client grundlegend konfigurieren und verwenden
- Die Rolle von OpenSSH-2-Rechnerschlüsseln verstehen
- GnuPG grundlegend konfigurieren und verwenden, inklusive Schlüsselrückruf
- SSH-Port-Tunnel (auch X11-Tunnel) verstehen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `ssh`
- `ssh-keygen`
- `ssh-agent`
- `ssh-add`
- `~/.ssh/id_rsa` und `id_rsa.pub`
- `~/.ssh/id_dsa` und `id_dsa.pub`
- `/etc/ssh/ssh_host_rsa_key` und `ssh_host_rsa_key.pub`
- `/etc/ssh/ssh_host_dsa_key` und `ssh_host_dsa_key.pub`
- `~/.ssh/authorized_keys`
- `/etc/ssh_known_hosts`
- `gpg`
- `~/.gnupg/*`



D

Kommando-Index

Dieser Anhang fasst alle im Text erklärten Kommandos zusammen und verweist auf deren Dokumentation sowie die Stellen im Text, wo die Kommandos eingeführt werden.

. Liest eine Datei mit Shell-Kommandos so ein, als ob sie auf der Kommandozeile eingegeben worden wäre bash(1) 139

adduser Komfortables Kommando zum Anlegen neuer Benutzerkonten (Debian) adduser(8) 188

alien Konvertiert verschiedene Software-Paketformate alien(1) 453

anacron Führt periodische Jobs aus, wenn der Computer nicht immer läuft anacron(8) 328

apropos Zeigt alle Handbuchseiten mit dem angegebenen Stichwort im „NAME“-Abschnitt apropos(1) 52

apt-get Komfortable Oberfläche für Debian-GNU/Linux-Paketverwaltung apt-get(8) 445

aptitude Komfortables Werkzeug zur Paketinstallation und -wartung (Debian) aptitude(8) 449

arp Erlaubt Zugriff auf den ARP-Cache (Abbildung von IP- auf MAC-Adressen) arp(8) 373

at Registriert Kommandos zur zeitversetzten Ausführung at(1) 322

atd Daemon für die zeitversetzte Ausführung von Kommandos über at atd(8) 324

atq Programm zur Abfrage der Warteschlangen für zeitversetzte Kommandoausführung atq(1) 324

atrm Storniert zeitversetzt auszuführende Kommandos atrm(1) 324

bash Die „Bourne-Again-Shell“, ein interaktiver Kommandointerpreter bash(1) 43

bg Lässt einen (angehaltenen) Prozess im Hintergrund weiterlaufen bash(1) 145

blkid Findet Attribute von blockorientierten Geräten und gibt sie aus blkid(8) 266

cat Hängt Dateien aneinander cat(1) 103

cd Wechselt das aktuelle Arbeitsverzeichnis der Shell bash(1) 71

cfdisk Plattenpartitionierungsprogramm mit textbildschirmorientierter Oberfläche cfdisk(8) 238

chage Setzt Kennwort-Attribute wie Ablaufdatum und Änderungsfristen chage(1) 189

chattr Setzt Dateiattribute für ext2- und ext3-Dateisysteme chattr(1) 206

chfn Erlaubt Benutzern das Ändern des GECOS-Felds in der Benutzerdatenbank chfn(1) 180

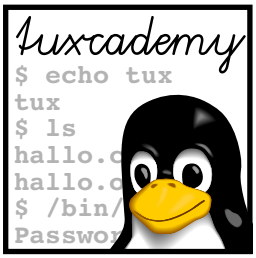
chgrp Setzt Gruppenzugehörigkeit von Dateien und Verzeichnissen chgrp(1) 199
chkconfig Schaltet Systemdienste ein oder aus (SUSE, Red Hat) chkconfig(8) 293
chmod Setzt Rechte für Dateien und Verzeichnisse chmod(1) 197
chown Setzt Eigentümer und Gruppenzugehörigkeit für Dateien und Verzeichnisse chown(1) 198
convmv Konvertiert Dateinamen zwischen Zeichenkodierungen convmv(1) 69
cp Kopiert Dateien cp(1) 78
cpio Dateiarchiv-Verwaltungsprogramm cpio(1) 461
crontab Programm zur Verwaltung von regelmäßig auszuführenden Kommandos crontab(1) 328
csh Die „C-Shell“, ein interaktiver Kommandointerpreter csh(1) 43
cut Extrahiert Felder oder Spalten aus seiner Eingabe cut(1) 122
date Gibt Datum und Uhrzeit aus date(1) 130, 46
dd „*Copy and convert*“, kopiert Dateien und Dateisysteme blockweise und macht einfache Konvertierungen dd(1) 268
debugfs Dateisystem-Debugger zur Reparatur schlimm verkorkster Dateisysteme. Nur für Gurus! debugfs(8) 255
dig Sucht Informationen im DNS (sehr komfortabel) dig(1) 416
dmesg Gibt den Inhalt des Kernel-Nachrichtenpuffers aus dmesg(8) 156, 282, 339
dnsmasq Ein einfacher DHCP- und cachender DNS-Server für kleine Installationen dnsmasq(8) 402
dpkg Verwaltungswerkzeug für Debian-GNU/Linux-Pakete dpkg(8) 438
dpkg-reconfigure Rekonfiguriert ein bereits installiertes Debian-Paket dpkg-reconfigure(8) 452
dumpe2fs Gibt interne Strukturen des ext2-Dateisystems aus. Nur für Gurus! dumpe2fs(8) 255
dumpreiserfs Gibt interne Strukturen des Reiser-Dateisystems aus. Nur für Gurus! dumpreiserfs(8) 258
e2fsck Prüft ext2- und ext3-Dateisysteme auf Konsistenz e2fsck(8) 253
e2label Ändert das Label eines ext2/3-Dateisystem e2label(8) 266
echo Gibt alle seine Parameter durch Leerzeichen getrennt auf der Standardausgabe aus bash(1), echo(1) 46
ed Primitiver zeilenorientierter Editor ed(1) 59
elvis Populärer „Klon“ des vi-Editors elvis(1) 58
env Gibt die Prozessumgebung aus oder startet Programme mit veränderter Umgebung env(1) 132
ex Leistungsfähiger zeilenorientierter Editor (eigentlich vi) vi(1) 59
exit Beendet eine Shell bash(1) 37
expand Ersetzt Tabulatorzeichen in der Eingabe durch äquivalente Leerzeichen expand(1) 111
export Definiert und verwaltet Umgebungsvariable bash(1) 131
fdisk Partitionierungswerkzeug für Platten und ähnliche Medien fdisk(8) 233
fg Holt einen Hintergrundprozess zurück in den Vordergrund bash(1) 145
file Rät den Typ einer Datei anhand des Inhalts file(1) 148
find Sucht nach Dateien, die bestimmte Kriterien erfüllen find(1), Info: find 86
fmt Umbricht die Zeilen der Eingabe auf eine bestimmte Breite fmt(1) 112
free Zeigt die Speicherauslastung und die Auslastung des Swap-Bereichs an free(1) 155
fsck Organisiert Dateisystemkonsistenzprüfungen fsck(8) 247
gdisk Partitionierungswerkzeug für GPT-Platten gdisk(8) 238
getent Ruft Einträge aus administrativen Datenbanken ab getent(1) 185, 417

