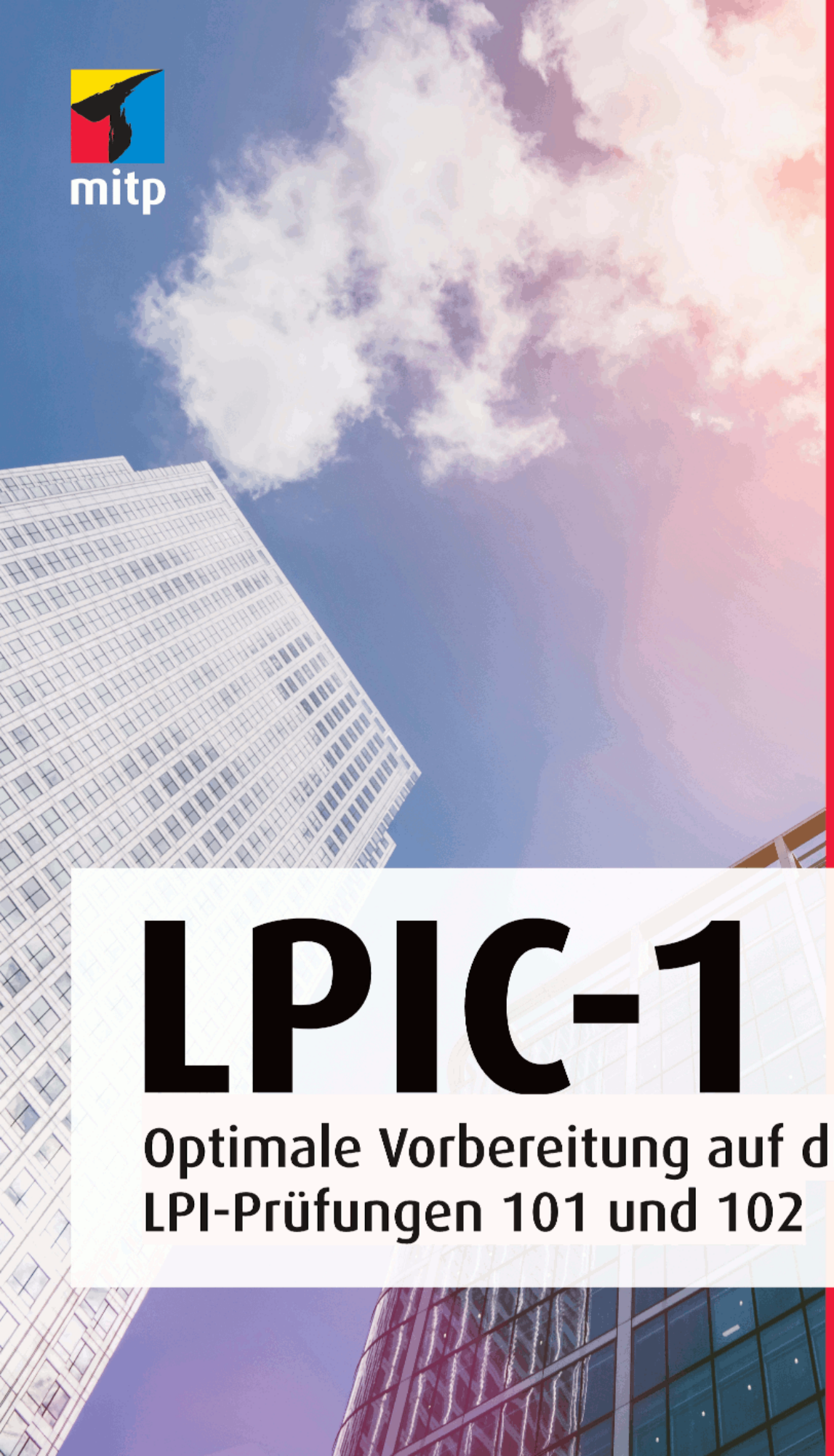

mitp

Anselm
Lingnau

7. Auflage

LPIC-1

Optimale Vorbereitung auf die LPI-Prüfungen 101 und 102

Inhaltsverzeichnis

1 Einleitung **17**
1.1 Die LPIC-1-Zertifizierung 18
1.1.1 Grundlegendes 18
1.1.2 Ablauf der Prüfungen 19
1.1.3 Prüfungsvorbereitung 21
1.2 Über dieses Buch 24
1.2.1 Zielgruppe 24
1.2.2 Aufbau und typografische Konventionen 24
1.2.3 Web-Seiten für dieses Buch 25

2 Dokumentation **27**
2.1 Überblick 28
2.2 Programminterne Hilfe 28
2.3 Die Handbuchseiten 28
2.4 Info-Seiten 31
2.5 Die HOWTOs 32
2.6 Weitere lokale Informationsquellen 32
2.7 Informationsquellen im Internet 33

3 Kommandos: Überblick und Dateiverwaltung **35**
3.1 Einleitung: Der Linux-Werkzeugkasten 36
3.2 Arbeit auf der Kommandozeile 37
3.2.1 Der Kommandointerpreter – Die Shell 37
3.2.2 Kommandos 39
3.2.3 Die Shell als komfortables Werkzeug 42
3.3 Umgang mit Dateien 46
3.3.1 Dateien benennen 46

3.3.2 Absolute und relative Pfadnamen . . . 47
3.3.3 Dateien und Verzeichnisse auflisten . . . 48
3.3.4 Kommandos für Verzeichnisse . . . 49
3.3.5 Grundlegender Umgang mit Dateien . . . 51
3.3.6 Aus eins mach zwei: Dateien verknüpfen . . . 55
3.4 Zugriffsrechte auf Dateien und Verzeichnisse . . . 59
3.4.1 Zugriffsrechte und ihre Bedeutung . . . 59
3.4.2 Die *umask* . . . 63
3.4.3 Dateieigentümer und Gruppe setzen . . . 64
3.4.4 Besondere Rechte für ausführbare Dateien . . . 65
3.4.5 Besondere Rechte für Verzeichnisse . . . 67
3.5 Suchen und Finden von Dateien . . . 69
3.5.1 Wo Dateien hingehören: Der *Filesystem Hierarchy Standard* . . . 69
3.5.2 Dateien finden . . . 71
3.5.3 Dateien finden – leicht gemacht . . . 73
3.5.4 Sonstige Kommandos . . . 75
3.6 Dateien archivieren und komprimieren . . . 76
3.6.1 Die Archivprogramme `tar` und `cpio` . . . 76
3.6.2 Komprimieren von Daten . . . 81
3.6.3 Blockweises Kopieren von Dateien und Partitionen mit `dd` . . . 85
3.6.4 Kryptografische Prüfsummen: `md5sum`, `sha1sum` & Co. . . . 86

4 Pipelines und Filter **89**
4.1 Ein-/Ausgabeumlenkung und Kommando-Pipelines . . . 90
4.1.1 Die Standardkanäle . . . 90
4.1.2 Standardkanäle umleiten . . . 91
4.1.3 Kommando-Pipelines . . . 94
4.1.4 Alternativen zu Pipelines . . . 96
4.2 Filter-Kommandos . . . 97
4.2.1 Mit Dateien arbeiten . . . 98
4.2.2 Zeichenmanipulation . . . 100
4.2.3 Spaltenmanipulation . . . 101
4.2.4 Zeilenmanipulation . . . 102

5 Reguläre Ausdrücke und Editoren **107**
5.1 Reguläre Ausdrücke . . . 108
5.1.1 Reguläre Ausdrücke: Die Grundlagen . . . 108
5.1.2 Reguläre Ausdrücke: Extras . . . 109
5.2 Dateien nach Textmustern durchsuchen – `grep` . . . 111

