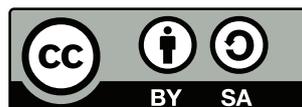
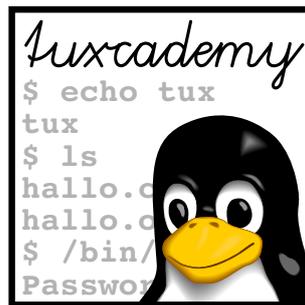




Version 4.0

# Linux-Administration I

## System und Benutzer



tuxcademy – Linux- und Open-Source-Lernunterlagen für alle  
[www.tuxcademy.org](http://www.tuxcademy.org) · [info@tuxcademy.org](mailto:info@tuxcademy.org)



Diese Schulungsunterlage ist inhaltlich und didaktisch auf Inhalte der Zertifizierungsprüfung LPI-101 (LPIC-1, Version 4.0) des Linux Professional Institute abgestimmt. Weitere Details stehen in Anhang B.

Das Linux Professional Institute empfiehlt keine speziellen Prüfungsvorbereitungsmaterialien oder -techniken – wenden Sie sich für Details an [info@lpi.org](mailto: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-Administration I System und Benutzer

Revision: adm1:738c572f2cee74ee:2015-08-08

adm1:9f8c99fa2fddd494:2015-08-08 1–13, B

adm1:HXwQaLzccKsXSawsn2uLLz

© 2015 Linup Front GmbH Darmstadt, Germany

© 2015 tuxcademy (Anselm Lingnau) Darmstadt, Germany

<http://www.tuxcademy.org> · [info@tuxcademy.org](mailto:info@tuxcademy.org)

Linux-Pinguin »Tux« © Larry Ewing (CC-BY-Lizenz)

Alle in dieser Dokumentation enthaltenen Darstellungen und Informationen wurden nach bestem Wissen erstellt und mit Sorgfalt getestet. Trotzdem sind Fehler nicht völlig auszuschließen. Das tuxcademy-Projekt haftet nach den gesetzlichen Bestimmungen bei Schadensersatzansprüchen, die auf Vorsatz oder grober Fahrlässigkeit beruhen, und, außer bei Vorsatz, nur begrenzt auf den vorhersehbaren, typischerweise eintretenden Schaden. Die Haftung wegen schuldhafter Verletzung des Lebens, des Körpers oder der Gesundheit sowie die zwingende Haftung nach dem Produkthaftungsgesetz bleiben unberührt. Eine Haftung über das Vorgenannte hinaus ist ausgeschlossen.

Die Wiedergabe von Warenbezeichnungen, Gebrauchsnamen, Handelsnamen und Ähnlichem in dieser Dokumentation berechtigt auch ohne deren besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne des Warenzeichen- und Markenschutzrechts frei seien und daher beliebig verwendet werden dürften. Alle Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt und sind möglicherweise eingetragene Warenzeichen Dritter.



Diese Dokumentation steht unter der »Creative Commons-BY-SA 4.0 International«-Lizenz. Sie dürfen sie vervielfältigen, verbreiten und öffentlich zugänglich machen, solange die folgenden Bedingungen erfüllt sind:

**Namensnennung** Sie müssen darauf hinweisen, dass es sich bei dieser Dokumentation um ein Produkt des tuxcademy-Projekts handelt.

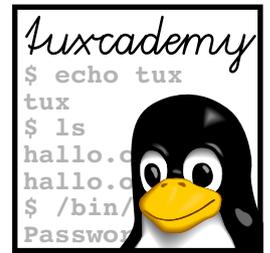
**Weitergabe unter gleichen Bedingungen** Sie dürfen die Dokumentation bearbeiten, abwandeln, erweitern, übersetzen oder in sonstiger Weise verändern oder darauf aufbauen, solange Sie Ihre Beiträge unter derselben Lizenz zur Verfügung stellen wie das Original.

Mehr Informationen und den rechtsverbindlichen Lizenzvertrag finden Sie unter <http://creativecommons.org/licenses/by-sa/4.0/>

Autoren: Thomas Erker, Anselm Lingnau

Technische Redaktion: Anselm Lingnau ([anselm@tuxcademy.org](mailto:anselm@tuxcademy.org))

Gesetzt in Palatino, Optima und DejaVu Sans Mono



# Inhalt

<b>1</b>	<b>Systemadministration</b>	<b>13</b>
1.1	Administration allgemein . . . . .	14
1.2	Das privilegierte root-Konto . . . . .	14
1.3	Administratorprivilegien erlangen. . . . .	16
1.4	Distributionsabhängige Administrationswerkzeuge . . . . .	19
<b>2</b>	<b>Benutzerverwaltung</b>	<b>23</b>
2.1	Grundlagen . . . . .	24
2.1.1	Wozu Benutzer? . . . . .	24
2.1.2	Benutzer und Gruppen. . . . .	25
2.1.3	»Natürliche Personen« und Pseudobenutzer . . . . .	27
2.2	Benutzer- und Gruppendaten . . . . .	28
2.2.1	Die Datei /etc/passwd. . . . .	28
2.2.2	Die Datei /etc/shadow. . . . .	31
2.2.3	Die Datei /etc/group . . . . .	34
2.2.4	Die Datei /etc/gshadow . . . . .	35
2.2.5	Das Kommando getent . . . . .	35
2.3	Benutzerkonten und Gruppeninformationen verwalten . . . . .	36
2.3.1	Benutzerkonten einrichten . . . . .	36
2.3.2	Das Kommando passwd . . . . .	38
2.3.3	Benutzerkonten löschen . . . . .	40
2.3.4	Benutzerkonten und Gruppenzuordnung ändern . . . . .	40
2.3.5	Die Benutzerdatenbank direkt ändern — vipw. . . . .	41
2.3.6	Anlegen, Ändern und Löschen von Gruppen. . . . .	41
<b>3</b>	<b>Zugriffsrechte</b>	<b>45</b>
3.1	Das Linux-Rechtekonzept . . . . .	46
3.2	Zugriffsrechte auf Dateien und Verzeichnisse. . . . .	46
3.2.1	Grundlagen. . . . .	46
3.2.2	Zugriffsrechte anschauen und ändern . . . . .	47
3.2.3	Dateieigentümer und Gruppe setzen – chown und chgrp . . . . .	48
3.2.4	Die <i>umask</i> . . . . .	49
3.3	Zugriffskontrolllisten (ACLs). . . . .	51
3.4	Eigentum an Prozessen. . . . .	52
3.5	Besondere Zugriffsrechte für ausführbare Dateien . . . . .	52
3.6	Besondere Zugriffsrechte für Verzeichnisse . . . . .	53
3.7	Dateiattribute . . . . .	55
<b>4</b>	<b>Prozessverwaltung</b>	<b>59</b>
4.1	Was ist ein Prozess? . . . . .	60
4.2	Prozesszustände . . . . .	61
4.3	Prozessinformationen – ps. . . . .	62
4.4	Prozesse im Baum – pstree. . . . .	63
4.5	Prozesse beeinflussen – kill und killall. . . . .	64
4.6	pgrep und pkill . . . . .	66
4.7	Prozessprioritäten – nice und renice . . . . .	67

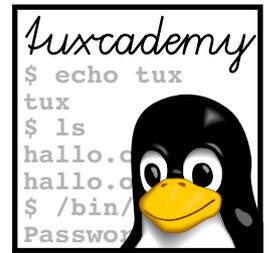
4.8	Weitere Befehle zur Prozessverwaltung – <code>nohup</code> , <code>top</code> . . . . .	69
<b>5</b>	<b>Hardware</b>	<b>71</b>
5.1	Grundlagen . . . . .	72
5.2	Linux und PCI (Express) . . . . .	73
5.3	USB. . . . .	75
5.4	Geräteeinbindung und Treiber . . . . .	78
5.4.1	Überblick. . . . .	78
5.4.2	Geräte und Treiber . . . . .	79
5.4.3	Das Verzeichnis <code>/sys</code> . . . . .	80
5.4.4	<code>udev</code> . . . . .	81
5.4.5	Geräteeinbindung und D-Bus . . . . .	82
<b>6</b>	<b>Platten (und andere Massenspeicher)</b>	<b>85</b>
6.1	Grundlagen . . . . .	86
6.2	Bussysteme für Massenspeicher . . . . .	86
6.3	Partitionierung. . . . .	89
6.3.1	Grundlagen. . . . .	89
6.3.2	Die traditionelle Methode (MBR) . . . . .	90
6.3.3	Die moderne Methode (GPT) . . . . .	91
6.4	Linux und Massenspeicher . . . . .	93
6.5	Platten partitionieren . . . . .	95
6.5.1	Prinzipielles. . . . .	95
6.5.2	Platten partitionieren mit <code>fdisk</code> . . . . .	97
6.5.3	Platten formatieren mit GNU <code>parted</code> . . . . .	100
6.5.4	<code>gdisk</code> . . . . .	102
6.5.5	Andere Partitionierungsprogramme . . . . .	102
6.6	Loop-Devices und <code>kpartx</code> . . . . .	103
6.7	Der Logical Volume Manager (LVM) . . . . .	105
<b>7</b>	<b>Dateisysteme: Aufzucht und Pflege</b>	<b>109</b>
7.1	Linux-Dateisysteme . . . . .	110
7.1.1	Überblick. . . . .	110
7.1.2	Die <code>ext</code> -Dateisysteme . . . . .	113
7.1.3	ReiserFS . . . . .	121
7.1.4	XFS . . . . .	123
7.1.5	Btrfs . . . . .	124
7.1.6	Noch mehr Dateisysteme . . . . .	126
7.1.7	Auslagerungsspeicher ( <i>swap space</i> ) . . . . .	127
7.2	Einbinden von Dateisystemen . . . . .	128
7.2.1	Grundlagen. . . . .	128
7.2.2	Der <code>mount</code> -Befehl . . . . .	128
7.2.3	Labels und UUIDs . . . . .	130
7.3	Das Programm <code>dd</code> . . . . .	132
7.4	Plattenkontingentierung (Quota) . . . . .	133
7.4.1	Überblick. . . . .	133
7.4.2	Kontingentierung für Benutzer ( <code>ext</code> und XFS). . . . .	134
7.4.3	Kontingentierung für Gruppen ( <code>ext</code> und XFS). . . . .	135
<b>8</b>	<b>Linux booten</b>	<b>139</b>
8.1	Grundlagen . . . . .	140
8.2	GRUB Legacy . . . . .	143
8.2.1	Grundlagen von GRUB. . . . .	143
8.2.2	Die Konfiguration von GRUB Legacy. . . . .	144
8.2.3	Installation von GRUB Legacy . . . . .	145
8.3	GRUB 2 . . . . .	146
8.3.1	Sicherheitsaspekte . . . . .	147

---

8.4	Kernelparameter . . . . .	148
8.5	Probleme beim Systemstart . . . . .	150
8.5.1	Fehlersuche . . . . .	150
8.5.2	Typische Probleme . . . . .	150
8.5.3	Rettungssysteme und Live-Distributionen . . . . .	152
<b>9</b>	<b>System-V-Init und der Init-Prozess</b>	<b>155</b>
9.1	Der Init-Prozess . . . . .	156
9.2	System-V-Init . . . . .	156
9.3	Upstart . . . . .	163
9.4	Herunterfahren des Systems . . . . .	165
<b>10</b>	<b>Systemd</b>	<b>171</b>
10.1	Überblick. . . . .	172
10.2	Unit-Dateien . . . . .	174
10.3	Typen von Units . . . . .	178
10.4	Abhängigkeiten . . . . .	179
10.5	Ziele . . . . .	180
10.6	Das Kommando <code>systemctl</code> . . . . .	183
10.7	Installation von Units . . . . .	186
<b>11</b>	<b>Dynamische Bibliotheken</b>	<b>189</b>
11.1	Bibliotheken. . . . .	190
11.2	Dynamische Bibliotheken in der Praxis . . . . .	192
11.3	Dynamische Bibliotheken installieren und finden . . . . .	195
11.4	Dynamische Bibliotheken in mehreren Versionen . . . . .	196
<b>12</b>	<b>Paketverwaltung mit Debian-Werkzeugen</b>	<b>199</b>
12.1	Überblick. . . . .	200
12.2	Das Fundament: <code>dpkg</code> . . . . .	200
12.2.1	Debian-Pakete . . . . .	200
12.2.2	Paketinstallation . . . . .	201
12.2.3	Pakete löschen . . . . .	202
12.2.4	Debian-Pakete und ihr Quellcode . . . . .	203
12.2.5	Informationen über Pakete . . . . .	204
12.2.6	Verifikation von Paketen . . . . .	206
12.3	Debian-Paketverwaltung der nächsten Generation . . . . .	207
12.3.1	APT . . . . .	207
12.3.2	Paketinstallation mit <code>apt-get</code> . . . . .	208
12.3.3	Informationen über Pakete . . . . .	209
12.3.4	<code>aptitude</code> . . . . .	211
12.4	Integrität von Debian-Paketen . . . . .	213
12.5	Die <code>debconf</code> -Infrastruktur . . . . .	214
12.6	<code>alien</code> : Pakete aus fremden Welten . . . . .	215
<b>13</b>	<b>Paketverwaltung mit RPM &amp; Co.</b>	<b>217</b>
13.1	Einleitung . . . . .	218
13.2	Paketverwaltung mit <code>rpm</code> . . . . .	219
13.2.1	Installation und Update . . . . .	219
13.2.2	Deinstallation von Paketen . . . . .	219
13.2.3	Datenbank- und Paketanfragen . . . . .	220
13.2.4	Verifikation von Paketen . . . . .	222
13.2.5	Das Programm <code>rpm2cpio</code> . . . . .	223
13.3	YUM . . . . .	223
13.3.1	Überblick. . . . .	223
13.3.2	Paketquellen . . . . .	223
13.3.3	Pakete installieren und entfernen mit YUM . . . . .	224
13.3.4	Informationen über Pakete . . . . .	226
13.3.5	Pakete nur herunterladen . . . . .	228

---

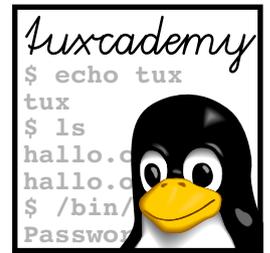
<b>A Musterlösungen</b>	<b>231</b>
<b>B LPIC-1-Zertifizierung</b>	<b>239</b>
B.1 Überblick . . . . .	239
B.2 Prüfung LPI-101 . . . . .	240
B.3 Prüfung LPI-102 . . . . .	240
B.4 LPI-Prüfungsziele in dieser Schulungsunterlage . . . . .	241
<b>C Kommando-Index</b>	<b>249</b>
<b>Index</b>	<b>253</b>



# Tabellenverzeichnis

3.1	Die wichtigsten Dateiattribute . . . . .	56
5.1	USB-Standards . . . . .	76
6.1	Verschiedene SCSI-Varianten . . . . .	88
6.2	Partitionstypen (hexadezimal) für Linux . . . . .	91
6.3	Partitionstyp-GUIDs für GPT (Auswahl) . . . . .	92
10.1	Gängige Ziele für systemd (Auswahl) . . . . .	181
10.2	Kompatibilitäts-Ziele für System-V-Init . . . . .	182

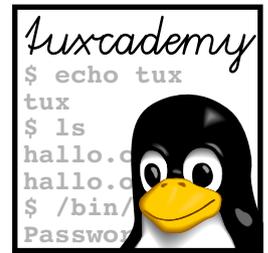




# Abbildungsverzeichnis

4.1	Verdeutlichung der Zusammenhänge zwischen den verschiedenen Prozesszuständen . . . . .	61
5.1	Ausgabe von lspci auf einem typischen x86-PC . . . . .	74
5.2	Das Programm usbview . . . . .	78
7.1	Die Datei /etc/fstab (Beispiel) . . . . .	129
9.1	Eine typische /etc/inittab-Datei (Auszug) . . . . .	157
9.2	Upstart-Konfigurationsdatei für Job rsyslog . . . . .	164
10.1	Eine systemd-Unit-Datei: console-getty.service . . . . .	175
12.1	Das Programm aptitude . . . . .	212





# Vorwort

Dieser Kurs ist eine Einführung in die Administration von Linux. Aufbauend auf Wissen über die Anwendung von Linux vermittelt er die wichtigsten theoretischen und praktischen Kenntnisse für Konfiguration und Betrieb eines freistehenden Linux-Rechners.

Der Kurs wendet sich an Benutzer, die über Kenntnisse in der Anwendung von Linux- oder Unix-Systemen etwa auf dem Niveau des Linup-Front-Kurses *Linux-Grundlagen für Anwender und Administratoren* verfügen und einen konzentrierten, aber umfassenden Einstieg in die Systemadministration suchen. Vorausgesetzt wird die sichere Anwendung der Shell und eines Texteditors sowie Erfahrung mit den gängigen Kommandozeilen-Werkzeugen eines Linux-Systems.

Zu den Themen dieser Unterlage gehören nebst einer Einführung in die Bedeutung und Probleme der Systemadministration die Grundlagen der Prozess-, Benutzer- und Zugriffsrechteverwaltung, Umgang mit Partitionen, Dateisystemen und Kontingierung, die gängigen Bootlader, die Vorgänge beim Systemstart und -stopp, PC-Hardware sowie Bibliotheks- und Paketverwaltung.

Der erfolgreiche Abschluss dieses Kurses oder vergleichbare Kenntnisse sind Voraussetzung für den erfolgreichen Besuch weiterer Linux-Kurse und für eine Zertifizierung beim *Linux Professional Institute*.

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!

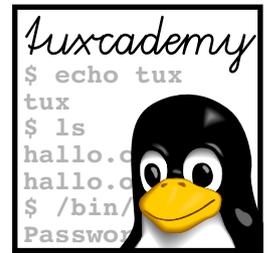


Die meisten Kapitel enthalten auch Übungsaufgaben, die mit dem »Bleistift«-Sinnbild am Absatzanfang gekennzeichnet sind. Die Aufgaben sind nummeriert und Musterlösungen für die wichtigsten befinden sich hinten in dieser Schulungsunterlage. Bei jeder Aufgabe ist in eckigen Klammern der Schwierigkeitsgrad 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





# 1

# Systemadministration

## Inhalt

1.1	Administration allgemein . . . . .	14
1.2	Das privilegierte root-Konto . . . . .	14
1.3	Administratorprivilegien erlangen. . . . .	16
1.4	Distributionsabhängige Administrationswerkzeuge . . . . .	19

## 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

## 1.1 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**

## 1.2 Das privilegierte root-Konto

**Superuser** 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.

**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« Zugriffsrechtssystem, 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 Programmierern 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 »böses« 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



**1.1 [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!



**1.2 [!1]** Warum sollten Sie als normaler Benutzer nicht auf dem root-Konto arbeiten?



**1.3 [1]** Wie sieht es mit der Rechtevergabe bei Ihrem Rechner zu Hause aus? Arbeiten Sie mit Administratorprivilegien?

## 1.3 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*).

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.

Auch auf Einzelplatzsystemen



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
```

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.

Protokoll



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.

**Verantwortlichkeit** 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.



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



**1.4 [2]** Welche Möglichkeiten gibt es, um Administratorrechte zu bekommen? Welche Möglichkeit ist die bessere? Warum?



**1.5 [!2]** Woran lässt sich bei einem normal konfigurierten System während des Betriebs erkennen, ob Sie unter dem root-Konto arbeiten?



**1.6 [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)?



**1.7 [!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!

## 1.4 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 Aspekte 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 kennen und die Linux-Administration von Grund auf neu lernen müßten. (Administrationswerkzeuge aus dritter Quelle wie Webmin haben dieses Problem nicht im selben Maße.)

## Übungen



**1.8** [!2] Stellt Ihre Distribution Ihnen ein Administrationswerkzeug (etwa YaST) zur Verfügung? Was können Sie damit alles machen?



**1.9** [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?

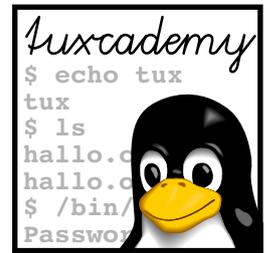


**1.10** [!1] Administrationswerkzeuge wie Webmin sind potentiell für jeden zugänglich, der einen Browser bedienen kann. Welche Vorteile und Nachteile ergeben sich daraus?

## 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.





# 2

## Benutzerverwaltung

### Inhalt

2.1	Grundlagen . . . . .	24
2.1.1	Wozu Benutzer? . . . . .	24
2.1.2	Benutzer und Gruppen. . . . .	25
2.1.3	»Natürliche Personen« und Pseudobenutzer . . . . .	27
2.2	Benutzer- und Gruppendaten . . . . .	28
2.2.1	Die Datei /etc/passwd. . . . .	28
2.2.2	Die Datei /etc/shadow. . . . .	31
2.2.3	Die Datei /etc/group . . . . .	34
2.2.4	Die Datei /etc/gshadow . . . . .	35
2.2.5	Das Kommando getent . . . . .	35
2.3	Benutzerkonten und Gruppeninformationen verwalten . . . . .	36
2.3.1	Benutzerkonten einrichten . . . . .	36
2.3.2	Das Kommando passwd . . . . .	38
2.3.3	Benutzerkonten löschen . . . . .	40
2.3.4	Benutzerkonten und Gruppenzuordnung ändern . . . . .	40
2.3.5	Die Benutzerdatenbank direkt ändern — vipw. . . . .	41
2.3.6	Anlegen, Ändern und Löschen von Gruppen. . . . .	41

### 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

## 2.1 Grundlagen

### 2.1.1 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 Herr 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 Tochtters Tagebuch (ehedem mit Schloss versehen) sollen neugierigen Augen vielleicht nicht ganz ohne weiteres zugänglich sein.



Den Umstand, dass Tochtters 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 Tochtters 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 anzurufen 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<sup>1</sup>.

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.

### 2.1.2 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

<sup>1</sup>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<sup>2</sup> 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
```

<sup>2</sup>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
```

### 2.1.3 »Natürliche Personen« und Pseudobnutzer

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.

Nach der Installation von Linux finden Sie in den Dateien /etc/passwd und /etc/group eine ganze Reihe solcher Pseudobnutzer. 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 Pseudobnutzer 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 Pseudobnutzer sind als Eigentümer von Dateien

Pseudobnutzer

Pseudobnutzer 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



2.1 [1] Wodurch unterscheidet der Betriebssystemkern verschiedene Benutzer und Gruppen?



2.2 [2] Was passiert, wenn eine UID zweimal mit unterschiedlichem Namen vergeben wird? Ist das erlaubt?



2.3 [1] Was versteht man unter Pseudobenzutzern? Nennen Sie Beispiele!



2.4 [2] (Beim zweiten Durcharbeiten.) Ist es akzeptabel, einen Benutzer in die Gruppe disk aufzunehmen, dem Sie nicht das root-Kennwort anvertrauen würden? Warum (nicht)?

## 2.2 Benutzer- und Gruppendaten

### 2.2.1 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 Pseudobenzutzer 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 2.6), 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; *shadow passwords* 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.



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 Benutzererkennung – 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 Pseudobnutzer 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 Erstausrüstung. 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	<i>Hugo aus der Personalabteilung</i>
/home/entw/susi	<i>Susi aus der Entwicklungsabteilung</i>
/home/vorst/heinz	<i>Heinz aus dem Vorstand</i>

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.

### 2.2.2 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 Milliarden ( $6,6 \cdot 10^{15}$ ) 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<sup>3)</sup> 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 verpöhen. 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 Ehegesponnes, 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

<sup>3)</sup>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*.

### 2.2.3 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.

#### 2.2.4 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.

#### 2.2.5 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

 2.5 [1] Welchen Wert finden Sie in der zweiten Spalte der Datei `/etc/passwd`? Warum finden Sie dort diesen Wert?

 2.6 [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?

 2.7 [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.)

## 2.3 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 Pseudob Benutzer. 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

### 2.3.1 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 (Abschnitt 7.4), 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.

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)`.

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.

### 2.3.2 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: »P5« 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 2.2.2.)

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 8.) Anschließend können Sie mit einem Editor das `<Kennwort>`-Feld des `root`-Eintrags in `/etc/passwd` löschen.

## Übungen



**2.8** [3] Ändern Sie das Kennwort von Benutzer `hugo`. Wie ändert sich die Datei `/etc/shadow`? Fragen Sie den Status zu diesem Kennwort ab.



**2.9** [!2] Der Benutzer `trottel` hat sein Kennwort vergessen. Wie können Sie ihm helfen?



**2.10** [!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!

### 2.3.3 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.

### 2.3.4 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 -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.

### 2.3.5 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



**2.11** [!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?



**2.12** [!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.



**2.13** [!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!



**2.14** [2] Ändern Sie die UID des Benutzers `test1`. Was müssen Sie außerdem tun?



**2.15** [2] Ändern Sie das Heimatverzeichnis für Benutzer `test1` um von `/home/test1` in `/home/user/test1`.

### 2.3.6 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 [-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 [-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 <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.

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:

```
gpasswd -a <Benutzer> <Gruppe>
```

fügt den `<Benutzer>` der `<Gruppe>` hinzu, und

```
gpasswd -d <Benutzer> <Gruppe>
```

entfernt ihn wieder daraus. Mit

```
gpasswd -A <Benutzer>,... <Gruppe>
```

Gruppenadministrator

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`.

## Übungen



**2.16** [2] Wozu werden Gruppen gebraucht? Geben Sie mögliche Beispiele an!



**2.17** [1] Können Sie ein Verzeichnis anlegen, auf das alle Mitglieder einer Gruppe Zugriff haben?



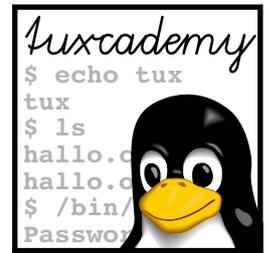
**2.18** [!2] Erstellen Sie eine zusätzliche Gruppe `test`. Mitglied dieser Gruppe soll nur `test1` sein. Setzen Sie ein Gruppenkennwort. Melden Sie sich als `test1` und `test2` an und wechseln Sie jeweils in die neue Gruppe!

## Kommandos in diesem Kapitel

<b>adduser</b>	Komfortables Kommando zum Anlegen neuer Benutzerkonten (Debian)		
		<code>adduser(8)</code>	38
<b>chage</b>	Setzt Kennwort-Attribute wie Ablaufdatum und Änderungsfristen		
		<code>chage(1)</code>	39
<b>chfn</b>	Erlaubt Benutzern das Ändern des GECOS-Felds in der Benutzerdatenbank		
		<code>chfn(1)</code>	30
<b>getent</b>	Ruft Einträge aus administrativen Datenbanken ab	<code>getent(1)</code>	35
<b>gpasswd</b>	Erlaubt Verwaltung von Gruppenmitglieder listen durch Gruppenadministratoren und das Setzen des Gruppenkennworts	<code>gpasswd(1)</code>	42
<b>groupadd</b>	Fügt Gruppen in die Gruppendatenbank ein	<code>groupadd(8)</code>	42
<b>groupdel</b>	Löscht Gruppen aus der Gruppendatenbank	<code>groupdel(8)</code>	42
<b>groupmod</b>	Ändert Gruppeneinträge in der Gruppendatenbank	<code>groupmod(8)</code>	42
<b>groups</b>	Gibt die Gruppen aus, in denen ein Benutzer Mitglied ist	<code>groups(1)</code>	26
<b>id</b>	Gibt UID und GIDs eines Benutzers aus	<code>id(1)</code>	26
<b>last</b>	Zeige zuletzt angemeldete Benutzer an	<code>last(1)</code>	26
<b>useradd</b>	Fügt neue Benutzerkonten hinzu	<code>useradd(8)</code>	37
<b>userdel</b>	Entfernt Benutzerkonten	<code>userdel(8)</code>	40
<b>usermod</b>	Modifiziert die Benutzerdatenbank	<code>usermod(8)</code>	40
<b>vigr</b>	Erlaubt das exklusive Ändern von <code>/etc/group</code> bzw. <code>/etc/gshadow</code>		
		<code>vipw(8)</code>	43

## 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.
- »Pseudob Benutzer« 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.



# 3

## Zugriffsrechte

### Inhalt

3.1	Das Linux-Rechtekonzept . . . . .	46
3.2	Zugriffsrechte auf Dateien und Verzeichnisse. . . . .	46
3.2.1	Grundlagen. . . . .	46
3.2.2	Zugriffsrechte anschauen und ändern . . . . .	47
3.2.3	Dateieigentümer und Gruppe setzen – chown und chgrp . . . . .	48
3.2.4	Die <i>umask</i> . . . . .	49
3.3	Zugriffskontrolllisten (ACLs). . . . .	51
3.4	Eigentum an Prozessen. . . . .	52
3.5	Besondere Zugriffsrechte für ausführbare Dateien . . . . .	52
3.6	Besondere Zugriffsrechte für Verzeichnisse . . . . .	53
3.7	Dateiattribute . . . . .	55

### 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 2)
- Kenntnisse über das Linux-Dateikonzept

## 3.1 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 <i>A</i> nicht ohne weiteres auf Dateien von Benutzer <i>B</i> 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
Zugriffsmodus	Einstellung der Rechte für eine Datei heißt auch deren <b>Zugriffsmodus</b> .
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.

## 3.2 Zugriffsrechte auf Dateien und Verzeichnisse

### 3.2.1 Grundlagen

Zugriffsrechte für Dateien	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. 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 <code>ls</code> -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 <code>cd</code> 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.

### 3.2.2 Zugriffsrechte anschauen und ändern

Informationen darüber, welche Rechte, Benutzer- und Gruppenzuordnung für eine Datei gelten, bekommen Sie mit »`ls -l`«: Informationen

```
$ 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 Rechtestelle 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«.