glxgears Zeigt sich drehende Zahnräder unter X11, mit OpenGL glxgears(1) 145
gpasswd Erlaubt Verwaltung von Gruppenmitgliederlisten durch Gruppenadministratoren und das Setzen des Gruppenkennworts gpasswd(1) 192
groff Programm zur druckreifen Aufbereitung von Texten groff(1) 52
groupadd Fügt Gruppen in die Gruppendatenbank ein groupadd(8) 192
groupdel Löscht Gruppen aus der Gruppendatenbank groupdel(8) 192
groupmod Ändert Gruppeneinträge in der Gruppendatenbank groupmod(8) 192
groups Gibt die Gruppen aus, in denen ein Benutzer Mitglied ist groups(1) 176
grub-md5-crypt Bestimmt MD5-verschlüsselte Kennwörter für GRUB Legacy grub-md5-crypt(8) 279
halt Fährt das System herunter halt(8) 298
hash Zeigt und verwaltet „gesehene“ Kommandos in der bash bash(1) 134
hd Abkürzung für hexdump hexdump(1) 107
head Zeigt den Anfang einer Datei an head(1) 105
help Zeigt Hilfe für bash-Kommandos bash(1) 46, 50
hexdump Gibt Dateiinhalte in hexadezimaler (oktaler, …) Form aus hexdump(1) 107
history Zeigt die zuletzt verwendeten bash-Kommandos an bash(1) 136
host Sucht Informationen im DNS host(1) 415
id Gibt UID und GIDs eines Benutzers aus id(1) 176, 38
ifconfig Konfiguriert Netzwerk-Schnittstellen ifconfig(8) 391
ifdown Schaltet eine Netzwerk-Schnittstelle aus (Debian) ifdown(8) 396
ifup Schaltet eine Netzwerk-Schnittstelle ein (Debian) ifup(8) 396
inetd Internet-Superserver, überwacht Ports und startet ggf. Dienste inetd(8) 377
info Zeigt GNU-Info-Seiten auf einem Textterminal an info(1) 54
initctl Zentrales Steuerungsprogramm für Upstart initctl(8) 297
insserv Aktiviert oder deaktiviert Init-Skripte (SUSE) insserv(8) 293
ip Verwaltet Netzwerkschnittstellen und Routing ip(8) 395
ipv6calc Hilfsprogramm für Berechnungen mit IPv6-Adressen ipv6calc(8) 386
jobs Berichtet über Hintergrundprozesse bash(1) 145
join Führt die Zeilen zweier Dateien „relational“ zusammen join(1) 125
kdesu Startet unter KDE ein Programm als anderer Benutzer KDE: help:/kdesu 37
kill Hält einen Hintergrundprozess an bash(1), kill(1) 145, 214
killall Schickt allen Prozessen mit passendem Namen ein Signal killall(1) 215
klogd Akzeptiert Protokollnachrichten des Systemkerns klogd(8) 156, 334, 338
konsole Ein „Terminalemulator“ für KDE KDE: help:/konsole 44
kpartx Erzeugt Blockgeräte aus Partitionstabellen kpartx(8) 239
ksh Die „Korn-Shell“, ein interaktiver Kommandointerpreter ksh(1) 43
last Zeige zuletzt angemeldete Benutzer an last(1) 176
less Zeigt Texte (etwa Handbuchseiten) seitenweise an less(1) 52, 86
ln Stellt („harte“ oder symbolische) Links her ln(1) 81
locate Sucht Dateien über ihren Namen in einer Dateinamensdatenbank locate(1) 90
logger Macht Einträge ins Systemprotokoll logger(1) 337
logout Beendet eine Sitzung („Abmelden“) bash(1) 36
logrotate Verwaltet, kürzt und „rotiert“ Protokolldateien logrotate(8) 347
logsurfer Programm zum Durchsuchen des Systemprotokolls nach wichtigen Ereignissen www.cert.dfn.de/eng/logsurf/ 338
losetup Erzeugt und verwaltet Loop-Devices losetup(8) 239
ls Listet Dateien oder den Inhalt von Verzeichnissen auf ls(1) 72

lsattr Zeigt Dateiattribute auf ext2- und ext3-Dateisystemen an lsattr(1) 206
lsblk Listet verfügbare Blockgeräte auf lsblk(8) 267
man Zeigt Handbuchseiten des Systems an man(1) 50
manpath Bestimmt den Suchpfad für Handbuchseiten manpath(1) 52
mesg Schaltet wall-Nachrichten für ein Terminal ein oder aus mesg(1) 299
mkdir Legt neue Verzeichnisse an mkdir(1) 74
mkdosfs Legt FAT-formatierte Dateisysteme an mkfs.vfat(8) 262
mke2fs Legt ein ext2- oder ext3-Dateisystem an mke2fs(8) 252
mkfifo Legt FIFOs (benannte Pipes) an mkfifo(1) 149
mkfs Organisiert das Anlegen von Dateisystemen mkfs(8) 246
mkfs.vfat Legt FAT-formatierte Dateisysteme an mkfs.vfat(8) 262
mkfs.xfs Legt XFS-formatierte Dateisysteme an mkfs.xfs(8) 259
mknod Legt Gerätedateien an mknod(1) 149
mkreiserfs Legt Reiser-Dateisysteme an mkreiserfs(8) 258
mkswap Initialisiert eine Auslagerungs-Partition oder -Datei mkswap(8) 263
more Zeigt Textdaten seitenweise an more(1) 86
mount Hängt ein Dateisystem in den Dateibaum ein mount(8), mount(2) 264
mv Verschiebt Dateien in andere Verzeichnisse oder benennt sie um mv(1) 80
netcat Sehr allgemeines Netzwerk-Client-Programm nc(8) 419
nice Startet Programme mit einem anderen *nice*-Wert nice(1) 217
nl Numeriert die Zeilen der Eingabe nl(1) 114
nmap Netzwerk-Portscanner, analysiert offene Ports auf Rechnern nmap(1) 414
nohup Startet ein Programm so, dass es immun gegen SIGHUP-Signale ist nohup(1) 219
od Zeigt die Bytes einer Datei in dezimaler, oktaler, hexadezimaler, ... Darstellung od(1) 106
parted Leistungsfähiges Partitionierungswerkzeug aus dem GNU-Projekt Info: parted 236
paste Fügt verschiedene Eingabedateien zeilenweise aneinander paste(1) 124
pgrep Sucht Prozesse anhand ihres Namens oder anderer Kriterien pgrep(1) 216
ping Prüft grundlegende Netzwerk-Konnektivität mit ICMP ping(8) 407
ping6 Prüft grundlegende Netzwerk-Konnektivität (für IPv6) ping(8) 408
pkill Signalisiert Prozessen anhand ihres Namens oder anderer Kriterien pkill(1) 217
pr Bereitet seine Eingabe zum Drucken auf – mit Kopfzeilen, Fußzeilen usw. pr(1) 113
ps Gibt Prozess-Statusinformationen aus ps(1) 212
pstree Gibt den Prozessbaum aus pstree(1) 213
pwd Gibt den Namen des aktuellen Arbeitsverzeichnisses aus pwd(1), bash(1) 72
reboot Startet den Rechner neu reboot(8) 298
reiserfsck Prüft ein Reiser-Dateisystem auf Konsistenz reiserfsck(8) 258
renice Ändert den *nice*-Wert laufender Prozesse renice(8) 218
reset Setzt den Bildschirmzeichensatz auf einen „vernünftigen“ Wert tset(1) 103
resize_reiserfs Ändert die Größe eines Reiser-Dateisystems resize_reiserfs(8) 258
rm Löscht Dateien oder Verzeichnisse rm(1) 80
rmdir Entfernt (leere) Verzeichnisse rmdir(1) 74
route Verwaltet die statische Routing-Tabelle im Linux-Kern route(8) 392
rpm Dient zur Paketverwaltung in vielen Linux-Systemen (Red Hat, SUSE, …) rpm(8) 456
rpm2cpio Wandelt RPM-Pakete in cpio-Archive um rpm2cpio(1) 461