5.3 Automatisiertes Editieren mit sed . . . 113
5.3.1 Einsatzgebiete . . . 113
5.3.2 Zeilenspezifikation . . . 114
5.3.3 sed-Kommandos . . . 115
5.4 Texte editieren mit dem Standard-Editor vi . . . 118
5.4.1 Überblick: Warum ausgerechnet vi? . . . 118
5.4.2 Grundlegende vi-Funktionen . . . 119
5.4.3 Erweiterte Funktionen . . . 122
5.4.4 Zusammengesetzte Kommandos . . . 124

6 Prozesse **127**
6.1 Was ist ein Prozess? . . . 128
6.2 Prozessinformationen . . . 131
6.3 Prozesse erzeugen und beenden . . . 133
6.4 Prozesse beeinflussen . . . 136
6.4.1 Signale . . . 136
6.4.2 Prioritäten . . . 139
6.4.3 Die Prozess-Steuerzentrale: top . . . 141
6.5 Systeminformationen abfragen . . . 141
6.6 Programme mehrmals ausführen – watch . . . 143
6.7 Sitzungen verwalten – screen und tmux . . . 144

7 Hardware **147**
7.1 Überblick . . . 148
7.2 Firmware . . . 148
7.3 Linux und PCI (Express) . . . 150
7.4 USB . . . 153
7.5 Massenspeicher . . . 155
7.5.1 Einführung . . . 155
7.5.2 IDE, ATA und SATA . . . 156
7.5.3 SCSI . . . 157
7.6 Geräte und Treiber . . . 159
7.6.1 Überblick . . . 159
7.6.2 Das Verzeichnis /sys . . . 161
7.6.3 udev . . . 163
7.6.4 Geräteeinbindung und D-Bus . . . 164

8 Plattenspeicher **167**
8.1 Partitionierung . . . 168
8.1.1 Überblick . . . 168

8.1.2 Die traditionelle Methode (MBR) . . . 169
8.1.3 Die moderne Methode (GPT) . . . 170
8.2 Linux und Massenspeicher . . . 173
8.3 Platten partitionieren . . . 175
8.3.1 Prinzipielles . . . 175
8.3.2 Platten partitionieren mit `fdisk` . . . 178
8.3.3 Platten formatieren mit GNU `parted` . . . 181
8.3.4 `gdisk` . . . 184
8.3.5 Andere Partitionierungsprogramme . . . 185
8.3.6 Auslagerungsspeicher (Swapspace) . . . 185
8.4 Linux-Dateisysteme . . . 187
8.4.1 Überblick . . . 187
8.4.2 Die ext-Dateisysteme . . . 190
8.4.3 XFS . . . 198
8.4.4 Btrfs . . . 200
8.4.5 Noch mehr Dateisysteme . . . 203
8.5 Logical Volume Manager (LVM) . . . 205
8.6 Ein- und Aushängen von Dateisystemen . . . 207
8.6.1 `mount` und `umount` . . . 207
8.6.2 Die Datei `/etc/fstab` . . . 210
8.6.3 Wechselmedien . . . 214
8.6.4 `/etc/fstab` und systemd . . . 216
8.7 Wartung von Dateisystemen . . . 217
8.7.1 Freien Platz bestimmen . . . 217
8.7.2 Belegten Platz bestimmen . . . 218

9 Systemstart und Init-System 221
9.1 Der Systemstart . . . 222
9.1.1 Firmware: BIOS vs. UEFI . . . 222
9.1.2 Bootlader und »früher Userspace« . . . 224
9.2 Bootlader und Bootmanager . . . 225
9.2.1 Was ist ein Bootlader? . . . 225
9.2.2 GRUB Legacy . . . 226
9.2.3 GRUB 2 . . . 230
9.2.4 Kernel-Parameter . . . 232
9.3 System-V-Init . . . 233
9.3.1 Grundlagen . . . 233
9.3.2 Die Datei `/etc/inittab` . . . 234
9.3.3 Runlevel . . . 237
9.3.4 Konfiguration der Runlevel . . . 238

9.3.5 Anhalten des Systems . . . 240
9.4 Upstart . . . 242
9.5 Systemd . . . 245
9.5.1 Grundlagen . . . 245
9.5.2 Unit-Dateien . . . 247
9.5.3 Ziele . . . 249
9.6 Klappen, Knöpfe und Batterien – acpid . . . 251
9.7 Problembehandlung beim Systemstart . . . 252

10 Software- und Paketverwaltung 255
10.1 Programmbibliotheken . . . 256
10.1.1 Wofür Bibliotheken? . . . 256
10.1.2 Suche nach Bibliotheken . . . 259
10.1.3 Individuelle Anpassungen . . . 260
10.1.4 Bibliotheksversionen . . . 260
10.2 Paketverwaltung mit Debian-Werkzeugen . . . 261
10.2.1 Einleitung . . . 261
10.2.2 Das Fundament: dpkg . . . 261
10.2.3 Informationen über Pakete . . . 265
10.2.4 Verifikation von Paketen . . . 268
10.2.5 Paketverwaltung der nächsten Generation . . . 269
10.2.6 aptitude . . . 275
10.2.7 Integrität von Debian-Paketen . . . 276
10.2.8 Die debconf-Infrastruktur . . . 278
10.3 Paketverwaltung mit RPM und YUM . . . 279
10.3.1 Einleitung . . . 279
10.3.2 Installation und Aktualisierung von Paketen . . . 280
10.3.3 Deinstallation von Paketen . . . 281
10.3.4 Datenbank- und Paketanfragen . . . 282
10.3.5 Verifikation von Paketen . . . 285
10.3.6 Das Programm rpm2cpio . . . 286
10.3.7 YUM . . . 287
10.3.8 Zypper . . . 292

11 Linux und Virtualisierung 297
11.1 Virtualisierung: Grundlagen . . . 298
11.2 Automatisierung . . . 299
11.3 Virtuelle Maschinen und Container generieren . . . 300
11.4 Das Programm cloud-initprg]cloud-init . . . 301

12 Shells und Skripte **303**
12.1 Die Shell als Arbeitsplatz . . . 304
12.1.1 Einleitung . . . 304
12.1.2 Shell-Variablen . . . 304
12.1.3 Ad-hoc-Konfiguration der Shell . . . 308
12.1.4 Aliase und Funktionen . . . 309
12.1.5 Tastaturlayout und Shortcuts . . . 310
12.1.6 Anmelde-Shells und interaktive Shells . . . 311
12.1.7 Änderungen dauerhaft machen . . . 313
12.2 Einfache Shell-Skripte . . . 315
12.2.1 Warum überhaupt Shell-Skripte? . . . 315
12.2.2 Shell-Skripte richtig zum Laufen bringen . . . 315
12.2.3 Rückgabewert als Steuergröße . . . 316
12.2.4 Bedingte Ausführung . . . 319
12.2.5 Schleifen . . . 320
12.2.6 Iteration . . . 322
12.2.7 Programme starten mit `exec` . . . 323

13 Die Grafikoberfläche X11 **325**
13.1 Grundlagen von X11 . . . 326
13.1.1 Überblick . . . 326
13.2 Installation und Konfiguration von X11 . . . 330
13.2.1 Installation . . . 330
13.2.2 Die Datei `xorg.conf` . . . 332
13.3 Display-Manager . . . 339
13.4 Arbeitsumgebungen . . . 342
13.5 Fernzugriff und Zugriffskontrolle . . . 343
13.5.1 X11-Protokoll . . . 343
13.5.2 Andere Verfahren . . . 344
13.6 Linux für Behinderte . . . 346
13.6.1 Einführung . . . 346
13.6.2 Tastatur, Maus und Joystick . . . 346
13.6.3 Die Bildschirmdarstellung . . . 348
13.7 Spracherkennung . . . 350