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: Befehl chmod

```
$ 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  
äquivalent zu ugo+w*

```
$ chmod a+w datei
```



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
```

### 3.2.3 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



**3.1** [!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?



**3.2** [4] Vergleichen Sie die Mechanismen, die verschiedene Dateibrowser (zum Beispiel Konqueror, Nautilus, ...) zum Setzen von Dateirechten, Eigentümer, Gruppe, ... anbieten. Gibt es nennenswerte Unterschiede?

### 3.2.4 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.

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 <i>umask</i> :	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 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.

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



**3.3 [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*?



**3.4 [2]** Vergewissern Sie sich, dass der Unterschied zwischen den Kommandos »chmod +x« und »chmod a+x« wirklich so aussieht wie beschrieben.

## 3.3 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 1003.1X-ACLs (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.

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.

### 3.4 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.

Prozesse haben auch 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 3.5) kann hier ebenfalls eingreifen.

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)
```

### 3.5 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.

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.

SGID-Bit

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.

chmod-Syntax

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.

ls-Ausgabe

## 3.6 Besondere Zugriffsrechte 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.)

SGID für Verzeichnisse

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):

<code># groupadd projekt</code>	<i>Neue Gruppe anlegen</i>
<code># usermod -a -G projekt hugo</code>	<i>hugo in die Gruppe</i>
<code># usermod -a -G projekt susi</code>	<i>susi auch</i>
<code>&lt;&lt;&lt;&lt;&lt;&lt;</code>	

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 3.6).

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 Programmablauf 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



**3.5 [2]** Was bedeutet das spezielle Zugriffsrecht »s«? Wo finden Sie es? Können Sie dieses Recht auf eine selbst erstellte Datei setzen?



**3.6 [!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?



**3.7 [2]** Was bedeutet das spezielle Zugriffsrecht »t«? Wo finden Sie es?



**3.8 [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.



**3.9 [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?

## 3.7 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 3.1. 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 3.1: Die wichtigsten Dateiattribute

Attribut	Bedeutung
A	<i>atime</i> wird nicht aktualisiert (interessant für mobile Rechner)
a	( <i>append-only</i> ) An die Datei kann nur angehängt werden
c	Dateiinhalte werden transparent komprimiert (nicht implementiert)
d	Datei wird von <i>dump</i> nicht gesichert
i	( <i>immutable</i> ) 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



**3.10** [!2] Vergewissern Sie sich, dass die **a**- und **i**-Optionen funktionieren wie behauptet.



**3.11** [2] Können Sie für eine gegebene Datei *alle* Striche in der **lsattr**-Ausgabe zum Verschwinden bringen?

## Kommandos in diesem Kapitel

<b>chattr</b>	Setzt Dateiattribute für ext2- und ext3-Dateisysteme	chattr(1)	56
<b>chgrp</b>	Setzt Gruppenzugehörigkeit von Dateien und Verzeichnissen	chgrp(1)	49
<b>chmod</b>	Setzt Rechte für Dateien und Verzeichnisse	chmod(1)	47
<b>chown</b>	Setzt Eigentümer und Gruppenzugehörigkeit für Dateien und Verzeichnisse	chown(1)	48
<b>lsattr</b>	Zeigt Dateiattribute auf ext2- und ext3-Dateisystemen an	lsattr(1)	56
<b>setfacl</b>	Erlaubt die Manipulation von ACLs	setfacl(1)	51
<b>star</b>	POSIX-kompatibles Archivprogramm mit ACL-Unterstützung	star(1)	51
<b>umask</b>	Stellt die <i>umask</i> (Rechtekontrolle für neue Dateien) ein	bash(1)	50

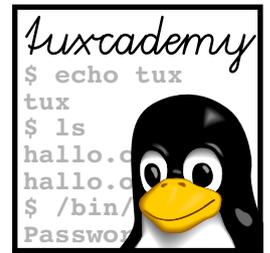
## Zusammenfassung

- 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.





# 4

## Prozessverwaltung

### Inhalt

4.1	Was ist ein Prozess? . . . . .	60
4.2	Prozesszustände . . . . .	61
4.3	Prozessinformationen – ps. . . . .	62
4.4	Prozesse im Baum – pstree. . . . .	63
4.5	Prozesse beeinflussen – kill und killall. . . . .	64
4.6	pgrep und pkill . . . . .	66
4.7	Prozessprioritäten – nice und renice . . . . .	67
4.8	Weitere Befehle zur Prozessverwaltung – nohup, top . . . . .	69

### 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

## 4.1 Was ist ein Prozess?

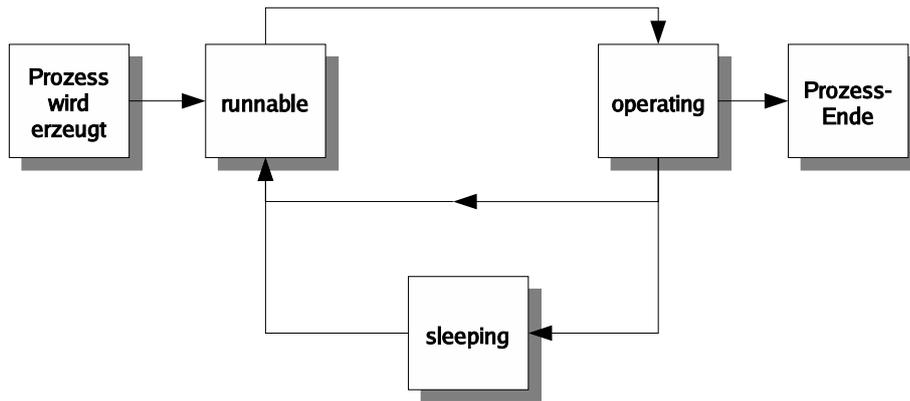
Ein Prozess ist im Wesentlichen ein »laufendes Programm«. Prozesse haben Code, der ausgeführt wird, und Daten, auf denen der Code operiert, aber auch diverse Attribute für die interne Verwaltung des Betriebssystems, zum Beispiel die Folgenden:

- |                                   |   |
|-----------------------------------|---|
| Prozessnummer                     | <ul style="list-style-type: none"> <li>• Die eindeutige Prozessnummer, kurz PID (engl. <i>process identity</i>), dient zur unmissverständlichen Identifizierung des Prozesses und kann zu jedem Zeitpunkt nur einem Prozess zugeordnet sein.</li> </ul>   |
| Prozessnummer des Elter-Prozesses | <ul style="list-style-type: none"> <li>• Alle Prozesse kennen die Prozessnummer des Elter-Prozesses, kurz PPID (engl. <i>parent process ID</i>). 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.</li> </ul>   |
| Benutzer Gruppen                  | <ul style="list-style-type: none"> <li>• 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 3.4.) Außerdem darf der Benutzer, dem der Prozess zugeordnet ist, den Prozess anhalten, beenden oder anderweitig beeinflussen. Die Benutzer- und Gruppenzuordnungen werden an Kindprozesse vererbt.</li> </ul>  |
| Priorität                         | <ul style="list-style-type: none"> <li>• Die CPU-Rechenzeit wird vom System in kleine »Scheiben« (engl. <i>time slices</i>) 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 <i>scheduler</i> (»Planer«) anhand der Priorität des Prozesses getroffen.</li> </ul>   |
|                                   | <p> 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.</p> |
| weitere Attribute                 | <ul style="list-style-type: none"> <li>• Ein Prozess hat weitere Attribute – ein aktuelles Verzeichnis, eine Prozessumgebung, ... –, die ebenfalls an Kindprozesse weitergegeben werden.</li> </ul>   |
| Prozessdateisystem                | <p>Die genannten Informationen können Sie im Verzeichnis <code>/proc</code> 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:</p>   |

```
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 | <p> 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 <code>jobs</code>, <code>bg</code></p> |
|--------------|--|



**Bild 4.1:** 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

 **4.1** [3] Wie können Sie sich die Umgebungsvariablen eines beliebigen Ihrer Prozesse anschauen? (*Tipp:* /proc-Dateisystem.)

 **4.2** [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«.)

## 4.2 Prozesszustände

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.

Prozesszustand  
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



**4.3 [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.)



**4.4 [4]** (Beim zweiten Lesen.) Können Sie absichtlich einen Zombie-Prozess erzeugen?

## 4.3 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



**4.5** [!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`?



**4.6** [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.

## 4.4 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
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.

## 4.5 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.

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.

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



4.7 [2] Welche Signale werden standardmäßig ignoriert? (Tipp: signal(7))

## 4.6 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



**4.8** [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.)



**4.9** [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 abzurechnen.

## 4.7 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).

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.)

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



**4.10** [2] Versuchen Sie, einem Prozess eine höhere Priorität zu geben. Möglicherweise funktioniert das nicht, warum? Überprüfen Sie den Prozesszustand mit *ps*.



**4.11** [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`).



**4.12** [2] Was ist der Unterschied zwischen »`ulimit -f`« und Plattenplatzkontingentierung?

## 4.8 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.

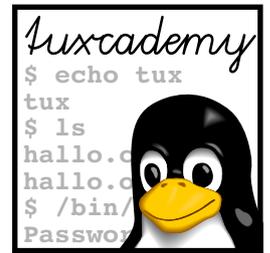
`top` vereint die Funktionen vieler Befehle zur Prozessverwaltung unter einer Oberfläche. Zudem bietet es nach Aufruf eine sich ständig aktualisierende Prozessstabelle. Interaktiv können verschiedene Operationen ausgeführt werden, eine Übersicht erhalten Sie mit `(h)`. Unter anderem ist es möglich, die Liste nach verschiedenen Kriterien zu sortieren, Signale an Prozesse zu versenden (`(k)`) und die Priorität zu verändern (`(r)`).

## Kommandos in diesem Kapitel

<code>kill</code>	Hält einen Hintergrundprozess an	<code>bash(1)</code> , <code>kill(1)</code>	64
<code>killall</code>	Schickt allen Prozessen mit passendem Namen ein Signal	<code>killall(1)</code>	65
<code>nice</code>	Startet Programme mit einem anderen <i>nice</i> -Wert	<code>nice(1)</code>	67
<code>nohup</code>	Startet ein Programm so, dass es immun gegen <code>SIGHUP</code> -Signale ist	<code>nohup(1)</code>	69
<code>pgrep</code>	Sucht Prozesse anhand ihres Namens oder anderer Kriterien	<code>pgrep(1)</code>	66
<code>pkill</code>	Signalisiert Prozessen anhand ihres Namens oder anderer Kriterien	<code>pkill(1)</code>	67
<code>ps</code>	Gibt Prozess-Statusinformationen aus	<code>ps(1)</code>	62
<code>pstree</code>	Gibt den Prozessbaum aus	<code>pstree(1)</code>	63
<code>renice</code>	Ändert den <i>nice</i> -Wert laufender Prozesse	<code>renice(8)</code>	68
<code>top</code>	Bildschirmorientiertes Programm zur Beobachtung und Verwaltung von Prozessen	<code>top(1)</code>	69
<code>ulimit</code>	Legt Ressourcenbegrenzungen für Prozesse fest	<code>bash(1)</code>	68

## 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.
- `ps tree` 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.



# 5

## Hardware

### Inhalt

5.1	Grundlagen . . . . .	72
5.2	Linux und PCI (Express) . . . . .	73
5.3	USB. . . . .	75
5.4	Geräteeinbindung und Treiber . . . . .	78
5.4.1	Überblick. . . . .	78
5.4.2	Geräte und Treiber . . . . .	79
5.4.3	Das Verzeichnis /sys. . . . .	80
5.4.4	udev . . . . .	81
5.4.5	Geräteeinbindung und D-Bus . . . . .	82

### Lernziele

- Grundlagen der PC-Hardwareunterstützung von Linux kennen
- BIOS, UEFI, PCI, USB und andere wichtige Begriffe einordnen können
- Grundbegriffe der dynamischen Peripherieunterstützung über udev kennen

### Vorkenntnisse

- Grundlegende Linux-Kenntnisse
- Grundlegende Kenntnisse von PC-Hardware und ihrer Ansteuerung bei anderen Betriebssystemen sind von Vorteil

## 5.1 Grundlagen

Linux unterstützt ein sehr breites Spektrum von Systemarchitekturen und -plattformen: Der Linux-Kern steht für alle heute gängigen Mikroprozessoren zur Verfügung und Linux läuft auf Rechnern von bescheidenen PDAs bis zu den größten Großrechnern. Linux ist auch fester Bestandteil vieler Geräte, die man nicht auf den ersten Blick damit in Verbindung bringen würde – Digital- und Videokameras, Router, Fernseher und Set-Top-Boxen, Navigationsgeräte und vieles andere mehr – und die zum Teil »ungewöhnliche« Hardware verwenden. Die mit Abstand meisten Linux-Rechner, die als »Computer« im eigentlichen Sinne gelten, basieren auf der von IBM und Intel begründeten x86-PC-Architektur.

 Die x86-PC-Architektur von IBM und Intel ist auch als einzige für die LPIC-1-Zertifizierung von Bedeutung, wobei man sagen muss, dass architektur-spezifische Themen inzwischen nur noch einen kleinen Teil der Lernziele ausmachen (früher war das mehr).

Architektur Computer verfügen heutzutage über einen oder mehrere Prozessoren (CPUs), Speicher für Programmablauf und Daten (RAM) und Hintergrundspeicher für Dateien (Platten, SSDs). Dazu kommen diverse Peripheriegeräte für Eingabe (Tastatur, Maus, Grafiktablett, Webcam, ...) und Ausgabe (Grafik»karte« – heute auch gerne Teil der CPU –, Audio, ...) und vieles andere mehr. Diese Peripheriegeräte sind entweder Teil der Hauptplatine (engl. *motherboard*) oder werden über verschiedene Schnittstellen (PCIe, USB, SATA, SCSI, ...) angebunden. Zur Vernetzung mit anderen Rechnern dienen Ethernet oder WLAN. Damit ein Rechner starten kann, braucht er ferner »Firmware« in einem nur lesbaren Speicher (ROM)<sup>1</sup>.

 Bei PCs heißt die Firmware entweder »BIOS« (bei älteren Systemen) oder »UEFI«. Früher – in der Zeit von MS-DOS – diente das BIOS dazu, die Verbindung zwischen dem eigentlichen Betriebssystem (DOS) und der Hardware des Rechners herzustellen. Heute wird es nur noch dazu gebraucht, den Rechner nach dem Einschalten zu initialisieren und ein »richtiges« Betriebssystem zu finden und zu starten. Moderne Betriebssysteme lassen das BIOS ansonsten komplett links liegen.

 UEFI ist eine modernere Implementierung derselben Idee, die mit einigen lästigen Einschränkungen des – aus den 1980er Jahren übrig gebliebenen – BIOS aufräumt und umfangreichere Möglichkeiten bietet, etwa wenn es um die Verwaltung mehrerer Betriebssysteme (oder Betriebssystemversionen) auf demselben Rechner oder die Sicherung der Systemsoftware gegen unerwünschte Manipulationen geht.

 Neue Rechner sind UEFI-basiert, können aber, wenn nötig, oft noch so tun, als hätten sie ein traditionelles BIOS, damit ältere Betriebssysteme unterstützt werden.

Während früher möglicherweise umfangreiche BIOS-Anpassungen nötig waren, damit ein Rechner zufriedenstellend mit Linux laufen konnte, ist das heute in der Regel kein Problem mehr. Sie müssen möglicherweise nur noch an ein paar Stellen eingreifen, etwa um Datum und Uhrzeit zu stellen.

Uhr des Rechners  Sie müssen sich entscheiden, ob die Uhr des Rechners in Zonenzeit (in Deutschland die »Mittleuropäische (Sommer-)Zeit«) oder in Weltzeit (UTC) laufen soll. Linux – das die Uhr des Rechners eigentlich nur beim Systemstart konsultiert – kann mit beiden umgehen, muss aber wissen, woran es ist. UTC ist auf reinen Linux-Systemen vorzuziehen, während Zonenzeit sinnvoll sein kann, wenn der Rechner alternativ noch mit anderen Betriebssystemen gestartet wird.



Bei BIOS-basierten Rechnern muss das BIOS zumindest die Platte kennen, von der aus das System gestartet werden soll. In der Regel können Sie angeben, welche Eigenschaften die verschiedenen Platten im System haben und ob das BIOS sie überhaupt sehen soll – diese Einstellungen haben keine Bedeutung für die spätere Benutzung der Platte(n) mit Linux, so dass es möglich ist, altertümliche BIOS-Implementierungen auszutricksen, die nicht mit den modernen großen Platten umgehen können, einfach indem Sie die Platte im BIOS deaktivieren. Linux kann später trotzdem darauf zugreifen, nur booten können Sie von einer solchen Platte nicht. Platten



Im BIOS können Sie für jede Platte eine Plattengeometrie eintragen, also die Anzahl von Schreib/Lese-Köpfen, Sektoren pro Zylinder und Zylindern pro Kopf der Platte. Heutige Platten haben auf den äußeren Zylindern mehr Sektoren als auf den inneren und folgen auch sonst nicht mehr dem »Zylinder/Kopf/Sektor«- oder »CHS«-Modell, sondern zählen die Sektoren einfach sequentiell von 0 bis ... – der LBA-Modus (engl. *linear block access*). Trotzdem behauptet jede Platte nach wie vor eine frei erfundene, der Plattenkapazität entsprechende Geometrie, um alte BIOSse und DOS glücklich zu machen. Bei modernen BIOS-Implementierungen können Sie Platten explizit in den »LBA«-Modus schalten; solange der Rechner mit den Standardeinstellungen bootet, ist das aber nicht nötig. Plattengeometrie  
LBA-Modus

In der Firmware können Sie oft auch bestimmte Arten von Peripherieunterstützung ein- und ausschalten: Peripherie

- Zuordnung serieller Schnittstellen aus Betriebssystemssicht (meist COM1: o. ä. genannt) zu existierenden seriellen Schnittstellen, IrDA-Port usw.
- Unterstützung für USB, insbesondere USB-Tastatur und -maus
- Interne Grafik, Sound, ...
- Energie- und Betriebszustandskontrolle (ACPI)

Es ist im Rahmen dieses Buchs nicht möglich (und für LPIC-1 glücklicherweise auch nicht nötig), hier detaillierte Vorgaben zu machen. Behalten Sie im Hinterkopf, dass es diese Einstellungen gibt und dass Sie, wenn irgendwelche Geräte unter Linux partout nicht zum Laufen kommen wollen, prüfen sollten, was die Firmware zu dem Thema zu sagen hat. Es kann sein, dass das betreffende Gerät im Firmware-Setup deaktiviert ist oder ein dort aktiviertes Gerät dem eigentlich gewünschten »im Weg steht«.

## 5.2 Linux und PCI (Express)

In den gut 30 Jahren, seit der erste IBM-PC auf den Markt kam, haben die »Innereien« der Hardware sich in fast allen Aspekten radikal geändert. Nicht zuletzt das dem Rechner zugrundeliegende Bussystem, das die CPU und den Speicher mit den wichtigsten Peripheriegeräten verbindet, hat seit dem ursprünglichen »ISA«-Bus einige Metamorphosen durchlaufen; der heutige Standard heißt **PCI Express** (PCIe) und hat seine Vorläufer fast völlig verdrängt. PCI Express



PCIe ist eine Weiterentwicklung von PCI, die viele Konzepte übernommen hat (etwa die Gerätecodes zur Hardwareerkennung, siehe unten), aber elektrisch völlig anders funktioniert.



PCIe ist konzeptuell im Gegensatz zu seinen Vorgängern ein serieller Bus, der unabhängige Punkt-zu-Punkt-Verbindungen zwischen verschiedenen Geräten erlaubt. Früher teilten sich alle Geräte einen parallelen Bus, was zu Einschränkungen bei der erreichbaren Geschwindigkeit führte.

<sup>1</sup>Technisch gesehen verwenden die allermeisten Computer statt »echtem« ROM heute Flash-ROM, das mit geeigneten Werkzeugen neu geschrieben werden kann. Das ändert aber nichts am Prinzip

```

# lspci
00:00.0 Host bridge: Intel Corp Core Processor DRAM Controller (rev 12)
00:01.0 PCI bridge: Intel Corp Core Processor PCI Express x16 Root Port
00:16.0 Communication controller: Intel Corporation 5 Series/3400▷
  < Series Chipset HECI Controller (rev 06)
00:16.3 Serial controller: Intel Corp 5 Series/3400 Series Chipset KT ▷
  < Controller (rev 06)
00:19.0 Ethernet controller: Intel Corporation 82577LM Gigabit Network▷
  < Connection (rev 06)
00:1a.0 USB controller: Intel Corporation 5 Series/3400 Series▷
  < Chipset USB2 Enhanced Host Controller (rev 06)
00:1b.0 Audio device: Intel Corporation 5 Series/3400 Series Chipset▷
  < High Definition Audio (rev 06)
<<<<<<
01:00.0 VGA compatible controller: NVIDIA Corporation GT218M▷
  < [NVS 3100M] (rev a2)
01:00.1 Audio device: NVIDIA Corporation High Definition Audio▷
  < Controller (rev a1)
<<<<<<

```

**Bild 5.1:** Ausgabe von lspci auf einem typischen x86-PC



Aus Geschwindigkeitsgründen ist es bei PCIe möglich, Datenpakete über mehrere Verbindungen (»Lanes«) parallel zu übertragen. Das ändert aber nichts an den Prinzipien.

#### Hardwareerkennung

Ein gravierender Vorteil von PCIe ist, dass eine automatische Hardwareerkennung möglich ist. Jedes angeschlossene Gerät kann einen Code melden, der dessen Natur, Hersteller und Modell angibt. Diese Information können Sie mit dem Kommando `lspci` abfragen (Bild 5.1). Am Anfang jeder Zeile ist die »PCI-ID« des jeweiligen Geräts angegeben, die seine Position auf dem Bus angibt.



Linux benutzt die Gerätecodes, um Treiber für die verschiedenen gefundenen Peripheriegeräte auszuwählen und zu konfigurieren. Der Kernel und die `udev`-Infrastruktur arbeiten dabei zusammen. Dazu später mehr.



Früher musste man im Wesentlichen raten, aber das konnte Probleme bis hin zum Systemabsturz geben, wenn ein Treiber das »richtige« falsche Gerät anpiekste. Im Zweifelsfall mussten Sie als Administrator hier mühsam manuelle Nachhilfe leisten!

`lspci` unterstützt einige interessante Optionen: »`lspci -v`« liefert eine ausführlichere Ausgabe:

```

# lspci -v
00:00.0 Host bridge: Intel Corp Core Processor DRAM Controller (rev 12)
  Subsystem: Hewlett-Packard Company Device 172b
  Flags: bus master, fast devsel, latency 0
  Capabilities: [e0] Vendor Specific Information: Len=0c <?>

00:01.0 PCI bridge: Intel Corporation Core Processor PCI Express x16▷
  < Root Port (rev 12) (prog-if 00 [Normal decode])
  Flags: bus master, fast devsel, latency 0
  Bus: primary=00, secondary=01, subordinate=01, sec-latency=0
  I/O behind bridge: 00005000-00005fff
  Memory behind bridge: d2000000-d30fffff
<<<<<<

```



Früher waren ähnliche Informationen auch in der »Datei« `/proc/pci` zu finden, die aber von neuen Kernels nicht mehr standardmäßig zur Verfügung gestellt wird. Die offizielle Methode, an die PCI-Daten zu kommen, ist `lspci`.