runlevel Zeigt den vorigen und den aktuellen Runlevel an runlevel(8) 292
scp Sicheres Dateikopierprogramm auf SSH-Basis scp(1) 427
sed Datenstromorientierter Texteditor, kopiert Eingabe unter Änderungen auf Ausgabe sed(1) 59
set Verwaltet Shellvariable bash(1) 132
setfacl Erlaubt die Manipulation von ACLs setfacl(1) 201
sfdisk Nichtinteraktives Partitionierungsprogramm sfdisk(8) 238
sftp Sicheres FTP-artiges Programm auf SSH-Basis sftp(1) 428
sgdisk Nichtinteraktives Partitionierungsprogramm für GPT-Platten sgdisk(8) 239
sh Die „Bourne-Shell“, ein interaktiver Kommandointerpreter sh(1) 43
shutdown Fährt das System herunter oder startet es neu, mit Verzögerung und Warnung an angemeldete Benutzer shutdown(8) 297
slocate Sucht Dateien über ihren Namen in einer Datenbank und beachtet dabei deren Zugriffsrechte slocate(1) 92
sort Sortiert die Zeilen seiner Eingabe sort(1) 117
source Liest eine Datei mit Shell-Kommandos so ein, als ob sie auf der Kommandozeile eingegeben worden wäre bash(1) 139
ssh „Secure Shell“, erlaubt sichere interaktive Sitzungen auf anderen Rechnern ssh(1) 424
ssh-add Akkreditiert private Schlüssel beim ssh-agent ssh-add(1) 430
ssh-agent Verwaltet private Schlüssel und Kennwörter für die SSH ssh-agent(1) 430
ssh-copy-id Kopiert öffentliche SSH-Schlüssel auf andere Rechner ssh-copy-id(1) 430
ssh-keygen Generiert und verwaltet Schlüssel für die SSH ssh-keygen(1) 429
sshd Server für das SSH-Protokoll (sicherer interaktiver Fernzugriff) sshd(8) 424
star POSIX-kompatibles Archivprogramm mit ACL-Unterstützung star(1) 201
su Startet eine Shell unter der Identität eines anderen Benutzers su(1) 167, 37
sudo Erlaubt normalen Benutzern das Aufrufen bestimmter Kommandos mit Administratorprivilegien sudo(8) 165, 37
swapoff Deaktiviert eine Auslagerungs-Partition oder -Datei swapoff(8) 263
swapon Aktiviert eine Auslagerungs-Partition oder -Datei swapon(8) 263
syslog-ng Bearbeitet Systemprotokoll-Meldungen (neuer und besser) syslog-ng(8) 343
syslogd Bearbeitet Systemprotokoll-Meldungen syslogd(8) 156, 334
systemctl Zentrales Steuerungsprogramm für systemd systemctl(1) 305, 315
tac Zeigt eine Datei von hinten nach vorne an tac(1) 104
tail Zeigt das Ende einer Datei an tail(1) 105, 338
tcpdump Netzwerk-Sniffer, protokolliert und analysiert Netzwerkverkehr tcpdump(1) 419
tcsh Die „Tenex-C-Shell“, ein interaktiver Kommandointerpreter tcsh(1) 43
tee Kopiert die Standardeingabe in die Standardausgabe und außerdem in Dateien tee(1) 101
telnet Erlaubt Verbindungen zu beliebigen TCP-Diensten, insbesondere TELNET (Fernzugriff) telnet(1) 418
test Wertet logische Ausdrücke auf der Kommandozeile aus test(1), bash(1) 141
top Bildschirmorientiertes Programm zur Beobachtung und Verwaltung von Prozessen top(1) 219
tr Tauscht Zeichen in der Standardeingabe gegen andere aus oder löscht sie tr(1) 109
tracepath Prüft die Wegleitung zu einer anderen Station, mit Pfad-MTU tracepath(8) 411