14 Systemverwaltung **351**
14.1 Benutzerkonten und Gruppen . . . 352
14.1.1 Einführung . . . 352
14.1.2 Benutzer- und Gruppendaten . . . 353
14.1.3 Benutzerkonten und Gruppeninformationen verwalten . . . 358

14.1.4 Das Kommando `getent` . . . 365
14.2 Das Systemprotokoll . . . 366
14.2.1 Das Problem . . . 366
14.2.2 Der rsyslog-Daemon . . . 366
14.2.3 Die Antiquität: Syslogd . . . 373
14.2.4 Die »nächste Generation«: Syslog-NG . . . 373
14.2.5 Die Protokolldateien . . . 374
14.2.6 Das Programm `logrotate` . . . 375
14.2.7 Protokollierung mit systemd . . . 376
14.3 Zeitgesteuerte Vorgänge . . . 382
14.3.1 Das Problem . . . 382
14.3.2 Einmalige Ausführung von Kommandos . . . 382
14.3.3 Wiederholte Ausführung von Kommandos . . . 385
14.3.4 Geplante Ausführung von Kommandos mit systemd . . . 390
14.4 Zeitverwaltung . . . 394
14.4.1 Uhren und Zeit unter Linux . . . 394
14.4.2 Zeitsynchronisation mit NTP . . . 395
14.4.3 Zeitsynchronisation mit chrony . . . 398
14.4.4 Zeitsynchronisation mit systemd . . . 399

15 Drucken **401**
15.1 Überblick . . . 402
15.2 CUPS . . . 403
15.3 Kommandos zum Drucken . . . 404
15.3.1 Dateien drucken: `lpr` und `lp` . . . 404
15.3.2 Verfolgen von Aufträgen . . . 407
15.3.3 Stornieren von Aufträgen . . . 408
15.3.4 Standardwerte für Druckoptionen . . . 409
15.4 CUPS-Konfiguration . . . 409
15.4.1 Grundlagen . . . 409
15.4.2 Installation und Konfiguration eines CUPS-Servers . . . 412

16 Internationalisierung und Lokalisierung **415**
16.1 Überblick . . . 416
16.2 Zeichencodierungen . . . 416
16.3 Spracheneinstellung unter Linux . . . 421
16.4 Lokalisierungs-Einstellungen . . . 422
16.5 Zeitzonen . . . 426

17 Netzwerkgrundlagen **431**

17.1 Grundlagen von TCP/IP 432
17.1.1 Das *Internet Protocol* – IP 432
17.1.2 Das *Internet Control Message Protocol* – ICMP 433
17.1.3 Das *Transmission Control Protocol* – TCP 433
17.1.4 Das *User Datagram Protocol* – UDP 435
17.1.5 IP-Adressen 435
17.1.6 Ports und Dienste 438
17.2 TCP/IP-Konfiguration 440
17.2.1 Netzwerkschnittstellen 440
17.2.2 Netzwerkrouten 451
17.2.3 Namensauflösung und DNS 455
17.2.4 Der Rechnername 458
17.3 Fehlersuche bei Netzproblemen 459
17.3.1 Lokale Probleme 459
17.3.2 `ping` 459
17.3.3 `traceroute` und `tracepath` 461
17.3.4 Dienste überprüfen mit `ss`, `netstat` und `nmap` 465
17.3.5 DNS testen mit `host` und `dig` 468
17.3.6 Andere nützliche Diagnosewerkzeuge 471
17.4 IPv6 473
17.4.1 Überblick 473
17.4.2 IPv6-Adressierung 474
17.4.3 IPv6-Konfiguration 477
17.4.4 IPv6-Fehlersuche 479

18 Wichtige Netzdienste **483**
18.1 Dienste starten mit `inetd` und `xinetd` 484
18.1.1 Überblick 484
18.1.2 Die Konfiguration des `inetd` 484
18.1.3 Der TCP-Wrapper `tcpd` 485
18.1.4 Der `xinetd` 487
18.2 Dienste starten mit systemd 488
18.3 Elektronische Post 489
18.3.1 Grundlagen 489
18.3.2 MTAs für Linux 490
18.3.3 Grundlegende Funktionen 491
18.3.4 Verwaltung der Nachrichtenwarteschlange 492
18.3.5 Lokale Zustellung, Aliase und benutzerspezifische Weiterleitung . . 493
18.4 Die Secure Shell 495
18.4.1 Überblick 495

18.4.2 Anmelden auf entfernten Rechnern mit ssh . . . 496
18.4.3 Andere nützliche Anwendungen: scp und sftp . . . 499
18.4.4 Client-Authentisierung über Schlüsselpaare . . . 499
18.4.5 Portweiterleitung über SSH . . . 503

19 Sicherheit **505**
19.1 Einführung . . . 506
19.2 Sicherheit im Dateisystem . . . 506
19.3 Benutzer und Dateien . . . 510
19.4 Ressourcenlimits . . . 513
19.5 Administratorprivilegien mit sudo . . . 516
19.6 Grundlegende Netzsicherheit . . . 519
19.7 Grundlagen von GnuPG . . . 521
19.7.1 Einführung . . . 521
19.7.2 GnuPG-Schlüssel generieren und verwalten . . . 523
19.7.3 Daten verschlüsseln und entschlüsseln . . . 528
19.7.4 Dateien signieren und Signaturen prüfen . . . 529
19.7.5 GnuPG-Konfiguration . . . 530
19.7.6 Der gpg-agent . . . 531

20 Prüfungsziele **533**
20.1 Vorbemerkung . . . 534
20.2 Thema 101: Systemarchitektur . . . 534
20.3 Thema 102: Linux-Installation und -Paketverwaltung . . . 535
20.4 Thema 103: GNU- und Unix-Kommandos . . . 536
20.5 Thema 104: Geräte, Dateisysteme, FHS . . . 537
20.6 Thema 105: Shells, Skripte und Datenverwaltung . . . 538
20.7 Thema 106: Oberflächen und Desktops . . . 538
20.8 Thema 107: Administrative Aufgaben . . . 538
20.9 Thema 108: Grundlegende Systemdienste . . . 539
20.10 Thema 109: Netz-Grundlagen . . . 540
20.11 Thema 110: Sicherheit . . . 540

Index **541**

Kapitel 1 Einleitung

In diesem Kapitel ...

- erläutern wir die Grundlagen der LPIC-Zertifizierung
- lernen Sie den Ablauf der LPIC-1-Prüfungen kennen
- erfahren Sie, wie Sie sich optimal auf die LPIC-1-Prüfungen vorbereiten können
- bekommen Sie den Aufbau dieses Buches erklärt
- werden Sie auf weitere Informationsquellen zum Thema »LPI« aufmerksam gemacht

1.1 Die LPIC-1-Zertifizierung

1.1.1 Grundlegendes

Ziel

Die LPIC-Prüfungen, entwickelt vom *Linux Professional Institute*, richten sich an Administratoren von Linux-Systemen und sollen deren Kompetenz im Umgang mit Linux und den dazugehörigen Werkzeugen bestätigen. Die Zertifizierung ist distributionsunabhängig und bietet drei Stufen (LPIC-1 bis LPIC-3). Dieses Buch dient zur Vorbereitung auf die LPIC-1-Zertifizierung, die aus zwei Teilprüfungen (101 und 102) besteht. LPIC-1 (*Junior Level Linux Professional*) soll bestätigen, dass Kandidaten die folgenden Themen beherrschen:

- Umgang mit der Linux-Kommandozeile und den wichtigsten Hilfsprogrammen, inklusive einem Texteditor;
- Grundkenntnisse von wichtigen Themen wie Shellprogrammierung, Lokalisierung oder Virtualisierung;
- Einfache Administrationsaufgaben: Umgang mit Protokolldateien, Verwalten von Benutzerdaten, Systemstart und -stopp;
- Installation eines Arbeitsplatzrechners (mit X11) und Anschließen an ein existierendes lokales Netz.