»`lspci -t`« liefert eine baumartige Darstellung der Verbindungen zwischen den einzelnen Komponenten: baumartige Darstellung

```
# lspci -t
-- [0000:ff]--+-00.0
|      +-00.1
|      +-02.0
|      +-02.1
|      +-02.2
|      \-02.3
\-[0000:00]--+-00.0
      +-01.0-[01]--+-00.0
      |      \-00.1
      +-16.0
      +-16.3
      +-19.0
<<<<<<
```

Hieran können Sie zum Beispiel sehen, dass die »PCI Bridge« des Chipsatzes (Gerät `0000:00:01.0`) die Verbindung zum »VGA compatible controller« (Gerät `0000:01:00.0`) und dem »Audio device« (Gerät `0000:01:00.1`, präziser gesagt der HDMI-Audioausgabe der Grafikkarte) darstellt. Auch der Ethernet-Adapter (Gerät `0000:01:19.0`) wird über PCIe angebunden.

»`lspci -n`« schließlich gibt die Gerätecodes direkt aus, statt deren Bedeutung in der Datenbank nachzuschlagen: Gerätecodes

```
# lspci -n
00:00.0 0600: 8086:0044 (rev 12)
00:01.0 0604: 8086:0045 (rev 12)
00:16.0 0780: 8086:3b64 (rev 06)
00:16.3 0700: 8086:3b67 (rev 06)
00:19.0 0200: 8086:10ea (rev 06)
00:1a.0 0c03: 8086:3b3c (rev 06)
00:1b.0 0403: 8086:3b56 (rev 06)
<<<<<<
```

## Übungen



**5.1** [!2] Untersuchen Sie die Hardwarestruktur Ihres Rechners mit Kommandos wie »`lspci -v`« oder »`lspci -t`«. Was können Sie Interessantes feststellen?

## 5.3 USB

Der *Universal Serial Bus* oder USB dient bei PCs heute praktisch zum Anschluss aller externen Peripheriegeräte, die nicht drahtlos (über WLAN oder Bluetooth) angesprochen werden. Ziel sind »altlastenfreie« (engl. *legacy-free*) Computer, die ohne die frühere Vielzahl von gerätespezifischen Anschlüssen für Tastatur, Maus, Drucker, Modems, ... auskommen.



USB verwendet traditionell ein »idiotensicheres« Verkabelungskonzept mit verschiedenen Buchsen für Rechner (»A«) und Peripheriegeräte (»B«) und entsprechenden Steckern an den Kabelenden, um Fehler zu verhindern. Außerdem erlaubt USB prinzipiell das Umstecken von Geräten im eingeschalteten Zustand (auch wenn das bei Festplatten oder USB-Sticks normalerweise keine gute Idee ist).

Tabelle 5.1: USB-Standards

Version	seit	max. Geschw.	Geräte
1.1	1998	1,5 MiBit/s	Tastaturen, Mäuse, Modems, Audio, ...
		12 MiBit/s	10-MBit-Ethernet, Platten, ...
2.0	2000	480 MiBit/s	Platten, 100-MiBit-Ethernet, Video ...
3.0	2008	5 GiBit/s	Platten, Grafik, ...
3.1	2013	10 GiBit/s	PCIe



Die neueste Mode (2015) sind die »USB-C«-Buchsen, die bei Rechnern und Peripheriegeräten gleich sind und bei denen es auch egal ist, wie herum Sie den Stecker in die Buchse stecken (was vor allem bei den USB-A-Steckern ein konstantes Ärgernis ist – Pro-Tipp, das USB-Logo auf dem Stecker zeigt beim Einstecken immer zu Ihnen hin). USB-C-Verbindungen sind nicht nur gut für USB, sondern auch noch für alle möglichen anderen Sachen (starker Ladestrom, Grafikschnelle und was einem sonst so in den Sinn kommen könnte), aber das System steckt noch in den Kinderschuhen.

USB-Controller USB ist heute Bestandteil aller neuen PCs. Das heißt, jeder PC enthält einen oder mehrere *USB-Controller*, deren jeder bis zu 127 USB-Geräte steuern kann.

USB-Hubs



Da PCs nicht so viele USB-Buchsen haben, erlaubt USB die baumartige Verschaltung von Geräten über *USB-Hubs*. An eine USB-Buchse im Rechner können Sie also einen Hub mit mehreren Buchsen anschließen; die Geräte am Hub teilen sich die Leitung zum Rechner. Natürlich können Sie an einen Hub auch weitere Hubs anschließen (auch wenn das in der Praxis nicht immer eine geniale Idee ist).

Der USB-Standard wurde seit seiner Einführung 1996 kontinuierlich weiterentwickelt, wobei neuere Versionen die mögliche maximale Übertragungsgeschwindigkeit immer weiter erhöhten. Aktuell (2015) ist die Version 3.1, mit der es prinzipiell möglich ist, externe Geräte mit PCIe-Busgeschwindigkeit anzusteuern (was zum Beispiel für externe Grafikkarten nett ist).



Neuere USB-Implementierungen sind prinzipiell rückwärtskompatibel. Sie können USB-2.0-Geräte auf einem USB-1.1-Bus betreiben, wobei natürlich nur die Geschwindigkeit von USB 1.1 zur Verfügung steht. Umgekehrt können Sie USB-1.1-Geräte an einen USB-2.0-Hub oder -Controller anschließen, aber sie werden davon nicht schneller – allerdings stehen sie auch »echten« USB-2.0-Geräten nicht im Weg herum.



USB-3.x-fähige Geräte unterstützen tatsächlich USB 2.0 und USB 3.x gleichzeitig über separate Adern im Kabel bzw. separate Kontakte an der Steckverbindung. Sie können einen USB-2.0-Stecker in eine USB-3.x-Buchse an Ihrem Rechner stecken und bekommen damit eine USB-2.0-Verbindung; umgekehrt können Sie auch einen USB-3.x-Stecker in eine USB-2.0-Buchse stecken, bekommen aber auch nur eine USB-2.0-Verbindung. Für »echtes« USB-3.x muss Ihr Rechner USB 3.x können, und Sie brauchen USB-3.x-Geräte und -Kabel.



Machen Sie einen Bogen um die in *c't* 13/2012 auf Seite 113 gezeigten Adapter von USB 3.0 auf Lüsterklemmen.



Die oben erwähnten neuen USB-C-Steckverbindungen sind etwa zeitgleich mit USB 3.1 angekündigt worden. Diese beiden Innovationen haben aber nichts miteinander zu tun. USB-C-Stecker und -Buchsen haben wie die bisherigen USB-3.x-Stecker und -Buchsen vom Typ A und B separate Leitungen für USB 1.1/2.0 und USB 3.x. Prinzipiell könnte man also zum Beispiel

USB-2.0-Platten mit USB-C-Buchsen anbieten. (Wir werden uns an Kabel mit einem USB-A- oder -B-Stecker am einen und einem USB-C-Stecker am anderen Ende gewöhnen müssen; Adapterkabel mit USB-C-Buchsen sind offiziell nicht erlaubt.)

Wenn ein USB-Gerät an den Bus angeschlossen wird, bekommt es eine Zahl zwischen 1 und 127 als Gerätenummer zugewiesen und der Rechner liest den »Deskriptor« des Geräts aus. Dieser gibt an, um was für eine Sorte Gerät es sich handelt, welche Arten von USB-Schnittstellen es unterstützt und Ähnliches. Insbesondere gehört jedes Gerät zu einer Klasse von Geräten, die vom Rechner ähnlich behandelt werden – etwa den *Human Interface Devices*, also Geräten, die Eingabemöglichkeiten für Menschen realisieren: Tastaturen, Mäuse, Steuerknüppel, aber auch Schalter, Regler, Telefontastaturen, Steuerräder, Datenhandschuhe und anderes. Klassen können Unterklassen haben, die ein Gerät noch weiter eingrenzen.

Aufzählung

Deskriptor

Klasse

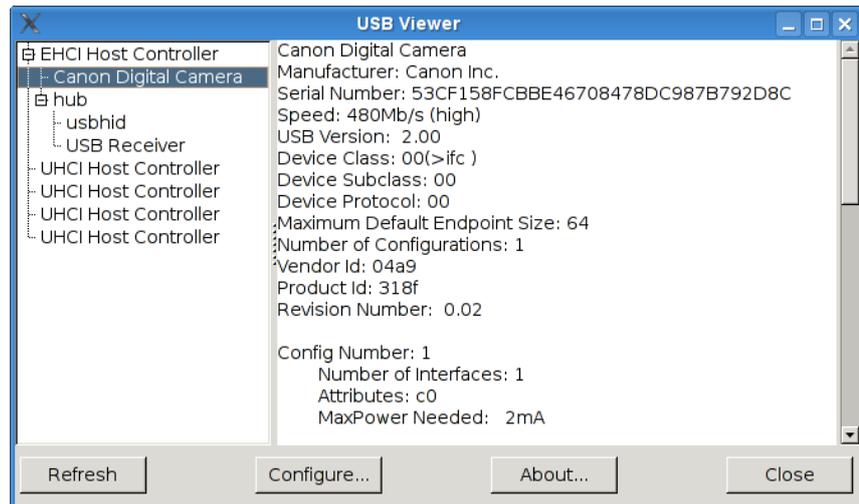
Unterklasse

Mit dem Kommando `lsusb` können Sie sich ein Bild davon machen, welche USB-Geräte gerade an Ihr System angeschlossen sind. Ähnlich wie `lspci` gibt es die Gerätenummern und -namen aus:

```
$ lsusb
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate >
  < Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 006: ID 04f2:b15e Chicony Electronics Co., Ltd
Bus 001 Device 008: ID 046d:0807 Logitech, Inc. Webcam B500
Bus 001 Device 010: ID 046a:0023 Cherry GmbH CyMotion Master Linux>
  < Keyboard G230
Bus 001 Device 009: ID 046d:c52b Logitech, Inc. Unifying Receiver
Bus 001 Device 007: ID 05e3:0608 Genesys Logic, Inc. USB-2.0 4-Port HUB
<<<<<<
```

Mit der Option `-v` wird das Programm deutlich »geschwätziger«:

```
# lsusb -v
Bus 002 Device 002: ID 8087:0020 Intel Corp. Integrated Rate >
  < Matching Hub
Device Descriptor:
  bLength                18
  bDescriptorType        1
  bcdUSB                  2.00
  bDeviceClass            9 Hub
  bDeviceSubClass         0 Unused
  bDeviceProtocol         1 Single TT
  bMaxPacketSize0         64
  idVendor                0x8087 Intel Corp.
  idProduct               0x0020 Integrated Rate Matching Hub
  bcdDevice               0.00
  iManufacturer          0
  iProduct                0
  iSerial                0
  bNumConfigurations      1
Configuration Descriptor:
  bLength                9
  bDescriptorType        2
  wTotalLength           25
  bNumInterfaces         1
  bConfigurationValue    1
  iConfiguration         0
  bmAttributes           0xe0
```



**Bild 5.2:** Das Programm usbview

Self Powered	
Remote Wakeup	
MaxPower	0mA
Interface Descriptor:	
bLength	9
bDescriptorType	4
bInterfaceNumber	0
bAlternateSetting	0

<<<<<

💡 Die Ausgabe von »lsusb -v« ist zwar ausführlich, aber auch relativ unübersichtlich. Das Programm usbview bereitet die Daten etwas freundlicher auf (Bild 5.2).

💡 Wenn Sie wissen wollen, woher lsusb seine enzyklopädische Kenntnis von USB-Peripherie bezieht: Schauen Sie mal in der Datei /var/lib/usbutils/usb.ids nach.

## 5.4 Geräteeinbindung und Treiber

### 5.4.1 Überblick

Die interessante Frage, die im Raum steht, ist natürlich, wie Linux mit nahezu beliebigen Peripheriegeräten umgehen kann – sowohl die in den Rechner fest eingebauten als auch solche, die während des Betriebs dazukommen oder verschwinden können.

Das Reden mit Peripheriegeräten ist die natürliche Aufgabe des Linux-Kerns. Sein Job ist es, angeschlossene Peripherie zu erkennen und zu initialisieren und Anwenderprogrammen in kontrollierter Form Zugriff auf sie zu erlauben. Dazu gehört selbstverständlich eine angemessene Rechtezuordnung, damit nur entsprechend privilegierte Benutzer direkten Zugriff auf Peripheriegeräte bekommen können.

💡 Im wirklichen Leben teilt sich der Linux-Kern die Arbeit der Geräteverwaltung gerne mit Programmen im »Userspace«. Das ist sinnvoll, da einerseits der Linux-Kern schlank bleibt und andererseits möglichst wenig Funktionalität fest im Kern verdrahtet wird, sondern in Anwenderprogrammen angesiedelt ist, wo sie leichter zu programmieren und an besondere Wünsche anzupassen ist.



Ein Beispiel: Einfache Drucker werden heute gerne über USB mit dem Rechner verbunden. Wenn ein neuer Drucker angestöpselt wird, merkt der Linux-Kern das und kann anhand der USB-Hersteller- und -Geräteinformationen herausfinden, dass es sich bei dem gerade erkannten Gerät um einen Drucker handelt und welches Modell welches Herstellers vorliegt. Resultat dieses Initialisierungsvorgangs ist eine Gerätedatei unter `/dev`, über die Anwenderprogramme mit dem Drucker Kontakt aufnehmen können. – Natürlich reden aber nicht beliebige Anwenderprogramme direkt mit dem Drucker, sondern der direkte Zugriff auf dessen Gerätedatei wird über geeignete Zugriffsrechte beschränkt auf das Druckersubsystem (CUPS). Die CUPS-Konfigurationsroutine kann auf die Informationen des Linux-Kerns über Marke und Modell des Druckers zurückgreifen und einen passenden Druckertreiber wählen. Anwenderprogramme – etwa LibreOffice – drucken Dokumente, indem sie sie in einem geeigneten Format (PostScript oder PDF) an CUPS übergeben. CUPS wandelt die Druckdaten dann in ein Format um, das der Drucker versteht, und reicht sie über die Gerätedatei an den Linux-Kern weiter, der sie dann via USB an den tatsächlichen Drucker leitet.

### 5.4.2 Geräte und Treiber

Linux unterstützt die vielfältige Landschaft von Peripheriegeräten, die für PCs zur Verfügung stehen, über Treiber, die in der Regel als ladbare Module ausgeführt werden. Diese ladbaren Module sind in Unterverzeichnissen von `/lib/modules` zu finden und können manuell mit dem Kommando `modprobe` in den Kernel geholt werden:

ladbare Module

```
# modprobe bla
```

zum Beispiel sucht und lädt das Modul `bla.ko`. Sollte das gewünschte Modul von der Funktionalität anderer Module abhängen, die aktuell nicht geladen sind, sorgt `modprobe` dafür, auch diese Module zu laden.



Linux verwendet ladbare Module nicht nur zur Implementierung von Gerätetreibern im eigentlichen Sinne, sondern auch für Netzwerkprotokolle, die Paketfilter-Infrastruktur und diverse andere Aspekte des Systems, die nicht auf jedem System oder nicht zu jeder Zeit benötigt werden. Module erleichtern auch die Kernel-Weiterentwicklung und sind darum bei Programmierern populär.

Das Kommando `lsmod` gibt einen Überblick über die aktuell geladenen Module:

```
$ lsmod
Module                Size  Used by
nfs                    213896  1
lockd                  54248  1 nfs
nfs_acl                 2912  1 nfs
sunrpc                 162144  10 nfs, lockd, nfs_acl
tun                     8292  1
michael_mic            2304  4
arc4                    1824  4
<<<<<<
```

Sie können auch versuchen, ein einmal geladenes Modul mit `»modprobe -r«` wieder zu entfernen:

```
# modprobe -r bla
```

(Ob das klappt, hängt von Ihrem Kernel ab.) Ist das zu entfernende Modul das einzige, das von einem anderen Modul abhängt, versucht `modprobe` dieses andere Modul ebenfalls zu entfernen.

Automatisches Laden bei Bedarf Sie werden wahrscheinlich nur selten in die Verlegenheit kommen, das Kommando `modprobe` manuell absetzen zu müssen, da der Kernel sich weitgehend selbst darum kümmern kann, Module bei Bedarf zu laden. Grundlage dafür ist die Datei `/etc/modprobe.conf` (oder die Dateien in `/etc/modprobe.d`), die Einträge der Form

<code>alias block-major-3-* ide_generic</code>	<i>Blockgerät</i>
<code>alias char-major-10-1 psmouse</code>	<i>zeichenorientiertes Gerät</i>
<code>alias net-pf-16-proto-8 scsi_transport_iscsi</code>	<i>Netzprotokoll</i>

(und verschiedenes mehr) enthalten kann. Diese Einträge verbinden Gerätedateien in `/dev` wie

```
$ ls -l /dev/hda1 /dev/psaux
brw-rw---- 1 root disk  3, 1 Jan 22 02:03 /dev/hda1
crw-rw---- 1 root root 10, 1 Jan 22 02:03 /dev/psaux
```

mit den dazugehörigen Treibern – `block-major-3-*` heißt nichts anderes, als dass für blockorientierte Gerätedateien mit der *major device number* 3 und beliebiger *minor device number* (etwa `/dev/hda1` in unserem Beispiel) das Treibermodul `ide_generic` zuständig ist, während Zugriffe auf `/dev/psaux`, eine zeichenorientierte Gerätedatei mit den Nummern (10, 1), vom über `char-major-10-1` benannten Treiber, namentlich `psmouse`, bearbeitet werden. In dem Moment, wo zum ersten Mal ein Zugriff auf die betreffende Gerätedatei passiert, sucht der Linux-Kernel das dazugehörige Treibermodul und bindet es ein.



Sie werden sich sicher fragen, wie ein Treiber wie `ide_generic` bei Bedarf von einer IDE-Platte gelesen werden kann, wenn Linux ihn dafür braucht, die Platte überhaupt anzusprechen. Die Antwort darauf besteht darin, dass dieser Treiber beim Booten von der IDE-Platte nicht wirklich aus `/lib/modules` gelesen wird, sondern der Linux-Kern ihn schon vorher der »initialen RAM-Disk« entnommen hat, die der Bootlader (über die Firmware, also ohne Linux-Beteiligung) zusammen mit dem Linux-Kern von der Platte holt. (Näheres steht in Kapitel 8).



Sie können mit `modprobe` und den dazugehörigen Konfigurationsdateien noch viele andere fremdartige und wundervolle Dinge tun, die den Rahmen dieser Diskussion sprengen würden. Mehr darüber erfahren Sie zum Beispiel aus der Linup-Front-Schulungsunterlage *Linux-Systemanpassungen*.

### 5.4.3 Das Verzeichnis `/sys`

Bis einschließlich Linux 2.4 stellte das `/proc`-Verzeichnis die Möglichkeit dar, auf Details der Kernel- und Systemkonfiguration zugreifen zu können. Den Kernel-Entwicklern missfiel jedoch die zunehmende Verwilderung der Einträge unter `/proc`, insbesondere derer, die gar nichts mit Prozessen (dem ursprünglichen Einsatzzweck) zu tun hatten. Aus diesem Grund entschloss man sich, diejenigen Aspekte von `/proc`, die nichts mit Prozessverwaltung zu tun haben, mittel- bis langfristig so weit wie möglich in ein neues Pseudodateisystem, `sysfs`, zu verlagern, für das auch strengere »Spielregeln« gelten sollten. Das `sysfs` wird gemeinhin unter `/sys` eingebunden und steht seit der Linux-Kernversion 2.6 zur Verfügung.

Insbesondere ermöglicht `/sys/bus` den Zugriff auf Geräte sortiert nach dem Anschlussstyp (engl. *Bus*; `pci`, `usb`, `scsi`, ...). `/sys/devices/` ermöglicht ebenfalls den Zugriff auf Geräte, lediglich die Sortierordnung ist eine andere (Gerätetyp, z. B. PCI-Bus-Adresse). Realisiert wird diese Redundanz durch (symbolische) Links:

```
$ ls -ld /sys/devices/pci0000:00/0000:00:07.2/usb1
drwxr-xr-x 6 root root 0 Jun 27 19:35▷
└─ /sys/devices/pci0000:00/0000:00:1a.0/usb1
$ ls -ld /sys/bus/usb/devices/usb1
```

```
lrwxrwxrwx 1 root root 0 Jun 27 19:35 /sys/bus/usb/devices/usb1>
< -> ../../../../devices/pci0000:00/0000:00:1a.0/usb1
```

Im Beispiel sehen Sie das Verzeichnis, über das Sie Zugriff auf die Daten des ersten USB-Anschlusses haben. Es handelt es sich in der Tat um dasselbe Verzeichnis – siehe Inodenummer. Ein solches Verzeichnis enthält diverse Dateien mit Informationen über das entsprechende Gerät:

```
$ ls /sys/bus/usb/devices/usb1
1-0:1.0      bmAttributes  devpath      remove
1-1         bMaxPacketSize0  driver       serial
authorized  bMaxPower     ep_00        speed
authorized_default  bNumConfigurations  idProduct    subsystem
avoid_reset_quirk  bNumInterfaces     idVendor     uevent
bcdDevice     busnum        manufacturer  urbnum
bConfigurationValue  configuration      maxchild     version
bDeviceClass  descriptors    power
bDeviceProtocol  dev           product
bDeviceSubClass  devnum        quirks
$ cat /sys/bus/usb/devices/usb1/product
EHCI Host Controller
```

Ein Nachteil des bisherigen Kernelkonzeptes war die Uneindeutigkeit der Zuordnung der Schnittstellen der Geräte, z. B. der Gerätedateien. Die »Durchnummerierung« der Geräte (z. B. sda, sdb, sdc, usw. oder eth0, eth1, usw.) war bis Kernel 2.4 nicht eindeutig nachvollziehbar. Das sysfs ermöglicht dagegen eine eindeutige Zuordnung von Schnittstelle und Gerät.

Zuordnung der Schnittstellen

Die Schnittstellen finden Sie in den Verzeichnissen /sys/block (blockorientierte Geräte) und /sys/class (zeichenorientierte Geräte, mehr oder weniger):

```
$ ls /sys/block
dm-0 dm-2 dm-4 dm-6 loop1 loop3 loop5 loop7 sr0
dm-1 dm-3 dm-5 loop0 loop2 loop4 loop6 sda
$ ls /sys/class
ata_device dmi mem regulator tty
ata_link firmware misc rfkill vc
ata_port graphics mmc_host rtc video4linux
backlight hidraw net scsi_device vtconsole
bdi hwmon pci_bus scsi_disk wmi
block i2c-adapter pcmcia_socket scsi_host
bluetooth ieee80211 power_supply sound
bsg input ppdev spi_master
dma leds printer thermal
```

Ein Vorteil: Hier tauchen nur die Geräte auf, die es tatsächlich im System gibt.

Die Zuordnung der Schnittstellen findet im Falle der blockorientierten Geräte mit Hilfe symbolischer Links statt; Im jeweiligen Unterverzeichnis von /sys/block finden Sie eine Datei namens device, etwa:

```
$ ls -l /sys/block/sda/device
lrwxrwxrwx 1 root root 0 Jun 27 19:45 /sys/block/sda/device>
< -> ../../../../0:0:0:0
```

#### 5.4.4 udev

In grauer Vergangenheit war es unter Linux-Distributoren üblich, bei der Installation ein /dev-Verzeichnis anzulegen, das mit Gerätedateien für alle Peripherie

unter der Sonne gefüllt war, um auf alle Eventualitäten vorbereitet zu sein. Inzwischen ist dieser Ansatz verpönt zugunsten der Idee, bei Bedarf dynamisch Gerätedateien nur für diejenigen Geräte zu erzeugen, die tatsächlich zur Verfügung stehen.

Die Linux-Infrastruktur hierfür heißt *udev*, kurz für *userspace /dev*. Sie wurde von Greg Kroah-Hartman und Kay Sievers (und anderen) implementiert und steht in der heutigen Form seit Kernel 2.6.13 zur Verfügung. *udev* besteht im wesentlichen aus einer Bibliothek namens *libudev*, die sich um die Namenszuweisung an Geräte kümmert, und einem Daemon namens *udevd*, der die eigentliche Arbeit macht bzw. delegiert.

Unter *udev* enthält das */dev*-Verzeichnis auf der Platte nur die allernötigsten Einträge. Beim Systemstart wird ziemlich früh ein *tmpfs*-Dateisystem (also eine RAM-Disk, siehe Abschnitt 7.1.6) über */dev* eingehängt, in dem *udev* dann die passenden Gerätedateien anlegt. Dazu wird der *udev* gestartet, der auf »uevents« lauscht, Ereignismeldungen des Kernels über hinzugekommene oder entfernte Geräte. Jede solche Meldung wird mit einem Regelsatz verglichen, der entscheidet, unter welchem Namen das Gerät in dem von *udev* verwalteten */dev*-Verzeichnis erscheinen soll, und auch zusätzliche Aktionen auslösen kann, etwa eine als Binärdatei vorliegende »Firmware« in das Gerät zu laden, um es zu initialisieren. Diese Regeln sind in */etc/udev/rules.d* zu finden.



Es spricht nichts dagegen, dass ein Gerät unter mehreren Namen in */dev* erscheint. Ihre Digitalkamera, die ansonsten über USB wie eine Festplatte angesprochen wird, könnte zum Beispiel außer */dev/sd»was-auch-immer«* den Namen */dev/camera* bekommen. Das ist alles eine Frage der Regeln.



*udev* kümmert sich auch um Geräte, die nicht mit Gerätedateien in */dev* korrespondieren, etwa Netzwerkkarten.