tracepath6 Entspricht tracepath, aber für IPv6 tracepath(8) 412
traceroute Prüft die Wegleitung zu einer anderen Station traceroute(8) 409
tune2fs Stellt Parameter von ext2- und ext3-Dateisystemen ein tunefs(8) 256, 267
type Bestimmt die Art eines Kommandos (intern, extern, Alias) bash(1) 46
ulimit Legt Ressourcenbegrenzungen für Prozesse fest bash(1) 218
umask Stellt die *umask* (Rechtekontrolle für neue Dateien) ein bash(1) 200
unexpand „Optimiert“ Tabulator- und Leerzeichen in der Eingabe unexpand(1) 111
uniq Ersetzt Folgen von gleichen Zeilen in der Eingabe durch die erste solche uniq(1) 121
unset Löscht Shell- oder Umgebungsvariable bash(1) 132
update-rc.d Installiert und entfernt System-V-Initskript-Links (Debian) update-rc.d(8) 293
updatedb Erstellt die Dateinamensdatenbank für locate updatedb(1) 91
uptime Gibt die Zeit seit dem letzten Systemstart sowie die CPU-Auslastung aus uptime(1) 154
useradd Fügt neue Benutzerkonten hinzu useradd(8) 187
userdel Entfernt Benutzerkonten userdel(8) 190
usermod Modifiziert die Benutzerdatenbank usermod(8) 190
vi Bildschirmorientierter Texteditor vi(1) 58
vigr Erlaubt das exklusive Ändern von /etc/group bzw. /etc/gshadow vipw(8) 193
vim Populärer „Klon“ des vi-Editors vim(1) 58
vol_id Bestimmt Dateisystemtypen und gibt Label und UUID aus vol_id(8) 266
wall Schreibt eine Nachricht auf die Terminals aller angemeldeten Benutzer wall(1) 299
wc Zählt Zeilen, Wörter und Zeichen in seiner Eingabe wc(1) 116
whatis Sucht Handbuchseiten mit dem gegebenen Stichwort in der Beschreibung whatis(1) 53
whereis Sucht ausführbare Programme, Handbuchseiten und Quellcode zu gegebenen Kommandos whereis(1) 134
which Sucht Programme in PATH which(1) 134
wireshark Paket-Sniffer, liest und analysiert Netzwerkverkehr (Ex-ethereal) wireshark(1) 420
xargs Konstruiert Kommandozeilen aus seiner Standardeingabe xargs(1), Info: find 89
xclock Zeigt eine Uhr an xclock(1x) 145
xconsole Zeigt Systemmeldungen in einem X-Fenster an xconsole(1) 334
xfs_info Entspricht xfs_growfs, aber ändert das Dateisystem nicht xfs_growfs(8) 260
xfs_mdrestore Überträgt einen Metadaten-Abzug in ein XFS-Dateisystem xfs_mdrestore(8) 259
xfs_metadump Erzeugt Metadaten-Abzüge von XFS-Dateisystemen xfs_metadump(8) 259
xfs_repair „Repariert“ XFS-Dateisysteme xfs_repair(8) 259
xinetd Verbesserter Internet-Superserver, überwacht Ports und startet ggf. Dienste xinetd(8) 377
xlogmaster X11-basiertes Systembeobachtungsprogramm xlogmaster(1), www.gnu.org/software/xlogmaster/ 338
xterm Ein „Terminalemulator“ für das X-Window-System xterm(1) 44
yum Komfortables Werkzeug zur Verwaltung von RPM-Paketen yum(8) 461
yumdownloader Lädt Pakete aus YUM-Paketquellen (ohne Installation) yumdownloader(8) 466



Index

Dieser Index verweist auf die wichtigsten Stichwörter in der Schulungsunterlage. Besonders wichtige Stellen für die einzelnen Stichwörter sind durch **fette** Seitenzahlen gekennzeichnet. Sortiert wird nur nach den Buchstaben im Indexeintrag; „`~/.bashrc`“ wird also unter „B“ eingeordnet.

., 71
., 139
.., 71
/, 70, 158, 231
_ (Umgebungsvariable), 323

`adduser`, 188
Administrationswerkzeuge, 164
`alias`, 46, 470
`alien`, 437, 453–454
    `--to-deb` (Option), 454
`anacron`, 321, 328–330, 485
    `-s` (Option), 330
    `-u` (Option), 485
`apropos`, 52, 54
`apt`, 438
`apt-cache`, 437, 447–449
`apt-get`, 437, 440, 445–447, 449–450, 461–462
    `dist-upgrade` (Option), 446–447
    `install` (Option), 446
    `remove` (Option), 447
    `source` (Option), 447
    `upgrade` (Option), 447
`apt-key`, 452
`aptitude`, 437–439, 449–450
`ar`, 439, 454
Arbeitsmodi, **60**
`arp`, 373
`at`, 311, 321–325, 327
    `-c` (Option), 324
    `-f` (Option), 323
    `-q` (Option), 324
AT&T, 16
ATA, 222
`atd`, 324–325
    `-b` (Option), 324
    `-d` (Option), 324
    `-l` (Option), 324
`atq`, 324
    `-q` (Option), 324
`atrm`, 324
`awk`, 108, 124

`bash`, 43, 47, 56, 72, 138–139, 146, 430, 511
    `-c` (Option), 138
`~/.bash_history`, 135
`batch`, 323–325
Bell Laboratories, 16
Benutzerdatenbank, 178, 181
    anderswo gelagert, 181
Benutzerkonten, **174**
Benutzername, **175**
Berkeley, 16
Bernstein, Daniel J., 426
`bg`, 145, 210
`/bin`, 45, 151, 153
`/bin/ls`, 134
`/bin/sh`, 180, 459, 480
`/bin/sh`, 326
`/bin/true`, 180
`blkid`, 266–267
blockorientierte Geräte, **152**
`/boot`, 150–151, 279
`/boot/grub`, 278
`/boot/grub/custom.cfg`, 279
`/boot/grub/grub.cfg`, 279
`/boot/grub/menu.lst`, 276
Bootmanager, **272**
Bootsektor, **272**
Bootskript, 291
Bottomley, James, 274
Bourne, Stephen L., 42
Broadcast-Adresse, **379**
BSD, 16
BSD-Lizenz, 20
`btrfs`, 261
`btrfs check`
    `--repair` (Option), 261

C, **16**
Canonical Ltd., 29

Card, Rémy, 249
cat, 99, 103–104, 106, 148, 185, 482
cc, 268
cd, 45, 71–72, 93, 196, 470
cfdisk, 238
chage, 189
chattr, 206, 482
  -R (Option), 206
chfn, 180
chgrp, 192, 199, 204
  -R (Option), 199
chkconfig, 293, 484
chmod, 87, 138, 165, 197–198, 200–201, 203–204, 206
  -R (Option), 198
  --reference=⟨*Name*⟩ (Option), 198
chown, 191, 198–199
  -R (Option), 199
chsh, 180
comm, 480
convmv, 69
cp, 78–81, 83–85, 264, 428, 478
  -a (Option), 85
  -i (Option), 79
  -L (Option), 85
  -l (Option), 83, 85
  -P (Option), 85
  -s (Option), 85
cpio, 275, 277, 461, 466, 512
cron, 91, 294, 311, 321–322, 325–330, 348, 429, 432, 485
crontab, 52, 135, 325, 327–328, 477, 485
  -e (Option), 328
  -l (Option), 328, 485
  -r (Option), 328, 485
  -u (Option), 328
csh, 43
cut, 122–124, 477, 480
  -c (Option), 122–123
  -d (Option), 123
  -f (Option), 123
  --output-delimiter (Option), 123
  -s (Option), 124