Höhere Stufen

Die LPIC-2-Zertifizierung wendet sich ebenfalls in zwei Teilprüfungen an den *Advanced Level Linux Professional*, während die LPIC-3-Zertifizierung auf den *Senior Level Linux Professional* zielt. Hier haben Sie die Auswahl zwischen verschiedenen Prüfungen, die jeweils einen Themenbereich sehr detailliert abdecken. Jede davon führt zu einem LPIC-3-Zertifikat für das betreffende Thema. Mit LPIC-2 und LPIC-3 beschäftigt sich dieses Buch *nicht*.

Herstellerzertifikate

Auf den LPIC-Zertifizierungen können weitere, herstellerspezifische Zertifizierungen aufbauen. Für die UCS-Distribution der Firma Univention wird beispielsweise eine »Univention Certified Professional«-Zertifizierung (UVCP) angeboten, die aus LPIC-1 zusammen mit einer Zusatzprüfung über Univention-spezifische Themen besteht.

CompTIA Linux+

Bis Oktober 2018 bot der Interessenverband CompTIA die LPIC-1-Zertifizierung unter dem Namen »CompTIA Linux+ Powered By

LPI« an, mit Prüfungen, die zu den LPIC-1-Prüfungen identisch waren. Diese Zusammenarbeit wurde allerdings wieder aufgegeben, und die aktuelle CompTIA-Linux+-Prüfung ist vom LPI unabhängig.

1.1.2 Ablauf der Prüfungen

Prüfungsabnahme

LPI-Prüfungen werden in Pearson-VUE-Testzentren computerbasiert abgenommen: Sie werden vor einen Rechner gesetzt, der Ihnen Fragen zur Beantwortung vorlegen wird. Die meisten LPI-Fragen sind »Ankreuzfragen« mit vorgegebenen Antworten, von denen eine oder manchmal mehrere richtig sein können. Ab und zu müssen Sie auch Text eingeben, meist Kommando- oder Dateinamen. Das Ganze dauert 90 Minuten pro Prüfung, und Sie sind ganz auf sich gestellt – Sie können kein Publikum befragen, niemanden anrufen und auch keine falschen Antworten ausschließen lassen. Die beiden LPIC-1-Prüfungen gibt es derzeit in Englisch, Deutsch und diversen anderen Sprachen. Englische Prüfungen sind übrigens eventuell 120 Minuten lang, weil Sie noch einige zusätzliche Fragen »außer Konkurrenz« gestellt bekommen. Das sind neue Fragen, deren Schwierigkeitsgrad noch eingestuft werden muss, bevor sie in die tatsächliche Prüfung kommen. Rechnen Sie damit, für jede Prüfung 200 US-Dollar (ca. 190 Euro) entrichten zu müssen.

Ankreuzfragen

Dauer

Sprachen

Kosten

➪ `http://www.pearsonvue.com/lpi/`

Sonderangebote

Das LPI bietet oft auf Messen und ähnlichen Großveranstaltungen (FrOSCon etc.) verbilligte Prüfungen an. Diese sind auf Papier und nicht am Computer und kosten etwa halb so viel wie die computerbasierten Prüfungen.

Ergebnisberechnung

Nach einer computerbasierten Prüfung erfahren Sie sofort, ob Sie bestanden haben. Die genaue Ergebnisberechnung ist kompliziert: Jede Frage hat eine (nicht veröffentlichte) Punktzahl, wobei die 60 Fragen jeder Prüfung sich zu etwas wie 800 Punkten aufaddieren (genau ist das nicht festgelegt). Sie müssen 500 Punkte erreichen, wobei die Anzahl der Fragen, die Sie dafür richtig beantworten müssen, von Prüfung zu Prüfung variiert. Ziel des LPI ist eine »faire« Abgrenzung zwischen qualifizierten Kandidaten (die bestehen sollten) und unqualifizierten Kandidaten (die durchfal-

len sollten). Es ist weder notwendig noch gewollt, dass Sie in der Prüfung die »volle Punktzahl« erreichen, und es gibt auch keine »Benotung« über ein »Bestanden« oder »Durchgefallen« hinaus.

Sie können die LPI-Prüfungen grundsätzlich in jeder beliebigen Reihenfolge angehen (Sie müssen also nicht unbedingt mit der Prüfung 101 anfangen, auch wenn sich das normalerweise anbietet).

Bestanden? Wenn alles geklappt hat: Herzlichen Glückwunsch! Nach der ersten bestandenen Prüfung für ein Zertifikat haben Sie fünf Jahre Zeit, die zweite Prüfung nachzuschieben. Sollten Sie eine der beiden Prüfungen vor Oktober 2018 abgelegt haben – dem Datum, zu dem die neuen, in diesem Buch besprochenen Prüfungsinhalte in Kraft getreten sind –, dann macht das nichts: Solange Sie die Fünfjahresfrist einhalten, können Sie Ihr Zertifikat komplettieren, indem Sie die andere Prüfung gemäß den aktuellen Inhalten bestehen.

Zertifikat Sobald Sie genug bestandene Prüfungen für ein Zertifikat haben, bekommen Sie diesen Umstand per E-Mail bestätigt und das Zertifikat nebst einer coolen Plastikkarte für Ihre Brieftasche automatisch geschickt (irgendwann später). Bei »Papierprüfungen« kann es vier bis sechs Wochen dauern, bis Sie Nachricht erhalten, ob Sie bestanden haben; die Fragebögen werden zentral beim LPI in Kanada ausgewertet.

Durchgefallen? Sollte es schiefgegangen sein (unwahrscheinlich, wenn Sie dieses Buch sorgfältig durchgearbeitet haben, aber man weiß ja nie), dann müssen Sie eine Woche warten, bevor Sie es wieder probieren dürfen. Wie Sie diese Zeit verbringen können, ohne sich zu langweilen, wissen Sie dann ja. Sollte es beim zweiten Mal auch nicht geklappt haben (ja, ja …), dann ist die Wartefrist 90 Tage.

Rezertifizierung Einmal bestandene LPIC-Zertifizierungen werden theoretisch niemals ungültig. Allerdings erklärt das LPI alle Zertifikate fünf Jahre nach dem Datum der Erteilung für »inaktiv« – was Sie verhindern oder rückgängig machen können, indem Sie die betreffenden Prüfungen (in dann mutmaßlich aktualisierter Form) noch einmal ablegen oder, solange Ihr aktuelles Zertifikat noch »aktiv« ist, ein »höherwertiges« Zertifikat (LPIC-2 für LPIC-1-Inhaber) erwerben. Eine einmal bestandene Prüfung darf man erst nach zwei Jahren

wiederholen. Dies deckt sich mit dem vom LPI angestrebten Turnus für die Überarbeitung der Prüfungsinhalte.

Das LPI behält sich vor, bei der Bestätigung des LPIC-Status eines Absolventen (die das LPI auf dessen Wunsch an potenzielle Arbeitgeber und ähnliche Dritte herausgibt) neben dem Versions- und Aktivierungsstand der bestandenen Zertifizierung auch den aktuellen Versionsstand anzugeben. Ein LPIC-1-Absolvent anno 2012 dürfte im Jahr 2021 gegenüber einem Absolventen mit 2019-Zertifikat also »alt aussehen«, was die Aktualität der Zertifizierung angeht.