Ein interessanter Nebeneffekt dieser Vorgehensweise ist, dass sie zwar eigentlich dafür gemeint ist, mit Geräten umgehen zu können, die irgendwann zur Laufzeit – genau genommen irgendwann nachdem der *udev* gestartet wurde – kommen und gehen (*hotplugging*). Dieselbe Methode wird aber auch als sogenanntes *coldplugging* beim ursprünglichen Systemstart angewendet, um die dann schon vorhandenen Geräte zu initialisieren. Natürlich können die *uevents*, die vom Kernel verschickt werden, um diese Geräte anzumelden, nicht bearbeitet werden, bevor *udev* läuft. Allerdings erlaubt der Kernel es, diese *uevents* bei Bedarf noch einmal zu verschicken, und genau das wird gemacht, nachdem *udev* zur Verfügung steht, um die Geräte im dynamische */dev*-Verzeichnis anzulegen. Damit besteht zwischen der Geräteinitialisierung per *hotplugging* und der per *coldplugging* kein Unterschied.



Um das zu ermöglichen, existiert unter */sys* im Eintrag für jedes Gerät eine Datei namens *uevent*. Sie müssen nur die Zeichenkette *add* in diese Datei schreiben, um den entsprechenden *uevent* auszulösen.



*udev* ist nicht der erste Versuch, eine entsprechende Infrastruktur in Linux zur Verfügung zu stellen. Das *devfs* von Richard Gooch tat etwas sehr Ähnliches, aber innerhalb des Kernels statt (wie *udev*) außerhalb. *devfs* fand nie viele Anhänger in der Gemeinde, da diverse Entscheidungen etwa über die Benennung von Geräten fest im *devfs* (und damit im Kernel) verdrahtet waren und nicht alle Entwickler damit übereinstimmten. *udev* exponiert die entsprechenden Regeln in änderbaren Konfigurationsdateien und ist damit wesentlich flexibler. Für das dynamische An- und Abmelden von Geräten gab es eine Infrastruktur namens *hotplug*, die aber von *udev* verdrängt wurde.

### 5.4.5 Geräteeinbindung und D-Bus

Anwenderprogramme Der letzte Baustein im Mosaik besteht darin, Anwenderprogramme ins Spiel zu bringen. Wie erreichen Sie, dass das Einstöpseln Ihrer Digitalkamera dazu führt,

dass Ihr Fotoverwaltungsprogramm gestartet wird und die neuen Bilder herunterlädt? Oder wie kommt das Icon für Ihre USB-Platte automatisch auf den Hintergrund Ihrer grafischen Arbeitsumgebung? Eine mögliche Antwort darauf lautet »udisks«.



Früher wurde hierfür eine Infrastruktur namens HAL (kurz für *hardware abstraction layer*) benutzt. HAL hat sich aber als zu kompliziert und mit Einschränkungen behaftet herausgestellt und wird nicht mehr weiterentwickelt.



Udisks ist eines der Nachfolgeprojekte, das sich speziell um Speichergeräte wie externe Platten und USB-Sticks kümmert; es gibt andere Projekte für andere Arten von Geräten. Große Teile von HAL sind auch in udev aufgegangen.

Grundlage von udisks und ähnlichen Projekten ist der D-Bus, ein einfaches System zur Interprozesskommunikation, das die meisten Arbeitsumgebungen unterstützen. Programme können beim D-Bus Dienste anmelden, die sie anderen Programmen anbieten wollen. Programme können auch auf Ereignisse warten. (Es macht nichts, wenn mehrere Programme auf dasselbe Ereignis warten; alle Interessenten werden benachrichtigt.) D-Bus hat zwei wesentliche Einsatzgebiete:

- Kommunikation von Programmen in derselben grafischen Arbeitsumgebung. Eine mögliche Anwendung wäre zum Beispiel, dass Ihr Telefonieprogramm bei einem eingehenden Anruf automatisch Ihr Musikabspielprogramm stumm schaltet.
- Kommunikation zwischen dem Betriebssystem und Ihrer grafischen Arbeitsumgebung.

Udisks besteht aus zwei Komponenten, einem Hintergrunddienst (`udisksd`) und einem Anwenderprogramm (`udisksctl`), das mit dem Hintergrunddienst kommunizieren kann. Der Hintergrunddienst ist über den D-Bus zugänglich und wird bei Bedarf gestartet, wenn ein Programm eine D-Bus-Anfrage an udisks stellt. Programme können auf diese Weise herausfinden, welche Speichergeräte zur Verfügung stehen, und Operationen ausführen, etwa Geräte einhängen oder aushängen. Dabei wird geprüft, ob der Benutzer die nötigen Privilegien hat; Benutzer könnten zum Beispiel an ihrem eigenen Arbeitsplatz USB-Sticks anstöpseln dürfen, aber nicht auf USB-Sticks an anderen Arbeitsplätzen zugreifen.

Hier ist ein Beispiel dafür, wie Sie mit `udisksctl` Informationen über einen USB-Stick abfragen können:

```
$ udisksctl status
MODEL                REVISION  SERIAL      DEVICE
-----
ST9320423AS          0002SDM1  5VH5TBTC    sda
hp    DVDROM GT30L    mP04       KZKA2LB3306 sr0
TOSHIBA TransMemory  1.00      7427EAB351F9CE110FE8E23E sdb
$ udisksctl info -b /dev/sdb
/org/freedesktop/UDisks2/block_devices/sdb:
org.freedesktop.UDisks2.Block:
  Configuration:      []
  CryptoBackingDevice:  '/'
  Device:              /dev/sdb
  DeviceNumber:       2064
  Drive:               '/org/freedesktop/UDisks2/drives/>
    < TOSHIBA_TransMemory_7427EAB351F9CE110FE8E23E'
  HintAuto:           true
  HintIconName:
  HintIgnore:         false
```

HintName:	
HintPartitionable:	true

Viele dieser Informationen stammen tatsächlich aus der udev-Datenbank und können vom Administrator über geeignete Regeln beeinflusst werden.



Fairerweise sollten wir noch anmerken, dass `udisks` eine »Minimallösung« darstellt. Die gängigen grafischen Arbeitsumgebungen wie KDE oder GNOME haben ähnliche Infrastrukturen, die sich zum Beispiel um eine direkte Einbindung von Speichergeräten in die grafische Oberfläche bemühen.

## Kommandos in diesem Kapitel

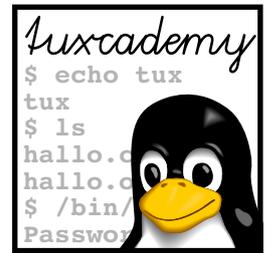
<code>lsmod</code>	Listet die geladenen Kernel-Module auf	<code>lsmod(8)</code>	79
<code>lspci</code>	Gibt Informationen über Geräte auf dem PCI-Bus aus	<code>lspci(8)</code>	74
<code>lsusb</code>	Listet alle angeschlossenen USB-Geräte auf	<code>lsusb(8)</code>	77
<code>modprobe</code>	Lädt Kernel-Module unter Berücksichtigung von Abhängigkeiten	<code>modprobe(8)</code>	79
<code>udev</code>	Daemon zur Behandlung von Kernel-uevents	<code>udev(8)</code>	82

## Zusammenfassung

- Die Firmware (BIOS/UEFI) organisiert den Bootvorgang eines Linux-Rechners, wird anschließend aber (außer in Ausnahmefällen) nicht mehr gebraucht.
- Typische für Linux relevante Firmware-Einstellungen umfassen zum Beispiel Datum und Uhrzeit, die Plattenparameter, die Bootreihenfolge und die Zuordnung von einigen Systemschnittstellen zu tatsächlicher Peripherie.
- Das `lspci`-Kommando gibt Aufschluss über den PCI(e)-Bus eines Rechners und die daran angeschlossenen Geräte.

## Literaturverzeichnis

- Hardware-HOWTO** Steven Pritchard. »Linux Hardware Compatibility HOWTO«, Januar 2004. <http://www.tldp.org/HOWTO/Hardware-HOWTO/>
- Linux-USB-Subsystem** Brad Hards. »The Linux USB sub-system«. <http://www.linux-usb.org/USB-guide/book1.html>
- PCI-HOWTO** Michael Will. »Linux PCI-HOWTO«, Juni 2001. <http://www.tldp.org/HOWTO/PCI-HOWTO.html>



# 6

## Platten (und andere Massenspeicher)

### Inhalt

6.1	Grundlagen . . . . .	86
6.2	Bussysteme für Massenspeicher . . . . .	86
6.3	Partitionierung. . . . .	89
6.3.1	Grundlagen. . . . .	89
6.3.2	Die traditionelle Methode (MBR) . . . . .	90
6.3.3	Die moderne Methode (GPT) . . . . .	91
6.4	Linux und Massenspeicher . . . . .	93
6.5	Platten partitionieren . . . . .	95
6.5.1	Prinzipielles. . . . .	95
6.5.2	Platten partitionieren mit fdisk . . . . .	97
6.5.3	Platten formatieren mit GNU parted . . . . .	100
6.5.4	gdisk . . . . .	102
6.5.5	Andere Partitionierungsprogramme . . . . .	102
6.6	Loop-Devices und kpartx . . . . .	103
6.7	Der Logical Volume Manager (LVM) . . . . .	105

### 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 (siehe Kapitel 5)

## 6.1 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*.

## 6.2 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« 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 durchnummerierter 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*<sup>1</sup> 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, einfallreicherweise Serial ATA se »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 siebenadriges Kabel für serielle Übertragung, eins pro Gerät.

<sup>1</sup>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<sup>2</sup>

 Außer der höheren Geschwindigkeit und der bequemerer 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<sup>3</sup>. 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.

<sup>2</sup>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.

<sup>3</sup>Was »klein« in diesem Kontext heißen soll, ist nirgendwo definiert, aber es muss etwas sein wie »kann noch von zwei Personen hochgehoben werden«.

Tabelle 6.1: 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 <sup>a</sup>
Ultra-160 SCSI <sup>b</sup>	16 Bit	160 MiB/s	16	LVD-Bus
Ultra-320 SCSI <sup>c</sup>	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 (vgl. Abschnitt 5.3) 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 lärmern 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 (Abschnitt 5.3) 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

-  **6.1** [1] Wie viele Platten oder SSDs enthält Ihr Computer? Mit welcher Kapazität? Wie sind sie an den Rechner angebunden (SATA, ...)?

## 6.3 Partitionierung

### 6.3.1 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.

### 6.3.2 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 – primäre Partitionen 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.



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<sup>4</sup>. 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.)

<sup>4</sup>Früher war es üblich, Partitionen über die Zylinder-, Kopf- und Sektoradresse der betreffenden Sektoren zu beschreiben, das ist aber schon lange verpönt.

**Tabelle 6.2:** Partitionstypen (hexadezimal) für Linux

Typ	Beschreibung
81	Linux-Daten
82	Linux-Auslagerungsspeicher ( <i>swap space</i> )
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 6.2.

### 6.3.3 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 6.3:** 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 aufgereichte 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 Partitions Grenzen 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 6.3 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.

## 6.4 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 7 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 7.2.3.)

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/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



**6.2** [!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?



**6.3** [!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.)



**6.4** [1] Stecken Sie einen USB-Stick in Ihren Computer. Prüfen Sie, ob in /dev neue Gerätedateien dazugekommen sind. Wenn ja, welche?

## 6.5 Platten partitionieren

### 6.5.1 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 6.7 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 7.

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 keinsten 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.

### 6.5.2 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` 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

```
# fdisk /dev/sdb Neue (leere) Platte
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 Swap-space, ...) 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 ...

Wie alle Linux-Kommandos kennt `fdisk` auch einige Kommandozeilenoptionen. Die wichtigsten davon lauten: Optionen

- 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



**6.5** [!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.



**6.6** [!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.

### 6.5.3 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
(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

```

*Upps.*

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

 **6.7** [!2] Wiederholen Sie Übung 6.5 mit parted statt fdisk, sowohl für das MBR- als auch das GPT-Schema.

 **6.8** [2] (Wenn Sie Kapitel 7 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?

### 6.5.4 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

 **6.9** [!2] Wiederholen Sie Übung 6.5 mit gdisk statt fdisk und generieren Sie eine GPT-Partitionstabelle.

 **6.10** [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.

### 6.5.5 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.

## 6.6 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.

Ein CD-ROM-Image erzeugen Sie aus einer existierenden CD-ROM einfach mit `dd`: CD-ROM-Image

```
# 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 7.

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



**6.11** [!2] Verwenden Sie die Test-»Platte« aus Übung 6.5. 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.

## 6.7 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

<b>cfdisk</b>	Plattenpartitionierungsprogramm mit textbildschirmorientierter Oberfläche	cfdisk(8)	102
<b>fdisk</b>	Partitionierungswerkzeug für Platten und ähnliche Medien	fdisk(8)	97
<b>gdisk</b>	Partitionierungswerkzeug für GPT-Platten	gdisk(8)	102
<b>kpartx</b>	Erzeugt Blockgeräte aus Partitionstabellen	kpartx(8)	103
<b>losetup</b>	Erzeugt und verwaltet Loop-Devices	losetup(8)	103
<b>parted</b>	Leistungsfähiges Partitionierungswerkzeug aus dem GNU-Projekt	Info: parted	100
<b>sfdisk</b>	Nichtinteraktives Partitionierungsprogramm	sfdisk(8)	102
<b>sgdisk</b>	Nichtinteraktives Partitionierungsprogramm für GPT-Platten	sgdisk(8)	103

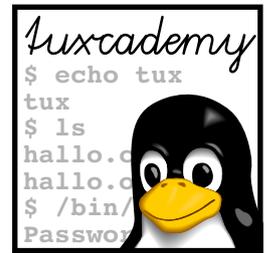
## Zusammenfassung

- 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`, `cgdisk` 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/>





# 7

## Dateisysteme: Aufzucht und Pflege

### Inhalt

7.1	Linux-Dateisysteme . . . . .	110
7.1.1	Überblick. . . . .	110
7.1.2	Die ext-Dateisysteme . . . . .	113
7.1.3	ReiserFS . . . . .	121
7.1.4	XFS. . . . .	123
7.1.5	Btrfs . . . . .	124
7.1.6	Noch mehr Dateisysteme . . . . .	126
7.1.7	Auslagerungsspeicher ( <i>swap space</i> ) . . . . .	127
7.2	Einbinden von Dateisystemen . . . . .	128
7.2.1	Grundlagen. . . . .	128
7.2.2	Der mount-Befehl . . . . .	128
7.2.3	Labels und UUIDs . . . . .	130
7.3	Das Programm dd. . . . .	132
7.4	Plattenkontingentierung (Quota) . . . . .	133
7.4.1	Überblick. . . . .	133
7.4.2	Kontingentierung für Benutzer (ext und XFS). . . . .	134
7.4.3	Kontingentierung für Gruppen (ext und XFS). . . . .	135

### 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 6)
- Vorkenntnisse über den Aufbau von Dateisystemen sind hilfreich

## 7.1 Linux-Dateisysteme

### 7.1.1 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.

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 8). 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`).

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.

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

- 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.



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.

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 7.2.2.



`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.

### 7.1.2 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 Gebäck.) 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 *log-strukturierte 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.

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:

ext4-Dateisystem

- 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 7.1.5) 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 -0 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
#
# umount /mnt
# cdrecord -data cdrom.img
... Sachen nach /mnt kopieren...
```

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.

Schritte Die Gerätedatei gibt das Blockgerät an, dessen Dateisystem geprüft werden soll. Wenn dieses Gerät kein ext-Dateisystem enthält, bricht das Kommando automatisch 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 gekennzeichnete 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 Rückgabewert Werte dieselbe Bedeutung haben.

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.:

```
# 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:

```
tune2fs [Optionen] <Gerät>
```

Die folgenden Optionen sind wichtig:

- c *<Anzahl>* setzt die maximale Anzahl der Einhängenvorgänge 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 Einhängenvorgängen. 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 7.2).



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



**7.1 [!2]** Generieren Sie auf einem geeigneten Medium (Plattenpartition, USB-Stick, mit `dd` angelegte Datei) ein ext4-Dateisystem.



**7.2 [2]** Ändern Sie die maximale Einhängezahl des in Übung 7.1 angelegten Dateisystems auf 30. Außerdem sollen 30% des Speicherplatzes für den Benutzer `test` reserviert sein.

### 7.1.3 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` **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



7.3 [!1] Wie lautet das Kommando, um die erste logische Partition der zweiten Platte im Reiser-Dateisystem zu formatieren?

### 7.1.4 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`.

### 7.1.5 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 6.7) 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 überholen 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<sup>1</sup>. 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

<sup>1</sup>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



7.4 [!1] Generieren Sie mit »mkfs -t btrfs« ein Btrfs-Dateisystem auf einer freien Partition.



7.5 [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.

### 7.1.6 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 7.2 zurück.)

Populär auf älteren Windows-Rechnern, USB-Sticks, Digitalkameras, MP3-Playern und anderen »Speichermedien« ohne große Ansprüche an Effizienz und 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/mmcb1k0p1
```

(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, 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 vermetzger, 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.

### 7.1.7 Auslagerungsspeicher (swap space)

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 7.2.2.

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`.

## 7.2 Einbinden von Dateisystemen

### 7.2.1 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 »drüberhängt« haben.

 Der Inhalt erscheint wieder, wenn Sie den Datenträger wieder mit `umount` aushängen. Trotzdem sollten Sie nichts über `/etc` und ähnliche Systemverzeichnisse drübermounten ...

### 7.2.2 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 7.1: 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	<i>Eine Möglichkeit ...</i>
# mount /home	<i>... und eine andere</i>

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 7.1. 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. *swap space*), 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
	xfv
	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 Reihenfolge 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 austöpseln) 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.

### 7.2.3 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.

Bei einem **Label** (engl. für Etikett, Schild) handelt es sich um einen frei vergeb- Label  
baren Text von maximal 16 Zeichen Länge, der im Superblock des Dateisystems  
abgelegt wird. Wenn Sie beim Anlegen des Dateisystems das Label vergessen ha-  
ben, so können Sie es jederzeit mit `e2label` nachtragen oder ändern. Das Komman-  
do

```
# 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
```

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:

UUID

```
$ uuidgen  
bea6383f-22a7-453f-8ef5-a5b895c8ccb0
```

und wird beim Anlegen eines Dateisystems automatisch und zufällig erzeugt. Da-  
durch wird sichergestellt, dass keine zwei Dateisysteme den gleichen UUID ha-  
ben. 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



7.6 [!2] Betrachten Sie die Einträge in den Dateien `/etc/fstab` und `/etc/mtab`. Wie unterscheiden sich diese?

## 7.3 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*<sup>2</sup>) 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 Festplatte eine Partitionstabelle, die mit der der alten übereinstimmt. Auch die

<sup>2</sup>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.

## 7.4 Plattenkontingentierung (Quota)

### 7.4.1 Überblick

Linux macht es möglich, die Maximalzahl von Inodes oder Datenblöcken pro Benutzer oder pro Gruppe zu beschränken. Dabei werden zwei Grenzen unterschieden: Die **weiche Grenze** (engl. *soft quota*) darf langfristig nicht überschritten werden. Sie können Benutzern aber einen »Dispositionskredit« einräumen, indem Sie die **harte Grenze** (engl. *hard quota*) auf einen höheren Wert setzen. Dann können sie kurzfristig Platz bis zur harten Grenze belegen, aber innerhalb einer bestimmten Frist muss der belegte Platz wieder unter die weiche Grenze gesenkt werden. Wird die harte Grenze erreicht oder läuft die Schonfrist ab, während mehr Platz belegt ist, als die weiche Grenze erlaubt, schlagen weitere Schreiboperationen fehl.

weiche Grenze

harte Grenze



Es reicht, ganz kurz unter die weiche Grenze abzutauchen; danach dürfen Sie wieder Platz bis zur harten Grenze belegen, und die Schonfrist zählt von vorne.

Kontingente können pro Dateisystem vergeben werden. Sie können also zum Beispiel eine Grenze für die Größe des Eingangspostfachs in `/var/mail` setzen, ohne die Heimatverzeichnisse der Benutzer einzuschränken, oder umgekehrt. Linux unterstützt Kontingente für alle gängigen Linux-Dateisysteme.

Kontingentvergabe



XFS tut, was Kontingente angeht, sein eigenes Ding – aber die aktuellen Linux-Werkzeuge zur Kontingentverwaltung können auch mit XFS umgehen. (Sollten Sie es mit einem ziemlich alten Linux zu tun bekommen, kann es sein, dass Sie statt dessen das Programm `xfs_quota` verwenden müssen.)



Btrfs hat seine eigene Infrastruktur für Kontingente, die sogenannten *quota groups* oder *qgroups*. Im Prinzip können Sie für einzelne Subvolumes Kontingente vergeben, die dann auch hierarchisch verwaltet werden. Das Dateisystem stellt sicher, dass ein Verzeichnis nie mehr Platz verbraucht, als die *quota group* in seinem Subvolume, die *quota group* im Subvolume darüber usw. angibt. Schnappschüsse zählen nicht, solange sie nicht geändert werden, da Schnappschüsse zunächst keinen zusätzlichen Platz verbrauchen. Studieren Sie `btrfs-qgroup(8)`.

## 7.4.2 Kontingentierung für Benutzer (ext und XFS)

Um Kontingente für Benutzer einzurichten, müssen Sie zunächst die Quota-Software installieren (die meisten Distributionen liefern sie mit). Anschließend können Sie die Dateisysteme, für die die Kontingentierung gelten soll, als solche kennzeichnen. Dies erreichen Sie durch die `mount`-Option `usrquota`, die Sie einem eingebundenen Dateisystem wie `/home` ad hoc durch

```
# mount -o remount,usrquota /home
```

mitgeben können. Dauerhaft versehen Sie ein Dateisystem damit durch einen entsprechenden Eintrag in `/etc/fstab`:

```
/dev/hda5 /home ext2 defaults,usrquota 0 2
```

Die Kontingent-Datenbank wird mit

```
# quotacheck -avu
```

neu initialisiert (achten Sie auf allfällige Warnmeldungen), *nachdem* das Dateisystem mit der Option `usrquota` eingebunden wurde. Das Programm `quotacheck` legt im Wurzelverzeichnis des Dateisystems die Datenbank-Datei `aquota.user` an, hier also `/home/aquota.user`. Am besten rufen Sie `quotacheck` auch im laufenden Betrieb periodisch auf, um die Datenbank zu »putzen«.

Danach sollten Sie das Kontingentierungssystem starten, damit die durch `quotacheck` angelegte Bestandsaufnahme bei jeder Dateisystem-Operation aktualisiert wird:

```
# quotaon -avu
```

Ihre Distribution enthält wahrscheinlich ein Init-Skript, das diesen Schritt beim Systemstart automatisch ausführt. Entsprechend wird das Kontingentierungssystem beim Herunterfahren des Rechners durch »`quotaoff -av`« deaktiviert.

Kontingente setzen

Die Kontingente für verschiedene Benutzer können Sie mit dem Kommando `edquota` setzen. `edquota` startet Ihren Lieblingseditor (gemäß der Umgebungsvariablen `EDITOR`) mit einer »Schablone«, in der Sie die weiche und harte Quota für die in Frage kommenden Dateisysteme setzen können:

```
# edquota -u hugo
```

*Kontingente für hugo*

Die Schonfrist in Tagen setzen Sie mit »`edquota -t`«.

Sie können sich auch auf die Kontingente eines »Musterbenutzers« beziehen:

```
# edquota -p tux hugo
```

setzt die Kontingente von `hugo` auf die für `tux` definierten. Dies macht es einfach, neu eingerichteten Benutzern vorgegebene Kontingente zuzuweisen, ohne diese manuell mit `edquota` einstellen zu müssen.

Kontingente abfragen

Benutzer können ihre Kontingente mit dem Kommando `quota` abfragen. »`quota -`

q« gibt eine Kurzmeldung aus, in der nur die Dateisysteme erwähnt werden, bei denen die weiche Grenze überschritten ist. Dieses Kommando eignet sich für Dateien wie `~/.profile`.

Das Kommando `repquota` erzeugt einen tabellarischen Überblick über den Plattenplatzverbrauch verschiedener Benutzer, zusammen mit möglicherweise aktiven Kontingenten:

```
# repquota -a
```

User	Block limits				File limits			
	used	soft	hard	grace	used	soft	hard	grace
root --	166512	0	0		19562	0	0	
tux --	2304	10000	12000		806	1000	2000	
hugo --	1192	5000	6000		389	500	1000	

Die `grace`-Spalte gibt die noch verbleibende Schonfrist an, wenn die weiche Grenze überschritten wurde.

### 7.4.3 Kontingentierung für Gruppen (ext und XFS)

Sie können auch Gruppenkontingente vergeben, die für alle Mitglieder einer Gruppe gemeinsam gelten. Damit diese Beschränkung effektiv wirksam wird, müssen Sie *alle* Gruppen mit einem Kontingent belegen, in denen die entsprechenden Benutzer Mitglied sind, andernfalls könnten sie ihre Kontingentierung durch Gruppenwechsel umgehen.

Gruppenkontingente

Die `mount`-Option für Gruppenkontingente ist `grpquota`, und die Datenbank-Datei heißt `aquota.group`. Um die Gruppenkontingente ansprechen zu können, müssen Sie bei den eben besprochenen Kommandos die Option `-u` durch die Option `-g` ersetzen oder ergänzen. So aktiviert

```
# quotaon -avg
```

jede Form von Kontingenten (Benutzer und Gruppe) auf jedem Dateisystem, während Ihnen

```
$ quota -vg
```

nur die Gruppen-Kontingente für die Gruppen anzeigt, in denen Sie Mitglied sind.

## Übungen



7.7 [2] Richten Sie Plattenplatz-Kontingente wie eben beschrieben ein: ein Dateisystem mit Benutzer-Quota, ein Dateisystem mit Gruppen-Quota und ein Dateisystem mit beiden.

Achten Sie auf die Ausgabe von `quotacheck`, `quotaon` und `quota`.



7.8 [2] Legen Sie für einen Benutzer und für eine Gruppe (in der dieser Benutzer Mitglied ist) jeweils Beschränkungen an.

Wie ändert sich die Ausgabe von `quota`, wenn Sie die weiche bzw. die harte Grenze erreichen. Bekommen Sie eine Warnung angezeigt?