Datagramme, **371**
date, 46, 130–131, 485
Dateiattribute, **205**
dd, 152, 218, 233, 239, 257, 260, 264, 268–269, 283
DEBCONF_FRONTEND (Umgebungsvariable), 452
DEBCONF_PRIORITY (Umgebungsvariable), 453
Debian Free Software Guidelines, 21
Debian-Projekt, **28**
debsums, 444–445, 450
debugfs, 255
  -w (Option), 255
Definitionen, **14**
demand paging, 205
/dev, 151, 390, 478
/dev/block, 230
/dev/fd0, 149
/dev/klog, 361
/dev/log, 334, 344, 354
/dev/mapper, 240
/dev/null, 152, 157, 319, 478
/dev/random, 109, 152, 475
/dev/scd0, 253
/dev/sda, 233
/dev/tty, 97
/dev/ttyS0, 166
/dev/urandom, 152
/dev/xconsole, 336
/dev/zero, 107, 152, 253
diff, 441
dig, 415–417
  -x (Option), 416
Dijkstra, Edsger, 281
dirs, 72
DISPLAY (Umgebungsvariable), 323, 431–432
dmesg, 156, 282, 339
  -c (Option), 339
  -n (Option), 339
dnsmasq, 402
domain (/etc/resolv.conf), 401
dpkg, 437–440, 443, 445, 448–450
  -a (Option), 439
  --configure (Option), 439
  --force-depends (Option), 439
  --force-overwrite (Option), 440
  -i (Option), 439
  --install (Option), 439
  -L (Option), 444
  -l (Option), 442
  --list (Option), 442
  --listfiles (Option), 444
  -P (Option), 440
  --purge (Option), 449
  -r (Option), 440
  -s (Option), 442, 444
  --search (Option), 444
  --status (Option), 442–443, 448
  --unpack (Option), 439
dpkg-reconfigure, 452–453
  -f (Option), 452
  --frontend (Option), 452
  -p (Option), 453
  --priority (Option), 453
dpkg-source, 441
dselect, 445, 449
dump, 205
dumpe2fs, 255
dumpreiserfs, 258

e2fsck, 253–255, 258
  -B (Option), 254

-b (Option), 254–255
-c (Option), 254
-f (Option), 254
-l (Option), 254
-p (Option), 254
-v (Option), 254
e2label, 266
e4defrag, 256
echo, 46, 75, 104–105, 130, 394, 470, 475
-n (Option), 130
ed, 59
EDITOR (Umgebungsvariable), 191, 328
egrep, 216
Einbenutzermodus, **294**
elvis, 58
env, 132
/etc, 152–153, 168, 264
/etc/anacrontab, 329
/etc/apt/apt.conf, 447
/etc/apt/sources.list, 446
/etc/apt/trusted.gpg, 452
/etc/at.allow, 325
/etc/at.deny, 325
/etc/at.deny, 325
/etc/cron.allow, 327, 485
/etc/cron.d, 327
/etc/cron.daily, 91, 327
/etc/cron.deny, 327, 485
/etc/cron.hourly, 327
/etc/crontab, 327–328
/etc/dpkg/dpkg.cfg, 439
/etc/filesystems, 265–266
/etc/fstab, 152–153, 158–160, 229, 233, 248, 256–257, 263, 265, 268–269, 291, 305–306, 483
/etc/group, 177, 179, 184–186, 190–193
/etc/grub.d, 279
/etc/grub.d/40_custom, 279
/etc/grub.inst, 278
/etc/gshadow, 184–185, 191–193, 514
/etc/hosts, 152, 402, 417
/etc/inetd.conf, 306
/etc/init, 295
/etc/init.d, 152
/etc/init.d/network, 397
/etc/init.d/networking, 396
/etc/inittab, 152, 288, 290–292, 298, 306, 308, 312, 314
/etc/issue, 153
/etc/logrotate.conf, 348–349
/etc/logrotate.d, 348, 487
/etc/machine-id, 361
/etc/magic, 148
/etc/modules.conf, 390
/etc/motd, 153
/etc/mtab, 153, 268, 483
/etc/network/interfaces, 396, 399, 412
/etc/network/options, 394
/etc/nologin, 298
/etc/nsswitch.conf, 185, 403, 417
/etc/passwd, 101, 127, 153, 160, 177–181, 184–186, 188, 190–191, 326–327, 418, 428, 481
/etc/protocols, 487
/etc/rc.d/init.d, 153
/etc/resolv.conf, 401
domain, 401
nameserver, 401
options, 402
search, 401
sortlist, 401
/etc/rpmrc, 456
/etc/rsyslog.conf, 334, 340, 342
/etc/securetty, 166
/etc/services, 377, 418, 487, 489
/etc/shadow, 92, 153, 178, 181–186, 189–191, 194, 202, 428, 474, 480–481
/etc/shells, 43, 180
/etc/skel, 187
/etc/ssh, 425
/etc/ssh/ssh_config, 431
/etc/ssh/sshd_config, 430–431
/etc/sysconfig, 169, 396–397
/etc/sysconfig/locate, 91
/etc/sysconfig/network, 396
/etc/sysconfig/network-scripts, 397–398
/etc/sysconfig/network-scripts/ifcfg-eth0, 397
/etc/sysconfig/network/config, 397
/etc/sysconfig/network/routes, 397
/etc/sysconfig/static-routes, 398
/etc/sysconfig/sysctl, 394
/etc/sysctl.conf, 395, 401
/etc/syslog-ng/syslog-ng.conf, 343
/etc/syslog.conf, 334, 337, 340, 346, 486
/etc/systemd/journald.conf, 355–357
/etc/udev/rules.d, 390
/etc/udev/rules.d/70-persistent-net.rules, 406
/etc/updatedb.conf, 91
/etc/yum.conf, 462
/etc/yum.repos.d, 462
ethereal, 420–421, 514
Ethernet, 364
EULA, 19
ex, 59, 62, 64–65
exit, 37, 43, 45, 138
expand, 111
-i (Option), 111
-t (Option), 111
export, 131–132
-n (Option), 132

FAQ, **55**
fdisk, 233–238
-l (Option), 235

-u (Option), 235
Festplatten
SCSI, 223
Festplattencache, 247
fg, 145, 210, 419
fgrep, 135
FHS, **149**
file, 148
Filterkommandos, **102**
find, 86–90, 474
-exec (Option), 89
-maxdepth (Option), 473
-name (Option), 474
-ok (Option), 89
-print (Option), 87, 89–90
-print0 (Option), 90
finger, 180
Flags, **375**
fmt, 112–114
-c (Option), 113
-w (Option), 112
Fox, Brian, 43
Fragmentierung, **371**
free, 155
Free Software Foundation, 18
Freeware, 20
fsck, 247–248, 253, 255–256, 259, 284
-A (Option), 248
-a (Option), 248
-f (Option), 248
-N (Option), 248
-p (Option), 248
-R (Option), 248
-s (Option), 248
-t (Option), 248, 259
-V (Option), 248
-v (Option), 248
fsck.ext2, 253
fsck.xfs, 259
FSF, 18