- Das LPI: `http://www.lpi.org/`, `http://www.lpice.eu/`
- Details über Prüfungen und deren Bewertung: `http://www.lpi.org/about-lpi/frequently-asked-questions/`
- Spielregeln für Nachprüfungen, Verfall von Prüfungen und Ähnliches: `http://www.lpi.org/about-lpi/policies/`

1.1.3 Prüfungsvorbereitung

In der Vorbereitungszeit Die LPIC-Prüfungen sind geschickt so angelegt, dass pures Büffeln nicht wirklich hilft. Sie sollten sich vor allem von der Vorstellung lösen, als Linux-Einsteiger mal ein paar Tage eine Schulung zu besuchen und anschließend die LPIC-1-Zertifizierung mit Bravour abzulegen. Viele der geprüften Zusammenhänge setzen eine Souveränität im Umgang mit dem Stoff voraus, die entweder Naturtalent oder über eine gewisse Zeit gewonnene Erfahrung bedingt. Was nicht heißen soll, dass Schulungen sinnlos sind, ganz im Gegenteil – manche lernen autodidaktisch aus Büchern wie diesem, während andere sich das nötige Wissen lieber unter fachkundiger Anleitung aneignen. In jedem Fall sollten Sie sich zwischen Lernen und Prüfung genug Zeit geben, um die Inhalte in eigener Regie zu erforschen und mit den dahinterstehenden Prinzipien vertraut zu werden. Linux ist ein auf den ersten Blick verwirrendes System, das aber trotzdem auf einigen ganz einfachen Grundlagen beruht. Wenn Sie diese Grundlagen beherrschen, ist der Rest viel besser zu verstehen.

Englisch

Polieren Sie Ihr Englisch auf. Selbst wenn Sie die Prüfungen auf Deutsch ablegen wollen, sollten Sie Englisch zumindest entziffern können, denn ein sehr großer Anteil der Systemdokumentation und viele fürs Hintergrundwissen interessante Web-Seiten stehen

nur in dieser Sprache zur Verfügung.

Prüfungsziele

Machen Sie sich mit den Prüfungen und den geforderten Inhalten vertraut. Auf den Web-Seiten des LPI finden Sie Themenlisten und Listen der wichtigsten Kommandos, Dateien und Schlagwörter für jedes Thema. Beachten Sie Anhang 20 dieses Buchs; dort haben wir zu den einzelnen Prüfungszielen Verständnisfragen zusammengestellt, die Sie im Geiste beantworten können, um sich selbst zu prüfen: Weiß ich das alles?

- Prüfungsziele für die Prüfungen 101 und 102 (unter anderem): `http://www.lpice.eu/de/lpi-zertifizierungsinhalte/lpic-1-ueberblick/`

Meilensteine

Setzen Sie sich Meilensteine. Dieses Buch behandelt den notwendigen Stoff in thematisch zusammenpassenden Abschnitten, auch da, wo die LPI-Prüfungsziele zwischen den Themen hin- und herspringen. Nehmen Sie sich den Stoff kapitelweise vor und folgen Sie auch den Querverweisen.

Internet

Verwenden Sie das Internet. Zu praktisch jedem der Prüfungsthemen gibt es zahlreiche Web-Seiten, die interessante Zusatzinformationen zu bieten haben. Wir haben auf einige relevante Seiten verwiesen, aber Sie können und sollten auch selber weitersuchen. Lesen Sie einen Linux-Nachrichtendienst wie LWN.net (`http://lwn.net/`), um ein Gefühl für die »Szene« zu bekommen.

Brain-Dumps

Machen Sie einen Bogen um sogenannte Brain-Dumps, also Sammlungen von Fragen und Antworten, an die andere Kandidaten sich nach ihrer Prüfung noch zu erinnern glauben, und die im Internet kursieren oder gar von windigen Händlern für teures Geld verkauft werden. Zum einen ist deren Qualität zum Teil hanebüchen, und zum anderen ist überhaupt nicht gesagt, dass die Fragen, die Sie in Ihrer Prüfung vorgelegt bekommen, irgend etwas mit dem Inhalt der Brain-Dumps zu tun haben, die Sie mühevoll auswendig gelernt haben. Das LPI schaufelt die Prüfungsfragen nämlich ziemlich zügig um und sucht gemeinerweise auch selbst nach Brain-Dump-Angeboten, um die darin enthaltenen Fragen besonders flott aufs Altenteil zu setzen. Hierin unterscheiden die LPI-Prüfungen sich von Führerscheinprüfungen und Einbürgerungstests. – Sie sollten sich außerdem dringend verkneifen, nach Ihrer Prüfung selber einen Brain-Dump anzufertigen und zu veröffentlichen. Vor der Prüfung müssen Sie nämlich

unterschreiben, dass Sie nichts über Ihre Fragen und Antworten weitererzählen, und das LPI behält sich vor, Ihnen gegebenenfalls mehr oder weniger nachdrücklich auf die Finger zu klopfen. Das ist die Sache wirklich nicht wert.

LPI-Registrierung

Vor der Prüfung Um LPI-Prüfungen ablegen zu können, müssen Sie beim LPI registriert sein (das machen Sie im Web). Sie erhalten eine Kennung, die Sie zur Prüfung parat haben müssen.

- LPI-Registrierung: `https://cs.lpi.org/caf/Xamman/register`

Testzentrum

Sie müssen mit dem Testzentrum Ihrer Wahl einen Termin ausmachen, an dem Sie die Prüfung ablegen können. Kommen Sie im eigenen Interesse ausgeruht, wach, mit Ihrer LPI-Kennung, mit vernünftigen Pegeln von Koffein und Alkohol im Blut und mit ausreichend Zeit, um einen Parkplatz finden, Verspätungen im ÖPNV tolerieren, allfällige Fragen »vor Ort« klären und noch einmal tief durchatmen zu können, bevor es losgeht.

In der Prüfung Sie müssen in 90 Minuten 60 Fragen beantworten. Rechnen Sie nicht damit, *alle* Fragen auf Anhieb korrekt beantworten zu können; das müssen Sie auch nicht. Es empfiehlt sich die folgende Strategie:

- Machen Sie zuerst einen »Schnelldurchlauf« durch alle Fragen und beantworten Sie alle, die Sie sicher richtig wissen, ohne lange überlegen zu müssen. Sie können andere Fragen zur »Wiedervorlage« markieren oder auch einfach vor- und zurückblättern. Machen Sie dann sukzessive weitere Durchgänge, in denen Sie sich pro Frage mehr und mehr Zeit geben, bis alle Fragen beantwortet sind.
- Lesen Sie die Fragen genau durch. Überlegen Sie, welche Antworten absurd sind und welche stimmen könnten. Passen Sie auf, dass Sie nichts Wichtiges übersehen haben. Es wird bei jeder Frage klargestellt, ob mehrere Antworten richtig sind.
- Halten Sie sich nicht zu lange an einzelnen Fragen fest, wenn Sie noch zu viele unbeantwortete Fragen haben.
- Wenn Sie (was wahrscheinlich ist) die Zeit dafür haben, dann gehen Sie alle Fragen noch einmal durch und überdenken Sie Ihre Antworten.

- Wenn Ihnen, womit aber absolut nicht zu rechnen ist, die Zeit davonläuft, dann verwenden Sie die letzten fünf Minuten, um zu »zocken« – die übrigen Antworten zu raten. Eine nicht beantwortete Frage ist immer falsch, aber eine geratene Antwort kann immerhin noch zufällig richtig sein, und es gibt keinen Abzug für falsche Antworten. Lassen Sie am Ende keine Fragen unbeantwortet.