## Kommandos in diesem Kapitel

<b>blkid</b>	Findet Attribute von blockorientierten Geräten und gibt sie aus	blkid(8)	130
<b>dd</b>	„Copy and convert“, kopiert Dateien und Dateisysteme blockweise und macht einfache Konvertierungen	dd(1)	132
<b>debugfs</b>	Dateisystem-Debugger zur Reparatur schlimm verkorkster Dateisysteme. Nur für Gurus!	debugfs(8)	119
<b>dumpe2fs</b>	Gibt interne Strukturen des ext2-Dateisystems aus. Nur für Gurus!	dumpe2fs(8)	119
<b>dumpreiserfs</b>	Gibt interne Strukturen des Reiser-Dateisystems aus. Nur für Gurus!	dumpreiserfs(8)	122
<b>e2fsck</b>	Prüft ext2- und ext3-Dateisysteme auf Konsistenz	e2fsck(8)	117
<b>e2label</b>	Ändert das Label eines ext2/3-Dateisystems	e2label(8)	130
<b>edquota</b>	Erlaubt Eingabe und Ändern von Plattenkontingenten	edquota(8)	134
<b>fsck</b>	Organisiert Dateisystemkonsistenzprüfungen	fsck(8)	111
<b>lsblk</b>	Listet verfügbare Blockgeräte auf	lsblk(8)	131
<b>mkdosfs</b>	Legt FAT-formatierte Dateisysteme an	mkfs.vfat(8)	126
<b>mke2fs</b>	Legt ein ext2- oder ext3-Dateisystem an	mke2fs(8)	116
<b>mkfs</b>	Organisiert das Anlegen von Dateisystemen	mkfs(8)	110
<b>mkfs.vfat</b>	Legt FAT-formatierte Dateisysteme an	mkfs.vfat(8)	126
<b>mkfs.xfs</b>	Legt XFS-formatierte Dateisysteme an	mkfs.xfs(8)	123
<b>mkreiserfs</b>	Legt Reiser-Dateisysteme an	mkreiserfs(8)	122
<b>mkswap</b>	Initialisiert eine Auslagerungs-Partition oder -Datei	mkswap(8)	127
<b>mount</b>	Hängt ein Dateisystem in den Dateibaum ein	mount(8), mount(2)	128
<b>reiserfsck</b>	Prüft ein Reiser-Dateisystem auf Konsistenz	reiserfsck(8)	122
<b>resize_reiserfs</b>	Ändert die Größe eines Reiser-Dateisystems	resize_reiserfs(8)	122
<b>swapoff</b>	Deaktiviert eine Auslagerungs-Partition oder -Datei	swapoff(8)	127
<b>swapon</b>	Aktiviert eine Auslagerungs-Partition oder -Datei	swapon(8)	127
<b>tune2fs</b>	Stellt Parameter von ext2- und ext3-Dateisystemen ein	tune2fs(8)	120, 131
<b>vol_id</b>	Bestimmt Dateisystemtypen und gibt Label und UUID aus	vol_id(8)	130
<b>xfs_info</b>	Entspricht xfs_growfs, aber ändert das Dateisystem nicht	xfs_growfs(8)	124
<b>xfs_mdrestore</b>	Überträgt einen Metadaten-Abzug in ein XFS-Dateisystem	xfs_mdrestore(8)	123
<b>xfs_metadump</b>	Erzeugt Metadaten-Abzüge von XFS-Dateisystemen	xfs_metadump(8)	123
<b>xfs_repair</b>	„Repariert“ XFS-Dateisysteme	xfs_repair(8)	123

## Zusammenfassung

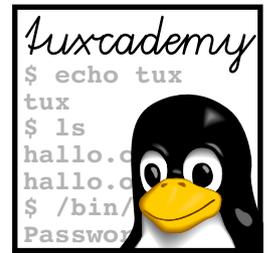
- Nach dem Partitionieren muss auf einer neuen Partition ein Dateisystem angelegt werden, bevor sie verwendet werden kann. Linux hält hierzu das Programm `mkfs` bereit (mit dateisystemspezifischen Hilfsprogrammen, die die eigentliche Arbeit machen).
- Unsauber ausgehängte Dateisysteme können Inkonsistenzen aufweisen. Bemerkte 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.
- Um Plattenplatz-Kontingente einzurichten, müssen Sie Dateisysteme entsprechend einbinden, die Datenbank initialisieren und Quota aktivieren.

- Kontingente werden mit edquota festgelegt und mit quota oder repquota überwacht.

## Literaturverzeichnis

**Quota-Mini-HOWTO** Ralf van Dooren. »Quota mini-HOWTO«, August 2003.  
<http://tldp.org/HOWTO/Quota.html>





# 8

## Linux booten

### Inhalt

8.1	Grundlagen . . . . .	140
8.2	GRUB Legacy . . . . .	143
8.2.1	Grundlagen von GRUB. . . . .	143
8.2.2	Die Konfiguration von GRUB Legacy. . . . .	144
8.2.3	Installation von GRUB Legacy . . . . .	145
8.3	GRUB 2 . . . . .	146
8.3.1	Sicherheitsaspekte . . . . .	147
8.4	Kernelparameter . . . . .	148
8.5	Probleme beim Systemstart . . . . .	150
8.5.1	Fehlersuche . . . . .	150
8.5.2	Typische Probleme . . . . .	150
8.5.3	Rettungssysteme und Live-Distributionen . . . . .	152

### 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

## 8.1 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 6 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.



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« 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

UEFI Secure Boot

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 Betriebssysteme 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.

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 9) oder `systemd` (Kapitel 10), 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



**8.1** [2] Wo auf einer MBR-partitionierten Festplatte darf sich der Bootlader befinden? Warum?

## 8.2 GRUB Legacy

### 8.2.1 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 8.3.1.

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 8.3) setzen.

Die grundlegende Funktionsweise von GRUB Legacy entspricht der in Abschnitt 8.1 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<sup>1</sup>.

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<sup>2</sup>. 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.

## 8.2.2 Die Konfiguration von GRUB Legacy

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
```

<sup>1</sup>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.

<sup>2</sup>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.

### 8.2.3 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 8.4.

`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)
```

## 8.3 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«.

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.)

Konfigurationsdatei

`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 8.2.2 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.

### 8.3.1 Sicherheitsaspekte

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.

Bootparameter?

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



**8.2 [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.



**8.3 [13]** 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?

## 8.4 Kernelparameter

Laufzeitkonfiguration 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 Kernelquellen 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.

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.

**hdx=noprobe** Veranlasst den Kernel, das festplattenartige Gerät `/dev/hdx` (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`

## 8.5 Probleme beim Systemstart

### 8.5.1 Fehlersuche

Normalerweise ist die Lage einfach: Sie schalten den Rechner ein, gehen gemütlich zur Kaffeemaschine (oder auch nicht – siehe Abschnitt 9.1), 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.

### 8.5.2 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 einstäpseln 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 `fdisk` 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 ultimativen Partitionstabellen-Super-GAU's 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 8.4) 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 9. Sie müssten dann aber aus dem Größten heraus sein.

### 8.5.3 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 Netzchnittstelle 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.

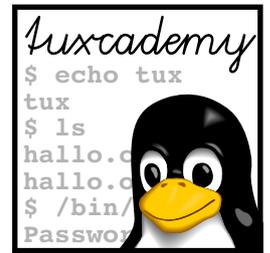
## Kommandos in diesem Kapitel

<code>dmesg</code>	Gibt den Inhalt des Kernel-Nachrichtenpuffers aus	<code>dmesg(8)</code>	150
<code>grub-md5-crypt</code>	Bestimmt MD5-verschlüsselte Kennwörter für GRUB Legacy	<code>grub-md5-crypt(8)</code>	147

## 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.





# 9

## System-V-Init und der Init-Prozess

### Inhalt

9.1	Der Init-Prozess . . . . .	156
9.2	System-V-Init . . . . .	156
9.3	Upstart . . . . .	163
9.4	Herunterfahren des Systems . . . . .	165

### 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 8)

## 9.1 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`.



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.

## 9.2 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.



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 10). Eine andere ist Upstart von Scott James Remnant (siehe Abschnitt 9.3).

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.

Die Syntax von `/etc/inittab` (Bild 9.1) 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 9.1:** 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 50 für tty50. 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 tauf frisch 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 9.4).

 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



**9.1 [2]** Können Sie herausfinden, wo Ihre Distribution die Skripte ablegt, die das Bootskript ausführt?



**9.2 [!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 Netzwerkservers
- 3 Mehrbenutzermodus mit Netzwerkservers
- 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 `5`) 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

 **9.3** [!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.

 **9.4** [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 `rcr.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/rcr.d`) ausgeführt. Dabei bekommen Links mit `»K«` am Anfang den Parameter `stop` und alle mit `»S«` den Parameter `start` übergeben.

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.

Dienste konfigurieren



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



**9.5** [!2] Was müssen Sie tun, damit der `syslog`-Dienst seine Konfiguration neu einliest?



**9.6** [1] Wie können Sie bequem die momentane Runlevel-Konfiguration überprüfen?



**9.7** [!2] Entfernen Sie den Dienst `cron` aus Runlevel 2.

Einbenutzermodus **Der Einbenutzermodus** Im **Einbenutzermodus** (Runlevel 5) kann nur der Systemadministrator an der Konsole arbeiten. Die Möglichkeit, mit `[Alt]+Funktions`-taste 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 5 gestartet werden und für einen geordneten (Mehrbenutzer-)Systembetrieb notwendig sind, so dass es nicht klug ist, aus dem Runlevel 5 wieder in einen Mehrbenutzer-Runlevel zu wechseln.

## Übungen



**9.8** [!1] Versetzen Sie das System in den Einbenutzermodus (*Tipp*: `telinit`). Was müssen Sie machen, um tatsächlich in den Einbenutzermodus zu gelangen?



**9.9 [1]** Überzeugen Sie sich davon, dass Sie im Einbenutzermodus wirklich der einzige Benutzer auf dem Rechner sind und dass keine Hintergrundprozesse laufen.

## 9.3 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 10) 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 röhelnder Festplatte im Hintergrund). Diese Arbeit beruhte auf System-V-Init und nicht Upstart.

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 9.2.

```

# 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 9.2:** 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 9.2 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.

## 9.4 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 9.1).

💡 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



**9.10** [!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)?



**9.11** [1] Was passiert, wenn Sie (als root) wall den Namen einer nicht existierenden Datei als Parameter übergeben?



**9.12** [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.)

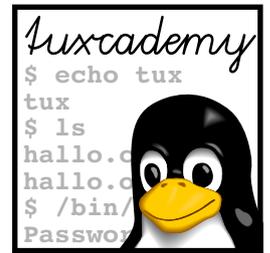
## Kommandos in diesem Kapitel

<b>chkconfig</b>	Schaltet Systemdienste ein oder aus (SUSE, Red Hat)	chkconfig(8)	161
<b>halt</b>	Fährt das System herunter	halt(8)	166
<b>initctl</b>	Zentrales Steuerungsprogramm für Upstart	initctl(8)	165
<b>insserv</b>	Aktiviert oder deaktiviert Init-Skripte (SUSE)	insserv(8)	161
<b>mesg</b>	Schaltet wall-Nachrichten für ein Terminal ein oder aus	mesg(1)	167
<b>reboot</b>	Startet den Rechner neu	reboot(8)	166
<b>runlevel</b>	Zeigt den vorigen und den aktuellen Runlevel an	runlevel(8)	160
<b>shutdown</b>	Fährt das System herunter oder startet es neu, mit Verzögerung und Warnung an angemeldete Benutzer	shutdown(8)	165
<b>update-rc.d</b>	Installiert und entfernt System-V-Initskript-Links (Debian)	update-rc.d(8)	161
<b>wall</b>	Schreibt eine Nachricht auf die Terminals aller angemeldeten Benutzer	wall(1)	167

## 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.





# 10

## Systemd

### Inhalt

10.1 Überblick. . . . .	172
10.2 Unit-Dateien . . . . .	174
10.3 Typen von Units . . . . .	178
10.4 Abhängigkeiten . . . . .	179
10.5 Ziele . . . . .	180
10.6 Das Kommando systemctl . . . . .	183
10.7 Installation von Units . . . . .	186

### 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 8)
- Wissen über System-V-Init (Kapitel 9)

## 10.1 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     Systemd verwendet »Units« als Abstraktion für zu verwaltende Systemaspekte  
Ziele     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 distributionspezifischen Shellskripten im Runlevel S erledigt wird. Ihre Verwendung kann den Systemstart deutlich beschleunigen und trägt in diesem Bereich zu einer distributionsübergreifenden Standardisierung bei.

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`.

Kompatibilität

Sie können das Kommando `systemctl` verwenden, um mit einem laufenden `systemd` zu interagieren, etwa um Dienste gezielt zu starten oder anzuhalten:

`systemctl`

```
# 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.



`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.

Transaktionen

## Übungen



**10.1** [!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.

## 10.2 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 10.1. Typisch ist die Einteilung in Abschnitte, die mit einem Titel in eckigen Klammern anfangen<sup>1</sup>. 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.)

<sup>1</sup>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 10.1:** Eine systemd-Unit-Datei: console-getty.service

 Optionen, die nicht in der Dokumentation stehen, werden von systemd mit einer Warnung angemerkert 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<sup>2</sup>. 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

<code>/etc/systemd/system</code>	<i>Lokale Konfiguration</i>
<code>/run/systemd/system</code>	<i>Dynamisch erzeugte Unit"-Dateien</i>
<code>/lib/systemd/system</code>	<i>Unit"-Dateien für Distributionspakete</i>

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.

<sup>2</sup>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 Dienstenamen 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 Protokollschemaschemata `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



**10.2 [!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?

## 10.3 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 10.2 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 10.5.
- .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



**10.3** [!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.

## 10.4 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



**10.4** [!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`?



**10.5** [2] Finden Sie in Ihrer Systemkonfiguration Beispiele für `Requires`-, `Wants`- und `Conflicts`-Abhängigkeiten, mit und ohne korrespondierende `Before`- oder `After`-Abhängigkeiten.

## 10.5 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 10.1:** 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 <b>Strg</b> + <b>Alt</b> + <b>Entf</b> 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 ( <i>nicht</i> 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 10.2:** 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 10.1

Im Interesse der Rückwärtskompatibilität zu System-V-Init definiert systemd einige Ziele, die mit den klassischen Runlevels korrespondieren. Betrachten Sie hierzu Tabelle 10.2.

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

 **10.6** [!2] Von welchen anderen Units hängt das multi-user.target ab? Hängen diese Units wiederum von anderen Units ab?

 **10.7** [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.)

 **10.8** [2] Starten Sie Ihr System mit »systemctl reboot« neu und dann anschließend nochmal mit shutdown. Würdigen Sie den Unterschied.

## 10.6 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



**10.9** [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.



**10.10** [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`?



**10.11** [1] Was ist der Vorteil von

```
# systemctl kill example.service
```

gegenüber

```
# killall example
```

(oder »`kill example`«)?

## 10.7 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).

Mit »`systemctl disable`« können Sie eine Unit wieder deaktivieren. Genau wie Unit deaktivieren 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.

Mit dem Kommando »`systemctl mask`« können Sie eine Unit »maskieren«, also Unit »maskieren« 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



**10.12** [!2] Was passiert, wenn Sie »systemctl disable cups« ausführen? (Betrachten Sie die ausgegebenen Kommandos.) Aktivieren Sie den Dienst anschließend wieder.



**10.13** [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)** 173, 183

## 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.

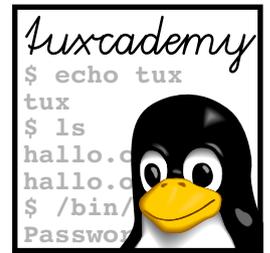
## Literaturverzeichnis

**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/>



# 11

## Dynamische Bibliotheken

### Inhalt

11.1 Bibliotheken. . . . .	190
11.2 Dynamische Bibliotheken in der Praxis . . . . .	192
11.3 Dynamische Bibliotheken installieren und finden . . . . .	195
11.4 Dynamische Bibliotheken in mehreren Versionen . . . . .	196

### Lernziele

- Dynamische Bibliotheken (*shared libraries*) identifizieren können
- Die typischen Speicherorte für dynamische Bibliotheken kennen
- Dynamische Bibliotheken verwalten können

### Vorkenntnisse

- Solide Kenntnisse der Linux-Kommandozeile
- Programmier-Erfahrung ist nützlich (aber nicht erforderlich)

## 11.1 Bibliotheken

Ein wichtiger Grundsatz beim Programmieren ist, das Rad nicht dauernd neu zu erfinden. Die meisten Programme haben eine Menge Code gemeinsam – von grundlegenden Funktionen wie dem Umgang mit Dateien und Zeichenketten bis zu Speziellerem wie der Verwaltung von Datums- und Zeitangaben. Es ist weder sinnvoll, diesen Code immer wieder neu zu schreiben, noch ist es vernünftig, ihn fest in jedes Programm zu integrieren. Ersteres, weil es ein Haufen Arbeit ist und dieselben Fehler sonst wieder und wieder korrigiert werden müßten; letzteres, weil es einen Haufen Platz auf der Platte kostet und extrem unbequem ist, *wenn* tatsächlich mal ein Fehler korrigiert wurde.

Die Antwort auf den ersten Einwand – der Code soll nicht immer wieder neu geschrieben werden müssen – sind Programmbibliotheken. Sie dienen dazu, oft gebrauchte Funktionalität zu sammeln, mit einer möglichst standardisierten und gut dokumentierten Schnittstelle zu versehen und anderen Programmen zur Verfügung zu stellen. Zentral für die Funktion eines Linux-Systems ist zum Beispiel die `libc`, die alle möglichen lebensnotwendigen Funktionen für Programme bündelt, die in der Programmiersprache C geschrieben sind.



In der Programmiersprache C geschrieben sind zumindest mittelbar praktisch alle Programme auf einem Linux-System: Sehr vieles ist direkt in C geschrieben, einiges ist in von C abgeleiteten Programmiersprachen wie C++ geschrieben, die gängigerweise auch die `libc` verwenden, und sogar Shellskripte und Programme in Programmiersprachen wie `awk`, Python oder Perl bauen auf der Shell oder Programmen wie `awk`, `python` und `perl` auf – und die sind in aller Regel C-Programme ...

Außer der `libc` gibt es auf einem typischen Linux-System einige hundert bis tausend andere Bibliotheken, die von mathematischen Funktionen über den Umgang mit verschiedenen Datenformaten bis hin zur Realisierung grafischer Oberflächen alle möglichen Anforderungen abdecken.

Wichtig sind Bibliotheken zunächst beim Programmieren: Im einfachsten Fall dient ein »Linker« dazu, den »Objektcode« – also die Ausgabe des Sprachübersetzers (engl. *compiler*), der ein zum Beispiel in C geschriebenes Programm in Befehle übersetzt, die der Prozessor ausführen kann – des eigentlichen Programms mit dem Objektcode der von diesem Programm verwendeten Bibliotheken zu einem »Maschinenprogramm« kombiniert, das Sie anschließend aufrufen können.



Betrachten Sie zur Illustration das folgende kleine C-Programm:

```
/* sqrttest.c */
#include <math.h>
#include <stdio.h>
int main(void)
{
    printf("%g\n", sin(3.141592654/2.0));
    return 0;
}
```

(Es dient dazu, die drängende Frage nach dem Wert von  $\sin(\pi/2)$  zu beantworten.) Der C-Compiler macht daraus ein Assembler-Programm, das ungefähr so anfängt:

```
; sqrttest.s
    .file "sqrttest.c"
    .section .rodata
.LC1:
    .string "%g\n"
    .align 8
```

```
.LC0:
    .long 1414677840
    .long 1073291771
    .text
.globl main
    .type main, @function
main:
    leal 4(%esp), %ecx
    andl $-16, %esp
    pushl -4(%ecx)
    pushl %ebp
<<<<<<
```

und der Assembler wiederum produziert eine Objektdatei, die »lesbar« dargestellt in etwa so losgeht:

```
7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
01 00 03 00 01 00 00 00 00 00 00 00 00 00 00 00
14 01 00 00 00 00 00 00 34 00 00 00 00 00 28 00
0b 00 08 00 8d 4c 24 04 83 e4 f0 ff 71 fc 55 89
e5 51 83 ec 14 dd 05 08 00 00 00 dd 1c 24 e8 fc
ff ff ff dd 5c 24 04 c7 04 24 00 00 00 00 e8 fc
<<<<<<
```

(insgesamt 952 Bytes, wenn Sie es wissen müssen). Diese Datei muss dann noch mit Hilfe des Linkers mit den Bibliotheken `libc` (für die Funktion `printf()`) und `libm` (für die Funktion `sin()`) zu einem Maschinenprogramm zusammengesetzt werden – erfreulicherweise geht das aber alles mit einem einzigen Kommando,

```
$ gcc -static sqrttest.c -lm -o sqrttest-static
```

Dieses Programm ist auf dem System des Autors lächerliche 560.608 Bytes kurz.

In diesem Fall sind Programm und Bibliotheken fest miteinander verbunden. Die Bibliotheken ersparen uns die lästige Wiederholung von Code, der für mehrere Programme interessant ist, aber wenn ein Problem in einer Bibliothek gefunden wird, müssen wir *alle* Programme neu »linken«, die die betreffende Bibliothek verwenden. Der Umstand, dass ein triviales Programm ein halbes Megabyte lang ist, ist auch irgendwie weniger als befriedigend.

Vorteile von Bibliotheken

An dieser Stelle treten dynamische Bibliotheken auf den Plan. Die Grundidee dahinter ist, das eigentliche Programm und die Bibliotheken nicht beim Linken dauerhaft miteinander zu verbinden, sondern erst wenn das Programm tatsächlich gestartet wird. Programm und Bibliotheken bleiben also voneinander unabhängig. Dies hat die folgenden Vorteile:

Dynamische Bibliotheken

- Solange die Schnittstelle der Bibliothek gleich bleibt (also die »Signaturen« der beteiligten Funktionen – die Anzahl und Typen der Aufrufparameter und Ergebniswerte – sich nicht ändern), kann die Bibliothek leicht ersetzt werden, ohne dass man die Programme neu linkt, die sie benutzen. Wenn ein Programm gestartet wird, holt es sich automatisch die neue Version der Bibliothek.
- Wenn  $n$  Programme die Bibliothek benutzen, braucht sie nicht  $n$ -mal Platz auf der Platte (als Bestandteil jedes dieser Programme). Eine Datei reicht.
- Tatsächlich reicht es in der Regel sogar aus, die Bibliothek nur einmal im Speicher zu haben. Alle Prozesse können sich eine Kopie teilen. Noch eine

Ersparnis! Deswegen spricht man übrigens auch von *shared libraries* – gemeinsam benutzbaren Bibliotheken.

Bezahlen tun Sie dafür übrigens mit Zeit: Einerseits dauert es einen Tick länger, ein Programm zu starten, das dynamische Bibliotheken benutzt, da die Verbindung zwischen dem eigentlichen Programm und den Bibliotheken beim Start stattfindet statt bei der Übersetzung zu einem Maschinenprogramm. Zum anderen kann es – je nach Prozessor und Programmiermodell – sein, dass dynamische Bibliotheken mit einem Laufzeitverlust gegenüber »statischen«, also fest mit dem Programm verbundenen Bibliotheken verbunden sind. Dieser bewegt sich jedoch normalerweise höchstens im kleinen einstelligen Prozentbereich und wird durch die oben genannten Vorteile mehr als aufgewogen.



Unser Beispiel von oben könnten Sie mit

```
$ gcc sqrttest.c -lm -o sqrttest-dynamic
```

gegen die dynamischen Bibliotheken linken. Das Ergebnis ist 6510 Bytes lang – schon ein Unterschied zu über 500.000!



Vielleicht finden Sie, dass Plattenplatz heute so billig ist, dass man über ein halbes Megabyte hier oder da keinen Schlaf verlieren sollte. Dynamische Bibliotheken wurden für Unix in den 1980er Jahren eingeführt, als Plattenplatz noch in Megabytes gemessen wurde und nicht in Terabytes; die Bibliotheken selbst waren zwar nicht ganz so groß, aber damals hat man den Unterschied noch viel deutlicher gemerkt. Im übrigen sind die anderen genannten Vorteile eher noch wichtiger.

Ehrenrettung von statisch gelinkten Programmen

Zur Ehrenrettung von statisch gelinkten Programmen sollten wir erwähnen, dass sie auch ihren Platz haben. Zum einen funktionieren sie ziemlich verlässlich auch dann, wenn das System ansonsten ziemlich am Boden liegt – sollte es Ihnen also gelingen, irgendwie die dynamische libc zu schrotten, dann werden Sie sich freuen, auch noch eine statisch gelinkte Version der busybox oder sash auf Ihrem System zu haben, denn für Reparaturarbeiten sind die bash und ihre Freunde wie cp und tar dann nicht mehr zu gebrauchen. (Die Alternative wäre natürlich Booten des Rettungssystems, aber das ist für Warmduscher.) Zum anderen sind statisch gelinkte Programme zwar riesengroß und unbequem, aber man weiß, was man hat<sup>1</sup> – wenn Sie also ein komplexes Programm als Maschinencode in Umlauf bringen wollen, dann machen Sie sich durch statisches Linken unabhängig von der Bibliotheksausstattung der Systeme, auf denen das Programm später laufen soll. (Das kann wiederum andere Probleme implizieren, aber eine ausführliche Diskussion würde hier zu weit führen.)

LSB



Ein großer Teil der Bemühungen um die LSB (*Linux Standard Base*), ein Übereinkommen zwischen verschiedenen Distributoren darüber, welche Umgebung für im Maschinencode verteilte Software von Drittanbietern (denken Sie an SAP, Oracle und Spiele) zur Verfügung stehen sollte, befasst sich damit, die Versionen und den Inhalt von wichtigen dynamischen Bibliotheken vorzugeben.

## 11.2 Dynamische Bibliotheken in der Praxis

Wie wir schon gesagt haben, ist der überwiegende Teil der Maschinenprogramme auf einem typischen Linux-System dynamisch gelinkt. Vergewissern können Sie sich mit dem Programm file:

<sup>1</sup>Entschuldigung an den Persil-Mann.

```
$ file sqrttest-dynamic
sqrttest-dynamic: ELF 32-bit LSB executable, Intel 80386, ▷
< version 1 (SYSV), dynamically linked (uses shared libs),▷
< for GNU/Linux 2.6.8, not stripped
```

Zum Vergleich:

```
$ file sqrttest-static
sqrttest-static: ELF 32-bit LSB executable, Intel 80386,▷
< version 1 (SYSV), statically linked, for GNU/Linux 2.6.8,▷
< not stripped
```



Das »not stripped« bezieht sich darauf, dass unser kleines (?) C-Programm noch Informationen enthält, die die Fehlersuche erleichtern sollen. Sie können es auf gerade knapp unter 500 kB abspecken, indem Sie »strip sqrttest-static« aufrufen.