Garrett, Matthew, 274
gated, 392–393
gcc, 68
gdisk, 238, 269, 283
gedit, 65
Gemeinfreiheit, 19
Gerätenummer, **152**
Gerhards, Rainer, 339
getent, 185, 417, 480
getfacl, 201
getmail_fetch, 433
getty, 312
glxgears, 145
GNOME, 65, 452
GNU, **18**
gpasswd, 192
-A (Option), 192
-a (Option), 192
-d (Option), 192
GPL, **18**
grep, 51, 96, 99, 103, 122, 151, 157, 185, 213, 216, 478, 487
-H (Option), 478
groff, 52, 54, 58
groupadd, 192
-g (Option), 192
groupdel, 192
groupmod, 190, 192
-g (Option), 192
-n (Option), 192
groups, 176
GRUB, **272**
Bootprobleme, 283
grub, 277
--device-map (Option), 278
lock (Option), 484
password (Option), 279
grub-install, 278
grub-md5-crypt, 279
grub-mkconfig, 279
grub.cfg, 279
Gruppe, **175**
administrative, 184
Administrator, 192
Kennwort, 184–185, 192
Gruppen, **165**
gzip, 350, 454
-6 (Option), 350

Hakim, Pascal, 328
halt, 298
hash, 134
-r (Option), 134
hd, 107
head, 105–106
-c (Option), 105
-n (Option), 105
-*n* (Option), 105
Heimatverzeichnis, **175**
hello, 438, 441
help, 46, 50, 134, 136
hexdump, 107–108, 127, 511
Hintergrund-Prozess, **144**
history, 136
-c (Option), 136
HOME (Umgebungsvariable), 327
/home, 44, 71, 85, 156–157, 180–181, 231
Homme, Kjetil Torgrim, 216
host, 415–417
-a (Option), 416
-l (Option), 416
-t (Option), 416
HOWTOs, **54**

-i, 472
IANA, 377
id, 38, 176, 179, 205, 479

-G (Option), 176
-g (Option), 176
-Gn (Option), 176
-n (Option), 176
-u (Option), 176
id_ed25519, 430
id_ed25519.pub, 430
id_rsa, 429
id_rsa.pub, 429–430
ifconfig, 391, 393, 395, 401, 406–407, 488
-a (Option), 406
ifdown, 396–398, 412
-a (Option), 396
ifup, 396–398, 412
-a (Option), 396
inetd, 377
info, 54
inhalt, 100
init, 152, 247, 280, 282, 290, 292, 484
Init-Skripte, 292, 298, 456
Parameter, 292
initctl, 297
initctl start, 297
initctl status, 297
initctl stop, 297
Inode-Nummern, **81**
insserv, 293–294, 484
ip, 395–396, 401, 406, 409
addr (Option), 395
addr add (Option), 395
brd + (Option), 395
help (Option), 395
link (Option), 395
local (Option), 395
route (Option), 395
ipv6calc, 386–387
ISOLINUX, 272

jobs, 145, 210
Johnson, Jeff, 456
join, 125
journalctl, 356–362
-b (Option), 359
-f (Option), 358
-k (Option), 359
--list-boots (Option), 359
-n (Option), 358
--no-pager (Option), 356
--output=verbose (Option), 360
-p (Option), 358–359
--since (Option), 359
-u (Option), 358–359
--until (Option), 359
journald, 359, 362
Journaling, 249
Joy, Bill, 59

kate, 65
KDE, 65, 452
kdesu, 37
Kennwörter, 175, 178, 181
ändern, 188
GRUB, 279
Gruppen-, 184–185, 192
vergeben, 188
Kernelmodule, **151**
kill, 145, 214–217, 294
killall, 215–217
-i (Option), 215
-l (Option), 215
-w (Option), 215
Kindprozess, **143**
klogd, 156, 334, 338–339, 343, 345
Knoppix, 29
Kok, Auke, 295
Kommandosubstitution, **98**
konsole, 44, 180
Korn, David, 42
kpartx, 239–240, 243
-v (Option), 240
Krafft, Martin F., 28
ksh, 43

Label, **266**
LANG (Umgebungsvariable), 117, 476
last, 176–177
LC_ALL (Umgebungsvariable), 117
LC_COLLATE (Umgebungsvariable), 117, 477
less, 52, 86, 98, 101, 185, 338, 356
/lib, 151
/lib/modules, 151
Linux Documentation Project, **54**
linux-0.01.tar.gz, 469
ln, 81–84, 149
-b (Option), 84
-f (Option), 84
-i (Option), 84
-s (Option), 84, 149
-v (Option), 84
locate, 90–93, 474, 514
-e (Option), 91
logger, 324, 337, 343, 347, 357
login, 167, 180, 298
LOGNAME (Umgebungsvariable), 326, 474
logout, 36
logrotate, 348–351, 357
-f (Option), 348
--force (Option), 348
logsurfer, 338
Lokale Netze, **367**
losetup, 239–240
-a (Option), 239
-f (Option), 239
lost+found, 157, 255
lpr, 113
ls, 53–54, 72–74, 76, 78, 81, 84, 98, 100, 102, 117, 122, 134, 151, 178,

191, 196–197, 206, 470–471, 474
-a (Option), 72
-d (Option), 73, 470
-F (Option), 72
-H (Option), 84–85
-i (Option), 81
-L (Option), 84–85
-l (Option), 72–73, 84, 178, 197, 206
-p (Option), 72
-U (Option), 100
lsattr, 206, 482
-a (Option), 206
-d (Option), 206
-R (Option), 206
LSB, 438
lsblk, 267
lsmod, 406, 488
lspci, 406
-k (Option), 406

mail, 180
MAILTO (Umgebungsvariable), 327
man, 50, 52–53, 77, 86, 155
-a (Option), 52
-f (Option), 53
-k (Option), 52
MANPATH (Umgebungsvariable), 52
manpath, 52
Maskierung, **46**
Mason, Chris, 246
Master Boot Record, **272**
Matilainen, Panu, 456
/media, 156
/media/cdrom, 156
/media/dvd, 156
/media/floppy, 156
mesg, 299
Minix, 16, 249
mkdir, 74, 148, 151
-p (Option), 74
mkdosfs, 262
mke2fs, 246, 252–253, 256
-F (Option), 253
mkfifo, 149
mkfs, 246–248, 252–253, 261–262, 273
-t (Option), 246, 252–253, 262
mkfs.btrfs
-d (Option), 261
-L (Option), 267
mkfs.vfat, 262
mkfs.xfs, 259–260
-l (Option), 259
mknod, 149
mkreiserfs, 258
mkswap, 263–264, 267
/mnt, 156, 253
more, 86
-l (Option), 86
-n ⟨*Zahl*⟩ (Option), 86
-s (Option), 86
Morton, Andrew, 23
mount, 135, 151, 233, 256, 264–266
-t (Option), 265
Multics, 16
Murdock, Ian, 28
mv, 80–82, 264, 478
-b (Option), 80
-f (Option), 80
-i (Option), 80
-R (Option), 81, 471
-u (Option), 80
-v (Option), 80

nameserver (/etc/resolv.conf), 401
NAT, 383
nc, 419
netcat, 419
netstat, 412–414
-l (Option), 413
-t (Option), 413
-u (Option), 413
Netzklassen, **381**
Netzmaske, **379**
Netzwerkadresse, **379**
newgrp, 184
nice, 217, 324
nl, 114–116
-b (Option), 115
-i (Option), 115
-n (Option), 115
-v (Option), 115
-w (Option), 115
nmap, 412, 414–415, 420
-A (Option), 415
nohup, 219
nohup.out, 219
Novell, 28
NSA, 165, 425
nslookup, 415

od, 106–107, 109, 476
-A (Option), 475
-N (Option), 107, 475
-t (Option), 106–107, 475
-v (Option), 107
Open Source, 18
OpenSSH, **424**
/opt, 153, 158, 231
options (/etc/resolv.conf), 402