1.2 Über dieses Buch

1.2.1 Zielgruppe

Dieses Buch richtet sich an Linux-Benutzer, die die LPIC-1-Zertifizierung anstreben. Es versucht, den Prüfungsstoff möglichst geordnet, knapp und verständlich darzustellen, ist aber kein Ersatz für herkömmliche Linux-Lehrbücher. Sie sollten es verwenden, um in der Vorbereitungszeit auf die Zertifizierungsprüfungen alles Nötige zu wiederholen, etwaige Lücken zu identifizieren und zu stopfen (wobei außer diesem Buch selbst auch andere Quellen sinnvoll sein können, auf die wir hinweisen) und die notwendige »Politur« anzubringen.

Für die Prüfung 101 bearbeiten Sie die Kapitel 2 bis 12.

Für die Prüfung 102 bearbeiten Sie die Kapitel 12 (nochmal) bis 19. (Dabei setzen wir voraus, dass Sie die Prüfung 101 schon erfolgreich abgelegt haben.)

1.2.2 Aufbau und typografische Konventionen

Struktur

Das Buch ist in Kapitel eingeteilt, die jeweils ein bestimmtes Thema behandeln, unabhängig davon, wie dieses Thema auf die LPI-Prüfungsziele verteilt ist. Obwohl dieses Buch kein Lehrbuch im traditionellen Sinne ist, haben wir uns bemüht, die Kapitel in eine didaktisch und auch für die Aufteilung in die beiden Teilprüfungen sinnvolle Reihenfolge zu bringen. In Kapitel 20 finden Sie die Prüfungsziele mit Verweisen auf die Kapitel, in denen ihre Inhalte besprochen werden. Dort ist auch die Gewichtung der einzelnen Prüfungsziele angegeben.

Randnotizen

Innerhalb des Textes finden Sie Randnotizen, die auf wichtige Stellen hinweisen. Die Randnotizen sollen es Ihnen einfacher machen, zu erkennen, worum es auf einer Seite geht, und den relevanten Absatz zu finden.

Dateinamen

Datei- und Programmnamen werden im Text in Maschinenschrift dargestellt: `/etc/passwd`, `grep`.

Tasten

Zu drückende Tasten werden wie A dargestellt; eine Taste, die mit der folgenden mit »+« verbunden ist, muss festgehalten werden, zum Beispiel Strg+C.

Beispiele

Beispiele – Kommandofolgen mit den Antworten des Systems oder Stücke aus Konfigurationsdateien – werden hellgrau hinterlegt gezeigt. Dabei sind Benutzereingaben fett, unsichtbare Eingaben wie Kennwörter dunkelgrau gedruckt:

```
login: hugo                                    Der Benutzername
Password: blafasel                             Das Kennwort
Typografisch notwendige Zeilenumbrüche, die im Original▷
◁ nicht vorhanden sind, werden wie hier gekennzeichnet.
```

Oft haben wir uninteressante Teile einer Systemausgabe aus Platzgründen weggelassen und durch »◁◁◁◁◁« ersetzt. Hin und wieder haben wir uns auch die Freiheit genommen, die Ausgabe von Programmen für die schmalen Spalten »passend zu machen«. Finden Sie, wo?

- Am Ende vieler Absätze verweisen Blöcke wie dieser auf andere Stellen im Buch oder auf interessante weitere Informationsquellen wie Bücher oder Web-Seiten.
- `https://www.lpic1-buch.de/`

1.2.3 Web-Seiten für dieses Buch

Zusatzinformationen zu vielen Themen in diesem Buch, Verweislisten zu den einzelnen Kapiteln und etwaige Fehlerberichtigungen finden Sie auf den Web-Seiten für dieses Buch unter `https://www.lpic1-buch.de/`.

Über Fragen, Meinungen, Kommentare und Vorschläge zu diesem Buch freuen wir uns unter

`info@lpic1-buch.de`

Wir wünschen Ihnen viel Spaß bei der Prüfungsvorbereitung und natürlich viel Erfolg in der Prüfung selbst!

https://www.lpic1-buch.de

Kapitel 3 Kommandos: Überblick und Dateiverwaltung

In diesem Kapitel ...

- ✔ lernen Sie die Philosophie der Kommandozeile kennen
- ✔ erleben Sie, dass sich auch mit der Kommandozeile gut arbeiten lässt
- ✔ lernen Sie, wie Sie grundlegende Dateioperationen ausführen können
- ✔ erfahren Sie einiges über Zugriffsrechte auf Dateien und Verzeichnisse
- ✔ lernen Sie fortgeschrittene Kommandos zum Archivieren und Komprimieren von Dateien kennen

3.1 Einleitung: Der Linux-Werkzeugkasten

Die Arbeit auf der Kommandozeile unterscheidet sich grundlegend vom Arbeiten unter der grafischen Oberfläche. Typisch für die grafische Oberfläche sind mächtige Programmpakete, die Komplettlösungen für ganze Problemklassen bereitstellen: The Gimp für Bildbearbeitung, LibreOffice für Büroaufgaben, Kontact für E-Mail, Firefox für den Zugriff auf und die Darstellung von Web-Seiten und so weiter. Zwar gibt es auch E-Mail-Programme und Web-Browser für die Kommandozeile, aber die meisten solcher Programme sind nicht nur eine abgespeckte Version der grafischen Variante. Typische Kommandozeilen-Programme lösen Probleme oft nicht direkt, sondern dienen vor allem als Werkzeug, um spezielle Problemlösungen zu generieren. Normal für die Kommandozeile sind also Programme wie `find`, das nicht viel mehr kann, als Dateien zu finden, oder `grep`, das nicht viel mehr kann, als Dateien nach bestimmten Texten zu durchsuchen. Typische Unix-Programme begnügen sich oft damit, nur eine Sache zu erledigen, diese aber richtig.

Werkzeugkastenprinzip

Dieses »Werkzeugkastenprinzip« ist eine der Grundlagen von Unix – und damit Linux. Das System verfügt über eine große Menge von Systemprogrammen, die jeweils eine (konzeptuell) simple Aufgabe erledigen. Diese Programme können dann von anderen Programmen als »Bausteine« verwendet werden und ersparen es den Autoren jener Programme, die entsprechende Funktionalität selbst zu entwickeln. So hat zum Beispiel nicht jedes Programm eine eigene Sortierfunktion, sondern viele Programme greifen auf das Kommando `sort` zurück. Dieser modulare Aufbau hat neben der Vereinfachung für die Programmierer, die nicht ständig neue Sortierroutinen schreiben oder zumindest in ihre Programme einbauen müssen, den Vorteil, dass bei einer Fehlerkorrektur oder Optimierung von `sort` alle Programme, die darauf zugreifen, ebenfalls profitieren, und das, ohne dass man sie explizit anpassen muss (meistens jedenfalls).

Ein Beispiel mag dies verdeutlichen: Der Aufruf von

```
$ man bash
```

zeigt die Handbuchseite des Programms `bash` an. In Wirklichkeit passiert aber Folgendes: `man` sucht und findet die Handbuchsei-

te /usr/share/man/man1/bash.1.gz. Da sie komprimiert ist, ruft es gzip auf, um sie zu dekomprimieren. Danach wird die Handbuchseite für die Ausgabe auf einem Terminal durch groff formatiert und durch less oder more seitenweise angezeigt.