Genau herausfinden, welche Bibliotheken sqrttest-dynamic benötigt, können Sie mit dem Kommando ldd:

```
$ ldd sqrttest-dynamic
linux-gate.so.1 => (0xb7f51000)
libm.so.6 => /lib/i686/cmov/libm.so.6 (0xb7f0b000)
libc.so.6 => /lib/i686/cmov/libc.so.6 (0xb7db0000)
/lib/ld-linux.so.2 (0xb7f52000)
```

Die Ausgabe nennt Ihnen die Namen der Bibliotheken nebst den Namen der Dateien, in denen sie gefunden wurden – falls möglich (ansonsten ist der Dateiname leer).



libm und libc haben wir schon diskutiert. /lib/ld-linux.so.2 ist der »dynamische Linker«, also das Programm, das beim Programmstart dafür sorgt, dass das eigentliche Programm mit den dynamischen Bibliotheken verbunden wird (wir kommen gleich darauf zurück).



Was aber ist linux-gate.so.1 und wofür ist es gut? Mit dieser Frage können Sie hervorragend testen, wieviel Ihre Kumpels *wirklich* über Linux wissen. Auffällig ist zuerst, dass ldd keine Datei gefunden hat, die mit dieser Bibliothek korrespondiert. Eigentlich sollte das Programm also gar nicht laufen. Wenn Sie suchen, werden Sie sogar finden, dass es auf dem System *überhaupt keine Datei gibt, die so heißt*. Das macht aber nichts, da der Kernel dafür sorgt, dass jedes Programm diese »virtuelle dynamische Bibliothek« einbindet. Gebraucht wird linux-gate.so.1, um auf moderneren x86-Prozessoren (das heißt heute »allen«) Systemaufrufe zu beschleunigen. Die Details sind zu unappetitlich, um sie hier zu erklären; lesen Sie [Pet08], aber nicht unmittelbar vor oder nach dem Essen.



Auf Intel- und AMD-basierten 64-Bit-Systemen erfüllt die »Bibliothek« linux-vdso.so.1 denselben Zweck wie linux-gate.so.1 auf 32-Bit-Systemen. Lassen Sie sich also nicht verwirren.

Typische Plätze, an denen Linux-Distributoren Bibliotheken (statische oder dynamische) unterbringen, sind die Verzeichnisse /lib und /usr/lib. Wie üblich ist ersteres eher für Bibliotheken gedacht, die sofort beim Systemstart benötigt werden, und letzteres für solche, die erst zum Tragen kommen, wenn alle Dateisysteme eingehängt sind.

Woran erkennen Sie nun eine dynamische Bibliothek? Zuerst mal am Dateinamen – die Dateinamen dynamischer Bibliotheken enden typischerweise auf .so, in

Bibliotheken: wo?

Dynamische Bibliotheken erkennen

der Regel noch gefolgt von einer Versionsnummer. (Die Dateinamen statischer Bibliotheken hören mit `.a` auf, und eine Versionsnummer kommt da nicht ins Spiel.) Wenn Sie es ganz genau wissen wollen, können Sie auch hier das Kommando `file` bemühen:

```
$ file /lib/libc-2.7.so
/lib/libc-2.7.so: ELF 32-bit LSB shared object, Intel 80386,▷
< version 1 (SYSV), dynamically linked (uses shared libs),▷
< for GNU/Linux 2.6.8, stripped
```

Schauen Sie genau hin – hier steht

```
ELF 32-bit LSB shared object
```

statt

```
ELF 32-bit LSB executable
```

– ein feiner Unterschied!



Bei statischen Bibliotheken passiert das Folgende:

```
$ file /usr/lib/libc.a
/usr/lib/libc.a: current ar archive
```

Man muss sich schon ein bisschen auskennen, um daraus den Schluss »Bibliothek!« zu ziehen – schauen Sie mal in `ar(1)`.

Dateinamen dynamischer Bibliotheken

Im »wirklichen Leben« kann der Dateiname einer dynamischen Bibliothek ungefähr so aussehen:

```
$ ls -l /usr/lib/libcurl.*
lrwxrwxrwx root root 16 Jan 22 01:23 /usr/lib/libcurl.so.4▷
< -> libcurl.so.4.1.0
-rw-r--r-- root root 271364 Dec 27 14:33 /usr/lib/libcurl.so.4.1.0
```

Das symbolische Link dient dazu, die Verbindung zwischen konkreten Programmen und der Bibliothek herzustellen. Ein Programm fordert nämlich immer eine bestimmte Version einer Bibliothek an – in unserem Beispiel etwa `libcurl.so.4`. Es hängt vom System ab, ob dies auf `/usr/lib/libcurl.so.4.1.0` oder `/usr/lib/libcurl.so.4.2.1` herausläuft, je nachdem, was installiert ist. (Die zugrundeliegende Annahme ist, dass alle Dateien, die sich als `libcurl.so.4` ausgeben, dieselbe Schnittstelle implementieren.) Mehr über Versionen dynamischer Bibliotheken steht in Abschnitt 11.4.



Übrigens können auch dynamische Bibliotheken von anderen dynamischen Bibliotheken abhängen. Informationen darüber verschafft Ihnen (natürlich) `ldd`:

```
$ ldd /usr/lib/libcurl.so.4.1.0
linux-gate.so.1 => (0xb7f6f000)
libidn.so.11 => /usr/lib/libidn.so.11 (0xb7edb000)
libssh2.so.1 => /usr/lib/libssh2.so.1 (0xb7eba000)
liblber-2.4.so.2 => /usr/lib/liblber-2.4.so.2 (0xb7eac000)
<<<<<<
```

(Was Sie zu sehen bekommen, wenn Sie `ldd` auf ein *Programm* anwenden, ist der transitive Abschluss über alle dynamische Bibliotheken, von denen das Programm selbst zusammen mit allen seinen dynamischen Bibliotheken abhängt.)

## Übungen



**11.1** [!1] Finden Sie in Ihrem System ein statisch gelinktes Programm (außer `sqrttest-static`).



**11.2** [1] Was sagt `ldd`, wenn Sie es auf ein statisch gelinktes Programm anwenden?



**11.3** [2] (Für Shell-Programmierer.) Welches Programm in `/usr/bin` auf Ihrem System ist gegen die meisten dynamischen Bibliotheken gelinkt?

## 11.3 Dynamische Bibliotheken installieren und finden

Wenn Sie ein Programm starten, muss der dynamische Linker `ld-linux.so` die nötigen Bibliotheken finden und laden. Im Programm selbst steht aber nur der Name der Bibliothek, nicht der Name der Datei, in der sie steht. Um zu vermeiden, dass der dynamische Linker das komplette Dateisystem (oder auch nur die Verzeichnisse `/lib` und `/usr/lib`) absuchen muss, steht in der Datei `/etc/ld.so.cache` ein »Index« aller bekannten dynamischen Bibliotheken. Der dynamische Linker findet nur diejenigen Bibliotheken, die in diesem Index verzeichnet sind.

Erzeugt wird der Index mit dem Programm `ldconfig`, das alle in `/etc/ld.so.conf` aufgeführten Verzeichnisse sowie die beiden Standard-Verzeichnisse `/lib` und `/usr/lib` nach Bibliotheken durchsucht. Alle gefundenen Bibliotheken werden in den Index übernommen. Außerdem kümmert sich `ldconfig` darum, die Hauptversionsnummer-Links für Bibliotheksdateien anzulegen.



`/lib` und `/usr/lib` werden immer nach Bibliotheken durchsucht, egal was in `/etc/ld.so.conf` steht.

Den aktuellen Zustand des Index können Sie sich mit

```
# ldconfig -p
909 libs found in cache `/etc/ld.so.cache'
  libzvt.so.2 (libc6) => /opt/gnome/lib/libzvt.so.2
  libz.so.1 (libc6) => /lib/libz.so.1
  libz.so (libc6) => /usr/lib/libz.so
<<<<<
```

anzeigen lassen. Dadurch wird kein neuer Index erzeugt, sondern nur der Inhalt von `ld.so.cache` in lesbarer Form ausgegeben. Findet sich eine Bibliothek nicht wie gewünscht hierin, so kann sie vom Linker auch nicht gefunden werden.

Wenn Sie eine neue dynamische Bibliothek verwenden wollen, die in einem der Standardverzeichnisse steht – außer `/lib` und `/usr/lib` typischerweise auch `/usr/local/lib` –, dann genügt ein einfacher Aufruf von

```
# ldconfig
```

dafür, `/etc/ld.so.cache` auf den aktuellen Stand zu bringen. Wenn Ihre Bibliothek nicht in einem der in `/etc/ld.so.conf` genannten Verzeichnisse steht, müssen Sie das Verzeichnis vorher noch dort eintragen.



Normalerweise sollte das Installationswerkzeug Ihrer Distribution sich darum kümmern, dass `ldconfig` aufgerufen wird, wenn Sie Pakete mit dynamischen Bibliotheken installieren. Der vorige Absatz gilt also vor allem dann, wenn Sie selbstgeschriebene Bibliotheken installieren wollen oder solche aus Softwarepaketen, die Sie vom Quellcode aus übersetzt haben.

Dynamischer Linker

`/etc/ld.so.cache`

`ldconfig`

`/etc/ld.so.conf`

Index-Inhalt

Neue dynamische Bibliotheken

LD\_LIBRARY\_PATH Wenn Sie keine Administrator-Rechte haben und darum `ldconfig` nicht aufrufen können, können Sie trotzdem Ihre eigenen dynamischen Bibliotheken verwenden. Der dynamische Linker durchsucht außer dem Cache in `/etc/ld.so.cache` nämlich auch die Verzeichnisse, die in der Umgebungsvariablen `LD_LIBRARY_PATH` aufgezählt sind. Die Syntax entspricht dabei der Variablen `PATH` – Verzeichnisnamen werden durch Doppelpunkte voneinander getrennt:

```
$ export LD_LIBRARY_PATH=$HOME/lib:/opt/bla/lib
```



Der dynamische Linker durchsucht erst den Inhalt der Verzeichnisse in `LD_LIBRARY_PATH` (falls vorhanden), dann den Inhalt von `/etc/ld.so.cache` und dann den Inhalt der Verzeichnisse `/lib` und `/usr/lib` (in dieser Reihenfolge – dies ist nur ein Sicherheitsnetz für den Fall, dass `/etc/ld.so.cache` nicht existiert). Bibliotheken, die in einem Verzeichnis in `LD_LIBRARY_PATH` stehen und so heißen wie »offizielle« Bibliotheken, werden also zuerst gefunden.



Die Dokumentation des dynamischen Linkers (`ld-linux.so(8)`) enthält viele interessante und spannende Optionen, mit denen Sie fremdartige und wundervolle Dinge tun können.

## Übungen



**11.4** [!2] Die Verzeichnisse in `LD_LIBRARY_PATH` werden nicht beachtet, wenn das zu startende Programm den Set-UID- oder Set-GID-Mechanismus (siehe Abschnitt 3.5) verwendet. Warum?

## 11.4 Dynamische Bibliotheken in mehreren Versionen

DLL-Hölle Anwender von Microsoft Windows kennen das Problem der »DLL-Hölle«<sup>2</sup>: Viele Softwarepakete kommen mit ihren eigenen Versionen wichtiger Systembibliotheken und schrecken nicht davor zurück, etwa schon vorhandene Versionen dieser Bibliotheken rücksichtslos mit ihren eigenen zu überschreiben. Wenn die überschriebene Version natürlich genau die war, die ein anderes Softwarepaket eingeschleppt hat, ist das Chaos nicht weit.

Software von Drittherstellern Unter Linux ist dieses Problem weitaus weniger ausgeprägt, da von Anfang an weniger Software von Drittherstellern als Maschinencode ausgeliefert wird. Binäre Software bekommen Sie in erster Linie von Ihrem Distributor, und andere Sachen übersetzen Sie aus dem Quellcode, wobei, soweit irgend möglich, auf die im System bereits vorhandenen Bibliotheken zurückgegriffen wird. Probleme ergeben sich viel eher daraus, dass die Linux-Softwarelandschaft sich ziemlich rapide weiterentwickelt und das natürlich auch für Bibliotheken gilt.

Versionsnummern Dynamische Bibliotheken bekommen aus diesem Grund Versionsnummern, die typischerweise bis zu dreiteilig sein können. Zum Beispiel haben wir vorhin über `libcurl.4.1.0` geredet. Diese Versionsnummern funktionieren wie folgt:

Hauptversionsnummer

- Die erste Zahl hinter dem Bibliotheksnamen ist die Hauptversionsnummer (engl. *major version number*). Sie sollte immer dann erhöht werden, wenn die Programmierschnittstelle, die die Bibliothek anderen Programmen anbietet, sich inkompatibel ändert. Zum Beispiel könnte eine Funktion, die bisher zwei Argumente erfordert hat, plötzlich drei benötigen, oder ein Argument, das bisher eine Ganzzahl war, könnte ein Zeiger auf eine Zeichenkette werden.



Es steht außer Zweifel, dass ein verantwortungsvoller Entwickler solche Änderungen, wenn irgend möglich, vermeiden wird, denn sie bedeuten, dass die Programme, die die Bibliothek verwenden, umgeschrieben werden müssen.

<sup>2</sup>DLLs oder *dynamically loadable libraries* sind das Windows-Pendant der dynamischen Bibliotheken von Linux.

- Die zweite Zahl (hinter dem zweiten Punkt) ist die Unterversionsnummer (engl. *minor version number*). Sie wird erhöht, wenn die Programmierschnittstelle erweitert wird, ohne dass sich an den existierenden Aufrufen etwas ändert. Zum Beispiel könnte eine ganz neue Funktion dazukommen. Ältere Programme können die Bibliothek trotzdem weiter verwenden, denn alles, was sie brauchen, ist nach wie vor identisch vorhanden; natürlich funktionieren neuere Programme, die die neu hinzugekommenen Aufrufe verwenden, nicht mit älteren Versionen der Bibliothek. Unterversionsnummer
- Die dritte Zahl (am Ende) ist der »Patchlevel«. Sie wird erhöht, wenn sich an der *Implementierung* der Bibliothek etwas ändert, das keinen Einfluss auf die Programmierschnittstelle hat. Zum Beispiel könnte ein Fehler in der Implementierung einer Funktion behoben werden. Die Tatsache, dass die Bibliothek dynamisch gelinkt wird, macht es möglich, einfach die fehlerhafte Bibliothek durch die korrigierte zu ersetzen – alle danach gestarteten Programme holen sich automatisch die neue Version. Patchlevel



Dass eine fehlerhafte Bibliothek durch eine neue ersetzt wird, hat keinen Einfluss auf Programme, die mit der fehlerhaften Bibliothek *schon laufen* – etwa Daemons. Solche Programme müssen Sie im Zweifel manuell neu starten, damit sie in den Genuss der Verbesserungen kommen.

Daemons



Leidtragende in diesem Verbesserungsprozess sind natürlich Programme, die sich auf die *Anwesenheit* des Fehlers in der Implementierung verlassen. Mitunter kommt es vor, dass ein Fehler sich in der Entwicklergemeinde herumspricht und die Programmierer ihn entweder ausnutzen (falls er irgendeinen positiven Nebeneffekt hat – sowas kommt vor) oder auf eine Weise um ihn herum programmieren, die nicht in Betracht zieht, dass der Fehler eines Tages nicht mehr vorhanden sein könnte<sup>3</sup>. In so einer Situation bedeutet eine Korrektur natürlich eine inkompatible Änderung und sollte sich im schlimmsten Fall sogar auf die Hauptversionsnummer auswirken.

Programme verlangen immer eine bestimmte Hauptversion einer Bibliothek, etwa `libcurl.so.4`, die über das symbolische Link dann auf eine konkrete Datei, zum Beispiel `/usr/lib/libcurl.so.4.1.0`, abgebildet wird.

Dadurch ist es ohne weiteres möglich, gleichzeitig Programme im System zu haben, die verschiedene (Haupt-)Versionen derselben Bibliothek verwenden. Sie können einfach die verschiedenen Bibliotheksdateien installiert lassen und dem dynamischen Linker die Arbeit aufbürden, für jedes Programm die richtige zu finden.

Mehrere Versionen gleichzeitig



Der LSB-Standard schreibt bestimmte Hauptversionen gängiger Bibliotheken vor und standardisiert jeweils eine Programmierschnittstelle für die betreffende Bibliothek. LSB-konforme Distributionen müssen diese Bibliotheken anbieten, aber diese Bibliotheken müssen nicht diejenigen sein, die der ganze Rest des Systems verwendet. Es ist also ohne weiteres möglich, eine Distribution aus der Sicht der Bibliotheksunterstützung LSB-konform zu machen, indem ein Satz der verlangten Bibliotheken in einem anderen Verzeichnis untergebracht wird, das LSB-konformen Programmen über Tricks mit dem dynamischen Linker (`LD_LIBRARY_PATH` lässt grüßen) anstatt der normalen Systemverzeichnisse für Bibliotheken untergeschoben wird.

<sup>3</sup>Der Autor dieser Zeilen hatte es vor langer Zeit mit einem C-Compiler zu tun, der bei Gleitkommandivisionen den Kehrwert des eigentlich richtigen Ergebnisses berechnete. (Fragen Sie nicht.) In einem Programmpaket, das ein Kollege geschrieben hatte, kam an strategischer Stelle genau eine solche Division vor, bei der (unkommentiert und längst vergessen) deswegen Dividend und Divisor vertauscht angegeben waren. Das nach dem Update auf einen korrigierten C-Compiler als Ursache diverser gruseliger Folgefehler zu identifizieren kostete uns ein paar Tage.



Dieselbe Strategie eignet sich natürlich auch für »gewöhnliche« Software von Drittanbietern, die sich nicht auf die LSB abstützt. Solange die Schnittstelle der libc zum Kernel sich nicht inkompatibel ändert (was die Kernel-Entwickler zum Glück tunlichst zu vermeiden suchen), kann ein Softwarepaket seine eigene libc mitbringen (plus im Zweifel natürlich alle möglichen anderen Bibliotheken) und sich so in seiner Bibliotheksausstattung vom unterliegenden Linux-System unabhängig machen. Ob das im wirklichen Leben immer eine gute Idee ist, ist eine andere Frage – immerhin muss man als Softwareanbieter dann unter Umständen unabhängige Fehlerkorrekturen in den Bibliotheken mit warten und an alle Kunden verteilen –, aber Tatsache ist, dass die »DLL-Hölle« unter Linux kein direktes Äquivalent hat. 1 : 0 für den Pinguin!

## Kommandos in diesem Kapitel

<b>busybox</b>	Eine Shell, die Versionen vieler Unix-Werkzeuge integriert hat		
		busybox(1)	192
<b>file</b>	Rät den Typ einer Datei anhand des Inhalts	file(1)	192
<b>ldconfig</b>	Baut den Cache für dynamische Bibliotheken auf	ldconfig(8)	195
<b>ldd</b>	Zeigt die dynamischen Bibliotheken für ein Programm an	ldd(1)	193
<b>sash</b>	„Stand-Alone Shell“ mit eingebauten Kommandos, zur Problembearbeitung	sash(8)	192
<b>strip</b>	Entfernt Symboltabellen von Objektdateien	strip(1)	193

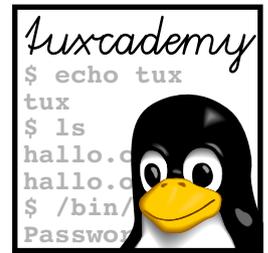
## Zusammenfassung

- Bibliotheken stellen oft gebrauchte Funktionalität in standardisierter Form mehreren Programmen zur Verfügung.
- Dynamische Bibliotheken sparen Speicher- und Plattenplatz und erleichtern die Wartung.
- Mit `file` können Sie herausfinden, ob ein Programm dynamisch gelinkt ist, und dynamische Bibliotheken zweifelsfrei als solche erkennen.
- `ldd` gibt die Namen der dynamischen Bibliotheken aus, die ein Programm (oder eine dynamische Bibliothek) benutzt.
- Dynamische Bibliotheken werden in `/lib`, `/usr/lib` und in den in `/etc/ld.so.conf` benannten Verzeichnissen gesucht. Das Kommando `ldconfig` baut einen Index auf.
- Mehrere Versionen derselben dynamischen Bibliothek können gleichzeitig installiert sein.

## Literaturverzeichnis

**Pet08** Johan Petersson. »What is linux-gate.so.1?«, August 2008.

<http://www.trilithium.com/johan/2005/08/linux-gate/>



# 12

## Paketverwaltung mit Debian-Werkzeugen

### Inhalt

12.1	Überblick	200
12.2	Das Fundament: dpkg	200
12.2.1	Debian-Pakete	200
12.2.2	Paketinstallation	201
12.2.3	Pakete löschen	202
12.2.4	Debian-Pakete und ihr Quellcode	203
12.2.5	Informationen über Pakete	204
12.2.6	Verifikation von Paketen	206
12.3	Debian-Paketverwaltung der nächsten Generation	207
12.3.1	APT	207
12.3.2	Paketinstallation mit apt-get	208
12.3.3	Informationen über Pakete	209
12.3.4	aptitude	211
12.4	Integrität von Debian-Paketen	213
12.5	Die debconf-Infrastruktur	214
12.6	alien: Pakete aus fremden Welten	215

### 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

## 12.1 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.

## 12.2 Das Fundament: `dpkg`

### 12.2.1 Debian-Pakete

Pakete In der Debian-Infrastruktur ist die Software auf dem System in Pakete eingeteilt.  
 Paketnamen 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 architekturenspezifische 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



**12.1 [2]** Holen Sie sich ein beliebiges Debian-Paket (etwa hello) und nehmen Sie es mit Hilfe von ar und tar auseinander. Finden Sie die Installationskripte? Welche Informationen stehen in control.tar.gz, welche in data.tar.gz?

### 12.2.2 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 205). 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



**12.2 [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<sup>1</sup>, 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`.



**12.3 [2]** Finden Sie die aktuelle Liste von virtuellen Paketen in Debian GNU/Linux. Wo wird diese abgelegt? Wann war die letzte Änderung?

### 12.2.3 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:

<sup>1</sup>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



**12.4** [1] Entfernen Sie das Paket aus Übung 12.2 wieder. Sorgen Sie dafür, dass auch allfällige Konfigurationsdateien mit verschwinden.

### 12.2.4 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



**12.5 [1]** Holen Sie sich den Quellcode des Pakets, das Sie in Übung 12.2 installiert haben, und packen Sie ihn aus. Werfen Sie einen Blick in das Unterverzeichnis `debian` in dem resultierenden Verzeichnis.

### 12.2.5 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<sup>2</sup>. `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 haben. 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 Installationskripte 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:

<sup>2</sup>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 (-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/mkcrampfs
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](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



**12.6 [3]** Wieviele Pakete sind auf Ihrem System installiert, deren Namen mit `lib` anfangen? Wieviele dieser Pakete haben die Priorität `required`?

### 12.2.6 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 böartigen Code verstecken. Auf das Thema »Integrität von Paketen« kommen wir in Abschnitt 12.4 zurück.

Schutz vor Eindringlingen

## Übungen



**12.7** [!2] Manipulieren Sie eine Datei in einem installierten Debian-Paket. (Suchen Sie sich ein nicht so wichtiges aus, etwa das aus Übung 12.2.) 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`.

## 12.3 Debian-Paketverwaltung der nächsten Generation

### 12.3.1 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).

### 12.3.2 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.

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.

Einfache 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.

»Intelligente« Aktualisierung

Den Quellcode zu einem Paket holen Sie sich mit dem Kommando »apt-get source«:

Quellcode

```
# apt-get source hello
```

Dies funktioniert auch, wenn das Binärpaket eines von mehreren ist, die aus demselben (anders heißen) 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



**12.8** [!1] Verwenden Sie apt-get, um das Paket hello zu installieren und anschließend wieder zu entfernen.



**12.9** [1] Laden Sie mit apt-get den Quellcode für das Paket hello herunter.

### 12.3.3 Informationen über Pakete

Ein nützliches Programm ist apt-cache, das die Paketquellen von apt-get durchsucht:

apt-cache

```
$ 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)
Normal packages: 25672
Pure virtual packages: 757
Single virtual packages: 1885
Mixed virtual packages: 267
Missing: 4784
Total distinct versions: 28955 (1506k)
```

*Alle Pakete im Cache  
Pakete, die es wirklich gibt  
Platzhalter für Funktionalität  
Nur eine Implementierung  
Mehrere Implementierungen  
Pakete in Abhängigkeiten, die es nicht (mehr?) gibt  
Paketversionen im Cache*

Total distinct descriptions: 28955 (695k)	
Total dependencies: 182689 (5115k)	<i>Anzahl der paarweisen Beziehungen</i>
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	

## Übungen



**12.10** [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«.)

### 12.3.4 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.

Verglichen mit `apt-get` und `dselect` bietet `aptitude` diverse Verbesserungen an. Unter anderem: Verbesserungen

- 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.

Mit `aptitude` kommen Sie in die interaktive Oberfläche (Bild 12.1). 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. interaktive Oberfläche

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. Paketliste

```

anselm@ceiidh: /home/anselm - Befehlsfenster - Konsole
Sitzung Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
Aktionen Rückgängig Paket Auflöser Suchen Optionen Ansichten Hilfe
C-T: Menü ?: Hilfe q: Beenden u: Update g: Download/Inst./Entf. von Paketen
aptitude 0.4.11.11
--- Aktualisierbare Pakete (62)
--- Neue Pakete (28)
--\ Installierte Pakete (2296)
--\ admin - Administrator-Werkzeuge (99)
--\ main - Die Debian-Distribution (99)
i acpi-support 0.109-10 0.109-10
i acpi-support-base 0.109-10 0.109-10
i acpid 1.0.8-1 1.0.8-1
i adduser 3.110 3.110
i anacron 2.3-13.1 2.3-13.1
Diese Pakete sind momentan auf Ihrem Rechner installiert.
Diese Gruppe enthält 2296 Pakete.

```

Bild 12.1: 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 **0**. **=** 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.