Packages.gz, 451–452
PAGER (Umgebungsvariable), 356
parted, 236–238
passwd, 178, 188–192, 202, 480
-l (Option), 189
-S (Option), 189
-u (Option), 189

passwd -n, 189
passwd -w, 189
passwd -x, 189
paste, 124–125
    -d (Option), 124
    -s (Option), 124–125
PATH (Umgebungsvariable), 71, 133–134, 139, 146, 477–478, 514
PDP-11, 16
perl, 124
pgrep, 216–217
    -a (Option), 216
    -d (Option), 216
    -f (Option), 216
    -G (Option), 216
    -l (Option), 216
    -n (Option), 216
    -o (Option), 216
    -P (Option), 217
    -t (Option), 217
    -u (Option), 217
ping, 372, 407–410, 488
    -a (Option), 408
    -b (Option), 408
    -c (Option), 408
    -f (Option), 408
    -I (Option), 408
    -i (Option), 408
    -n (Option), 408
    -s (Option), 408
ping6, 408–409
Pipeline, **101**
Pipes, **101**
pkill, 216–217, 318
    --signal (Option), 217
Poettering, Lennart, 288, 304
popd, 72
Portnummern, **375**
Ports, 376
Portscanner, **414**
pppd, 397
pr, 113–114
präemptives Multitasking, **211**
primäre Gruppe, **179**
printf, 108
Priorität, **217**
/proc, 154–155, 158, 210–212, 482
/proc/cpuinfo, 154
/proc/devices, 154
/proc/dma, 154
/proc/filesystems, 265
/proc/interrupts, 154
/proc/ioports, 154
/proc/kcore, 154, 479
/proc/kmsg, 338–339
/proc/loadavg, 154
/proc/meminfo, 155
/proc/mounts, 155
/proc/scsi, 155
/proc/swaps, 263–264
/proc/sys/kernel/pid_max, 482
Prozesszustand, **211**
Prüfungsziele, **495**
ps, 155, 202, 212–214, 216
    a (Option), 212–213
    ax (Option), 213
    -C (Option), 213
    --forest (Option), 212, 214
    --help (Option), 212
    -l (Option), 212–213
    -o (Option), 213
    p (Option), 216
    r (Option), 213
    T (Option), 213
    U (Option), 213
    -u (Option), 202
    x (Option), 213
Pseudobenutzer, 177
Pseudogeräte, **152**
pstree, 213–214
    -G (Option), 214
    -p (Option), 214
    -u (Option), 214
Public-Domain-Software, 19
Puffer, **59**
pushd, 72
pwconv, 186
pwd, 72, 93
Python, 452

Qt, 21
quota, 294

Ramey, Chet, 43
rc, 293
rcnetwork, 397
reboot, 298
Red Hat, 22
Referenzzähler, **81**
registered ports, **377**
Reiser, Hans, 257–258
reiserfsck, 258
Release, 451
Release.gpg, 451
Remnant, Scott James, 288, 295
renice, 218
reset, 103
resize_reiserfs, 258
Ritchie, Dennis, 16, 203
rm, 46, 80–82, 84, 89, 196, 338, 470–472
    -f (Option), 81
    -i (Option), 80–81, 472
    -r (Option), 81
    -v (Option), 81
rmdir, 74–75, 471
    -p (Option), 75
rmmod, 488

/root, 150, 156–157
route, 392, 394, 396–397, 409
    -host (Option), 394
    -net (Option), 394
routed, 392
Routing, **374**
rpm, 438, 454, 456–458, 461, 465
    -a (Option), 458
    -c (Option), 459
    -d (Option), 459
    -e (Option), 457
    -f (Option), 458
    -h (Option), 457
    -i (Option), 457, 459
    -l (Option), 459–460
    --nodeps (Option), 457–458
    -p (Option), 458
    --provides (Option), 460
    -q (Option), 458
    -qi (Option), 465
    --requires (Option), 459
    --test (Option), 457–458
    -V (Option), 460
    -v (Option), 456, 459
    -vv (Option), 456
    --whatprovides (Option), 459
    --whatrequires (Option), 460
rpm2cpio, 461
~/.rpmrc, 456
Rückgabewert, **137**, **143**, **211**
/run/log/journal, 355
Runlevel, **288**, 298
    Bedeutung, 291
    konfigurieren, 293
    wechseln, 292
runlevel, 292, 318, 484

/sbin, 151, 153
/sbin/init, 288
Schattenkennwörter, 178, 181
Scheidler, Balazs, 343
scp, 427–428, 430
    -r (Option), 428
search (/etc/resolv.conf), 401
sed, 59
SELinux, 165
set, 132
setfacl, 201
sfdisk, 238–239, 269, 283
sftp, 428, 430
sgdisk, 239
sh, 43
shadow passwords, **178**, 181
SHELL (Umgebungsvariable), 326
Shellskript, **138**
Shellskripte, 126
Shellvariable, **131**
shutdown, 165, 290, 297–299, 315
    -c (Option), 484
    -h (Option), 298
    -r (Option), 298
Shuttleworth, Mark, 29–30
Sievers, Kay, 288, 304
Signale, **214**
SkoleLinux, 29
sleep, 217, 433, 478
slocate, 92, 474
Snowden, Edward, 425
sort, 102, 117–119, 121–122, 128, 135, 140, 156, 375, 477, 480
    -b (Option), 119–120
    -f (Option), 477
    -k (Option), 117
    -n (Option), 121
    -r (Option), 120
    -t (Option), 120
    -u (Option), 477, 522
-u, 122
sortlist (/etc/resolv.conf), 401
source, 139
/srv, 156, 232
~/.ssh, 429
ssh, 177, 375, 418, 424–428, 430–434, 488–489
    -f (Option), 433
    -KR (Option), 433
    -L (Option), 432–433
    -N (Option), 432
    -R (Option), 432–433
    -X (Option), 431
ssh-add, 430–431
    -D (Option), 430
ssh-agent, 430–431
ssh-copy-id, 430
ssh-keygen, 425, 429–431
    -f (Option), 425
    -l (Option), 425
    -t ed25519 (Option), 430
~/.ssh/authorized_keys, 430
~/.ssh/config, 426, 431
~/.ssh/known_hosts, 426–427, 488
~/.ssh/ssh_config, 427
sshd, 216, 418, 424, 431
Stallman, Richard M., 18
Standardkanäle, **96**
star, 201
sticky bit, **204**
su, 37, 39, 167–168, 177, 323, 328, 337, 479, 481
Subnetting, **382**
sudo, 37, 165, 167, 479
Superblock, **246**
Superuser, **164**
SUSE, 22
SuSEconfig, 169, 396
Swap-Partition, **263**
swapoff, 263