Damit dieser Werkzeugkastenansatz erfolgreich funktionieren kann, muss das System aber einige Bedingungen erfüllen:

- Programme müssen miteinander kombiniert werden können. Dies wird zum einen durch das Konzept der Pipeline erreicht, das es erlaubt, eine Art Fließbandverarbeitung zu realisieren (Abschnitt 4.1). Zum anderen stellt die Kommandozeile durch ihre Programmiermöglichkeiten die Infrastruktur bereit, komplexe Kombinationen von Kommandos zu bilden (Abschnitt 12.2).
- Programme müssen miteinander kommunizieren können, sie müssen die gleiche Sprache sprechen. Unter Unix haben sich dafür (zeilenorientierte) Textdateien etabliert. Nur weil die meisten Werkzeuge Textdateien lesen und schreiben (Abschnitt 4.2), sind sie miteinander kompatibel.
- Wenn Sie mehrere Werkzeuge kombinieren müssen, um Ihr jeweiliges Problem zu lösen, kann dies schnell in viel Tipperei ausarten. Die Kommandozeile muss Sie so unterstützen, dass Sie effektiv arbeiten können.

➪ »Opening the Software Toolbox« in der GNU-coreutils-Dokumentation (aufzurufen mit »info coreutils 'Opening the software toolbox'«)

➪ B. W. Kernighan, R. Pike, *The Unix Programming Environment* (Prentice Hall, Inc., 1984), http://cm.bell-labs.com/cm/cs/upe/

3.2 Arbeit auf der Kommandozeile

3.2.1 Der Kommandointerpreter – Die Shell

Was ist eine Shell? Für den Anwender ist eine direkte Kommunikation mit dem Betriebssystem nicht möglich. Das geht nur über Programme, die die Systemaufrufe des Systemkerns ansprechen. Irgendwie müssen Sie also solche Programme starten können. Diese Aufgabe übernimmt die Shell, ein besonderes Anwen-

dungsprogramm, das (meistens) Kommandos von der Tastatur liest und diese – zum Beispiel – als Programmaufrufe interpretiert und ausführt. Die Shell stellt damit eine Art »Oberfläche« für den Computer dar, die das eigentliche Betriebssystem wie eine Muschelschale (engl. *shell*) umschließt, das dadurch nicht direkt sichtbar ist. Die Shell selbst ist nur ein Programm unter vielen, das auf das Betriebssystem zugreift.

Bourne-Shell

C-Shell

Korn-Shell

Schon das allererste Unix – Ende der 1960er Jahre – hatte eine Shell. Die älteste Shell, die heute noch außerhalb von Museen zu finden ist, wurde Mitte der 1970er Jahre für »Unix Version 7« von Stephen L. Bourne entwickelt. Diese nach ihm benannte Bourne-Shell enthält alle grundlegenden Funktionen und erfreute sich weiter Verbreitung, ist heute aber nur mehr sehr selten in ihrer ursprünglichen Form anzutreffen. Daneben zählen die C-Shell, an der *University of California in Berkeley* entstanden und (extrem vage) an die Programmiersprache C angelehnt, sowie die zur Bourne-Shell weitgehend kompatible, aber mit einem größeren Funktionsumfang ausgestattete Korn-Shell (von David Korn, ebenfalls bei AT&T) zu den klassischen Unix-Shells.

Bourne-Again-Shell

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

Da die Shell – ungeachtet ihrer zentralen Bedeutung – ein normales Anwendungsprogramm ist, kann sie leicht durch andere Programme ersetzt werden. Wenn im System mehrere verschiedene Shells zur Verfügung stehen (der Regelfall), lässt sich mit folgenden Befehlen zwischen diesen umschalten:

`sh` für eine einfache, an die klassische Bourne-Shell angelehnte Shell (falls vorhanden – bei Linux ist auch `sh` oft die Bash).

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

`ksh` für die Korn-Shell.

`csh` für die C-Shell.

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

Aktuelle Shell?

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

Shell verlassen

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

Bash als Standard

Alle erwähnten Shells unterscheiden sich mehr oder weniger in Syntax und Programmierung, die Auswahl bleibt letztendlich den persönlichen Vorlieben des Anwenders überlassen. Da sich jedoch, wie erwähnt, auf Linux-Systemen die Bash als Standard etabliert hat und diese Shell auch für die LPIC-1-Prüfung vorausgesetzt wird, konzentrieren wir uns auf diese Variante.

3.2.2 Kommandos

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

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

Eingabeaufforderung

Die Shell zeigt eine Eingabeaufforderung (engl. *prompt*) auf dem Bildschirm an. Die Eingabeaufforderung ist frei konfigurierbar; traditionell besteht sie für normale Benutzer aus »$ « oder »% « und für den Systemadministrator aus einem »# «. (Viele Linux-Distributionen verwenden aufwendigere Eingabeaufforderungen.)

Sollte die Shell am Ende einer Eingabezeile noch nicht zufrieden sein, beispielsweise weil Sie Anführungszeichen aufgemacht, aber nicht geschlossen haben, so macht sie dies durch eine andere Eingabeaufforderung (»> «) kenntlich:

```
$ echo 'Anfang
> Ende'
Anfang
Ende
```

Wie ist ein Kommando aufgebaut? Unter einem Kommando verstehen wir grundsätzlich eine längere Folge von Zeichen, die durch die Eingabetaste abgeschlossen und danach von der Shell ausgewertet wird. Kommandos sind zumeist der englischen Sprache entlehnt und ergeben in ihrer Gesamtheit eine eigene »Kommandosprache«. Diese Sprache muss gewissen Regeln, einer Syntax, gehorchen, damit sie von der Shell »verstanden« werden kann.

Syntax

Wörter

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

Erstes Wort: Kommando

Groß- und Kleinschreibung

Für DOS- und Windows-Anwender ergibt sich hier ein möglicher Stolperstein, da die Shell zwischen Groß- und Kleinschreibung unterscheidet. Linux-Kommandos werden in der Regel komplett kleingeschrieben und nicht anders verstanden.

Parameter

Alle weiteren Wörter in der Kommandozeile sind Parameter, die das Kommando genauer spezifizieren. Die Shell übergibt die zur Trennung dienenden Leerzeichen jedoch bei der Kommandoausführung nicht mit, sie dienen lediglich zur Abgrenzung der Parameter vom Kommando. Deutlich wird dies an einem kleinen Beispiel.

Das Kommando `echo` gibt alle übergebenen Parameter, getrennt durch jeweils ein Leerzeichen, auf dem Bildschirm aus, also:

```
$ echo schauen wir mal
schauen wir mal
$ echo ein   riesiges    loch
ein riesiges loch
```

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

Optionen

- Parameter, die mit einem Minuszeichen »-« beginnen, heißen Optionen. Diese können auftreten, müssen es aber nicht, sie sind eben »optional«. Bildlich gesprochen handelt es sich dabei um »Schalter«, mit denen Sie Aspekte des jeweiligen Kommandos ein- oder ausschalten können. Möchten Sie mehrere Optionen übergeben, können diese (meistens) hinter einem Minus-

zeichen zusammengefasst werden, die Parameterfolge »`-a -l -F`« entspricht also »`-alF`«. Viele Programme haben mehr Optionen, als sich leicht merkbar auf Buchstaben abbilden lassen, oder unterstützen aus Gründen der besseren Lesbarkeit »lange Optionen« (oft zusätzlich zu »normalen« Optionen, die dasselbe tun). Lange Optionen werden mit zwei Minuszeichen[1] eingeleitet und können nicht zusammengefasst werden: »`bla --fasel --blubb`«.

Lange Optionen

- Parameter ohne einleitendes Minuszeichen nennen wir »Argumente«. Dies sind oftmals die Namen der Dateien, die mit dem entsprechenden Kommando bearbeitet werden sollen.