## 12.4 Integrität von Debian-Paketen

Das Programm `debsums` dient dazu, die Integrität der Dateien in einem einzelnen Paket zu überprüfen (Abschnitt 12.2.6). 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<sup>+</sup>07, Kapitel 7]. GnuPG erklären wir in der Linup-Front-Schulungsunterlage *Linux-Administration II*.

## 12.5 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 Installationsskripte eines Pakets zugreifen können. Zur Manipulation der Datenbank unterstützt `debconf` modulare Bedienoberflächen, 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 Bedienoberfläche.



Mit der Option `--frontend` (oder `-f`) können Sie sich für dieses Mal eine andere Bedienoberfläche 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



**12.11** [1] Wie können Sie die voreingestellte Bedienungsoberfläche von `debconf` dauerhaft ändern?

## 12.6 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
```

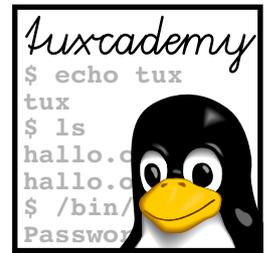
Zum Auseinandernehmen und Zusammensetzen von RPM-Paketen muss das Programm `rpm` installiert sein (das es für Debian GNU/Linux als Paket gibt); zum Zusammensetzen von `deb`-Paketen brauchen Sie ein paar einschlägige Debian-Pakete, die in `alien(1p)` aufgelistet sind. (Auseinandernehmen können Sie `deb`-Pakete, wie in Abschnitt 12.2.1 angedeutet, auf fast allen Linux-Systemen mit »Bordmitteln« wie `tar`, `gzip` und `ar`.)

## Kommandos in diesem Kapitel

<code>alien</code>	Konvertiert verschiedene Software-Paketformate	<code>alien(1)</code>	215
<code>apt-get</code>	Komfortable Oberfläche für Debian-GNU/Linux-Paketverwaltung	<code>apt-get(8)</code>	207
<code>aptitude</code>	Komfortables Werkzeug zur Paketinstallation und -wartung (Debian)	<code>aptitude(8)</code>	211
<code>dpkg</code>	Verwaltungswerkzeug für Debian-GNU/Linux-Pakete	<code>dpkg(8)</code>	200
<code>dpkg-reconfigure</code>	Rekonfiguriert ein bereits installiertes Debian-Paket	<code>dpkg-reconfigure(8)</code>	214

## Literaturverzeichnis

F<sup>+</sup>07 Javier Fernández-Sanguino Peña, et al. »Securing Debian Manual«, 2007.  
<http://www.debian.org/doc/manuals/securing-debian-howto/>



# 13

## Paketverwaltung mit RPM & Co.

### Inhalt

13.1	Einleitung . . . . .	218
13.2	Paketverwaltung mit rpm . . . . .	219
13.2.1	Installation und Update . . . . .	219
13.2.2	Deinstallation von Paketen . . . . .	219
13.2.3	Datenbank- und Paketanfragen . . . . .	220
13.2.4	Verifikation von Paketen . . . . .	222
13.2.5	Das Programm rpm2cpio. . . . .	223
13.3	YUM . . . . .	223
13.3.1	Überblick. . . . .	223
13.3.2	Paketquellen . . . . .	223
13.3.3	Pakete installieren und entfernen mit YUM . . . . .	224
13.3.4	Informationen über Pakete . . . . .	226
13.3.5	Pakete nur herunterladen . . . . .	228

### 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

## 13.1 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.

## 13.2 Paketverwaltung mit rpm

### 13.2.1 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.

### 13.2.2 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 `.rpm.save` an. (Was als Konfigurationsdatei gilt, legt das jeweilige RPM-Paket selbst fest.)

### 13.2.3 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:<sup>1</sup>

```
$ 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.

<sup>1</sup>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 openssl
```

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.

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 Abhängigkeiten

```
$ 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:<sup>2</sup>

```
$ rpm -q --requires saint
web_browser
/bin/rm
/bin/sh
/usr/bin/perl
<<<<<<
```

<sup>2</sup>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.

### 13.2.4 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 Schadensbegrenzung, beispielsweise nach einem Dateisystemcrash.

### 13.2.5 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



**13.1** [2] Benutzen Sie `rpm2cpio` und `cpio`, um eine Liste der Dateien in einem RPM-Paket anzuzeigen.

## 13.3 YUM

### 13.3.1 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.

### 13.3.2 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.

### 13.3.3 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 Verarbeitungstest 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.

### 13.3.4 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
<<<<<<
```

### 13.3.5 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 heruntergeladen *würde*.



Mit der Option --source lädt yumdownloader keine Binär-, sondern Quellcode-RPMs herunter.

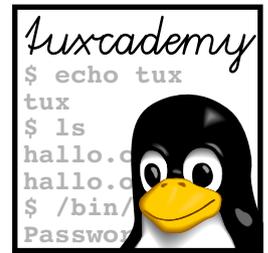
## Kommandos in diesem Kapitel

<b>cpio</b>	Dateiarchiv-Verwaltungsprogramm	cpio(1)	223
<b>rpm</b>	Dient zur Paketverwaltung in vielen Linux-Systemen (Red Hat, SUSE, ...)	rpm(8)	218
<b>rpm2cpio</b>	Wandelt RPM-Pakete in cpio-Archive um	rpm2cpio(1)	223
<b>yum</b>	Komfortables Werkzeug zur Verwaltung von RPM-Paketen	yum(8)	223
<b>yumdownloader</b>	Lädt Pakete aus YUM-Paketquellen (ohne Installation)	yumdownloader(8)	228

## 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.

**1.1** 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.

**1.2** Als root dürfen Sie alles, daher ist es sehr einfach, das System zu beschädigen, wenn Sie sich z. B. versehentlich vertippen.

**1.3** 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.

**1.4** 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.

**1.5** Die Eingabeaufforderung sieht oft anders aus. Außerdem hilft z. B. der Befehl id weiter.

**1.6** 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.

**1.7** 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.

**1.10** 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.

**2.1** Durch deren numerische UID und GID.

**2.2** 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.

**2.3** Die UIDs von Pseudobenzern werden von (Dienst-)Programmen genutzt, die dadurch exakt definierte Zugriffsrechte bekommen.

**2.4** 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 3.5) 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 ver-raten würden.

**2.5** 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.

**2.6** 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.

**2.7** Benutzen Sie `getent`, `cut` und `sort`, um Listen der Benutzernamen für die Datenbanken zu generieren und `comm`, um die beiden Listen zu vergleichen.

**2.8** 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?).

**2.9** Sie geben ihm mit `»passwd trottel«` als `root` ein neues Kennwort, denn lesen können Sie sein altes auch mit `root`-Rechten nicht.

**2.10** Verwenden Sie dazu den Befehl `»passwd -n 7 -x 14 -w 2 hugo«`. Testen können Sie die Einstellungen mit `»passwd -S«`.

**2.11** 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 ...

**2.12** 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).

**2.13** Verwenden Sie zum Löschen den Befehl `userdel`. Um die Dateien zu löschen verwenden Sie das Kommando `»find / -uid <UID> -delete«`.

**2.14** 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.

**2.15** Sie können dazu unter anderem `/etc/passwd` mit `vi` editieren, oder Sie verwenden `usermod`.

**2.16** 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`).

**2.17** 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.

**2.18** 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 Kommando »newgrp test«. Nur wenn Sie nicht Mitglied der entsprechenden Gruppe sind, werden Sie nach dem Kennwort gefragt.

**3.1** 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.

**3.3** 077 beziehungsweise »u=rwx,g=«.

**3.5** 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.

**3.6** 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.

**3.7** 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.

**3.9** Mit der Bash funktioniert das nicht (jedenfalls nicht ohne Weiteres). Für andere Shells können wir hier nicht sprechen.

**3.11** 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.)

**4.1** 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«.

**4.2** 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.)

**4.4** 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

$ kill -STOP 12345                        In einem anderen Fenster:

$ ps u | grep sleep                       Warten
hugo 12346 0.0 0.0 3612 456 pts/2  Z 18:19 0:00 sleep 20
```

**4.5** Konsultieren Sie `ps(1)`.

**4.6** Probieren Sie

```
$ ps -o pid,ppid,state,cmd
```

**4.7** 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.

**4.8** Etwas wie

```
$ pgrep -u hugo
```

sollte dafür ausreichen.

**4.10** Verwenden Sie z. B. das Kommando `renice -10 PID`. Sie können allerdings nur als `root` negative *Nice*-Werte vergeben.

**4.12** Das Ressourcenlimit funktioniert nur für einzelne Prozesse, die Kontingenzierung gilt für den insgesamt von einem Benutzer (oder einer Gruppe) verbrauchten Plattenplatz.

**6.2** sda1, sda2, sda5, sda6 sowie sdb1, sdb5, sdb6, sdb7.

**7.2** Verwenden Sie tune2fs mit den Optionen -c, -u und -m.

**7.3** mkreiserfs /dev/sdb5

**7.6** In /etc/fstab werden alle häufig verwendeten Dateisysteme mit ihren Mountpunkten eingetragen, in /etc/mtab sind alle momentan eingehängten Dateisysteme eingetragen.

**8.1** 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).

**8.2** Siehe Text der Unterlage.

**8.3** 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.

**9.3** 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.

**9.4** 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.

**9.5** Rufen Sie dazu das syslog-Init-Skript mit dem Parameter restart oder reload auf.

**9.6** Zum Beispiel mit »chkconfig -l«.

**9.7** 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.

**9.8** Sie sollten damit rechnen, dass der Rechner Sie nach dem root-Kennwort fragt.

### 9.10 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.

### 9.11 Der Dateiname wird als Meldung verschickt.

**10.4** 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 distributionspezifischen Paketverwaltungswerkzeugs) und ermöglicht die problemlose Aktualisierung von Unit-Dateien durch die Distribution.

**10.10** 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
```

**10.11** 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.

**10.13** Gar nicht (»systemctl mask« liefert eine Fehlermeldung). Sie müssen den Dienst deaktivieren und dann die Unit-Datei löschen, verschieben oder umbenennen.

### 11.3 Herausfinden können Sie das mit einem Shellskript analog zu

```
for f in /usr/bin/*
do
    printf "%4u %s\n" $(ldd $f 2>/dev/null | wc -l) $f
done | sort -nr | head
```

(Das `2>/dev/null` unterdrückt die Fehlermeldung, die `ldd` ausspuckt, wenn Sie ihm ein Shellskript statt eines Maschinenprogramms vorsetzen.) Spitzenreiter auf dem System des Autors ist `mplayer` mit immerhin 118 Bibliotheken.

**11.4** Das wäre eine kapitale Sicherheitslücke, da Sie versuchen könnten, dem Programm über ein privates Verzeichnis in `LD_LIBRARY_PATH` eine Version einer gängigen Bibliothek unterzuschieben, die ein »trojanisches Pferd« enthält. Sie können zum Beispiel mit an Sicherheit grenzender Wahrscheinlichkeit davon ausgehen, dass ein Programm wie `passwd` eine C-Funktion wie `open()` aufrufen wird. Wenn es Ihnen gelänge, dass diese Funktion nicht aus der `libc` genommen wird, sondern aus Ihrer eigenen Bibliothek, dann könnten Sie im Grunde mit `root`-Rechten machen, was Sie wollen. Das ist natürlich nicht erwünscht.

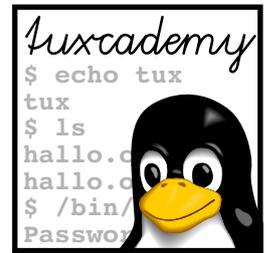
### 12.11 Versuchen Sie mal

```
# dpkg-reconfigure debconf
```

### 13.1 Das geht am einfachsten mit etwas wie