swapon, 263–264
Symbolische Links, **83**
/sys, 155
/sys/bus/scsi/devices, 230
/sys/class/scsi_host, 230
sysctl, 400
syslog, 294, 484
Syslog-NG, 343
syslog-ng, 343
syslog.conf, 337
syslogd, 155–156, 282, 294, 324, 326, 334, 336–341, 343–344, 346–348, 357, 486
-r (Option), 336, 344, 486
systemctl, 305, 314–320, 354–355, 485
--full (Option), 317
--kill-who (Option), 316
-l (Option), 317, 355
--lines (Option), 317
-n (Option), 317
--now (Option), 319
-s (Option), 316
--signal (Option), 316
-t (Option), 315, 317
systemd, 317
systemd-escape, 310
-p (Option), 310
-u (Option), 310
systemd-journald, 355–357
SYSTEMD_LESS (Umgebungsvariable), 356
SYSTEMD_PAGER (Umgebungsvariable), 356
Systemlast, 323

tac, 104, 106, 476
-b (Option), 104
-s (Option), 104
tail, 105–106, 338, 358, 475
-c (Option), 105
-f (Option), 105, 338, 358
-n (Option), 105
-*n* (Option), 105
Tanenbaum, Andrew S., 16
tar, 201, 275, 439, 454, 456, 479
tcpdump, 419–420, 434
tcsh, 43
tee, 101–102, 475
-a (Option), 101
teilnehmer0.dat, 122
telinit, 290, 292, 294
q (Option), 290
telnet, 418–419
TERM (Umgebungsvariable), 86, 323
Terminierung, **225**
test, 46, 141, 470
-f (Option), 478
Thawte, 30
Thompson, Ken, 16
/tmp, 156, 158, 191, 204, 232, 481
top, 219
Torvalds, Linus, 16, 20, 22–23
touch, 191
tr, 109–112
-c (Option), 110, 476
-s (Option), 111, 476
tracepath, 409, 411–412
tracepath6, 412
traceroute, 409–412, 487
-6 (Option), 411
-I (Option), 410
-M tcp (Option), 410
-p (Option), 410
-T (Option), 410
traceroute6, 411–412
Treibernummer, **152**
Ts'o, Theodore, 251
tune2fs, 254, 256–257, 267, 483
-c (Option), 483
-L (Option), 267
-l (Option), 254
-m (Option), 483
-u (Option), 483
Tweedie, Stephen, 249
type, 46, 134
TZ (Umgebungsvariable), 130
Tzur, Itai, 328

Ubuntu, 29
UID, **175**
ulimit, 218
-a (Option), 218
-d (Option), 218
-f (Option), 218
-n (Option), 218
-t (Option), 218
-u (Option), 218
-v (Option), 218
umask, 200, 205
-S (Option), 200
Umgebungsvariable, **131**
_, 323
DEBCONF_FRONTEND, 452
DEBCONF_PRIORITY, 453
DISPLAY, 323, 431–432
EDITOR, 191, 328
HOME, 327
LANG, 117, 476
LC_ALL, 117
LC_COLLATE, 117, 477
LOGNAME, 326, 474
MAILTO, 327
MANPATH, 52
PAGER, 356
PATH, 71, 133–134, 139, 146, 477–478, 514
SHELL, 326
SYSTEMD_LESS, 356
SYSTEMD_PAGER, 356
TERM, 86, 323

TZ, 130
VISUAL, 191, 328
umount, 160, 264
uname, 177
-r (Option), 177
unexpand, 111–112
-a (Option), 111
uniq, 121
Unix, **16**
unset, 132
update-grub, 279
update-rc.d, 293–294
updatedb, 91–92, 474
uptime, 154
Urheberrecht, 19
useradd, 186–188, 190–192, 481
userdel, 190, 192
-r (Option), 190
usermod, 190, 192, 481
/usr, 149, 153–154
/usr/bin, 45, 150, 153
/usr/bin/test, 134
/usr/lib, 153
/usr/lib/rpm, 456
/usr/local, 153, 156, 231, 457
/usr/local/bin, 150
/usr/sbin, 153
/usr/share, 153
/usr/share/doc, 154
/usr/share/file, 148
/usr/share/file/magic, 148
/usr/share/info, 154
/usr/share/man, 52, 154
/usr/share/zoneinfo, 130
/usr/src, 154
UUID, **267**

van de Ven, Arjan, 295
/var, 155–156, 158, 232
/var/lib/dpkg/info, 445
/var/log, 155, 337, 355, 362
/var/log/auth.log, 479
/var/log/journal, 355
/var/log/messages, 167, 282, 356, 479, 486
/var/log/syslog, 282, 479
/var/mail, 84, 155, 190
/var/spool, 158
/var/spool/atjobs, 324
/var/spool/atspool, 324
/var/spool/cron, 155
/var/spool/cron/allow, 327
/var/spool/cron/crontabs, 325, 327
/var/spool/cron/deny, 327, 485
/var/spool/cups, 155
/var/tmp, 156, 158
verbindungsloses Protokoll, **371**
Verisign, 30
vi, 58–60, 62–64, 66, 83, 169, 191, 328
vigr, 191, 193
-s (Option), 193
vim, 58, 65
vimtutor, 470
vipw, 191, 193, 481
-s (Option), 191
VISUAL (Umgebungsvariable), 191, 328
vmlinuz, 151
vol_id, 266
Volkerding, Patrick, 27
von Münchhausen, Karl Friedrich Hieronymus, 272

w, 238
wall, 299–301, 512
-n (Option), 300
--nobanner (Option), 300
wc, 99, 116–117, 128, 476, 478
wc -l, 140
Webmin, 169
Weitverkehrsnetze, **367**
well-known ports, **377**
whatis, 53
whereis, 134, 477
which, 134, 477
wireshark, 420, 434
write, 300
Wurzelverzeichnis, **150**

Xandros, 29
xargs, 89–90
-0 (Option), 90
-r (Option), 89
.Xauthority, 432
xclock, 145, 212
-update 1 (Option), 145
xconsole, 334
xfs_copy, 260
xfs_growfs, 270, 514
xfs_info, 260
xfs_mdrestore, 259
xfs_metadump, 259
xfs_repair, 259
-n (Option), 259
xfsdump, 260
xfsrestore, 260
xinetd, 377
xlogmaster, 338
xterm, 44, 135, 180

YUM, 461
yum, 461–466
--disablerepo (Option), 462
--enablerepo= (Option), 462
--obsoletes (Option), 463
yumdownloader, 466
--resolve (Option), 466
--source (Option), 466
--urls (Option), 466

zeichenorientierte Geräte, **152**
Zombie, **211**
zsh, 188
Zugriffsmodus, **196**