Argumente

Arten von Kommandos In Shells gibt es prinzipiell zwei Arten von Kommandos:

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

Externe Kommandos (zum Beispiel `sort`, `who` oder `ls`) Das sind ausführbare Dateien, die im Dateisystem üblicherweise in den Verzeichnissen `/bin` oder `/usr/bin` abgelegt sind. Auch vom Benutzer selbst erstellte Kommandos gehören zu dieser Kategorie.

Extern oder intern?

Um die Art eines Kommandos herauszufinden, können Sie das Kommando `type` benutzen. Übergeben Sie hier den Befehlsnamen als Argument, wird die Art des Kommandos oder der entsprechende Verzeichnispfad auf dem Bildschirm ausgegeben, etwa

```
$ type echo
echo is a shell builtin
$ type sort
sort is /usr/bin/sort
```

[1] Zumindest ist dies bei Programmen des GNU-Projekts so. Bei Programmen, die keine kurzen Optionen unterstützen, können auch lange Optionen mit nur einem Minuszeichen vorkommen.

3.2.3 Die Shell als komfortables Werkzeug

Wer zum ersten Mal von einem grafisch orientierten Betriebssystem aus kommend mit einer Kommandozeile in Berührung kommt, empfindet in der Regel ein gewisses Unbehagen und weiß gar nicht so recht, was er tun soll. Dabei verfügt die Bash über viele hilfreiche Funktionen, die das Arbeiten mit ihr so angenehm machen können, dass sie eine ernstzunehmende Alternative zu grafischen Benutzeroberflächen darstellt:

Kommandoeditor Die Kommandozeilen lassen sich wie mit einem Texteditor bearbeiten. Die Schreibmarke kann also in der Zeile hin- und herbewegt werden, Zeichen können beliebig gelöscht oder hinzugefügt werden usw., bis die Eingabe durch Betätigen der Eingabetaste beendet wird. Dabei imitiert die Bash das Verhalten des Editors `emacs`, wahlweise auch das des `vi`, und deckt damit die beiden bedeutendsten Editoren unter Linux ab.

➪ Abschnitt »READLINE« in `bash`(1)

Kommandoabbruch Bei den vielen Linux-Kommandos kann es durchaus vorkommen, dass ein Name verwechselt oder ein falscher Parameter übergeben wurde. Aus diesem Grund können Sie ein Kommando während der Durchführung abbrechen. Hierzu müssen Sie nur die Tasten Strg+c gleichzeitig drücken. Die Reaktion auf diese Tastenkombination kann unter Umständen etwas auf sich warten lassen.

Die Vorgeschichte Die Shell nimmt alle Tastatureingaben des Anwenders entgegen und merkt sich diese. Sie können sich dann mittels der Cursortasten ↑ und ↓ durch die zuletzt eingegebenen Befehle bewegen. Diese Liste wird beim ordnungsgemäßen Verlassen des Systems in der versteckten Datei `~/.bash_history` gespeichert und steht nach dem nächsten Anmelden wieder zur Verfügung. Die Liste kann im Übrigen auch mit Strg+r durchsucht werden.

`history` Das Kommando `history` gibt die komplette Vorgeschichte aus (mit Zeilennummern verbrämt). »`history 20`« gibt nur die letzten 20 Kommandos aus (Sie verstehen schon). Die Zeilennummern können Sie verwenden, um Kommandos aus der Vorgeschichte direkt zu recyceln, indem Sie etwas wie

```
$ history
  ⋘⋘⋘⋘
    333  make install
  ⋘⋘⋘⋘
$ !333
make install
⋘⋘⋘⋘
```

eingeben. Über !-Ausdrücke können Sie Kommandos aus der Vorgeschichte in zahllosen fremdartigen und wundervollen Variationen weiterverwenden. Lesen Sie die Details in der Bash-Dokumentation nach und staunen Sie – eingedenk der Tatsache, dass in der LPI-101-Prüfung nur die elementaren Grundlagen gefragt werden. Auch das `history`-Kommando selbst hat übrigens die eine oder andere Option zu bieten.

- ➪ Abschnitte »HISTORY« und »HISTORY EXPANSION« in `bash(1)`
- ➪ »`help history`«

Komplettierung von Kommando- bzw. Dateinamen

Autovervollständigung Eine große Arbeitserleichterung ist die Fähigkeit der Bash zur automatischen Komplettierung von Kommando- bzw. Dateinamen. Durch Drücken der Tab-Taste werden unvollständige Eingaben vollendet, sofern der gewünschte Kommando- oder Dateiname eindeutig identifiziert werden kann. Für die Kommandonamenvervollständigung zieht die Bash neben den internen Kommandos alle ausführbaren Dateien in den Verzeichnissen heran, die in der Variable `PATH` aufgezählt sind; für die Dateinamenvervollständigung betrachtet sie die im aktuellen bzw. angegebenen Verzeichnis befindlichen Dateien. Existieren hierbei mehrere Dateien, deren Bezeichnungen gleich beginnen, vervollständigt die Shell die Eingabe so weit wie möglich und gibt durch ein akustisches Signal zu erkennen, dass der Datei- bzw. Kommandoname noch immer unvollendet sein kann. Ein erneutes Drücken von Tab listet dann die übrigen Möglichkeiten auf.

Befehlsliste

Die Eingabe eines einzelnen Buchstabens mit zweimaligem Drücken der Tab-Taste liefert eine Liste aller verfügbaren Befehle mit dem gewünschten Anfangsbuchstaben. Dies ist zum Beispiel hilfreich, um sich die Schreibweise selten gebrauchter Befehle ins Gedächtnis zurückzurufen.

Tastaturkürzel	Funktion
↑ bzw. ↓	durch zuletzt eingegebene Kommandos blättern
Strg+r	History durchsuchen
← bzw. →	Cursor in Kommandozeile bewegen
Pos1 oder Strg+a	Cursor an Zeilenanfang setzen
Ende oder Strg+e	Cursor an Zeilenende setzen
⇐ bzw. Entf	Zeichen vor bzw. nach Cursor löschen
Strg+k	bis Zeilenende löschen
Strg+l	Bildschirm löschen
Strg+c	Kommando abbrechen
Strg+d	Eingabe abbrechen (in der Login-Shell: Abmelden)

Tabelle 3.1: Tastaturkürzel innerhalb der Bash (Auswahl)

In der Ausgabe blättern Neue Zeilen werden erwartungsgemäß immer unterhalb der letzten ausgegeben. Der Bildschirm lässt jedoch nur die Darstellung einer bestimmten Zeilenanzahl zu. Wird diese überschritten, verschwinden die ältesten Zeilen nach oben aus dem sichtbaren Bereich. Das bedeutet jedoch nicht, dass das System diese vergessen hätte. Es verfügt nämlich über einen Speicher, der die Inhalte mehrerer kompletter Bildschirme aufnehmen kann. Möchten Sie sich also weiter zurückliegende Darstellungen noch einmal anschauen, können Sie mit den Tasten ⇑+Bild ↑ bzw. ⇑+Bild ↓ zwischen den Bildschirmseiten vor- und zurückblättern. Eingaben sind aber stets nur am Ende der letzten Seite, also in der aktuellen Zeile, möglich. – Das ist eigentlich keine Leistung der Bash, sondern eine des Treibers für virtuelle Konsolen oder des grafischen Terminalemulators. Der Umfang des gespeicherten alten Materials hängt darum von diesem Programm ab und kann deutlich verschieden ausfallen.

Tastaturkürzelübersicht Tabelle 3.1 gibt eine Übersicht über die wichtigsten in der Bash verwendeten Tastaturkürzel.

Sonderzeichen Die Schreibweise eines Kommandos ist bei der Interpretation durch die Shell entscheidend. So ist es z. B. ein Unterschied, ob ein Kommando groß- oder kleingeschrieben wird (in