```
$ rpm2cpio <Paket> | cpio -t
```





# B

## LPIC-1-Zertifizierung

### B.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.

## B.2 Prüfung LPI-101

Die folgende Tabelle zeigt die Prüfungsziele der Prüfung LPI-101 (Version 4.0) und die Unterlagen, 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	GRD1	ADM1
101.1	2	Hardware-Einstellungen ermitteln und konfigurieren	–	5–6
101.2	3	Das System starten	–	8–10
101.3	3	Runlevel/Boot-Targets wechseln und das System anhalten oder neu starten	–	9–10
102.1	2	Festplattenaufteilung planen	–	6
102.2	2	Einen Boot-Manager installieren	–	8
102.3	1	Shared Libraries verwalten	–	11
102.4	3	Debian-Paketverwaltung verwenden	–	12
102.5	3	RPM- und YUM-Paketverwaltung verwenden	–	13
103.1	4	Auf der Kommandozeile arbeiten	3–4	–
103.2	3	Textströme mit Filtern verarbeiten	8	–
103.3	4	Grundlegende Dateiverwaltung	6, 11	7.3
103.4	4	Ströme, Pipes und Umleitungen verwenden	8	–
103.5	4	Prozesse erzeugen, überwachen und beenden	9.6	4
103.6	2	Prozess-Ausführungsprioritäten ändern	–	4
103.7	2	Textdateien mit regulären Ausdrücken durchsuchen	7–8	–
103.8	3	Grundlegendes Editieren von Dateien mit vi	5, 7	–
104.1	2	Partitionen und Dateisysteme anlegen	–	6–7
104.2	2	Die Integrität von Dateisystemen sichern	–	7
104.3	3	Das Ein- und Aushängen von Dateisystemen steuern	–	7
104.4	1	Platten-Quotas verwalten	–	7.4
104.5	3	Dateizugriffsrechte und -eigentümerschaft verwalten	–	3
104.6	2	Harte und symbolische Links anlegen und ändern	6	–
104.7	2	Systemdateien finden und Dateien am richtigen Ort platzieren	6, 10	–

## B.3 Prüfung LPI-102

Die folgende Tabelle zeigt die Prüfungsziele der Prüfung LPI-102 (Version 4.0) und die Unterlagen, 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	ADM1	GRD2	ADM2
105.1	4	Die Shell-Umgebung anpassen und verwenden	–	1–2	–
105.2	4	Einfache Skripte anpassen oder schreiben	–	2–5	–
105.3	2	SQL-Datenverwaltung	–	8	–
106.1	2	X11 installieren und konfigurieren	–	11	–
106.2	1	Einen Display-Manager einrichten	–	11	–
106.3	1	Hilfen für Behinderte	–	12	–
107.1	5	Benutzer- und Gruppenkonten und dazugehörige Systemdateien verwalten	2	–	–
107.2	4	Systemadministrationsaufgaben durch Einplanen von Jobs automatisieren	–	9	–
107.3	3	Lokalisierung und Internationalisierung	–	10	–
108.1	3	Die Systemzeit verwalten	–	–	8
108.2	3	Systemprotokollierung	–	–	1–2
108.3	3	Grundlagen von Mail Transfer Agents (MTAs)	–	–	11
108.4	2	Drucker und Druckvorgänge verwalten	–	–	9
109.1	4	Grundlagen von Internet-Protokollen	–	–	3–4
109.2	4	Grundlegende Netz-Konfiguration	–	–	4–5, 7
109.3	4	Grundlegende Netz-Fehlersuche	–	–	4–5, 7
109.4	2	Clientseitiges DNS konfigurieren	–	–	4
110.1	3	Administrationsaufgaben für Sicherheit durchführen	2	–	4–5, 13
110.2	3	Einen Rechner absichern	2	–	4, 6–7, 13
110.3	3	Daten durch Verschlüsselung schützen	–	–	10, 12

## B.4 LPI-Prüfungsziele in dieser Schulungsunterlage

### 101.1 Hardware-Einstellungen ermitteln und konfigurieren

**Gewicht** 2

**Beschreibung** Kandidaten sollten in der Lage sein, die wesentliche Hardware eines Systems zu bestimmen und zu konfigurieren.

#### Wichtigste Wissensgebiete

- Integrierte Peripheriegeräte aktivieren und deaktivieren
- Systeme mit oder ohne externe Peripheriegeräte wie Tastaturen konfigurieren
- Die verschiedenen Arten von Massenspeicher unterscheiden
- Die Unterschiede zwischen Coldplug- und Hotplug-Geräten kennen
- Hardwareressourcen für Geräte ermitteln
- Werkzeuge und Hilfsprogramme, um verschiedene Hardware-Informationen aufzulisten (z.B. `lsusb`, `lspci` usw.)
- Werkzeuge und Hilfsprogramme, um USB-Geräte zu manipulieren
- Konzeptuelles Verständnis von `sysfs`, `udev`, `dbus`

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `/sys/`
- `/proc/`
- `/dev/`
- `modprobe`
- `lsmod`
- `lspci`
- `lsusb`

### 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
- Verständnis von SysV-Init und systemd
- Wissen über grundlegende Eigenschaften von Upstart
- Ereignisse beim Systemstart in den Protokolldateien nachschlagen

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- |             |              |             |
|-------------|--------------|-------------|
| • dmesg     | • Systemkern | • SysV-Init |
| • BIOS      | • initramfs  | • systemd   |
| • Bootlader | • init       |             |

### 101.3 Runlevel/Boot-Targets wechseln und das System anhalten oder neu starten

**Gewicht** 3

**Beschreibung** Kandidaten sollten in der Lage sein, den SysV-Init-Runlevel oder das systemd-Boot-Target 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/Boot-Targets zu benachrichtigen und Prozesse korrekt anzuhalten. Dieses Prüfungsziel umfasst ferner das Einstellen des Standard-SysV-Init-Runlevels oder systemd-Boot-Targets. Es enthält außerdem grundlegendes Wissen über Upstart als Alternative zu SysV-Init oder systemd.

#### Wichtigste Wissensgebiete

- Den Standard-Runlevel oder das Standard-Boot-Target setzen
- Zwischen Runlevels/Boot-Targets wechseln, einschließlich dem Einbenutzermodus
- Systemhalt und Neustart von der Kommandozeile
- Benutzer vor einem Runlevel/Boot-Target-Wechsel oder anderem größerem Ereignis benachrichtigen
- Prozesse korrekt beenden

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- |                |                 |                     |
|----------------|-----------------|---------------------|
| • /etc/inittab | • telinit       | • /usr/lib/systemd/ |
| • shutdown     | • systemd       | • wall              |
| • init         | • systemctl     |                     |
| • /etc/init.d/ | • /etc/systemd/ |                     |

### 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)
- /home-Dateisystem
- /boot-Dateisystem
- /var-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:

- menu.lst, grub.cfg und grub.conf
- grub-install
- grub-mkconfig
- MBR

## 102.3 Shared Libraries verwalten

**Gewicht** 1

**Beschreibung** Kandidaten sollten in der Lage sein, die Shared Libraries zu bestimmen, von denen ausführbare Programme abhängen, und diese bei Bedarf zu installieren.

**Wichtigste Wissensgebiete**

- Shared Libraries identifizieren
- Die typischen Orte für Systembibliotheken identifizieren
- Shared Libraries laden

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- ldd
- ldconfig
- /etc/ld.so.conf
- LD\_LIBRARY\_PATH

## 102.4 Debian-Paketverwaltung verwenden

**Gewicht** 3

**Beschreibung** Kandidaten sollten in der Lage sein, Pakete mit den Debian-Paketwerkzeugen zu verwalten.

**Wichtigste Wissensgebiete**

- Debian-Binärpakete installieren, aktualisieren und entfernen
- Pakete finden, die bestimmte Dateien oder Bibliotheken enthalten und installiert sind oder nicht
- Paketinformationen bestimmen wie Version, Inhalt, Abhängigkeiten, Integrität des Pakets und Installationsstatus (ob das Paket installiert ist oder nicht)

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- /etc/apt/sources.list
- dpkg
- dpkg-reconfigure
- apt-get
- apt-cache
- aptitude

## 102.5 RPM- und YUM-Paketverwaltung verwenden

**Gewicht** 3

**Beschreibung** Kandidaten sollten in der Lage sein, Pakete mit den RPM- und YUM-Werkzeugen zu verwalten.

**Wichtigste Wissensgebiete**

- Pakete mit RPM und YUM installieren, erneut installieren, aktualisieren und entfernen
- Informationen über RPM-Pakete bestimmen wie Version, Status, Abhängigkeiten, Integrität und Signaturen
- Herausfinden, welche Dateien ein Paket zur Verfügung stellt, und herausfinden, aus welchem Paket eine bestimmte Datei kommt

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- rpm
- rpm2cpio
- /etc/yum.conf
- /etc/yum.repos.d/
- yum
- yumdownloader

## 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 und Verzeichnisse rekursiv kopieren
- Dateien 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
- xz
- Dateisuchmuster

## 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:

- |        |          |           |
|--------|----------|-----------|
| • &    | • nohup  | • pgrep   |
| • bg   | • ps     | • pkill   |
| • fg   | • top    | • killall |
| • jobs | • free   | • screen  |
| • kill | • uptime |           |

## 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 | • renice |
| • ps   | • top    |

## 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

- MBR-Partitionstabellen verwalten
- Verschiedene mkfs-Kommandos verwenden, um Partitionen zu installieren und verschiedene Dateisysteme anzulegen wie
  - ext2/ext3/ext4
  - xfs
  - vfat
- Wissen von der Existenz von ReiserFS und Btrfs
- Grundlagen von gdisk und parted mit GPT

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- |         |          |          |
|---------|----------|----------|
| • fdisk | • parted | • mkswap |
| • gdisk | • mkfs   |          |

## 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     | • mke2fs   | • xfs-Werkzeuge (etwa xfs_metadump und xfs_info) |
| • df     | • debugfs  |  |
| • fsck   | • dumpe2fs |  |
| • e2fsck | • tune2fs  |  |

## 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 | • mount  |
| • /media/    | • 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   | • repquota |
| • edquota | • 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 | • chown |
| • umask | • chgrp |

## 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 | • chage    | • passwd  |
| • /etc/shadow | • groupadd | • useradd |
| • /etc/group  | • groupdel | • userdel |
| • /etc/skel/  | • groupmod | • usermod |

## 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 Altern von Kennwörtern setzen oder ändern
- Mit nmap und netstat offene Ports auf einem System finden können
- Grenzen für Benutzeranmeldungen, Prozesse und Speicherverbrauch setzen
- Feststellen, welche Benutzer sich angemeldet haben oder gerade angemeldet sind
- Grundlegende Konfiguration und Gebrauch von sudo

Hier ist eine auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- find
- passwd
- fuser
- lsof
- nmap
- chage
- netstat
- sudo
- /etc/sudoers
- su
- usermod
- ulimit
- who, w, last

## 110.2 Einen Rechner absichern

**Gewicht** 3

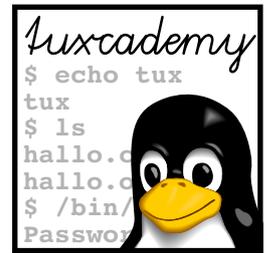
**Beschreibung** Kandidaten sollten wissen, wie sie grundlegende Rechnersicherheit konfigurieren können.

### 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



# C

## Kommando-Index

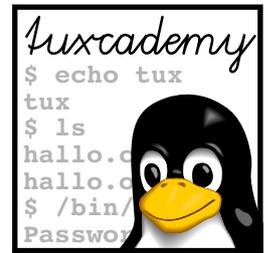
Dieser Anhang fasst alle im Text erklärten Kommandos zusammen und verweist auf deren Dokumentation sowie die Stellen im Text, wo die Kommandos eingeführt werden.

<b>adduser</b>	Komfortables Kommando zum Anlegen neuer Benutzerkonten (Debian)	
		adduser(8) 38
<b>alien</b>	Konvertiert verschiedene Software-Paketformate	alien(1) 215
<b>apt-get</b>	Komfortable Oberfläche für Debian-GNU/Linux-Paketverwaltung	
		apt-get(8) 207
<b>aptitude</b>	Komfortables Werkzeug zur Paketinstallation und -wartung (Debian)	
		aptitude(8) 211
<b>blkid</b>	Findet Attribute von blockorientierten Geräten und gibt sie aus	
		blkid(8) 130
<b>busybox</b>	Eine Shell, die Versionen vieler Unix-Werkzeuge integriert hat	
		busybox(1) 192
<b>cfdisk</b>	Plattenpartitionierungsprogramm mit textbildschirmorientierter Oberfläche	
		cfdisk(8) 102
<b>chage</b>	Setzt Kennwort-Attribute wie Ablaufdatum und Änderungsfristen	
		chage(1) 39
<b>chattr</b>	Setzt Dateiattribute für ext2- und ext3-Dateisysteme	chattr(1) 56
<b>chfn</b>	Erlaubt Benutzern das Ändern des GECOS-Felds in der Benutzerdatenbank	chfn(1) 30
<b>chgrp</b>	Setzt Gruppenzugehörigkeit von Dateien und Verzeichnissen	
		chgrp(1) 49
<b>chkconfig</b>	Schaltet Systemdienste ein oder aus (SUSE, Red Hat)	
		chkconfig(8) 161
<b>chmod</b>	Setzt Rechte für Dateien und Verzeichnisse	chmod(1) 47
<b>chown</b>	Setzt Eigentümer und Gruppenzugehörigkeit für Dateien und Verzeichnisse	chown(1) 48
<b>cpio</b>	Dateiarchiv-Verwaltungsprogramm	cpio(1) 223
<b>dd</b>	„Copy and convert“, kopiert Dateien und Dateisysteme blockweise und macht einfache Konvertierungen	dd(1) 132
<b>debugfs</b>	Dateisystem-Debugger zur Reparatur schlimm verkorkster Dateisysteme. Nur für Gurus!	debugfs(8) 119
<b>dmesg</b>	Gibt den Inhalt des Kernel-Nachrichtenpuffers aus	dmesg(8) 150
<b>dpkg</b>	Verwaltungswerkzeug für Debian-GNU/Linux-Pakete	dpkg(8) 200
<b>dpkg-reconfigure</b>	Rekonfiguriert ein bereits installiertes Debian-Paket	
		dpkg-reconfigure(8) 214
<b>dumpe2fs</b>	Gibt interne Strukturen des ext2-Dateisystems aus. Nur für Gurus!	
		dumpe2fs(8) 119

<b>dumpriserfs</b>	Gibt interne Strukturen des Reiser-Dateisystems aus. Nur für Gurus!	dumpriserfs(8)	122
<b>e2fsck</b>	Prüft ext2- und ext3-Dateisysteme auf Konsistenz	e2fsck(8)	117
<b>e2label</b>	Ändert das Label eines ext2/3-Dateisystems	e2label(8)	130
<b>edquota</b>	Erlaubt Eingabe und Ändern von Plattenkontingenten	edquota(8)	134
<b>fdisk</b>	Partitionierungswerkzeug für Platten und ähnliche Medien	fdisk(8)	97
<b>file</b>	Rät den Typ einer Datei anhand des Inhalts	file(1)	192
<b>fsck</b>	Organisiert Dateisystemkonsistenzprüfungen	fsck(8)	111
<b>gdisk</b>	Partitionierungswerkzeug für GPT-Platten	gdisk(8)	102
<b>getent</b>	Ruft Einträge aus administrativen Datenbanken ab	getent(1)	35
<b>gpasswd</b>	Erlaubt Verwaltung von Gruppenmitgliederlisten durch Gruppenadministratoren und das Setzen des Gruppenkennworts	gpasswd(1)	42
<b>groupadd</b>	Fügt Gruppen in die Gruppendatenbank ein	groupadd(8)	42
<b>groupdel</b>	Löscht Gruppen aus der Gruppendatenbank	groupdel(8)	42
<b>groupmod</b>	Ändert Gruppeneinträge in der Gruppendatenbank	groupmod(8)	42
<b>groups</b>	Gibt die Gruppen aus, in denen ein Benutzer Mitglied ist	groups(1)	26
<b>grub-md5-crypt</b>	Bestimmt MD5-verschlüsselte Kennwörter für GRUB Legacy	grub-md5-crypt(8)	147
<b>halt</b>	Führt das System herunter	halt(8)	166
<b>id</b>	Gibt UID und GIDs eines Benutzers aus	id(1)	26
<b>initctl</b>	Zentrales Steuerungsprogramm für Upstart	initctl(8)	165
<b>insserv</b>	Aktiviert oder deaktiviert Init-Skripte (SUSE)	insserv(8)	161
<b>kill</b>	Hält einen Hintergrundprozess an	bash(1), kill(1)	64
<b>killall</b>	Schickt allen Prozessen mit passendem Namen ein Signal	killall(1)	65
<b>kpartx</b>	Erzeugt Blockgeräte aus Partitionstabellen	kpartx(8)	103
<b>last</b>	Zeigt zuletzt angemeldete Benutzer an	last(1)	26
<b>ldconfig</b>	Baut den Cache für dynamische Bibliotheken auf	ldconfig(8)	195
<b>ldd</b>	Zeigt die dynamischen Bibliotheken für ein Programm an	ldd(1)	193
<b>losetup</b>	Erzeugt und verwaltet Loop-Devices	losetup(8)	103
<b>lsattr</b>	Zeigt Dateiattribute auf ext2- und ext3-Dateisystemen an	lsattr(1)	56
<b>lsblk</b>	Listet verfügbare Blockgeräte auf	lsblk(8)	131
<b>lsmod</b>	Listet die geladenen Kernel-Module auf	lsmod(8)	79
<b>lspci</b>	Gibt Informationen über Geräte auf dem PCI-Bus aus	lspci(8)	74
<b>lsusb</b>	Listet alle angeschlossenen USB-Geräte auf	lsusb(8)	77
<b>mesg</b>	Schaltet wall-Nachrichten für ein Terminal ein oder aus	mesg(1)	167
<b>mkdosfs</b>	Legt FAT-formatierte Dateisysteme an	mkfs.vfat(8)	126
<b>mke2fs</b>	Legt ein ext2- oder ext3-Dateisystem an	mke2fs(8)	116
<b>mkfs</b>	Organisiert das Anlegen von Dateisystemen	mkfs(8)	110
<b>mkfs.vfat</b>	Legt FAT-formatierte Dateisysteme an	mkfs.vfat(8)	126
<b>mkfs.xfs</b>	Legt XFS-formatierte Dateisysteme an	mkfs.xfs(8)	123
<b>mkreiserfs</b>	Legt Reiser-Dateisysteme an	mkreiserfs(8)	122
<b>mkswap</b>	Initialisiert eine Auslagerungs-Partition oder -Datei	mkswap(8)	127
<b>modprobe</b>	Lädt Kernel-Module unter Berücksichtigung von Abhängigkeiten	modprobe(8)	79
<b>mount</b>	Hängt ein Dateisystem in den Dateibaum ein	mount(8), mount(2)	128
<b>nice</b>	Startet Programme mit einem anderen <i>nice</i> -Wert	nice(1)	67
<b>nohup</b>	Startet ein Programm so, dass es immun gegen SIGHUP-Signale ist	nohup(1)	69
<b>parted</b>	Leistungsfähiges Partitionierungswerkzeug aus dem GNU-Projekt	Info: parted	100
<b>pgrep</b>	Sucht Prozesse anhand ihres Namens oder anderer Kriterien	pgrep(1)	66
<b>pkill</b>	Signalisiert Prozessen anhand ihres Namens oder anderer Kriterien	pkill(1)	67
<b>ps</b>	Gibt Prozess-Statusinformationen aus	ps(1)	62
<b>pstree</b>	Gibt den Prozessbaum aus	pstree(1)	63

<b>reboot</b>	Startet den Rechner neu	reboot(8)	166
<b>reiserfsck</b>	Prüft ein Reiser-Dateisystem auf Konsistenz	reiserfsck(8)	122
<b>renice</b>	Ändert den <i>nice</i> -Wert laufender Prozesse	renice(8)	68
<b>resize_reiserfs</b>	Ändert die Größe eines Reiser-Dateisystems	resize_reiserfs(8)	122
<b>rpm</b>	Dient zur Paketverwaltung in vielen Linux-Systemen (Red Hat, SUSE, ...)	rpm(8)	218
<b>rpm2cpio</b>	Wandelt RPM-Pakete in <i>cpio</i> -Archive um	rpm2cpio(1)	223
<b>runlevel</b>	Zeigt den vorigen und den aktuellen Runlevel an	runlevel(8)	160
<b>sash</b>	„Stand-Alone Shell“ mit eingebauten Kommandos, zur Problembearbeitung	sash(8)	192
<b>setfacl</b>	Erlaubt die Manipulation von ACLs	setfacl(1)	51
<b>sfdisk</b>	Nichtinteraktives Partitionierungsprogramm	sfdisk(8)	102
<b>sgdisk</b>	Nichtinteraktives Partitionierungsprogramm für GPT-Platten	sgdisk(8)	103
<b>shutdown</b>	Führt das System herunter oder startet es neu, mit Verzögerung und Warnung an angemeldete Benutzer	shutdown(8)	165
<b>star</b>	POSIX-kompatibles Archivprogramm mit ACL-Unterstützung	star(1)	51
<b>strip</b>	Entfernt Symboltabellen von Objektdateien	strip(1)	193
<b>su</b>	Startet eine Shell unter der Identität eines anderen Benutzers	su(1)	17
<b>sudo</b>	Erlaubt normalen Benutzern das Aufrufen bestimmter Kommandos mit Administratorprivilegien	sudo(8)	15
<b>swapoff</b>	Deaktiviert eine Auslagerungs-Partition oder -Datei	swapoff(8)	127
<b>swapon</b>	Aktiviert eine Auslagerungs-Partition oder -Datei	swapon(8)	127
<b>systemctl</b>	Zentrales Steuerungsprogramm für <i>systemd</i>	systemctl(1)	173, 183
<b>top</b>	Bildschirmorientiertes Programm zur Beobachtung und Verwaltung von Prozessen	top(1)	69
<b>tune2fs</b>	Stellt Parameter von <i>ext2</i> - und <i>ext3</i> -Dateisystemen ein	tune2fs(8)	120, 131
<b>udev</b>	Daemon zur Behandlung von Kernel-uevents	udev(8)	82
<b>ulimit</b>	Legt Ressourcenbegrenzungen für Prozesse fest	bash(1)	68
<b>umask</b>	Stellt die <i>umask</i> (Rechtekontrolle für neue Dateien) ein	bash(1)	50
<b>update-rc.d</b>	Installiert und entfernt System-V-Initkript-Links (Debian)	update-rc.d(8)	161
<b>useradd</b>	Fügt neue Benutzerkonten hinzu	useradd(8)	37
<b>userdel</b>	Entfernt Benutzerkonten	userdel(8)	40
<b>usermod</b>	Modifiziert die Benutzerdatenbank	usermod(8)	40
<b>vigr</b>	Erlaubt das exklusive Ändern von <i>/etc/group</i> bzw. <i>/etc/gshadow</i>	vipw(8)	43
<b>vol_id</b>	Bestimmt Dateisystemtypen und gibt Label und UUID aus	vol_id(8)	130
<b>wall</b>	Schreibt eine Nachricht auf die Terminals aller angemeldeten Benutzer	wall(1)	167
<b>xfs_info</b>	Entspricht <i>xfs_growfs</i> , aber ändert das Dateisystem nicht	xfs_growfs(8)	124
<b>xfs_mdrestore</b>	Überträgt einen Metadaten-Abzug in ein XFS-Dateisystem	xfs_mdrestore(8)	123
<b>xfs_metadump</b>	Erzeugt Metadaten-Abzüge von XFS-Dateisystemen	xfs_metadump(8)	123
<b>xfs_repair</b>	„Repariert“ XFS-Dateisysteme	xfs_repair(8)	123
<b>yum</b>	Komfortables Werkzeug zur Verwaltung von RPM-Paketen	yum(8)	223
<b>yumdownloader</b>	Lädt Pakete aus YUM-Paketquellen (ohne Installation)	yumdownloader(8)	228





# 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.

- /, 95
- adduser, 38
- Administrationswerkzeuge, 14
- alien, 199, 215–216
  - to-deb (Option), 216
- apt, 200
- apt-cache, 199, 209–211
- apt-get, 199, 202, 207–209, 211–212, 223–224
  - dist-upgrade (Option), 208–209
  - install (Option), 208
  - remove (Option), 209
  - source (Option), 209
  - upgrade (Option), 209
- apt-key, 214
- aptitude, 199–201, 211–212
- aquota.group, 135
- aquota.user, 134
- ar, 201, 216
- at, 179
- ATA, 86
- awk, 190
  
- bash, 192
- Benutzerdatenbank, 28, 31
  - anderswo gelagert, 31
- Benutzerkonten, **24**
- Benutzername, **25**
- bg, 60
- /bin/sh, 30, 221, 232
- /bin/true, 30
- blkid, 130–131
- /boot, 147
- /boot/grub, 146
- /boot/grub/custom.cfg, 147
- /boot/grub/grub.cfg, 147
- /boot/grub/menu.lst, 144
- Bootmanager, **140**
- Bootsektor, **140**
- Bootskript, 159
- Bottomley, James, 142
  
- btrfs, 125
- btrfs check
  - repair (Option), 125
- busybox, 192
  
- Card, Rémy, 113
- cat, 35, 234
- cc, 132
- cd, 46
- cfdisk, 102
- chage, 39
- chattr, 56, 234
  - R (Option), 56
- chfn, 30
- chgrp, 42, 49, 54
  - R (Option), 49
- chkconfig, 161, 236
- chmod, 15, 47–48, 50–51, 53–54, 56
  - R (Option), 48
  - reference=<Name> (Option), 48
- chown, 41, 48–49
  - R (Option), 49
- chsh, 30
- comm, 232
- cp, 128, 192
- cpio, 143, 145, 223, 228, 251
- cron, 162, 179
- cut, 232
  
- D-Bus, 83
- Dateiattribute, **55**
- dd, 68, 97, 103, 121, 124, 128, 132–133, 151
- DEBCONF\_FRONTEND (Umgebungsvariable), 214
- DEBCONF\_PRIORITY (Umgebungsvariable), 215
- debsums, 206–207, 212
- debugfs, 119
  - w (Option), 119
- Definitionen, **12**
- demand paging, 55

- /dev, 81–82
- /dev/block, 94
- /dev/mapper, 104
- /dev/null, 187
- /dev/psaux, 80
- /dev/scd0, 117
- /dev/sda, 97
- /dev/ttyS0, 16
- /dev/zero, 117
- device, 81
- diff, 203
- Dijkstra, Edsger, 149
- dmesg, 150
- dpkg, 199–202, 205, 207, 210–212
  - a (Option), 201
  - configure (Option), 201
  - force-depends (Option), 201
  - force-overwrite (Option), 202
  - i (Option), 201
  - install (Option), 201
  - L (Option), 206
  - l (Option), 204
  - list (Option), 204
  - listfiles (Option), 206
  - P (Option), 202
  - purge (Option), 211
  - r (Option), 202
  - s (Option), 204, 206
  - search (Option), 206
  - status (Option), 204–205, 210
  - unpack (Option), 201
- dpkg-reconfigure, 214–215
  - f (Option), 214
  - frontend (Option), 214
  - p (Option), 215
  - priority (Option), 215
- dpkg-source, 203
- dselect, 207, 211
- dump, 55
- dumpe2fs, 119
- dumpreiserfs, 122
- e2fsck, 117–119, 122
  - B (Option), 118
  - b (Option), 118–119
  - c (Option), 118
  - f (Option), 118
  - l (Option), 118
  - p (Option), 118
  - v (Option), 118
- e2label, 130
- e4defrag, 120
- EDITOR (Umgebungsvariable), 41, 134
- edquota, 134
  - t (Option), 134
- egrep, 66
- Einbenutzermodus, **162**
- /etc, 18, 128
- /etc/apt/apt.conf, 209
- /etc/apt/sources.list, 208
- /etc/apt/trusted.gpg, 214
- /etc/dpkg/dpkg.cfg, 201
- /etc/filesystems, 129–130
- /etc/fstab, 93, 97, 112, 120–121, 127, 129, 132–134, 159, 173–174, 236
- /etc/group, 27, 29, 34–36, 40–43
- /etc/grub.d, 147
- /etc/grub.d/40\_custom, 147
- /etc/grub.inst, 146
- /etc/gshadow, 34–35, 41–43, 251
- /etc/inetd.conf, 174
- /etc/init, 163
- /etc/inittab, 156, 158–160, 166, 174, 176, 180, 182
- /etc/ld.so.cache, 195–196
- /etc/ld.so.conf, 195
- /etc/modprobe.conf, 79
- /etc/modprobe.d, 79
- /etc/mtab, 132, 236
- /etc/nologin, 166
- /etc/nsswitch.conf, 35
- /etc/passwd, 27–31, 34–36, 38, 40–41, 233
- /etc/rpmrc, 218
- /etc/securetty, 16
- /etc/shadow, 28, 31–36, 39–41, 44, 52, 232–233
- /etc/shells, 30
- /etc/skel, 37
- /etc/sysconfig, 19
- /etc/udev/rules.d, 82
- /etc/yum.conf, 224
- /etc/yum.repos.d, 224
- fdisk, 97–102
  - l (Option), 99
  - u (Option), 99
- Festplatten
  - Geometrie, 73
  - SCSI, 87
- Festplattencache, 111
- fg, 60
- file, 192–193
- finger, 30
- fsck, 111–112, 117, 119–120, 123, 152
  - A (Option), 112
  - a (Option), 112
  - f (Option), 112
  - N (Option), 112
  - p (Option), 112
  - R (Option), 112
  - s (Option), 112
  - t (Option), 112, 123
  - V (Option), 112
  - v (Option), 112
- fsck.ext2, 117
- fsck.xfs, 123
- Garrett, Matthew, 142

- gdisk, 102, 133, 151
- getent, 35, 232
- getfacl, 51
- getty, 180
- GNOME, 214
- Gooch, Richard, 82
- gpasswd, 42
  - A (Option), 42
  - a (Option), 42
  - d (Option), 42
- grep, 35, 63, 66
- groupadd, 42
  - g (Option), 42
- groupdel, 42
- groupmod, 40, 42
  - g (Option), 42
  - n (Option), 42
- groups, 26
- GRUB, 140**
  - Bootprobleme, 151
- grub, 145
  - device-map (Option), 146
  - lock (Option), 236
  - password (Option), 147
- grub-install, 146
- grub-md5-crypt, 147
- grub-mkconfig, 147
- grub.cfg, 147
- Gruppe, 25**
  - administrative, 34
  - Administrator, 42
  - Kennwort, 34–35, 42
- Gruppen, 15**
- gzip, 216
  
- halt, 166
- harte Grenze, 133
- Heimatverzeichnis, 25
- hello, 200, 203
- /home, 30–31, 95
- Homme, Kjetil Torgrim, 66
  
- id, 26, 29, 55, 231
  - G (Option), 26
  - g (Option), 26
  - Gn (Option), 26
  - n (Option), 26
  - u (Option), 26
- init, 111, 148, 150, 158, 160, 236
- Init-Skripte, 160, 166, 218
  - Parameter, 160
- initctl, 165
- initctl start, 165
- initctl status, 165
- initctl stop, 165
- insserv, 161–162, 236
- ISOLINUX, 140
  
- jobs, 60
  
- Johnson, Jeff, 218
- Journaling, 113
  
- KDE, 214
- Kennwörter, 25, 28, 31
  - ändern, 38
  - GRUB, 147
  - Gruppen-, 34–35, 42
  - vergeben, 38
- kill, 64–67, 162
- killall, 65–67
  - i (Option), 65
  - l (Option), 65
  - w (Option), 65
- Kok, Auke, 163
- konsole, 30
- kpartx, 103–104, 107
  - v (Option), 104
- Kroah-Hartman, Greg, 82
  
- Label, 130**
- last, 26–27
- ld-linux.so, 195
- ld.so.cache, 195
- LD\_LIBRARY\_PATH (Umgebungsvariable), 195–197, 237
- ldconfig, 195
  - p (Option), 195
- ldd, 193–195, 237
- less, 35
- /lib, 193, 195–196
- /lib/modules, 79–80
- login, 17, 30, 166
- losetup, 103–104
  - a (Option), 103
  - f (Option), 103
- lost+found, 119
- ls, 28, 41, 46–47, 56
  - l (Option), 28, 47, 56
- lsattr, 56, 234
  - a (Option), 56
  - d (Option), 56
  - R (Option), 56
- LSB, 200
- lsblk, 131
- lsmod, 79
- lspci, 74–75, 77
  - n (Option), 75
  - t (Option), 75
  - v (Option), 74–75
- lsusb, 77–78
  - v (Option), 77
  
- mail, 30
- Mason, Chris, 110
- Master Boot Record, 140
- Matilainen, Panu, 218
- mesg, 167
- Minix, 113

- mkdosfs, 126
- mke2fs, 110, 116–117, 120
  - F (Option), 117
- mkfs, 110–112, 116–117, 125–126, 141
  - t (Option), 110, 116–117, 126
- mkfs.btrfs
  - d (Option), 125
  - L (Option), 131
- mkfs.vfat, 126
- mkfs.xfs, 123–124
  - l (Option), 123
- mkreiserfs, 122
- mkswap, 127–128, 131
- /mnt, 117
- modprobe, 79–80
  - r (Option), 79
- mount, 97, 120, 128–130
  - t (Option), 129
  - usrquota (Option), 134
- mv, 128
  
- newgrp, 34
- nice, 67
- nohup, 69
- nohup.out, 69
- NSA, 15
  
- /opt, 95
  
- Packages.gz, 213–214
- parted, 100–102
- passwd, 28, 38–42, 52, 232, 237
  - l (Option), 39
  - S (Option), 39
  - u (Option), 39
- passwd -n, 39
- passwd -w, 39
- passwd -x, 39
- PATH (Umgebungsvariable), 195
- PCI Express, 73
- perl, 190
- pgrep, 66–67
  - a (Option), 66
  - d (Option), 66
  - f (Option), 66
  - G (Option), 66
  - l (Option), 66
  - n (Option), 66
  - o (Option), 66
  - P (Option), 67
  - t (Option), 67
  - u (Option), 67
- pkill, 66–67, 186
  - signal (Option), 67
- Poettering, Lennart, 156, 172
- präemptives Multitasking, 61
- primäre Gruppe, 29
- Priorität, 67
- /proc, 60–62, 80, 234
  - /proc/filesystems, 129
  - /proc/pci, 74
  - /proc/swaps, 127–128
  - /proc/sys/kernel/pid\_max, 234
  - ~/.profile, 134
- Prozesszustand, 61
- Prüfungsziele, 239
- ps, 52, 62–64, 66
  - a (Option), 62–63
  - ax (Option), 63
  - C (Option), 63
  - forest (Option), 62, 64
  - help (Option), 62
  - l (Option), 62–63
  - o (Option), 63
  - p (Option), 66
  - r (Option), 63
  - T (Option), 63
  - U (Option), 63
  - u (Option), 52
  - x (Option), 63
- Pseudobeneutzer, 27
- pstree, 63–64
  - G (Option), 64
  - p (Option), 64
  - u (Option), 64
- pwconv, 36
- Python, 214
- python, 190
  
- quota, 134–135, 162
  - q (Option), 134
- quotacheck, 134
- quotaoff, 134
  
- rc, 161
- reboot, 166
- Reiser, Hans, 121–122
- reiserfsck, 122
- Release, 213
- Release.gpg, 213
- Remnant, Scott James, 156, 163
- renice, 68
- repquota, 134
- resize\_reiserfs, 122
- Ritchie, Dennis, 53
- rm, 46
- rpm, 200, 216, 218–220, 223, 227
  - a (Option), 220
  - c (Option), 221
  - d (Option), 221
  - e (Option), 219
  - f (Option), 220
  - h (Option), 219
  - i (Option), 219, 221
  - l (Option), 221–222
  - nodeps (Option), 219–220
  - p (Option), 220
  - provides (Option), 222

- q (Option), 220
- qi (Option), 227
- requires (Option), 221
- test (Option), 219–220
- V (Option), 222
- v (Option), 218, 221
- vv (Option), 218
- whatprovides (Option), 221
- whatrequires (Option), 222
- rpm2cpio, 223
- ~/ .rpmrc, 218
- Rückgabewert, **61**
- Runlevel, **156**, 166
  - Bedeutung, 159
  - konfigurieren, 161
  - wechseln, 160
- runlevel, 160, 186, 236
- sash, 192
- /sbin/init, 156
- Schattenkennwörter, 28, 31
- SELinux, 15
- setfacl, 51
- sfdisk, 102–103, 133, 151
- sgdisk, 103
- shadow passwords*, **28**, 31
- shutdown, 15, 158, 165–167, 183
  - c (Option), 236
  - h (Option), 166
  - r (Option), 166
- Sievers, Kay, 82, 156, 172
- Signale, **64**
- sleep, 67
- sort, 232
- /srv, 96
- ssh, 27
- sshd, 66
- star, 51
- sticky bit*, **54**
- strip, 193
- su, 17, 19, 27, 231, 233
- sudo, 15, 17–18, 231
- Superblock, **110**
- Superuser, **14**
- SuSEconfig, 19
- Swap-Partition, **127**
- swapoff, 127
- swapon, 127–128
  - /sys, 82
  - /sys/block, 81
  - /sys/bus, 80
  - /sys/bus/scsi/devices, 94
  - /sys/class, 81
  - /sys/class/scsi\_host, 94
  - /sys/devices/, 80
- syslog, 162, 236
- syslogd, 150, 162
- systemctl, 173, 182–188, 237
  - full (Option), 185
  - kill-who (Option), 184
  - l (Option), 185
  - lines (Option), 185
  - n (Option), 185
  - now (Option), 187
  - s (Option), 184
  - signal (Option), 184
  - t (Option), 183, 185
- systemd, 185
- systemd-escape, 178
  - p (Option), 178
  - u (Option), 178
- tar, 51, 143, 192, 201, 216, 218, 231
- telinit, 158, 160, 162
  - q (Option), 158
- Terminierung, **89**
- /tmp, 41, 54, 96, 233
- top, 69
- touch, 41
- Ts'o, Theodore, 115
- tune2fs, 118, 120–121, 131, 235
  - c (Option), 235
  - L (Option), 131
  - l (Option), 118
  - m (Option), 235
  - u (Option), 235
- Tweedie, Stephen, 113
- udev, 82
- udisksctl, 83
- udisksd, 83
- UID, **25**
- ulimit, 68
  - a (Option), 68
  - d (Option), 68
  - f (Option), 68
  - n (Option), 68
  - t (Option), 68
  - u (Option), 68
  - v (Option), 68
- umask, 50, 55
  - S (Option), 50
- Umgebungsvariable
  - DEBCONF\_FRONTEND, 214
  - DEBCONF\_PRIORITY, 215
  - EDITOR, 41, 134
  - LD\_LIBRARY\_PATH, 195–197, 237
  - PATH, 195
  - VISUAL, 41
- umount, 128
- uname, 27
  - r (Option), 27
- update-grub, 147
- update-rc.d, 161–162
- usbview, 78
- useradd, 36–38, 40–42, 233
- userdel, 40, 42
  - r (Option), 40

usermod, 40, 42, 233  
/usr/lib, 193, 195–196  
/usr/lib/rpm, 218  
/usr/local, 95, 219  
/usr/local/lib, 195  
**UUID, 131**

van de Ven, Arjan, 163  
/var, 96  
/var/lib/dpkg/info, 207  
/var/lib/usbutils/usb.ids, 78  
/var/log/auth.log, 231  
/var/log/messages, 17, 150, 231  
/var/log/syslog, 150, 231  
/var/mail, 40, 133  
vi, 20, 41  
vigr, 41, 43  
    -s (Option), 43  
vipw, 41, 43, 233  
    -s (Option), 41  
VISUAL (Umgebungsvariable), 41  
vol\_id, 130  
von Münchhausen, Karl Friedrich  
    Hieronymus, 140

w, 102  
wall, 167–169, 250  
    -n (Option), 168  
    --nobanner (Option), 168  
Webmin, 19  
weiche Grenze, 133  
write, 168

xclock, 62  
xfs\_copy, 124  
xfs\_growfs, 136, 251  
xfs\_info, 124  
xfs\_mdrestore, 123  
xfs\_metadump, 123  
xfs\_quota, 133  
xfs\_repair, 123  
    -n (Option), 123  
xfsdump, 124  
xfsrestore, 124  
xterm, 30

**YUM, 223**  
yum, 223–228  
    --disablerepo (Option), 224  
    --enablerepo= (Option), 224  
    --obsoletes (Option), 225  
yumdownloader, 228  
    --resolve (Option), 228  
    --source (Option), 228  
    --urls (Option), 228

**Zombie, 61**  
zsh, 38  
Zugriffsmodus, 46